RM0492
Reference manual

STM32H503 line Arm®-based 32-bit MCUs

Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32H503xx microcontrollers memory and peripherals.

For ordering information, mechanical and electrical device characteristics, refer to the corresponding datasheets.

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

STM32H5 series microcontrollers include ST state-of-the-art patented technology.

Related documents

- STM32H503xx datasheet (DS14053)
- STM32H503CB/EB/KB/RB device errata (ES0561)
- STM32 Cortex®-M33 MCUs programming manual (PM0264)

Contents

1 Documentation conventions 65

1.1 General information 65

1.2 List of abbreviations for registers 65

1.3 Register reset value 65

1.4 Glossary 66

2 Memory and bus architecture 67

2.1 System architecture 67

2.1.1 Fast C-bus 68

2.1.2 S-bus 68

2.1.3 GPDMA1 and GPDMA2 buses 68

2.1.4 Bus matrix 69

2.1.5 AHB/APB bridges 69

2.2 Memory organization 70

2.2.1 Introduction 70

2.2.2 Memory map and register boundary addresses 71

2.2.3 Embedded SRAM 76

2.2.4 Flash memory overview 76

3 System security 77

3.1 Key security features 77

3.2 Resource isolation 77

3.2.1 Temporal isolation using secure hide protection (HDP) 77

3.2.2 Resource isolation using Cortex privileged mode 78

3.2.3 Resource isolation using GTZC privilege control 79

3.3 Secure execution 82

3.3.1 Memory protection unit (MPU) 82

3.3.2 Embedded flash memory write protection 82

3.3.3 Tamper detection and response 83

3.4 Unique ID 85

3.5 Product life-cycle 85

3.5.1 Life-cycle management 87

3.5.2 Recommended product settings 88

3.6 Software intellectual property protection and collaborative development . . . 89
3.6.1 Software intellectual property protection . . . 90
3.6.2 Other software intellectual property protections . . . 90

4 Boot modes . . . 91
4.1 STM32H503 boot modes . . . 91

5 RAMs configuration controller (RAMCFG) . . . 92
5.1 Introduction . . . 92
5.2 RAMCFG main features . . . 92
5.3 RAMCFG functional description . . . 92
5.3.1 Internal SRAMs features . . . 92
5.3.2 Error code correction (SRAM1, SRAM2, BKPSRAM) . . . 93
5.3.3 Write protection (SRAM2) . . . 95
5.3.4 Software erase . . . 95
5.4 RAMCFG low-power modes . . . 95
5.5 RAMCFG interrupts . . . 96
5.6 RAMCFG registers . . . 96
5.6.1 RAMCFG memory x control register (RAMCFG_MxCR) . . . 96
5.6.2 RAMCFG memory x interrupt enable register (RAMCFG_MxIER) . . . 97
5.6.3 RAMCFG memory interrupt status register (RAMCFG_MxISR) . . . 98
5.6.4 RAMCFG memory x ECC single error address register (RAMCFG_MxSEAR) . . . 98
5.6.5 RAMCFG memory x ECC double error address register (RAMCFG_MxDEAR) . . . 99
5.6.6 RAMCFG memory x interrupt clear register x (RAMCFG_MxICR) . . . 99
5.6.7 RAMCFG memory 2 write protection register 1 (RAMCFG_M2WPR1) . . . 100
5.6.8 RAMCFG memory x ECC key register (RAMCFG_MxECCKEYR) . . . 100
5.6.9 RAMCFG memory x erase key register (RAMCFG_MxERKEYR) . . . 101
5.6.10 RAMCFG register map . . . 101

6 Global privilege controller (GTZC) . . . 104
6.1 Introduction . . . 104
6.2 GTZC main features . . . 104
6.3 GTZC implementation . . . 105

6.4 GTZC functional description 107
6.4.1 GTZC block diagram 107
6.4.2 Illegal unprivileged access 107
6.4.3 Privilege controller (TZSC) 107
6.4.4 Memory protection controller - block based (MPCBB) 108
6.4.5 Power-on/reset state 108
6.5 GTZC1 TZSC registers 109
6.5.1 GTZC1 TZSC privilege configuration register 1 (GTZC1_TZSC_PRIVCFGR1) 109
6.5.2 GTZC1 TZSC privilege configuration register 2 (GTZC1_TZSC_PRIVCFGR2) 111
6.5.3 GTZC1 TZSC privilege configuration register 3 (GTZC1_TZSC_PRIVCFGR3) 112
6.5.4 GTZC1 TZSC BKPSRAM subregion A watermark configuration register (GTZC1_TZSC_MPCWM4ACFGR) 113
6.5.5 GTZC1 TZSC BKPSRAM subregion A watermark register (GTZC1_TZSC_MPCWM4AR) 114
6.5.6 GTZC1 TZSC register map 115
6.6 GTZC1 MPCBBz registers (z = 1 to 2) 116
6.6.1 GTZC1 SRAMz MPCBB privileged configuration for super-block x register (GTZC1_MPCBBz_PRIVCFGRx) (z = 1 to 2) 116
6.6.2 GTZC1 MPCBBz register map (z = 1 to 2) 116

7 Embedded flash memory (FLASH) 117
7.1 Introduction 117
7.2 FLASH main features 117
7.3 FLASH functional description 118
7.3.1 FLASH block diagram 118
7.3.2 FLASH signals 118
7.3.3 Flash memory architecture and usage 120
7.3.4 FLASH read operations 122
7.3.5 FLASH program operations 125
7.3.6 FLASH erase operations 127
7.3.7 FLASH parallel operations 129
7.3.8 Flash memory error protections 130
7.3.9 OTP and RO memory access 130
7.3.10 Flash bank swapping 132
7.3.11 FLASH reset and clocks 134

7.4 FLASH option bytes . . . 136
7.4.1 About option bytes . . . 136
7.4.2 Option-byte loading . . . 136
7.4.3 Option-byte modification . . . 137
7.4.4 Option-byte overview . . . 138
7.4.5 Description of user and system option bytes . . . 140
7.4.6 Description of data protection option bytes . . . 141
7.4.7 Description of boot address option bytes . . . 142
7.4.8 Specific rules for modifying option bytes . . . 142
7.5 FLASH security and protections . . . 143
7.5.1 Hide protection (HDP) . . . 144
7.5.2 Privileged flash memory area protection . . . 146
7.5.3 Flash memory registers privileged and unprivileged modes . . . 147
7.5.4 Flash memory banks attributes in case of bank swap . . . 148
7.5.5 Flash memory configuration protection . . . 149
7.5.6 Write protection . . . 150
7.5.7 Life cycle management . . . 150
7.5.8 Product state transitions . . . 151
7.5.9 One-time-programmable and read-only memory protections . . . 152
7.6 System memory . . . 153
7.6.1 System memory introduction . . . 153
7.6.2 RSS user functions . . . 153
7.7 FLASH low-power modes . . . 155
7.8 FLASH error management . . . 156
7.8.1 Introduction . . . 156
7.8.2 Write protection error (WRPERR) . . . 156
7.8.3 Programming sequence error (PGSERR) . . . 158
7.8.4 Strobe error (STRBERR) . . . 158
7.8.5 Inconsistency error (INCERR) . . . 159
7.8.6 Error correction code error (ECCC/ECCD) . . . 159
7.8.7 Option byte change error (OPTCHANGEERR) . . . 160
7.8.8 Miscellaneous HardFault errors . . . 161
7.9 FLASH interrupts . . . 161
7.10 FLASH registers . . . 162
7.10.1 FLASH access control register (FLASH_ACR) . . . 162
7.10.2 FLASH key register (FLASH_NSKEYR) . . . 163

7.10.3 FLASH option key register (FLASH_OPTKEYR) 164
7.10.4 FLASH operation status register (FLASH_OPSR) 165
7.10.5 FLASH option control register (FLASH_OPTCR) 165
7.10.6 FLASH status register (FLASH_NSSR) 167
7.10.7 FLASH control register (FLASH_NSCR) 168
7.10.8 FLASH clear control register (FLASH_NSCCR) 171
7.10.9 FLASH privilege configuration register (FLASH_PRIVCFGR) 172
7.10.10 FLASH HDP extension register (FLASH_HDPEXTR) 172
7.10.11 FLASH option status register (FLASH_OPTSR_CUR) 173
7.10.12 FLASH option status register (FLASH_OPTSR_PRG) 175
7.10.13 FLASH option status register 2 (FLASH_OPTSR2_CUR) 176
7.10.14 FLASH option status register 2 (FLASH_OPTSR2_PRG) 177
7.10.15 FLASH unique boot entry register (FLASH_NSBOOTR_CUR) 178
7.10.16 FLASH unique boot entry address (FLASH_NSBOOTR_PRG) 179
7.10.17 FLASH OTP block lock (FLASH_OTPBLR_CUR) 179
7.10.18 FLASH OTP block lock (FLASH_OTPBLR_PRG) 180
7.10.19 FLASH privilege register for bank 1 (FLASH_PRIVBB1R1) 180
7.10.20 FLASH write sector protection for Bank1 (FLASH_WRP1R_CUR) . . . 181
7.10.21 FLASH write sector protection for Bank1 (FLASH_WRP1R_PRG) . . . 181
7.10.22 FLASH HDP Bank1 register (FLASH_HDP1R_CUR) 182
7.10.23 FLASH HDP Bank1 register (FLASH_HDP1R_PRG) 182
7.10.24 FLASH ECC correction register (FLASH_ECCCORR) 183
7.10.25 FLASH ECC detection register (FLASH_ECCDETR) 184
7.10.26 FLASH ECC data (FLASH_ECCDR) 185
7.10.27 FLASH privilege register for Bank2 (FLASH_PRIVBB2R1) 185
7.10.28 FLASH write sector protection for Bank2 (FLASH_WRP2R_CUR) . . . 185
7.10.29 FLASH write sector protection for Bank2 (FLASH_WRP2R_PRG) . . . 186
7.10.30 FLASH HDP Bank2 register (FLASH_HDP2R_CUR) 187
7.10.31 FLASH HDP Bank2 register (FLASH_HDP2R_PRG) 187
7.10.32 FLASH register map 188

8 Instruction cache (ICACHE) 192

8.1 ICACHE introduction 192
8.2 ICACHE main features 192
8.3 ICACHE implementation 193
8.4 ICACHE functional description 193
8.4.1 ICACHE block diagram 194

8.4.2 ICACHE reset and clocks . . . 194
8.4.3 ICACHE TAG memory . . . 195
8.4.4 Direct-mapped ICACHE (1-way cache) . . . 196
8.4.5 ICACHE enable . . . 197
8.4.6 Cacheable and noncacheable traffic . . . 197
8.4.7 Cacheable accesses . . . 198
8.4.8 ICACHE maintenance . . . 198
8.4.9 ICACHE performance monitoring . . . 199
8.4.10 ICACHE boot . . . 199
8.5 ICACHE low-power modes . . . 199
8.6 ICACHE error management and interrupts . . . 200
8.7 ICACHE registers . . . 200
8.7.1 ICACHE control register (ICACHE_CR) . . . 200
8.7.2 ICACHE status register (ICACHE_SR) . . . 201
8.7.3 ICACHE interrupt enable register (ICACHE_IER) . . . 202
8.7.4 ICACHE flag clear register (ICACHE_FCR) . . . 202
8.7.5 ICACHE hit monitor register (ICACHE_HMONR) . . . 203
8.7.6 ICACHE miss monitor register (ICACHE_MMONR) . . . 203
8.7.7 ICACHE register map . . . 204

9 Power control (PWR) . . . 205
9.1 Introduction . . . 205
9.2 PWR main features . . . 205
9.3 PWR pins and internal signals . . . 206
9.4 PWR power supplies and supply domains . . . 207
9.4.1 External power supplies . . . 207
9.4.2 Internal regulator . . . 208
9.4.3 Power-up and power-down power sequences . . . 209
9.4.4 Independent analog peripherals supply . . . 209
9.4.5 USB transceivers supply . . . 210
9.4.6 Independent I/O supply rail . . . 210
9.4.7 Backup domain . . . 210
9.5 PWR system supply voltage regulation . . . 212
9.5.1 LDO embedded regulator . . . 212
9.5.2 V_{CORE} supply versus reset, voltage scaling, and low-power modes . . . 212
9.5.3 Embedded voltage regulator operating modes . . . 212

9.6 PWR power supply and temperature supervision 213
9.6.1 Power-on reset (POR)/power-down reset (PDR)/ 213
9.6.2 Brownout reset (BOR) 213
9.6.3 Programmable voltage detector (PVD) 214
9.6.4 Analog voltage detector (AVD) 216
9.6.5 VDDIO2 voltage monitor (IO2VM) 216
9.6.6 Backup domain voltage monitoring 217
9.6.7 Temperature monitoring 217
9.7 PWR power management 218
9.7.1 Voltage scaling 218
9.7.2 Power management examples 219
9.8 Power modes 220
9.8.1 Slowing down system clocks 223
9.8.2 Peripheral clock gating 223
9.8.3 Low-power modes 224
9.8.4 Sleep mode 225
9.8.5 Stop mode 226
9.8.6 Standby mode 228
9.8.7 Power modes output pins 230
9.9 PWR privileged protection 231
9.10 PWR interrupts 232
9.11 PWR registers 232
9.11.1 PWR power mode control register (PWR_PMCR) 232
9.11.2 PWR status register (PWR_PMSR) 234
9.11.3 PWR voltage scaling control register (PWR_VOSCR) 234
9.11.4 PWR voltage scaling status register (PWR_VOSSR) 235
9.11.5 PWR backup domain control register (PWR_BDCR) 236
9.11.6 PWR disable backup protection control register (PWR_DBPCR) 237
9.11.7 PWR backup domain status register (PWR_BDSR) 237
9.11.8 PWR supply configuration control register (PWR_SCCR) 238
9.11.9 PWR voltage monitor control register (PWR_VMCR) 239
9.11.10 PWR voltage monitor status register (PWR_VMSR) 240
9.11.11 PWR wakeup status clear register (PWR_WUSCR) 241
9.11.12 PWR wakeup status register (PWR_WUSR) 241
9.11.13 PWR wakeup configuration register (PWR_WUCR) 242
9.11.14 PWR I/O retention register (PWR_IORETR) 243

9.11.15 PWR privilege configuration register (PWR_PRIVCFGR) 243
9.11.16 PWR register map 244

10 Reset and clock control (RCC) 246

10.1 Introduction 246
10.2 RCC pins and internal signals 246
10.3 RCC reset functional description 246
10.3.1 Power reset 246
10.3.2 System reset 247
10.3.3 Backup domain reset 248
10.3.4 Reset source identification 248
10.4 RCC clocks functional description 249
10.4.1 HSE clock 252
10.4.2 HSI clock 253
10.4.3 CSI oscillator 254
10.4.4 HSI48 clock 256
10.4.5 PLL description 256
10.4.6 LSE clock 260
10.4.7 LSI clock 261
10.4.8 System clock (SYSCLK) selection 262
10.4.9 Handling clock generators in stop and standby modes 262
10.4.10 Clock security system (CSS) 264
10.4.11 Clock output generation (MCO1/MCO2) 265
10.4.12 Kernel clock selection 265
10.4.13 Clock measurement with TIMx and LPTIMx 268
10.4.14 Internal oscillator calibration 268
10.4.15 RTC and TAMP clock 268
10.4.16 Timer clock 269
10.4.17 Watchdog clock 269
10.4.18 Peripherals clock gating 269
10.4.19 Bus clock gating 270
10.5 RCC privilege protection modes 270
10.6 RCC low-power modes 272
10.7 RCC interrupts 273
10.8 RCC registers 274
10.8.1 RCC clock control register (RCC_CR) 274

10.8.2 RCC HSI calibration register (RCC_HSICFGR) 277
10.8.3 RCC clock recovery RC register (RCC_CRRCR) 277
10.8.4 RCC CSI calibration register (RCC_CSICFGR) 278
10.8.5 RCC clock configuration register 1 (RCC_CFGR1) 278
10.8.6 RCC CPU domain clock configuration register 2 (RCC_CFGR2) 281
10.8.7 RCC PLL clock source selection register (RCC_PLL1CFGR) 284
10.8.8 RCC PLL clock source selection register (RCC_PLL2CFGR) 286
10.8.9 RCC PLL1 dividers register (RCC_PLL1DIVR) 288
10.8.10 RCC PLL1 fractional divider register (RCC_PLL1FRACR) 289
10.8.11 RCC PLL1 dividers register (RCC_PLL2DIVR) 290
10.8.12 RCC PLL2 fractional divider register (RCC_PLL2FRACR) 291
10.8.13 RCC clock source interrupt enable register (RCC_CIER) 292
10.8.14 RCC clock source interrupt flag register (RCC_CIFR) 293
10.8.15 RCC clock source interrupt clear register (RCC_CICR) 294
10.8.16 RCC AHB1 reset register (RCC_AHB1RSTR) 296
10.8.17 RCC AHB2 peripheral reset register (RCC_AHB2RSTR) 297
10.8.18 RCC APB1 peripheral low reset register (RCC_APB1LRSTR) 298
10.8.19 RCC APB1 peripheral high reset register (RCC_APB1HRSTR) 300
10.8.20 RCC APB2 peripheral reset register (RCC_APB2RSTR) 300
10.8.21 RCC APB3 peripheral reset register (RCC_APB3RSTR) 301
10.8.22 RCC AHB1 peripherals clock register (RCC_AHB1ENR) 302
10.8.23 RCC AHB2 peripheral clock register (RCC_AHB2ENR) 303
10.8.24 RCC APB1 peripheral clock register (RCC_APB1LENR) 305
10.8.25 RCC APB1 peripheral clock register (RCC_APB1HENR) 307
10.8.26 RCC APB2 peripheral clock register (RCC_APB2ENR) 307
10.8.27 RCC APB3 peripheral clock register (RCC_APB3ENR) 308
10.8.28 RCC AHB1 sleep clock register (RCC_AHB1LPENR) 309
10.8.29 RCC AHB2 sleep clock register (RCC_AHB2LPENR) 311
10.8.30 RCC APB1 sleep clock register (RCC_APB1LLPENR) 312
10.8.31 RCC APB1 sleep clock register (RCC_APB1HLPENR) 314
10.8.32 RCC APB2 sleep clock register (RCC_APB2LPENR) 315
10.8.33 RCC APB3 sleep clock register (RCC_APB3LPENR) 316
10.8.34 RCC kernel clock configuration register (RCC_CCIPR1) 317
10.8.35 RCC kernel clock configuration register (RCC_CCIPR2) 318
10.8.36 RCC kernel clock configuration register (RCC_CCIPR3) 318
10.8.37 RCC kernel clock configuration register (RCC_CCIPR4) 319
10.8.38 RCC kernel clock configuration register (RCC_CCIPR5) 321

10.8.39 RCC backup domain control register (RCC_BDCR) . . . 322
10.8.40 RCC reset status register (RCC_RSR) . . . 324
10.8.41 RCC privilege configuration register (RCC_PRIVCFGR) . . . 325
10.8.42 RCC register map . . . 326

11 Clock recovery system (CRS) . . . 331
11.1 CRS introduction . . . 331
11.2 CRS main features . . . 331
11.3 CRS implementation . . . 331
11.4 CRS functional description . . . 332
11.4.1 CRS block diagram . . . 332
11.4.2 CRS internal signals . . . 332
11.4.3 Synchronization input . . . 333
11.4.4 Frequency error measurement . . . 333
11.4.5 Frequency error evaluation and automatic trimming . . . 334
11.4.6 CRS initialization and configuration . . . 335
11.5 CRS in low-power modes . . . 336
11.6 CRS interrupts . . . 336
11.7 CRS registers . . . 336
11.7.1 CRS control register (CRS_CR) . . . 336
11.7.2 CRS configuration register (CRS_CFGR) . . . 338
11.7.3 CRS interrupt and status register (CRS_ISR) . . . 339
11.7.4 CRS interrupt flag clear register (CRS_ICR) . . . 341
11.7.5 CRS register map . . . 341

12 General-purpose I/Os (GPIO) . . . 343
12.1 Introduction . . . 343
12.2 GPIO main features . . . 343
12.3 GPIO functional description . . . 343
12.3.1 General-purpose I/O (GPIO) . . . 345
12.3.2 I/O pin alternate function multiplexer and mapping . . . 346
12.3.3 I/O port control registers . . . 346
12.3.4 I/O port data registers . . . 347
12.3.5 I/O data bitwise handling . . . 347
12.3.6 GPIO locking mechanism . . . 347
12.3.7 I/O alternate function input/output . . . 348

12.3.8 External interrupt/wakeup lines . . . 348
12.3.9 Input configuration . . . 348
12.3.10 Output configuration . . . 349
12.3.11 Alternate function configuration . . . 349
12.3.12 Analog configuration . . . 350
12.3.13 Using the HSE or LSE oscillator pins as GPIOs . . . 350
12.3.14 Using the GPIO pins in the RTC supply domain . . . 351
12.3.15 I/Os state retention during standby mode . . . 351
12.3.16 Privileged and unprivileged modes . . . 351
12.3.17 High-speed low-voltage mode (HSLV) . . . 351
12.3.18 I/O compensation cell . . . 351
12.4 GPIO registers . . . 352
12.4.1 GPIO x port mode register (GPIOx_MODER) (x = A to D, H) . . . 352
12.4.2 GPIO x port output type register (GPIOx_OTYPER) (x = A to D, H) . . . 352
12.4.3 GPIO x port output speed register (GPIOx_OSPEEDR) (x = A to D, H) . . . 353
12.4.4 GPIO x port pull-up/pull-down register (GPIOx_PUPDR) (x = A to D, H) . . . 354
12.4.5 GPIO x port input data register (GPIOx_IDR) (x = A to D, H) . . . 354
12.4.6 GPIO x port output data register (GPIOx_ODR) (x = A to D, H) . . . 355
12.4.7 GPIO x port bit set/reset register (GPIOx_BSRR) (x = A to D, H) . . . 355
12.4.8 GPIO x port configuration lock register (GPIOx_LCKR) (x = A to D, H) . . . 356
12.4.9 GPIO x alternate function low register (GPIOx_AFRL) (x = A to D, H) . . . 357
12.4.10 GPIO x alternate function high register (GPIOx_AFRH) (x = A to D, H) . . . 357
12.4.11 GPIO x port bit reset register (GPIOx_BRR) (x = A to D, H) . . . 358
12.4.12 GPIO x high-speed low-voltage register (GPIOx_HSLVR) (x = A to D, H) . . . 359
12.4.13 GPIO register map . . . 359

13 System configuration, boot, and security (SBS) . . . 361

13.1 SBS introduction . . . 361

13.2 SBS main features . . . 361
13.3 SBS functional description . . . 362
13.3.1 SBS block diagram . . . 362
13.3.2 SBS signals . . . 362
13.3.3 SBS reset and clocks . . . 363
13.3.4 SBS system configuration . . . 363
13.3.5 SBS boot control . . . 364
13.3.6 SBS debug control . . . 366
13.4 SBS interrupts . . . 369
13.5 SBS registers . . . 369
13.5.1 SBS temporal isolation control register (SBS_HDPLCR) . . . 369
13.5.2 SBS temporal isolation status register (SBS_HDPLSR) . . . 369
13.5.3 SBS debug control register (SBS_DBGCR) . . . 370
13.5.4 SBS debug lock register (SBS_DBGLOCKR) . . . 371
13.5.5 SBS product mode and configuration register (SBS_PMCR) . . . 371
13.5.6 SBS FPU interrupt mask register (SBS_FPUIMR) . . . 372
13.5.7 SBS memory erase status register (SBS_MESR) . . . 372
13.5.8 SBS compensation cell for I/Os control and status register (SBS_CCCSR) . . . 373
13.5.9 SBS compensation cell for I/Os value register (SBS_CCVALR) . . . 374
13.5.10 SBS compensation cell for I/Os software code register (SBS_CCSWCR) . . . 375
13.5.11 SBS Class B register (SBS_CFGR2) . . . 375
13.5.12 SBS CPU lock register (SBS_CNSLCKR) . . . 376
13.5.13 SBS flift ECC NMI mask register (SBS_ECCNMIR) . . . 377
13.5.14 SBS register map . . . 377

14 Peripherals interconnect matrix . . . 379
14.1 Introduction . . . 379
14.2 Connection summary . . . 379
14.3 Interconnection details . . . 381
14.3.1 Master to slave interconnection for timers . . . 381
14.3.2 Triggers to ADCs . . . 382
14.3.3 ADC analog watchdogs as triggers to timers . . . 382
14.3.4 Triggers to DAC . . . 383
14.3.5 Triggers to DTS . . . 383
14.3.6 LP timer to LP timer connection . . . 384

14.3.7 Internal clock sources to timers . . . 384
14.3.8 Triggers to low-power timers . . . 385
14.3.9 RTC wakeup as inputs to timers . . . 385
14.3.10 Blanking sources to comparator . . . 386
14.3.11 Comparators as inputs, trigger or break signals to timers . . . 386
14.3.12 System errors as break signals to timers . . . 387
14.3.13 Triggers to GPDMA1/2 . . . 387
14.3.14 Internal analog signals to analog peripherals . . . 388
14.3.15 Clock source for the DAC sample and hold mode . . . 389
14.3.16 Internal tamper sources . . . 389
14.3.17 Output from tamper to RTC . . . 390

15 General purpose direct memory access controller (GPDMA) . . . 391

15.1 GPDMA introduction . . . 391
15.2 GPDMA main features . . . 391
15.3 GPDMA implementation . . . 392
15.3.1 GPDMA instances . . . 392
15.3.2 GPDMA channels . . . 392
15.3.3 GPDMA in low-power modes . . . 393
15.3.4 GPDMA requests . . . 393
15.3.5 GPDMA block requests . . . 397
15.3.6 GPDMA channels with peripheral early termination . . . 398
15.3.7 GPDMA triggers . . . 398
15.4 GPDMA functional description . . . 400
15.4.1 GPDMA block diagram . . . 400
15.4.2 GPDMA channel state and direct programming without any linked-list 400
15.4.3 GPDMA channel suspend and resume . . . 401
15.4.4 GPDMA channel abort and restart . . . 402
15.4.5 GPDMA linked-list data structure . . . 403
15.4.6 Linked-list item transfer execution . . . 406
15.4.7 GPDMA channel state and linked-list programming in run-to-completion mode . . . 407
15.4.8 GPDMA channel state and linked-list programming in link step mode 410
15.4.9 GPDMA channel state and linked-list programming . . . 417
15.4.10 GPDMA FIFO-based transfers . . . 419
15.4.11 GPDMA transfer request and arbitration . . . 426
15.4.12 GPDMA triggered transfer . . . 430

15.4.13 GPDMA circular buffering with linked-list programming 431
15.4.14 GPDMA transfer in peripheral flow-control mode 433
15.4.15 GPDMA privileged/unprivileged channel 434
15.4.16 GPDMA error management 434
15.5 GPDMA in debug mode 436
15.6 GPDMA in low-power modes 436
15.7 GPDMA interrupts 437
15.8 GPDMA registers 438
15.8.1 GPDMA privileged configuration register (GPDMA_PRIVCFGR) 438
15.8.2 GPDMA masked interrupt status register (GPDMA_MISR) 439
15.8.3 GPDMA channel x linked-list base address register (GPDMA_CxLBAR) 439
15.8.4 GPDMA channel x flag clear register (GPDMA_CxFCR) 440
15.8.5 GPDMA channel x status register (GPDMA_CxSR) 441
15.8.6 GPDMA channel x control register (GPDMA_CxCR) 442
15.8.7 GPDMA channel x transfer register 1 (GPDMA_CxTR1) 444
15.8.8 GPDMA channel x transfer register 2 (GPDMA_CxTR2) 448
15.8.9 GPDMA channel x block register 1 (GPDMA_CxBR1) 452
15.8.10 GPDMA channel x alternate block register 1 (GPDMA_CxBR1) 453
15.8.11 GPDMA channel x source address register (GPDMA_CxSAR) 456
15.8.12 GPDMA channel x destination address register (GPDMA_CxDAR) . . 457
15.8.13 GPDMA channel x transfer register 3 (GPDMA_CxTR3) 458
15.8.14 GPDMA channel x block register 2 (GPDMA_CxBR2) 459
15.8.15 GPDMA channel x linked-list address register (GPDMA_CxLLR) 460
15.8.16 GPDMA channel x alternate linked-list address register (GPDMA_CxLLR) 462
15.8.17 GPDMA register map 463

16 Nested vectored interrupt controller (NVIC) 465
16.1 NVIC main features 465
16.2 SysTick calibration value register 465
16.3 Interrupt and exception vectors 465

17 Extended interrupts and event controller (EXTI) 470
17.1 EXTI main features 470
17.2 EXTI block diagram 470
17.2.1 EXTI connections between peripherals and CPU 471

17.2.2 EXTI interrupt/event mapping 472
17.3 EXTI functional description 473
17.3.1 EXTI configurable event input wakeup 473
17.3.2 EXTI mux selection 474
17.4 EXTI functional behavior 474
17.5 EXTI event protection 475
17.5.1 EXTI privilege protection 476
17.6 EXTI registers 476
17.6.1 EXTI rising trigger selection register (EXTI_RTSR1) 476
17.6.2 EXTI falling trigger selection register (EXTI_FTSR1) 477
17.6.3 EXTI software interrupt event register (EXTI_SWIER1) 478
17.6.4 EXTI rising edge pending register (EXTI_RPR1) 478
17.6.5 EXTI falling edge pending register (EXTI_FPR1) 479
17.6.6 EXTI privilege configuration register (EXTI_PRIVCFGR1) 479
17.6.7 EXTI rising trigger selection register 2 (EXTI_RTSR2) 480
17.6.8 EXTI falling trigger selection register 2 (EXTI_FTSR2) 481
17.6.9 EXTI software interrupt event register 2 (EXTI_SWIER2) 481
17.6.10 EXTI rising edge pending register 2 (EXTI_RPR2) 482
17.6.11 EXTI falling edge pending register 2 (EXTI_FPR2) 483
17.6.12 EXTI privilege configuration register 2 (EXTI_PRIVCFGR2) 484
17.6.13 EXTI external interrupt selection register (EXTI_EXTICR1) 485
17.6.14 EXTI external interrupt selection register (EXTI_EXTICR2) 486
17.6.15 EXTI external interrupt selection register (EXTI_EXTICR3) 487
17.6.16 EXTI external interrupt selection register (EXTI_EXTICR4) 488
17.6.17 EXTI CPU wakeup with interrupt mask register (EXTI_IMR1) 490
17.6.18 EXTI CPU wakeup with event mask register (EXTI_EMR1) 491
17.6.19 EXTI CPU wakeup with interrupt mask register 2 (EXTI_IMR2) 492
17.6.20 EXTI CPU wakeup with event mask register 2 (EXTI_EMR2) 493
17.6.21 EXTI register map 494

18 Cyclic redundancy check calculation unit (CRC) 496
18.1 Introduction 496
18.2 CRC main features 496
18.3 CRC functional description 497
18.3.1 CRC block diagram 497
18.3.2 CRC internal signals 497

18.3.3 CRC operation . . . 497
18.4 CRC registers . . . 499
18.4.1 CRC data register (CRC_DR) . . . 499
18.4.2 CRC independent data register (CRC_IDR) . . . 499
18.4.3 CRC control register (CRC_CR) . . . 500
18.4.4 CRC initial value (CRC_INIT) . . . 501
18.4.5 CRC polynomial (CRC_POL) . . . 501
18.4.6 CRC register map . . . 502

19 Analog-to-digital converters (ADC1) . . . 503
19.1 Introduction . . . 503
19.2 ADC main features . . . 503
19.3 ADC implementation . . . 505
19.4 ADC functional description . . . 506
19.4.1 ADC block diagram . . . 506
19.4.2 ADC pins and internal signals . . . 507
19.4.3 ADC clocks . . . 510
19.4.4 ADC connectivity . . . 512
19.4.5 Slave AHB interface . . . 513
19.4.6 ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN) . . . 513
19.4.7 Single-ended and differential input channels . . . 514
19.4.8 Calibration (ADCAL, ADCALDIF, ADC_CALFACT) . . . 514
19.4.9 ADC on-off control (ADEN, ADDIS, ADRDY) . . . 517
19.4.10 Constraints when writing the ADC control bits . . . 518
19.4.11 Channel selection (SQRx, JSQRx) . . . 519
19.4.12 Channel-wise programmable sampling time (SMPR1, SMPR2) . . . 520
19.4.13 Single conversion mode (CONT = 0) . . . 522
19.4.14 Continuous conversion mode (CONT = 1) . . . 522
19.4.15 Starting conversions (ADSTART, JADSTART) . . . 523
19.4.16 ADC timing . . . 524
19.4.17 Stopping an ongoing conversion (ADSTP, JADSTP) . . . 525
19.4.18 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN,JEXTSEL, JEXTEN) . . . 526
19.4.19 Injected channel management . . . 528
19.4.20 Discontinuous mode (DISCEN, DISCNUM, JDISCEN) . . . 529
19.4.21 Queue of context for injected conversions . . . 530

19.4.22 Programmable resolution (RES) - fast conversion mode 538
19.4.23 End of conversion, end of sampling phase (EOC, JEOC, EOSMP) . . 539
19.4.24 End of conversion sequence (EOS, JEOS) 539
19.4.25 Timing diagrams example (single/continuous modes, hardware/software triggers) 540
19.4.26 Data management 542
19.4.27 Dynamic low-power features 548
19.4.28 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx) 553
19.4.29 Oversampler 557
19.4.30 Temperature sensor 563
19.4.31 VBAT supply monitoring 565
19.4.32 Monitoring the internal voltage reference 565
19.4.33 Monitoring the supply voltage 567
19.5 ADC interrupts 567
19.6 ADC registers 569
19.6.1 ADC interrupt and status register (ADC_ISR) 569
19.6.2 ADC interrupt enable register (ADC_IER) 571
19.6.3 ADC control register (ADC_CR) 573
19.6.4 ADC configuration register (ADC_CFGR) 577
19.6.5 ADC configuration register 2 (ADC_CFGR2) 581
19.6.6 ADC sample time register 1 (ADC_SMPR1) 583
19.6.7 ADC sample time register 2 (ADC_SMPR2) 584
19.6.8 ADC watchdog threshold register 1 (ADC_TR1) 585
19.6.9 ADC watchdog threshold register 2 (ADC_TR2) 586
19.6.10 ADC watchdog threshold register 3 (ADC_TR3) 587
19.6.11 ADC regular sequence register 1 (ADC_SQR1) 587
19.6.12 ADC regular sequence register 2 (ADC_SQR2) 588
19.6.13 ADC regular sequence register 3 (ADC_SQR3) 589
19.6.14 ADC regular sequence register 4 (ADC_SQR4) 590
19.6.15 ADC regular data register (ADC_DR) 591
19.6.16 ADC injected sequence register (ADC_JSQR) 591
19.6.17 ADC offset y register (ADC_OFRy) 594
19.6.18 ADC injected channel y data register (ADC_JDRy) 595
19.6.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR) 596
19.6.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR) 596
19.6.21 ADC Differential mode selection register (ADC_DIFSEL) 597

19.6.22 ADC calibration factors (ADC_CALFACT) . . . 598
19.6.23 ADC option register (ADC_OR) . . . 598
19.7 ADC common registers . . . 599
19.7.1 ADC common control register (ADC_CCR) . . . 599
19.7.2 ADC hardware configuration register (ADC_HWCFGR0) . . . 600
19.7.3 ADC version register (ADC_VERR) . . . 601
19.7.4 ADC identification register (ADC_IPDR) . . . 601
19.7.5 ADC size identification register (ADC_SIDR) . . . 602
19.8 ADC register map . . . 603

20 Digital temperature sensor (DTS) . . . 607
20.1 Introduction . . . 607
20.2 DTS main features . . . 607
20.3 DTS functional description . . . 608
20.3.1 DTS block diagram . . . 608
20.3.2 DTS internal signals . . . 608
20.3.3 DTS block operation . . . 609
20.3.4 Operating modes . . . 609
20.3.5 Calibration . . . 609
20.3.6 Prescaler . . . 609
20.3.7 Temperature measurement principles . . . 610
20.3.8 Sampling time . . . 611
20.3.9 Quick measurement mode . . . 611
20.3.10 Trigger input . . . 612
20.3.11 On-off control and ready flag . . . 612
20.3.12 Temperature measurement sequence . . . 613
20.4 DTS low-power modes . . . 614
20.5 DTS interrupts . . . 614
20.5.1 Temperature window comparator . . . 614
20.5.2 Synchronous interrupt . . . 614
20.5.3 Asynchronous wakeup . . . 614
20.6 DTS registers . . . 615
20.6.1 Temperature sensor configuration register 1 (DTS_CFGR1) . . . 615
20.6.2 Temperature sensor T0 value register 1 (DTS_T0VALR1) . . . 617
20.6.3 Temperature sensor ramp value register (DTS_RAMPVALR) . . . 617
20.6.4 Temperature sensor interrupt threshold register 1 (DTS_ITR1) . . . 618

20.6.5 Temperature sensor data register (DTS_DR) 618
20.6.6 Temperature sensor status register (DTS_SR) 619
20.6.7 Temperature sensor interrupt enable register (DTS_ITENR) 620
20.6.8 Temperature sensor clear interrupt flag register (DTS_ICIFR) 621
20.6.9 Temperature sensor option register (DTS_OR) 622
20.6.10 DTS register map 623

21 Digital-to-analog converter (DAC) 624
21.1 Introduction 624
21.2 DAC main features 624
21.3 DAC implementation 625
21.4 DAC functional description 626
21.4.1 DAC block diagram 626
21.4.2 DAC pins and internal signals 627
21.4.3 DAC clocks 628
21.4.4 DAC channel enable 628
21.4.5 DAC data format 628
21.4.6 DAC conversion 630
21.4.7 DAC output voltage 631
21.4.8 DAC trigger selection 632
21.4.9 DMA requests 632
21.4.10 Noise generation 633
21.4.11 Triangle-wave generation 634
21.4.12 DAC channel modes 635
21.4.13 DAC channel buffer calibration 638
21.4.14 Dual DAC channel conversion modes (if dual channels are available) 640
21.5 DAC in low-power modes 644
21.6 DAC interrupts 644
21.7 DAC registers 644
21.7.1 DAC control register (DAC_CR) 644
21.7.2 DAC software trigger register (DAC_SWTRGR) 648
21.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1) 648
21.7.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1) 649

21.7.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1) . . . 649
21.7.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2) . . . 650
21.7.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2) . . . 650
21.7.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2) . . . 651
21.7.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD) . . . 651
21.7.10 Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD) . . . 652
21.7.11 Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD) . . . 652
21.7.12 DAC channel1 data output register (DAC_DOR1) . . . 653
21.7.13 DAC channel2 data output register (DAC_DOR2) . . . 653
21.7.14 DAC status register (DAC_SR) . . . 654
21.7.15 DAC calibration control register (DAC_CCR) . . . 655
21.7.16 DAC mode control register (DAC_MCR) . . . 656
21.7.17 DAC channel1 sample and hold sample time register (DAC_SHSR1) . . . 657
21.7.18 DAC channel2 sample and hold sample time register (DAC_SHSR2) . . . 658
21.7.19 DAC sample and hold time register (DAC_SHHR) . . . 658
21.7.20 DAC sample and hold refresh time register (DAC_SHRR) . . . 659
21.7.21 DAC register map . . . 660

22 Comparator (COMP) . . . 663

22.1 Introduction . . . 663
22.2 COMP main features . . . 663
22.3 COMP functional description . . . 664
22.3.1 COMP block diagram . . . 664
22.3.2 COMP pins and internal signals . . . 664
22.3.3 COMP reset and clocks . . . 666
22.3.4 Comparator LOCK mechanism . . . 666
22.3.5 Hysteresis . . . 666
22.3.6 Comparator output blanking function . . . 667
22.3.7 COMP startup stabilization time . . . 667
22.3.8 COMP power and speed modes . . . 667

22.4 COMP low-power modes . . . 668
22.5 COMP interrupts . . . 668
22.5.1 Interrupt through EXTI . . . 668
22.5.2 Interrupt through the NVIC of the CPU . . . 669
22.6 SCALER function . . . 669
22.7 COMP registers . . . 670
22.7.1 Comparator status register (COMP_SR) . . . 670
22.7.2 Comparator interrupt clear flag register (COMP_ICFR) . . . 671
22.7.3 Comparator configuration register 1 (COMP_CFGR1) . . . 671
22.7.4 Comparator configuration register 2 (COMP_CFGR2) . . . 673
22.7.5 COMP register map . . . 674

23 Operational amplifiers (OPAMP) . . . 675
23.1 Introduction . . . 675
23.2 OPAMP main features . . . 675
23.3 OPAMP functional description . . . 675
23.3.1 OPAMP reset and clocks . . . 675
23.3.2 Initial configuration . . . 676
23.3.3 Signal routing . . . 676
23.3.4 OPAMP modes . . . 676
23.3.5 Calibration . . . 683
23.4 OPAMP low-power modes . . . 685
23.5 OPAMP PGA gain . . . 685
23.6 OPAMP registers . . . 685
23.6.1 OPAMP1 control/status register (OPAMP1_CSR) . . . 685
23.6.2 OPAMP1 trimming register in normal mode (OPAMP1_OTR) . . . 687
23.6.3 OPAMP1 trimming register in high-speed mode (OPAMP1_HSOTR) . 688
23.6.4 OPAMP register map . . . 688

24 True random number generator (RNG) . . . 689
24.1 Introduction . . . 689
24.2 RNG main features . . . 689
24.3 RNG functional description . . . 690
24.3.1 RNG block diagram . . . 690
24.3.2 RNG internal signals . . . 690
24.3.3 Random number generation . . . 690

24.3.4 RNG initialization . . . 693
24.3.5 RNG operation . . . 694
24.3.6 RNG clocking . . . 696
24.3.7 Error management . . . 696
24.3.8 RNG low-power use . . . 697
24.4 RNG interrupts . . . 698
24.5 RNG processing time . . . 698
24.6 RNG entropy source validation . . . 699
24.6.1 Introduction . . . 699
24.6.2 Validation conditions . . . 699
24.7 RNG registers . . . 700
24.7.1 RNG control register (RNG_CR) . . . 700
24.7.2 RNG status register (RNG_SR) . . . 702
24.7.3 RNG data register (RNG_DR) . . . 703
24.7.4 RNG noise source control register (RNG_NSCR) . . . 704
24.7.5 RNG health test control register (RNG_HTCR) . . . 705
24.7.6 RNG register map . . . 705

25 Hash processor (HASH) . . . 706
25.1 Introduction . . . 706
25.2 HASH main features . . . 706
25.3 HASH implementation . . . 707
25.4 HASH functional description . . . 707
25.4.1 HASH block diagram . . . 707
25.4.2 HASH internal signals . . . 707
25.4.3 About secure hash algorithms . . . 708
25.4.4 Message data feeding . . . 708
25.4.5 Message digest computing . . . 709
25.4.6 Message padding . . . 711
25.4.7 HMAC operation . . . 712
25.4.8 HASH suspend/resume operations . . . 714
25.4.9 HASH DMA interface . . . 716
25.4.10 HASH error management . . . 716
25.4.11 HASH processing time . . . 716
25.5 HASH interrupts . . . 717
25.6 HASH registers . . . 717

25.6.1 HASH control register (HASH_CR) 717
25.6.2 HASH data input register (HASH_DIN) 719
25.6.3 HASH start register (HASH_STR) 720
25.6.4 HASH digest registers 721
25.6.5 HASH interrupt enable register (HASH_IMR) 723
25.6.6 HASH status register (HASH_SR) 723
25.6.7 HASH context swap registers 724
25.6.8 HASH register map 725

26 Advanced-control timers (TIM1) 727

26.1 TIM1 introduction 727
26.2 TIM1 main features 728
26.3 TIM1 functional description 729
26.3.1 Block diagram 729
26.3.2 TIM1 pins and internal signals 730
26.3.3 Time-base unit 734
26.3.4 Counter modes 736
26.3.5 Repetition counter 748
26.3.6 External trigger input 749
26.3.7 Clock selection 750
26.3.8 Capture/compare channels 754
26.3.9 Input capture mode 757
26.3.10 PWM input mode 758
26.3.11 Forced output mode 759
26.3.12 Output compare mode 759
26.3.13 PWM mode 761
26.3.14 Asymmetric PWM mode 769
26.3.15 Combined PWM mode 770
26.3.16 Combined 3-phase PWM mode 771
26.3.17 Complementary outputs and dead-time insertion 772
26.3.18 Using the break function 775
26.3.19 Bidirectional break inputs 781
26.3.20 Clearing the tim_ocxref signal on an external event 782
26.3.21 6-step PWM generation 784
26.3.22 One-pulse mode 785
26.3.23 Retriggerable One-pulse mode 787
26.3.24 Pulse on compare mode 788

26.3.25 Encoder interface mode . . . 790
26.3.26 Direction bit output . . . 807
26.3.27 UIF bit remapping . . . 808
26.3.28 Timer input XOR function . . . 808
26.3.29 Interfacing with Hall sensors . . . 808
26.3.30 Timer synchronization . . . 810
26.3.31 ADC triggers . . . 815
26.3.32 DMA burst mode . . . 815
26.3.33 TIM1 DMA requests . . . 816
26.3.34 Debug mode . . . 816
26.4 TIM1 low-power modes . . . 817
26.5 TIM1 interrupts . . . 817
26.6 TIM1 registers . . . 818
26.6.1 TIM1 control register 1 (TIM1_CR1) . . . 818
26.6.2 TIM1 control register 2 (TIM1_CR2) . . . 819
26.6.3 TIM1 slave mode control register (TIM1_SMCR) . . . 823
26.6.4 TIM1 DMA/interrupt enable register (TIM1_DIER) . . . 827
26.6.5 TIM1 status register (TIM1_SR) . . . 828
26.6.6 TIM1 event generation register (TIM1_EGR) . . . 831
26.6.7 TIM1 capture/compare mode register 1 (TIM1_CCMR1) . . . 832
26.6.8 TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1) . . . 834
26.6.9 TIM1 capture/compare mode register 2 (TIM1_CCMR2) . . . 837
26.6.10 TIM1 capture/compare mode register 2 [alternate] (TIM1_CCMR2) . . . 838
26.6.11 TIM1 capture/compare enable register (TIM1_CCER) . . . 841
26.6.12 TIM1 counter (TIM1_CNT) . . . 845
26.6.13 TIM1 prescaler (TIM1_PSC) . . . 845
26.6.14 TIM1 autoreload register (TIM1_ARR) . . . 846
26.6.15 TIM1 repetition counter register (TIM1_RCR) . . . 846
26.6.16 TIM1 capture/compare register 1 (TIM1_CCR1) . . . 847
26.6.17 TIM1 capture/compare register 2 (TIM1_CCR2) . . . 847
26.6.18 TIM1 capture/compare register 3 (TIM1_CCR3) . . . 848
26.6.19 TIM1 capture/compare register 4 (TIM1_CCR4) . . . 849
26.6.20 TIM1 break and dead-time register (TIM1_BDTR) . . . 850
26.6.21 TIM1 capture/compare register 5 (TIM1_CCR5) . . . 854
26.6.22 TIM1 capture/compare register 6 (TIM1_CCR6) . . . 855

26.6.23 TIM1 capture/compare mode register 3 (TIM1_CCMR3) 856
26.6.24 TIM1 timer deadtime register 2 (TIM1_DTR2) 857
26.6.25 TIM1 timer encoder control register (TIM1_ECR) 858
26.6.26 TIM1 timer input selection register (TIM1_TISEL) 859
26.6.27 TIM1 alternate function option register 1 (TIM1_AF1) 860
26.6.28 TIM1 alternate function register 2 (TIM1_AF2) 863
26.6.29 TIM1 DMA control register (TIM1_DCR) 865
26.6.30 TIM1 DMA address for full transfer (TIM1_DMAR) 867
26.6.31 TIM1 register map 867

27 General-purpose timers (TIM2/TIM3) 870

27.1 TIM2/TIM3 introduction 870
27.2 TIM2/TIM3 main features 870
27.3 TIM2/TIM3 implementation 871
27.4 TIM2/TIM3 functional description 872
27.4.1 Block diagram 872
27.4.2 TIM2/TIM3 pins and internal signals 873
27.4.3 Time-base unit 876
27.4.4 Counter modes 878
27.4.5 Clock selection 889
27.4.6 Capture/compare channels 893
27.4.7 Input capture mode 895
27.4.8 PWM input mode 896
27.4.9 Forced output mode 897
27.4.10 Output compare mode 897
27.4.11 PWM mode 899
27.4.12 Asymmetric PWM mode 907
27.4.13 Combined PWM mode 908
27.4.14 Clearing the tim_ocxref signal on an external event 909
27.4.15 One-pulse mode 911
27.4.16 Retriggerable one-pulse mode 912
27.4.17 Pulse on compare mode 913
27.4.18 Encoder interface mode 915
27.4.19 Direction bit output 933
27.4.20 UIF bit remapping 934
27.4.21 Timer input XOR function 934
27.4.22 Timers and external trigger synchronization 934

27.4.23 Timer synchronization . . . 938
27.4.24 ADC triggers . . . 943
27.4.25 DMA burst mode . . . 944
27.4.26 TIM2/TIM3 DMA requests . . . 945
27.4.27 Debug mode . . . 945
27.4.28 TIM2/TIM3 low-power modes . . . 945
27.4.29 TIM2/TIM3 interrupts . . . 946
27.5 TIM2/TIM3 registers . . . 947
27.5.1 TIMx control register 1 (TIMx_CR1)(x = 2, 3) . . . 947
27.5.2 TIMx control register 2 (TIMx_CR2)(x = 2, 3) . . . 948
27.5.3 TIMx slave mode control register (TIMx_SMCR)(x = 2, 3) . . . 950
27.5.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2, 3) . . . 954
27.5.5 TIMx status register (TIMx_SR)(x = 2, 3) . . . 955
27.5.6 TIMx event generation register (TIMx_EGR)(x = 2, 3) . . . 957
27.5.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 2, 3) . . . 958
27.5.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2, 3) . . . 960
27.5.9 TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 2, 3) . . . 962
27.5.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2, 3) . . . 963
27.5.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2, 3) . . . 966
27.5.12 TIM3 counter (TIM3_CNT) . . . 967
27.5.13 TIM2 counter (TIM2_CNT) . . . 968
27.5.14 TIMx prescaler (TIMx_PSC)(x = 2, 3) . . . 968
27.5.15 TIM3 autoreload register (TIM3_ARR) . . . 969
27.5.16 TIM2 autoreload register (TIM2_ARR) . . . 969
27.5.17 TIM3 capture/compare register 1 (TIM3_CCR1) . . . 970
27.5.18 TIM2 capture/compare register 1 (TIM2_CCR1) . . . 971
27.5.19 TIM3 capture/compare register 2 (TIM3_CCR2) . . . 971
27.5.20 TIM2 capture/compare register 2 (TIM2_CCR2) . . . 972
27.5.21 TIM3 capture/compare register 3 (TIM3_CCR3) . . . 973
27.5.22 TIM2 capture/compare register 3 (TIM2_CCR3) . . . 974
27.5.23 TIM3 capture/compare register 4 (TIM3_CCR4) . . . 975
27.5.24 TIM2 capture/compare register 4 (TIM2_CCR4) . . . 976
27.5.25 TIMx timer encoder control register (TIMx_ECR)(x = 2, 3) . . . 977
27.5.26 TIMx timer input selection register (TIMx_TISEL)(x = 2, 3) . . . 978
27.5.27 TIMx alternate function register 1 (TIMx_AF1)(x = 2, 3) . . . 979

27.5.28 TIMx alternate function register 2 (TIMx_AF2)(x = 2, 3) 980
27.5.29 TIMx DMA control register (TIMx_DCR)(x = 2, 3) 981
27.5.30 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2, 3) 982
27.5.31 TIMx register map 983

28 Basic timers (TIM6/TIM7) 986
28.1 TIM6/TIM7 introduction 986
28.2 TIM6/TIM7 main features 986
28.3 TIM6/TIM7 functional description 987
28.3.1 TIM6/TIM7 block diagram 987
28.3.2 TIM6/TIM7 internal signals 987
28.3.3 TIM6/TIM7 clocks 988
28.3.4 Time-base unit 988
28.3.5 Counting mode 990
28.3.6 UIF bit remapping 997
28.3.7 ADC triggers 998
28.3.8 TIM6/TIM7 DMA requests 998
28.3.9 Debug mode 998
28.3.10 TIM6/TIM7 low-power modes 998
28.3.11 TIM6/TIM7 interrupts 998
28.4 TIM6/TIM7 registers 999
28.4.1 TIMx control register 1 (TIMx_CR1)(x = 6 to 7) 999
28.4.2 TIMx control register 2 (TIMx_CR2)(x = 6 to 7) 1001
28.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7) 1001
28.4.4 TIMx status register (TIMx_SR)(x = 6 to 7) 1002
28.4.5 TIMx event generation register (TIMx_EGR)(x = 6 to 7) 1002
28.4.6 TIMx counter (TIMx_CNT)(x = 6 to 7) 1002
28.4.7 TIMx prescaler (TIMx_PSC)(x = 6 to 7) 1003
28.4.8 TIMx autoreload register (TIMx_ARR)(x = 6 to 7) 1003
28.4.9 TIMx register map 1004

29 Low-power timer (LPTIM) 1005
29.1 Introduction 1005
29.2 LPTIM main features 1005
29.3 LPTIM implementation 1005
29.4 LPTIM functional description 1007

29.4.1 LPTIM block diagram . . . 1007
29.4.2 LPTIM pins and internal signals . . . 1007
29.4.3 LPTIM input and trigger mapping . . . 1009
29.4.4 LPTIM reset and clocks . . . 1010
29.4.5 Glitch filter . . . 1010
29.4.6 Prescaler . . . 1011
29.4.7 Trigger multiplexer . . . 1011
29.4.8 Operating mode . . . 1012
29.4.9 Timeout function . . . 1014
29.4.10 Waveform generation . . . 1014
29.4.11 Register update . . . 1015
29.4.12 Counter mode . . . 1016
29.4.13 Timer enable . . . 1017
29.4.14 Timer counter reset . . . 1017
29.4.15 Encoder mode . . . 1018
29.4.16 Repetition counter . . . 1019
29.4.17 Capture/compare channels . . . 1020
29.4.18 Input capture mode . . . 1021
29.4.19 PWM mode . . . 1023
29.4.20 DMA requests . . . 1025
29.4.21 Debug mode . . . 1026
29.5 LPTIM low-power modes . . . 1026
29.6 LPTIM interrupts . . . 1026
29.7 LPTIM registers . . . 1027
29.7.1 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1, 2) . . . 1028
29.7.2 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1, 2) . . . 1030
29.7.3 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1, 2) . . . 1032
29.7.4 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1, 2) . . . 1033
29.7.5 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1, 2) . . . 1035
29.7.6 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1, 2) . . . 1036
29.7.7 LPTIM configuration register (LPTIM_CFGR) . . . 1038
29.7.8 LPTIM control register (LPTIM_CR) . . . 1041

29.7.9 LPTIM compare register 1 (LPTIM_CCR1) 1042
29.7.10 LPTIM autoreload register (LPTIM_ARR) 1042
29.7.11 LPTIM counter register (LPTIM_CNT) 1043
29.7.12 LPTIM configuration register 2 (LPTIM_CFGR2) 1043
29.7.13 LPTIM repetition register (LPTIM_RCR) 1044
29.7.14 LPTIM capture/compare mode register 1 (LPTIM_CCMR1) 1045
29.7.15 LPTIM compare register 2 (LPTIM_CCR2) 1047
29.7.16 LPTIM register map 1048

30 Independent watchdog (IWDG) 1050

30.1 Introduction 1050
30.2 IWDG main features 1050
30.3 IWDG implementation 1050
30.4 IWDG functional description 1051
30.4.1 IWDG block diagram 1051
30.4.2 IWDG internal signals 1052
30.4.3 Software and hardware watchdog modes 1052
30.4.4 Window option 1053
30.4.5 Debug 1056
30.4.6 Register access protection 1056
30.5 IWDG low power modes 1057
30.6 IWDG interrupts 1057
30.7 IWDG registers 1059
30.7.1 IWDG key register (IWDG_KR) 1060
30.7.2 IWDG prescaler register (IWDG_PR) 1060
30.7.3 IWDG reload register (IWDG_RLR) 1061
30.7.4 IWDG status register (IWDG_SR) 1061
30.7.5 IWDG window register (IWDG_WINR) 1063
30.7.6 IWDG early wake-up interrupt register (IWDG_EWCR) 1063
30.7.7 IWDG register map 1065

31 System window watchdog (WWDG) 1066

31.1 Introduction 1066
31.2 WWDG main features 1066
31.3 WWDG implementation 1066
31.4 WWDG functional description 1067

31.4.1 WWDG block diagram 1067
31.4.2 WWDG internal signals 1067
31.4.3 Enabling the watchdog 1068
31.4.4 Controlling the down-counter 1068
31.4.5 How to program the watchdog timeout 1068
31.4.6 Debug mode 1069
31.5 WWDG interrupts 1070
31.6 WWDG registers 1070
31.6.1 WWDG control register (WWDG_CR) 1070
31.6.2 WWDG configuration register (WWDG_CFR) 1071
31.6.3 WWDG status register (WWDG_SR) 1072
31.6.4 WWDG register map 1072

32 Real-time clock (RTC) 1073
32.1 Introduction 1073
32.2 RTC main features 1073
32.3 RTC functional description 1073
32.3.1 RTC block diagram 1073
32.3.2 RTC pins and internal signals 1075
32.3.3 GPIOs controlled by the RTC and TAMP 1076
32.3.4 RTC privilege protection modes 1079
32.3.5 Clock and prescalers 1080
32.3.6 Real-time clock and calendar 1081
32.3.7 Calendar ultra-low power mode 1082
32.3.8 Programmable alarms 1082
32.3.9 Periodic auto-wake-up 1082
32.3.10 RTC initialization and configuration 1083
32.3.11 Reading the calendar 1086
32.3.12 Resetting the RTC 1087
32.3.13 RTC synchronization 1087
32.3.14 RTC reference clock detection 1088
32.3.15 RTC smooth digital calibration 1089
32.3.16 Timestamp function 1091
32.3.17 Calibration clock output 1091
32.3.18 Tamper and alarm output 1092
32.4 RTC low-power modes 1092

32.5 RTC interrupts . . . 1093
32.6 RTC registers . . . 1093
32.6.1 RTC time register (RTC_TR) . . . 1094
32.6.2 RTC date register (RTC_DR) . . . 1095
32.6.3 RTC subsecond register (RTC_SSR) . . . 1096
32.6.4 RTC initialization control and status register (RTC_ICSR) . . . 1096
32.6.5 RTC prescaler register (RTC_PRER) . . . 1098
32.6.6 RTC wake-up timer register (RTC_WUTR) . . . 1099
32.6.7 RTC control register (RTC_CR) . . . 1099
32.6.8 RTC privilege mode control register (RTC_PRIVCFGR) . . . 1103
32.6.9 RTC write protection register (RTC_WPR) . . . 1105
32.6.10 RTC calibration register (RTC_CALR) . . . 1105
32.6.11 RTC shift control register (RTC_SHIFTR) . . . 1107
32.6.12 RTC timestamp time register (RTC_TSTR) . . . 1108
32.6.13 RTC timestamp date register (RTC_TSDR) . . . 1109
32.6.14 RTC timestamp subsecond register (RTC_TSSSR) . . . 1109
32.6.15 RTC alarm A register (RTC_ALRMAR) . . . 1110
32.6.16 RTC alarm A subsecond register (RTC_ALRMASSR) . . . 1111
32.6.17 RTC alarm B register (RTC_ALRMBR) . . . 1112
32.6.18 RTC alarm B subsecond register (RTC_ALRMBSSR) . . . 1113
32.6.19 RTC status register (RTC_SR) . . . 1114
32.6.20 RTC masked interrupt status register (RTC_MISR) . . . 1116
32.6.21 RTC status clear register (RTC_SCR) . . . 1117
32.6.22 RTC alarm A binary mode register (RTC_ALRABINR) . . . 1118
32.6.23 RTC alarm B binary mode register (RTC_ALRBBINR) . . . 1118
32.6.24 RTC register map . . . 1120

33 Tamper and backup registers (TAMP) . . . 1122
33.1 Introduction . . . 1122
33.2 TAMP main features . . . 1122
33.3 TAMP functional description . . . 1123
33.3.1 TAMP block diagram . . . 1123
33.3.2 TAMP pins and internal signals . . . 1124
33.3.3 GPIOs controlled by the RTC and TAMP . . . 1126
33.3.4 TAMP register write protection . . . 1126
33.3.5 Backup registers protection zones . . . 1127
33.3.6 TAMP privilege protection modes . . . 1127

33.3.7 Tamper detection 1127
33.3.8 TAMP backup registers and other device secrets erase 1128
33.3.9 Tamper detection configuration and initialization 1130
33.4 TAMP low-power modes 1136
33.5 TAMP interrupts 1137
33.6 TAMP registers 1137
33.6.1 TAMP control register 1 (TAMP_CR1) 1137
33.6.2 TAMP control register 2 (TAMP_CR2) 1139
33.6.3 TAMP control register 3 (TAMP_CR3) 1140
33.6.4 TAMP filter control register (TAMP_FLTCR) 1142
33.6.5 TAMP active tamper control register 1 (TAMP_ATCR1) 1143
33.6.6 TAMP active tamper seed register (TAMP_ATSEEDR) 1145
33.6.7 TAMP active tamper output register (TAMP_ATOR) 1146
33.6.8 TAMP active tamper control register 2 (TAMP_ATCR2) 1147
33.6.9 TAMP configuration register (TAMP_CFGR) 1148
33.6.10 TAMP privilege configuration register (TAMP_PRIVCFGR) 1149
33.6.11 TAMP interrupt enable register (TAMP_IER) 1150
33.6.12 TAMP status register (TAMP_SR) 1151
33.6.13 TAMP masked interrupt status register (TAMP_MISR) 1153
33.6.14 TAMP status clear register (TAMP_SCR) 1155
33.6.15 TAMP monotonic counter 1 register (TAMP_COUNT1R) 1157
33.6.16 TAMP resources protection configuration register (TAMP_RPCFGR) 1157
33.6.17 TAMP backup x register (TAMP_BKPxR) 1158
33.6.18 TAMP register map 1159

34 Inter-integrated circuit interface (I2C) 1161

34.1 Introduction 1161
34.2 I2C main features 1161
34.3 I2C implementation 1162
34.4 I2C functional description 1162
34.4.1 I2C block diagram 1163
34.4.2 I2C pins and internal signals 1163
34.4.3 I2C clock requirements 1164
34.4.4 I2C mode selection 1164
34.4.5 I2C initialization 1165
34.4.6 I2C reset 1169

34.4.7 I2C data transfer . . . 1170
34.4.8 I2C target mode . . . 1172
34.4.9 I2C controller mode . . . 1181
34.4.10 I2C_TIMINGR register configuration examples . . . 1192
34.4.11 SMBus specific features . . . 1194
34.4.12 SMBus initialization . . . 1196
34.4.13 SMBus I2C_TIMEOUTR register configuration examples . . . 1198
34.4.14 SMBus target mode . . . 1199
34.4.15 SMBus controller mode . . . 1202
34.4.16 Wake-up from Stop mode on address match . . . 1205
34.4.17 Error conditions . . . 1206
34.5 I2C in low-power modes . . . 1208
34.6 I2C interrupts . . . 1208
34.7 I2C DMA requests . . . 1209
34.7.1 Transmission using DMA . . . 1209
34.7.2 Reception using DMA . . . 1209
34.8 I2C debug modes . . . 1209
34.9 I2C registers . . . 1210
34.9.1 I2C control register 1 (I2C_CR1) . . . 1210
34.9.2 I2C control register 2 (I2C_CR2) . . . 1213
34.9.3 I2C own address 1 register (I2C_OAR1) . . . 1215
34.9.4 I2C own address 2 register (I2C_OAR2) . . . 1215
34.9.5 I2C timing register (I2C_TIMINGR) . . . 1216
34.9.6 I2C timeout register (I2C_TIMEOUTR) . . . 1217
34.9.7 I2C interrupt and status register (I2C_ISR) . . . 1218
34.9.8 I2C interrupt clear register (I2C_ICR) . . . 1221
34.9.9 I2C PEC register (I2C_PECR) . . . 1222
34.9.10 I2C receive data register (I2C_RXDR) . . . 1222
34.9.11 I2C transmit data register (I2C_TXDR) . . . 1223
34.9.12 I2C register map . . . 1224

35 Improved inter-integrated circuit (I3C) . . . 1225
35.1 Introduction . . . 1225
35.2 I3C main features . . . 1225
35.3 I3C implementation . . . 1227
35.3.1 I3C instantiation . . . 1227

35.3.2 I3C wake-up from low-power mode(s) . . . 1227
35.3.3 I3C FIFOs . . . 1227
35.3.4 I3C triggers . . . 1227
35.3.5 I3C interrupt(s) . . . 1227
35.3.6 I3C MIPI® support . . . 1228
35.4 I3C block diagram . . . 1229
35.5 I3C pins and internal signals . . . 1229
35.6 I3C reset and clocks . . . 1230
35.6.1 I3C reset . . . 1230
35.6.2 I3C clocks and requirements . . . 1230
35.7 I3C peripheral state and programming . . . 1232
35.7.1 I3C peripheral state . . . 1232
35.7.2 I3C controller state and programming sequence . . . 1232
35.7.3 I3C target state and programming sequence . . . 1237
35.8 I3C registers and programming . . . 1241
35.8.1 I3C register set, as controller/target . . . 1241
35.8.2 I3C registers and fields use versus peripheral state, as controller . . . 1242
35.8.3 I3C registers and fields usage versus peripheral state, as target . . . 1245
35.9 I3C bus transfers and programming . . . 1247
35.9.1 I3C command set (CCCs), as controller/target . . . 1247
35.9.2 I3C broadcast/direct CCC transfer (except ENTDAA, RSTACT), as controller . . . 1251
35.9.3 I3C broadcast ENTDAA CCC transfer, as controller . . . 1253
35.9.4 I3C broadcast/direct RSTACT CCC transfer, as controller . . . 1253
35.9.5 I3C broadcast/direct CCC transfer (except ENTDAA, DEFTGTS, DEFGRPA), as target . . . 1255
35.9.6 I3C broadcast ENTDAA CCC transfer, as target . . . 1257
35.9.7 I3C broadcast DEFTGTS CCC transfer, as target . . . 1258
35.9.8 I3C broadcast DEFGRPA CCC transfer, as target . . . 1259
35.9.9 I3C direct GETSTATUS CCC response, as target . . . 1260
35.9.10 I3C private read/write transfer, as controller . . . 1261
35.9.11 I3C private read/write transfer, as target . . . 1262
35.9.12 Legacy I2C read/write transfer, as controller . . . 1263
35.9.13 I3C IBI transfer, as controller/target . . . 1264
35.9.14 I3C hot-join request transfer, as controller/target . . . 1265
35.9.15 I3C controller-role request transfer, as controller/target . . . 1266
35.10 I3C FIFOs management, as controller . . . 1267

35.10.1 C-FIFO management, as controller 1267
35.10.2 TX-FIFO management, as controller 1268
35.10.3 RX-FIFO management, as controller 1271
35.10.4 S-FIFO management, as controller 1273
35.11 I3C FIFOs management, as target 1275
35.11.1 RX-FIFO management, as target 1275
35.11.2 TX-FIFO management, as target 1276
35.12 I3C error management 1279
35.12.1 Controller error management 1279
35.12.2 Target error management 1281
35.13 I3C wake-up from low-power mode(s) 1282
35.13.1 Wake-up from Stop 1282
35.14 I3C in low-power modes 1285
35.15 I3C interrupts 1286
35.16 I3C registers 1287
35.16.1 I3C message control register (I3C_CR) 1287
35.16.2 I3C message control register [alternate] (I3C_CR) 1289
35.16.3 I3C configuration register (I3C_CFGR) 1291
35.16.4 I3C receive data byte register (I3C_RDR) 1296
35.16.5 I3C receive data word register (I3C_RDWR) 1296
35.16.6 I3C transmit data byte register (I3C_TDR) 1297
35.16.7 I3C transmit data word register (I3C_TDWR) 1298
35.16.8 I3C IBI payload data register (I3C_IBIDR) 1300
35.16.9 I3C target transmit configuration register (I3C_TGTTDR) 1301
35.16.10 I3C status register (I3C_SR) 1302
35.16.11 I3C status error register (I3C_SER) 1303
35.16.12 I3C received message register (I3C_RMR) 1305
35.16.13 I3C event register (I3C_EVR) 1306
35.16.14 I3C interrupt enable register (I3C_IER) 1310
35.16.15 I3C clear event register (I3C_CEVR) 1312
35.16.16 I3C own device characteristics register (I3C_DEVR0) 1314
35.16.17 I3C device x characteristics register (I3C_DEVRx) 1316
35.16.18 I3C maximum read length register (I3C_MAXRLR) 1318
35.16.19 I3C maximum write length register (I3C_MAXWLR) 1319
35.16.20 I3C timing register 0 (I3C_TIMINGR0) 1320
35.16.21 I3C timing register 1 (I3C_TIMINGR1) 1321

35.16.22 I3C timing register 2 (I3C_TIMINGR2) 1323
35.16.23 I3C bus characteristics register (I3C_BCR) 1324
35.16.24 I3C device characteristics register (I3C_DCR) 1325
35.16.25 I3C get capability register (I3C_GETCAPR) 1326
35.16.26 I3C controller-role capability register (I3C_CRCAPR) 1327
35.16.27 I3C get max data speed register (I3C_GETMXDSR) 1328
35.16.28 I3C extended provisioned ID register (I3C_EPIDR) 1330
35.16.29 I3C register map 1331

36 Universal synchronous/asynchronous receiver transmitter (USART/UART) 1334

36.1 Introduction 1334
36.2 USART main features 1334
36.3 USART extended features 1335
36.4 USART implementation 1335
36.5 USART functional description 1337
36.5.1 USART block diagram 1337
36.5.2 USART pins and internal signals 1337
36.5.3 USART clocks 1339
36.5.4 USART character description 1339
36.5.5 USART FIFOs and thresholds 1342
36.5.6 USART transmitter 1342
36.5.7 USART receiver 1345
36.5.8 USART baud rate generation 1352
36.5.9 Tolerance of the USART receiver to clock deviation 1354
36.5.10 USART auto baud rate detection 1355
36.5.11 USART multiprocessor communication 1357
36.5.12 USART Modbus communication 1359
36.5.13 USART parity control 1360
36.5.14 USART LIN (local interconnection network) mode 1361
36.5.15 USART synchronous mode 1363
36.5.16 USART single-wire half-duplex communication 1367
36.5.17 USART receiver timeout 1367
36.5.18 USART smartcard mode 1368
36.5.19 USART IrDA SIR ENDEC block 1372
36.5.20 Continuous communication using USART and DMA 1375
36.5.21 RS232 hardware flow control and RS485 driver enable 1377

36.5.22 USART low-power management 1380
36.6 USART in low-power modes 1383
36.7 USART interrupts 1383
36.8 USART registers 1386
36.8.1 USART control register 1 (USART_CR1) 1386
36.8.2 USART control register 1 [alternate] (USART_CR1) 1390
36.8.3 USART control register 2 (USART_CR2) 1393
36.8.4 USART control register 3 (USART_CR3) 1397
36.8.5 USART control register 3 [alternate] (USART_CR3) 1401
36.8.6 USART baud rate register (USART_BRR) 1405
36.8.7 USART guard time and prescaler register (USART_GTPR) 1405
36.8.8 USART receiver timeout register (USART_RTOR) 1406
36.8.9 USART request register (USART_RQR) 1407
36.8.10 USART interrupt and status register (USART_ISR) 1408
36.8.11 USART interrupt and status register [alternate] (USART_ISR) 1414
36.8.12 USART interrupt flag clear register (USART_ICR) 1419
36.8.13 USART receive data register (USART_RDR) 1420
36.8.14 USART transmit data register (USART_TDR) 1421
36.8.15 USART prescaler register (USART_PRESC) 1421
36.8.16 USART register map 1422

37 Low-power universal asynchronous receiver transmitter (LPUART) 1424
37.1 Introduction 1424
37.2 LPUART main features 1424
37.3 LPUART implementation 1425
37.4 LPUART functional description 1426
37.4.1 LPUART block diagram 1426
37.4.2 LPUART pins and internal signals 1427
37.4.3 LPUART clocks 1428
37.4.4 LPUART character description 1428
37.4.5 LPUART FIFOs and thresholds 1430
37.4.6 LPUART transmitter 1430
37.4.7 LPUART receiver 1434
37.4.8 LPUART baud rate generation 1438
37.4.9 Tolerance of the LPUART receiver to clock deviation 1439

37.4.10 LPUART multiprocessor communication 1440
37.4.11 LPUART parity control 1442
37.4.12 LPUART single-wire half-duplex communication 1443
37.4.13 Continuous communication using DMA and LPUART 1443
37.4.14 RS232 hardware flow control and RS485 driver enable 1446
37.4.15 LPUART low-power management 1448
37.5 LPUART in low-power modes 1451
37.6 LPUART interrupts 1452
37.7 LPUART registers 1453
37.7.1 LPUART control register 1 (LPUART_CR1) 1453
37.7.2 LPUART control register 1 [alternate] (LPUART_CR1) 1456
37.7.3 LPUART control register 2 (LPUART_CR2) 1459
37.7.4 LPUART control register 3 (LPUART_CR3) 1461
37.7.5 LPUART control register 3 [alternate] (LPUART_CR3) 1464
37.7.6 LPUART baud rate register (LPUART_BRR) 1467
37.7.7 LPUART request register (LPUART_RQR) 1467
37.7.8 LPUART interrupt and status register (LPUART_ISR) 1468
37.7.9 LPUART interrupt and status register [alternate] (LPUART_ISR) . . . 1472
37.7.10 LPUART interrupt flag clear register (LPUART_ICR) 1475
37.7.11 LPUART receive data register (LPUART_RDR) 1476
37.7.12 LPUART transmit data register (LPUART_TDR) 1477
37.7.13 LPUART prescaler register (LPUART_PRESC) 1477
37.7.14 LPUART register map 1478

38 Serial peripheral interface (SPI) 1480
38.1 Introduction 1480
38.2 SPI main features 1480
38.3 SPI implementation 1481
38.4 SPI functional description 1482
38.4.1 SPI block diagram 1482
38.4.2 SPI pins and internal signals 1483
38.4.3 SPI communication general aspects 1484
38.4.4 Communications between one master and one slave 1484
38.4.5 Standard multislave communication 1487
38.4.6 Multimaster communication 1490
38.4.7 Slave select (SS) pin management 1490

38.4.8 Ready pin (RDY) management . . . 1494
38.4.9 Communication formats . . . 1494
38.4.10 Configuring the SPI . . . 1496
38.4.11 Enabling the SPI . . . 1497
38.4.12 SPI data transmission and reception procedures . . . 1498
38.4.13 Disabling the SPI . . . 1502
38.4.14 Communication using DMA (direct memory addressing) . . . 1503
38.5 SPI specific modes and control . . . 1505
38.5.1 TI mode . . . 1505
38.5.2 SPI error flags . . . 1505
38.5.3 CRC computation . . . 1509
38.6 SPI in low-power modes . . . 1510
38.7 SPI interrupts . . . 1510
38.8 I2S main features . . . 1512
38.9 I2S functional description . . . 1512
38.9.1 I2S general description . . . 1512
38.9.2 Pin sharing with SPI function . . . 1513
38.9.3 Bitfields usable in I2S/PCM mode . . . 1513
38.9.4 Slave and master modes . . . 1514
38.9.5 Supported audio protocols . . . 1514
38.9.6 Additional serial interface flexibility . . . 1520
38.9.7 Startup sequence . . . 1522
38.9.8 Stop sequence . . . 1524
38.9.9 Clock generator . . . 1525
38.9.10 Internal FIFOs . . . 1527
38.9.11 FIFO status flags . . . 1528
38.9.12 Handling of underrun situation . . . 1528
38.9.13 Handling of overrun situation . . . 1529
38.9.14 Frame error detection . . . 1530
38.9.15 DMA interface . . . 1532
38.9.16 Programing examples . . . 1532
38.10 I2S interrupts . . . 1535
38.11 SPI/I2S registers . . . 1535
38.11.1 SPI/I2S control register 1 (SPI_CR1) . . . 1535
38.11.2 SPI/I2S control register 2 (SPI_CR2) . . . 1537
38.11.3 SPI/I2S configuration register 1 (SPI_CFG1) . . . 1538

38.11.4 SPI/I2S configuration register 2 (SPI_CFG2) 1541
38.11.5 SPI/I2S interrupt enable register (SPI_IER) 1543
38.11.6 SPI/I2S status register (SPI_SR) 1544
38.11.7 SPI/I2S interrupt/status flags clear register (SPI_IFCR) 1547
38.11.8 SPI/I2S transmit data register (SPI_TXDR) 1548
38.11.9 SPI/I2S receive data register (SPI_RXDR) 1548
38.11.10 SPI/I2S polynomial register (SPI_CRCPOLY) 1549
38.11.11 SPI/I2S transmitter CRC register (SPI_TXCRC) 1549
38.11.12 SPI/I2S receiver CRC register (SPI_RXCRC) 1550
38.11.13 SPI/I2S underrun data register (SPI_UDRDR) 1551
38.11.14 SPI/I2S configuration register (SPI_I2SCFGR) 1551
38.11.15 SPI/I2S register map 1553

39 FD controller area network (FDCAN) 1555

39.1 Introduction 1555
39.2 FDCAN main features 1557
39.3 FDCAN functional description 1558
39.3.1 FDCAN block diagram 1558
39.3.2 FDCAN pins and internal signals 1559
39.3.3 Bit timing 1560
39.3.4 Operating modes 1561
39.3.5 Error management 1570
39.3.6 Message RAM 1571
39.3.7 FIFO acknowledge handling 1580
39.3.8 FDCAN Rx FIFO element 1580
39.3.9 FDCAN Tx buffer element 1582
39.3.10 FDCAN Tx event FIFO element 1584
39.3.11 FDCAN standard message ID filter element 1585
39.3.12 FDCAN extended message ID filter element 1586
39.4 FDCAN registers 1588
39.4.1 FDCAN core release register (FDCAN_CREL) 1588
39.4.2 FDCAN endian register (FDCAN_ENDN) 1588
39.4.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP) 1588
39.4.4 FDCAN test register (FDCAN_TEST) 1589
39.4.5 FDCAN RAM watchdog register (FDCAN_RWD) 1590
39.4.6 FDCAN CC control register (FDCAN_CCCR) 1591
39.4.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP) . 1592

39.4.8 FDCAN timestamp counter configuration register (FDCAN_TSCC) . 1594
39.4.9 FDCAN timestamp counter value register (FDCAN_TSCV) 1594
39.4.10 FDCAN timeout counter configuration register (FDCAN_TOCC) . . . 1595
39.4.11 FDCAN timeout counter value register (FDCAN_TOCV) 1596
39.4.12 FDCAN error counter register (FDCAN_ECR) 1596
39.4.13 FDCAN protocol status register (FDCAN_PSR) 1597
39.4.14 FDCAN transmitter delay compensation register (FDCAN_TDCR) . . 1599
39.4.15 FDCAN interrupt register (FDCAN_IR) 1599
39.4.16 FDCAN interrupt enable register (FDCAN_IE) 1602
39.4.17 FDCAN interrupt line select register (FDCAN_ILS) 1604
39.4.18 FDCAN interrupt line enable register (FDCAN_ILE) 1605
39.4.19 FDCAN global filter configuration register (FDCAN_RXGFC) 1605
39.4.20 FDCAN extended ID and mask register (FDCAN_XIDAM) 1607
39.4.21 FDCAN high-priority message status register (FDCAN_HPMS) 1607
39.4.22 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S) 1608
39.4.23 CAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A) 1609
39.4.24 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S) 1609
39.4.25 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A) 1610
39.4.26 FDCAN Tx buffer configuration register (FDCAN_TXBC) 1610
39.4.27 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS) 1611
39.4.28 FDCAN Tx buffer request pending register (FDCAN_TXBRP) 1611
39.4.29 FDCAN Tx buffer add request register (FDCAN_TXBAR) 1612
39.4.30 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR) . . 1613
39.4.31 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO) 1613
39.4.32 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF) . . 1614
39.4.33 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE) 1614
39.4.34 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE) 1615
39.4.35 FDCAN Tx event FIFO status register (FDCAN_TXEFS) 1615
39.4.36 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA) . . . 1616
39.4.37 FDCAN CFG clock divider register (FDCAN_CKDIV) 1616
39.4.38 FDCAN register map 1617

40 Universal serial bus full-speed host/device interface (USB) 1621

40.1 Introduction 1621
40.2 USB main features 1621

40.3 USB implementation 1621
40.4 USB functional description 1622
40.4.1 USB block diagram 1622
40.4.2 USB pins and internal signals 1622
40.4.3 USB reset and clocks 1623
40.4.4 General description and Device mode functionality 1623
40.4.5 Description of USB blocks used in both Device and Host modes . . . 1624
40.4.6 Description of host frame scheduler (HFS) specific to Host mode . . 1625
40.5 Programming considerations for Device and Host modes 1626
40.5.1 Generic USB Device programming 1626
40.5.2 System and power-on reset 1626
40.5.3 Double-buffered endpoints and usage in Device mode 1633
40.5.4 Double buffered channels: usage in Host mode 1635
40.5.5 Isochronous transfers in Device mode 1636
40.5.6 Isochronous transfers in Host mode 1637
40.5.7 Suspend/resume events 1638
40.6 USB registers 1642
40.6.1 USB control register (USB_CNTR) 1642
40.6.2 USB interrupt status register (USB_ISTR) 1645
40.6.3 USB frame number register (USB_FNR) 1649
40.6.4 USB Device address (USB_DADDR) 1649
40.6.5 USB LPM control and status register (USB_LPMCSR) 1650
40.6.6 USB battery charging detector (USB_BCDR) 1651
40.6.7 USB endpoint/channel n register (USB_CHEPnR) 1652
40.6.8 USB register map 1661
40.7 USBSRAM registers 1662
40.7.1 Channel/endpoint transmit buffer descriptor n (USB_CHEP_TXRXBD_n) 1663
40.7.2 Channel/endpoint receive buffer descriptor n [alternate] (USB_CHEP_TXRXBD_n) 1663
40.7.3 Channel/endpoint receive buffer descriptor n (USB_CHEP_RXTXBD_n) 1665
40.7.4 Channel/endpoint transmit buffer descriptor n [alternate] (USB_CHEP_RXTXBD_n) 1666
40.7.5 USBSRAM register map 1667

41 Debug support (DBG) 1668

41.1 Introduction 1668

41.2 DBG functional description 1669
41.2.1 DBG block diagram 1669
41.2.2 DBG pins and internal signals 1669
41.2.3 DBG reset and clocks 1670
41.2.4 DBG power domains 1670
41.2.5 Debug and low-power modes 1670
41.2.6 Security 1671
41.2.7 Debug authentication 1672
41.3 Serial-wire and JTAG debug port (SWJ-DP) 1673
41.3.1 JTAG debug port 1674
41.3.2 Serial-wire debug port 1676
41.3.3 Debug port registers 1677
41.3.4 Debug port register map and reset values 1684
41.4 Access ports 1685
41.4.1 Access port registers 1685
41.4.2 Access port register map 1692
41.5 ROM tables 1693
41.5.1 System ROM table registers 1696
41.5.2 System ROM table register map 1700
41.5.3 MCU ROM table registers 1701
41.5.4 MCU ROM table register map 1706
41.5.5 Processor ROM table registers 1707
41.5.6 Processor ROM table register map 1711
41.6 Data watchpoint and trace unit (DWT) 1712
41.6.1 DWT registers 1713
41.6.2 DWT register map 1727
41.7 Instrumentation trace macrocell (ITM) 1730
41.7.1 ITM registers 1731
41.7.2 ITM register map 1739
41.8 Breakpoint unit (BPU) 1741
41.8.1 BPU registers 1741
41.8.2 BPU register map 1747
41.9 Embedded trace macrocell (ETM) 1748
41.9.1 ETM registers 1749
41.9.2 ETM register map 1774
41.10 Trace port interface unit (TPIU) 1778

41.10.1 TPIU registers 1779
41.10.2 TPIU register map 1789
41.11 Cross-trigger interface (CTI) 1791
41.11.1 CTI registers 1792
41.11.2 CTI register map 1803
41.12 Microcontroller debug unit (DBGMCU) 1805
41.12.1 Device ID 1805
41.12.2 Low-power mode emulation 1805
41.12.3 Peripheral clock freeze 1806
41.12.4 DBGMCU registers 1807
41.12.5 DBGMCU register map 1820
41.13 References 1823

42 Device electronic signature 1824
42.1 Unique device ID register (96 bits) 1824
42.2 Flash size data register 1825
42.3 Package data register 1826

43 Important security notice 1827

44 Revision history 1828

List of tables

Table 1. Memory map and peripheral register boundary addresses . . . 72
Table 2. DMA channel use (privilege) . . . 80
Table 3. Internal tampers in TAMP . . . 83
Table 4. Effect of low-power modes on TAMP . . . 85
Table 5. Main product life-cycle transitions . . . 86
Table 6. Typical product life-cycle phases . . . 87
Table 7. Software intellectual property protection with PRODUCT_STATE . . . 90
Table 8. STM32H503 boot modes . . . 91
Table 9. Internal SRAMs features . . . 93
Table 10. Effect of low-power modes on RAMCFG . . . 95
Table 11. RAMCFG interrupt requests . . . 96
Table 12. RAMCFG register map and reset values . . . 101
Table 13. GTZC implementation . . . 105
Table 14. GTZC features . . . 105
Table 15. GTZC1 subblock address offset . . . 106
Table 16. MPCWM resource assignment . . . 106
Table 17. MPCBB resource assignment . . . 106
Table 18. GTZC1 TZSC register map and reset values . . . 115
Table 19. GTZC1 MPCBBz register map and reset values (z = 1 to 2) . . . 116
Table 20. FLASH recommended number of wait states and programming delay . . . 123
Table 21. Flash memory OTP organization . . . 131
Table 22. Read-only public data organization . . . 132
Table 23. Flash memory read-only organization . . . 132
Table 24. memory map and the swapping option . . . 133
Table 25. Recommended reactions to FLASH_OPSR contents . . . 135
Table 26. Option-byte organization . . . 138
Table 27. Specific OB modifying rules overview . . . 142
Table 28. OB modifiable in closed product . . . 143
Table 29. HDP protected definition . . . 145
Table 30. Secure hide protection . . . 145
Table 31. HDP protections summary . . . 146
Table 32. Privilege protection summary . . . 147
Table 33. Privilege and mass or bank erase . . . 147
Table 34. Privilege configuration register access conditions . . . 147
Table 35. Flash interface register protection summary . . . 150
Table 36. Product states, debug states and debug policy . . . 150
Table 37. PRODUCT_STATE transitions . . . 152
Table 38. OTP/RO access constraints . . . 153
Table 39. C defined macro for RSS services . . . 153
Table 40. NSSlib interface function list . . . 153
Table 41. Effect of low-power modes on the embedded flash memory . . . 155
Table 42. Locating ECC failure . . . 160
Table 43. Flash interrupt request . . . 162
Table 44. Register map and reset value table . . . 188
Table 45. ICACHE features . . . 193
Table 46. TAG memory dimensioning parameters for n-way set associative operating mode (default) . . . 195
Table 47. TAG memory dimensioning parameters for direct-mapped cache mode . . . 196

Table 48. ICACHE cacheability for AHB transaction . . . 197
Table 49. ICACHE interrupts . . . 200
Table 50. ICACHE register map and reset values . . . 204
Table 51. PWR input/output pins . . . 206
Table 52. PWR internal input/output signals . . . 206
Table 53. Analog switched recommended configuration . . . 210
Table 54. Low-power mode summary . . . 221
Table 55. Functionalities depending on the working mode . . . 222
Table 56. Sleep mode . . . 225
Table 57. Stop mode . . . 227
Table 58. Standby mode . . . 230
Table 59. Power modes output states versus MCU power modes . . . 231
Table 60. PWR privilege configuration summary . . . 231
Table 61. PWR interrupt requests . . . 232
Table 62. PWR register map and reset values . . . 244
Table 63. RCC input/output signals connected to package pins or balls . . . 246
Table 64. Reset source identification (RCC_RSR) . . . 249
Table 65. STOPWUCK and STOPKERWUCK description . . . 263
Table 66. HSIKERON and CSIKERON behavior . . . 263
Table 67. Kernel clock distribution overview . . . 266
Table 68. RCC privilege configuration summary . . . 270
Table 69. Interrupt sources and control . . . 273
Table 70. RCC register map and reset values . . . 326
Table 71. CRS features . . . 331
Table 72. CRS internal input/output signals . . . 332
Table 73. CRS interconnection . . . 333
Table 74. Effect of low-power modes on CRS . . . 336
Table 75. Interrupt control bits . . . 336
Table 76. CRS register map and reset values . . . 341
Table 77. Port bit configuration . . . 344
Table 78. GPIO register map and reset values . . . 359
Table 79. SBS internal input/output signals . . . 362
Table 80. HDPL encoded values . . . 365
Table 81. SBS boot logic . . . 366
Table 82. Debug authentication logic . . . 368
Table 83. SBS register map and reset values . . . 377
Table 84. Peripherals interconnect matrix . . . 379
Table 85. GPDMA1/2 channel implementation . . . 393
Table 86. GPDMA1/2 wake-up in low-power modes . . . 393
Table 87. Programmed GPDMA1/2 request . . . 393
Table 88. Programmed GPDMA1/2 request as a block request . . . 398
Table 89. GPDMA1/2 channel with peripheral early termination . . . 398
Table 90. Programmed GPDMA1/2 request with peripheral early termination . . . 398
Table 91. Programmed GPDMA1/2 trigger . . . 398
Table 92. Programmed GPDMA source/destination burst . . . 419
Table 93. Programmed data handling . . . 424
Table 94. Effect of low-power modes on GPDMA . . . 436
Table 95. GPDMA interrupt requests . . . 437
Table 96. GPDMA register map and reset values . . . 463
Table 97. STM32H503xx vector table . . . 466
Table 98. EXTI pin overview . . . 471
Table 99. EVG pin overview . . . 471

Table 100. EXTI line connections . . . 472
Table 101. Masking functionality . . . 475
Table 102. Register protection overview . . . 475
Table 103. EXTI register map sections . . . 476
Table 104. EXTI register map and reset values . . . 494
Table 105. CRC internal input/output signals . . . 497
Table 106. CRC register map and reset values . . . 502
Table 107. ADC features . . . 505
Table 108. ADC input/output pins . . . 507
Table 109. ADC internal input/output signals . . . 507
Table 110. ADC interconnection . . . 507
Table 111. Configuring the trigger polarity for regular external triggers . . . 526
Table 112. Configuring the trigger polarity for injected external triggers . . . 527
Table 113. TSAR timings depending on resolution . . . 539
Table 114. Offset computation versus data resolution . . . 542
Table 115. Analog watchdog channel selection . . . 553
Table 116. Analog watchdog 1 comparison . . . 554
Table 117. Analog watchdog 2 and 3 comparison . . . 555
Table 118. Maximum output results versus N and M (gray cells indicate truncation) . . . 558
Table 119. ADC interrupts . . . 568
Table 120. ADC global register map . . . 603
Table 121. ADC register map and reset values . . . 603
Table 122. ADC register map and reset values (master and slave ADC common registers) . . . 605
Table 123. DTS internal input/output signals . . . 608
Table 124. Sampling time configuration . . . 611
Table 125. Trigger configuration . . . 612
Table 126. Temperature sensor behavior in low-power modes . . . 614
Table 127. Interrupt control bits . . . 615
Table 128. DTS register map and reset values . . . 623
Table 129. DAC features . . . 625
Table 130. DAC input/output pins . . . 627
Table 131. DAC internal input/output signals . . . 627
Table 132. DAC interconnection . . . 628
Table 133. Data format (case of 12-bit data) . . . 630
Table 134. HFSEL description . . . 631
Table 135. Sample and refresh timings . . . 636
Table 136. Channel output modes summary . . . 638
Table 137. Effect of low-power modes on DAC . . . 644
Table 138. DAC interrupts . . . 644
Table 139. DAC register map and reset values . . . 660
Table 140. COMP1 input/output internal signals . . . 664
Table 141. COMP1 blanking source selection . . . 665
Table 142. COMP1 noninverting input assignment . . . 665
Table 143. COMP1 inverting input assignment . . . 665
Table 144. COMP1 blanking source assignment . . . 665
Table 145. Comparator behavior in low-power modes . . . 668
Table 146. Interrupt through EXTI control bits . . . 668
Table 147. Interrupt through NVIC control bits . . . 669
Table 148. COMP register map and reset values . . . 674
Table 149. Possible connections for OPAMP . . . 676
Table 150. Operating modes and calibration . . . 683

Table 151. Effect of low-power modes on the OPAMP . . . 685
Table 152. OPAMP register map and reset values . . . 688
Table 153. RNG internal input/output signals . . . 690
Table 154. RNG interrupt requests . . . 698
Table 155. RNG initialization times . . . 699
Table 156. RNG configurations . . . 699
Table 157. Configuration selection . . . 700
Table 158. RNG register map and reset map . . . 705
Table 159. HASH internal input/output signals . . . 707
Table 160. Information on supported hash algorithms . . . 708
Table 161. Hash processor outputs . . . 711
Table 162. Processing time (in clock cycle) . . . 716
Table 163. HASH interrupt requests . . . 717
Table 164. HASH1 register map and reset values . . . 725
Table 165. TIM input/output pins . . . 730
Table 166. TIM internal input/output signals . . . 730
Table 167. Interconnect to the tim_ti1 input multiplexer . . . 732
Table 168. Interconnect to the tim_ti2 input multiplexer . . . 732
Table 169. Interconnect to the tim_ti3 input multiplexer . . . 732
Table 170. Interconnect to the tim_ti4 input multiplexer . . . 732
Table 171. Internal trigger connection . . . 732
Table 172. Interconnect to the tim_etr input multiplexer . . . 733
Table 173. Timer break interconnect . . . 733
Table 174. Timer break2 interconnect . . . 733
Table 175. System break interconnect . . . 733
Table 176. CCR and ARR register change dithering pattern . . . 767
Table 177. CCR register change dithering pattern in center-aligned PWM mode . . . 768
Table 178. Behavior of timer outputs versus tim_brk/tim_brk2 inputs . . . 780
Table 179. Break protection disarming conditions . . . 782
Table 180. Counting direction versus encoder signals (CC1P = CC2P = 0) . . . 791
Table 181. Counting direction versus encoder signals and polarity settings . . . 795
Table 182. DMA request . . . 816
Table 183. Effect of low-power modes on TIM1 . . . 817
Table 184. Interrupt requests . . . 817
Table 185. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature . . . 844
Table 186. TIM1 register map and reset values . . . 867
Table 187. STM32H503xx general purpose timers . . . 871
Table 188. TIM input/output pins . . . 873
Table 189. TIM internal input/output signals . . . 873
Table 190. Interconnect to the tim_ti1 input multiplexer . . . 874
Table 191. Interconnect to the tim_ti2 input multiplexer . . . 874
Table 192. Interconnect to the tim_ti3 input multiplexer . . . 875
Table 193. Interconnect to the tim_ti4 input multiplexer . . . 875
Table 194. TIMx internal trigger connection . . . 875
Table 195. Interconnect to the tim_etr input multiplexer . . . 875
Table 196. CCR and ARR register change dithering pattern . . . 906
Table 197. CCR register change dithering pattern in center-aligned PWM mode . . . 907
Table 198. Counting direction versus encoder signals(CC1P = CC2P = 0) . . . 916
Table 199. Counting direction versus encoder signals and polarity settings . . . 921
Table 200. DMA request . . . 945
Table 201. Effect of low-power modes on TIM2/TIM3 . . . 945

Table 202. Interrupt requests . . . 946
Table 203. Output control bit for standard tim_ocx channels . . . 967
Table 204. TIM2/TIM3 register map and reset values . . . 983
Table 205. TIM internal input/output signals . . . 987
Table 206. TIMx_ARR register change dithering pattern . . . 997
Table 207. DMA request . . . 998
Table 208. Effect of low-power modes on TIM6/TIM7 . . . 998
Table 209. Interrupt request . . . 998
Table 210. TIMx register map and reset values . . . 1004
Table 211. STM32H503xx LPTIM features . . . 1006
Table 212. LPTIM1/2 input/output pins . . . 1007
Table 213. LPTIM1/2 internal signals . . . 1008
Table 214. LPTIM1/2 external trigger connections . . . 1009
Table 215. LPTIM1/2 input 1 connections . . . 1009
Table 216. LPTIM1/2 input 2 connections . . . 1009
Table 217. LPTIM1/2 input capture 1 connections . . . 1009
Table 218. LPTIM1/2 input capture 2 connections . . . 1010
Table 219. Prescaler division ratios . . . 1011
Table 220. Encoder counting scenarios . . . 1018
Table 221. Input capture Glitch filter latency (in counter step unit) . . . 1022
Table 222. Effect of low-power modes on the LPTIM . . . 1026
Table 223. Interrupt events . . . 1027
Table 224. LPTIM register map and reset values . . . 1048
Table 225. IWDG features . . . 1050
Table 226. IWDG delays versus actions . . . 1051
Table 227. IWDG internal input/output signals . . . 1052
Table 228. Effect of low power modes on IWDG . . . 1057
Table 229. IWDG interrupt request . . . 1059
Table 230. IWDG register map and reset values . . . 1065
Table 231. WWDG features . . . 1066
Table 232. WWDG internal input/output signals . . . 1067
Table 233. WWDG interrupt requests . . . 1070
Table 234. WWDG register map and reset values . . . 1072
Table 235. RTC input/output pins . . . 1075
Table 236. RTC internal input/output signals . . . 1075
Table 237. RTC interconnection . . . 1076
Table 238. RTC pin PC13 configuration . . . 1076
Table 239. RTC_OUT mapping . . . 1079
Table 240. Effect of low-power modes on RTC . . . 1092
Table 241. RTC pins functionality over modes . . . 1093
Table 242. Interrupt requests . . . 1093
Table 243. RTC register map and reset values . . . 1120
Table 244. TAMP input/output pins . . . 1124
Table 245. TAMP internal input/output signals . . . 1124
Table 246. TAMP interconnection . . . 1125
Table 247. Device resource x tamper protection . . . 1129
Table 248. Active tamper output change period . . . 1133
Table 249. Minimum ATPER value . . . 1134
Table 250. Active tamper filtered pulse duration . . . 1135
Table 251. Effect of low-power modes on TAMP . . . 1136
Table 252. TAMP pins functionality over modes . . . 1136
Table 253. Interrupt requests . . . 1137

Table 254. TAMP register map and reset values . . . 1159
Table 255. I2C implementation . . . 1162
Table 256. I2C input/output pins . . . 1163
Table 257. I2C internal input/output signals . . . 1164
Table 258. Comparison of analog and digital filters . . . 1166
Table 259. I²C-bus and SMBus specification data setup and hold times . . . 1168
Table 260. I2C configuration . . . 1172
Table 261. I²C-bus and SMBus specification clock timings . . . 1183
Table 262. Timing settings for f_{I2CCLK} of 8 MHz . . . 1193
Table 263. Timing settings for f_{I2CCLK} of 16 MHz . . . 1193
Table 264. SMBus timeout specifications . . . 1195
Table 265. SMBus with PEC configuration . . . 1197
Table 266. TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms . . . 1198
Table 267. TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms . . . 1198
Table 268. TIMEOUTA[11:0] for maximum t_{IDLE} of 50 µs . . . 1198
Table 269. Effect of low-power modes to I2C . . . 1208
Table 270. I2C interrupt requests . . . 1208
Table 271. I2C register map and reset values . . . 1224
Table 272. I3C wake-up . . . 1227
Table 273. I3C FIFOs implementation . . . 1227
Table 274. I3C interrupt(s) . . . 1227
Table 275. I3C peripheral controller/target features versus MIPI v1.1 . . . 1228
Table 276. I3C input/output pins . . . 1229
Table 277. I3C internal input/output signals . . . 1229
Table 278. I3C register usage . . . 1241
Table 279. I3C registers/fields usage versus controller state . . . 1242
Table 280. I3C registers/fields usage versus target state . . . 1245
Table 281. List of supported I3C CCCs, as controller/target . . . 1248
Table 282. I3C controller error management . . . 1279
Table 283. I3C target error management . . . 1281
Table 284. Effect of low-power modes . . . 1285
Table 285. I3C interrupt requests . . . 1286
Table 286. I3C register map and reset values . . . 1331
Table 287. Instance implementation on STM32H503xx . . . 1335
Table 288. USART/LPUART features . . . 1335
Table 289. USART/UART input/output pins . . . 1338
Table 290. USART internal input/output signals . . . 1339
Table 291. Noise detection from sampled data . . . 1351
Table 292. Tolerance of the USART receiver when BRR [3:0] = 0000 . . . 1355
Table 293. Tolerance of the USART receiver when BRR[3:0] is different from 0000 . . . 1355
Table 294. USART frame formats . . . 1360
Table 295. Effect of low-power modes on the USART . . . 1383
Table 296. USART interrupt requests . . . 1384
Table 297. USART register map and reset values . . . 1422
Table 298. Instance implementation on STM32H503xx . . . 1425
Table 299. USART/LPUART features . . . 1425
Table 300. LPUART input/output pins . . . 1427
Table 301. LPUART internal input/output signals . . . 1427
Table 302. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32.768 kHz . . . 1438
Table 303. Tolerance of the LPUART receiver . . . 1439
Table 305. Effect of low-power modes on the LPUART . . . 1451
Table 306. LPUART interrupt requests . . . 1452

Table 307. LPUART register map and reset values . . . 1478
Table 308. SPI features . . . 1481
Table 309. SPI/I2S input/output pins . . . 1483
Table 310. SPI internal input/output signals . . . 1484
Table 311. Effect of low-power modes on the SPI . . . 1510
Table 312. SPI wake-up and interrupt requests . . . 1511
Table 313. Bitfields usable in PCM/I2S mode . . . 1513
Table 314. WS and CK level before SPI/I2S is enabled when AFCNTR = 1 . . . 1522
Table 315. Serial data line swapping . . . 1522
Table 316. CLKGEN programming examples for usual I2S frequencies . . . 1526
Table 317. I2S interrupt requests . . . 1535
Table 318. SPI register map and reset values . . . 1553
Table 319. CAN subsystem I/O signals . . . 1559
Table 320. CAN subsystem I/O pins . . . 1559
Table 321. DLC coding in FDCAN . . . 1563
Table 322. Possible configurations for frame transmission . . . 1577
Table 323. Rx FIFO element . . . 1580
Table 324. Rx FIFO element description . . . 1580
Table 325. Tx buffer and FIFO element . . . 1582
Table 326. Tx buffer element description . . . 1582
Table 327. Tx event FIFO element . . . 1584
Table 328. Tx event FIFO element description . . . 1584
Table 329. Standard message ID filter element . . . 1585
Table 330. Standard message ID filter element field description . . . 1586
Table 331. Extended message ID filter element . . . 1586
Table 332. Extended message ID filter element field description . . . 1587
Table 333. FDCAN register map and reset values . . . 1617
Table 334. STM32H503xx USB implementation . . . 1621
Table 335. USB input/output pins . . . 1622
Table 336. Double-buffering buffer flag definition . . . 1634
Table 337. Bulk double-buffering memory buffers usage (Device mode) . . . 1634
Table 338. Bulk double-buffering memory buffers usage (Host mode) . . . 1636
Table 339. Isochronous memory buffers usage . . . 1637
Table 340. Isochronous memory buffers usage . . . 1638
Table 341. Resume event detection . . . 1640
Table 342. Resume event detection for host . . . 1641
Table 343. Reception status encoding . . . 1659
Table 344. Endpoint/channel type encoding . . . 1659
Table 345. Endpoint/channel kind meaning . . . 1659
Table 346. Transmission status encoding . . . 1659
Table 347. USB register map and reset values . . . 1661
Table 348. Definition of allocated buffer memory . . . 1664
Table 349. USBSRAM register map and reset values . . . 1667
Table 350. SPIRIT address block DBG . . . 1668
Table 351. JTAG/Serial-wire debug port pins . . . 1669
Table 352. Trace port pins . . . 1669
Table 353. Single-wire trace port pins . . . 1670
Table 354. Authentication signal states . . . 1671
Table 355. Life cycle state and debug states . . . 1671
Table 356. JTAG-DP data registers . . . 1675
Table 357. Packet request . . . 1677
Table 358. ACK response . . . 1677

Table 359. Data transfer . . . 1677
Table 360. Debug port registers . . . 1678
Table 361. Debug port register map and reset values . . . 1684
Table 362. MEM-AP registers . . . 1686
Table 363. Access port register map and reset values . . . 1692
Table 364. System ROM table . . . 1693
Table 365. MCU ROM table . . . 1694
Table 366. Processor ROM table . . . 1694
Table 367. System ROM table register map and reset values . . . 1700
Table 368. MCU ROM table register map and reset values . . . 1706
Table 369. CPU ROM table register map and reset values . . . 1711
Table 370. DWT register map and reset values . . . 1727
Table 371. ITM register map and reset values . . . 1739
Table 372. BPU register map and reset values . . . 1747
Table 373. ETM register map and reset values . . . 1774
Table 374. TPIU register map and reset values . . . 1789
Table 375. CTI inputs . . . 1791
Table 376. CTI outputs . . . 1791
Table 377. CTI register map and reset values . . . 1803
Table 378. Peripheral clock freeze control bits . . . 1806
Table 379. Peripheral behaviour in debug mode . . . 1807
Table 380. DBGMCU register map and reset values . . . 1820
Table 381. Document revision history . . . 1828

List of figures

Figure 1. System architecture . . . 68
Figure 2. Memory map . . . 71
Figure 3. Flash memory secure HDP area . . . 78
Figure 4. Device life-cycle security . . . 86
Figure 5. Product life-cycle . . . 88
Figure 6. Collaborative development principle . . . 89
Figure 7. SRAM1 and SRAM2 with ECC memory map . . . 94
Figure 8. GTZC privilege architecture . . . 105
Figure 9. GTZC block diagram . . . 107
Figure 10. Watermark memory protection controller (region x/subregion A) . . . 108
Figure 11. MPCBB block diagram . . . 108
Figure 12. Flash block diagram (simplified) . . . 118
Figure 13. Embedded flash memory organization . . . 120
Figure 14. Embedded flash memory usage . . . 121
Figure 15. Flash bank swapping sequence . . . 134
Figure 16. HDP in user flash memory . . . 145
Figure 17. Protection attributes in case of bank swap illustration . . . 149
Figure 18. ICACHE block diagram . . . 194
Figure 19. ICACHE TAG and data memories functional view . . . 196
Figure 20. Power supply overview with LDO . . . 207
Figure 21. System supply configurations . . . 208
Figure 22. Power-on reset (POR)/power-down reset (PDR) waveform . . . 213
Figure 23. BOR thresholds . . . 214
Figure 24. PVD thresholds . . . 215
Figure 25. AVD thresholds . . . 216
Figure 26. VBAT thresholds . . . 217
Figure 27. Temperature thresholds . . . 218
Figure 28. Dynamic voltage scaling in Run mode . . . 220
Figure 29. I/O states in Standby mode . . . 229
Figure 30. Simplified diagram of the reset circuit . . . 247
Figure 31. Clock tree . . . 251
Figure 32. HSE/ LSE clock sources . . . 252
Figure 33. HSE/ LSE bypass . . . 252
Figure 34. HSI calibration flow . . . 254
Figure 35. CSI calibration flow . . . 255
Figure 36. PLL block diagram . . . 257
Figure 37. PLLs initialization flowchart . . . 260
Figure 38. CRS block diagram . . . 332
Figure 39. CRS counter behavior . . . 334
Figure 40. Structure of three-volt or five-volt tolerant GPIO (TT or FT) . . . 344
Figure 41. Input floating/pull-up/pull-down configurations . . . 348
Figure 42. Output configuration . . . 349
Figure 43. Alternate function configuration . . . 350
Figure 44. High-impedance analog configuration . . . 350
Figure 45. SBS block diagram . . . 362
Figure 46. Compensation cell management . . . 363
Figure 47. Compensation cell use . . . 364
Figure 48. SBS boot control . . . 364

Figure 49. SBS debug control . . . 367
Figure 50. GPDMA block diagram . . . 400
Figure 51. GPDMA channel direct programming without linked-list (GPDMA_CxLLR = 0) . . . 401
Figure 52. GPDMA channel suspend and resume sequence . . . 402
Figure 53. GPDMA channel abort and restart sequence . . . 403
Figure 54. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x . . . 404
Figure 55. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x . . . 405
Figure 56. GPDMA dynamic linked-list data structure of a linear addressing channel x . . . 406
Figure 57. GPDMA dynamic linked-list data structure of a 2D addressing channel x . . . 406
Figure 58. GPDMA channel execution and linked-list programming in run-to-completion mode (GPDMA_CxCR.LSM = 0) . . . 408
Figure 59. Inserting a LLIn with an auxiliary GPDMA channel y . . . 410
Figure 60. GPDMA channel execution and linked-list programming in link step mode (GPDMA_CxCR.LSM = 1) . . . 412
Figure 61. Building LLIn+1: GPDMA dynamic linked-lists in link step mode . . . 413
Figure 62. Replace with a new LLIn' in register file in link step mode . . . 414
Figure 63. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 1) . . . 415
Figure 64. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 2) . . . 416
Figure 65. GPDMA channel execution and linked-list programming . . . 418
Figure 66. Programmed 2D addressing . . . 421
Figure 67. GPDMA arbitration policy . . . 428
Figure 68. Trigger hit, memorization, and overrun waveform . . . 431
Figure 69. GPDMA circular buffer programming: update of the memory start address with a linear addressing channel . . . 432
Figure 70. Shared GPDMA channel with circular buffering: update of the memory start address with a linear addressing channel . . . 433
Figure 71. EXTI block diagram . . . 471
Figure 72. Configurable event trigger logic CPU wakeup . . . 473
Figure 73. EXTI mux GPIO selection . . . 474
Figure 74. CRC calculation unit block diagram . . . 497
Figure 75. ADC block diagram . . . 506
Figure 76. ADC clock scheme . . . 511
Figure 77. ADC1 connectivity . . . 512
Figure 78. ADC calibration . . . 515
Figure 79. Updating the ADC calibration factor . . . 516
Figure 80. Mixing single-ended and differential channels . . . 517
Figure 81. Enabling / disabling the ADC . . . 518
Figure 82. Bulb mode timing diagram . . . 521
Figure 83. Analog-to-digital conversion time . . . 524
Figure 84. Stopping ongoing regular conversions . . . 525
Figure 85. Stopping ongoing regular and injected conversions . . . 526
Figure 86. Trigger selection . . . 527
Figure 87. Injected conversion latency . . . 529
Figure 88. Example of JSQR queue of context (sequence change) . . . 532
Figure 89. Example of JSQR queue of context (trigger change) . . . 532
Figure 90. Example of JSQR queue of context with overflow before conversion . . . 533
Figure 91. Example of JSQR queue of context with overflow during conversion . . . 533
Figure 92. Example of JSQR queue of context with empty queue (case JQM = 0) . . . 534

Figure 93. Example of JSQR queue of context with empty queue (JQM = 1) . . . 535
Figure 94. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion. . . . 535
Figure 95. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion and a new trigger occurs . . . 536
Figure 96. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs outside an ongoing conversion. . . . 536
Figure 97. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 1) . . . 537
Figure 98. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 0). . . . 537
Figure 99. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 1). . . . 538
Figure 100. Single conversions of a sequence, software trigger . . . 540
Figure 101. Continuous conversion of a sequence, software trigger. . . . 540
Figure 102. Single conversions of a sequence, hardware trigger . . . 541
Figure 103. Continuous conversions of a sequence, hardware trigger . . . 541
Figure 104. Right alignment (offset disabled, unsigned value) . . . 543
Figure 105. Right alignment (offset enabled, signed value). . . . 544
Figure 106. Left alignment (offset disabled, unsigned value) . . . 544
Figure 107. Left alignment (offset enabled, signed value) . . . 545
Figure 108. Example of overrun (OVRMOD = 0) . . . 546
Figure 109. Example of overrun (OVRMOD = 1) . . . 547
Figure 110. AUTODLY = 1, regular conversion in continuous mode, software trigger . . . 550
Figure 111. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0) . . . 550
Figure 112. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1) . . . 551
Figure 113. AUTODLY = 1, regular continuous conversions interrupted by injected conversions . . . 552
Figure 114. AUTODLY = 1 in auto- injected mode (JAUTO = 1) . . . 552
Figure 115. Analog watchdog guarded area . . . 553
Figure 116. ADCy_AWDx_OUT signal generation (on all regular channels). . . . 555
Figure 117. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by software) . . . 556
Figure 118. ADCy_AWDx_OUT signal generation (on a single regular channel) . . . 556
Figure 119. ADCy_AWDx_OUT signal generation (on all injected channels) . . . 556
Figure 120. 20-bit to 16-bit result truncation . . . 558
Figure 121. Numerical example with 5-bit shift and rounding . . . 558
Figure 122. Triggered regular oversampling mode (TROVS bit = 1) . . . 560
Figure 123. Regular oversampling modes (4x ratio) . . . 561
Figure 124. Regular and injected oversampling modes used simultaneously . . . 562
Figure 125. Triggered regular oversampling with injection . . . 562
Figure 126. Oversampling in auto-injected mode . . . 563
Figure 127. Temperature sensor channel block diagram . . . 564
Figure 128. VBAT channel block diagram . . . 565
Figure 129. VREFINT channel block diagram . . . 566
Figure 130. Temperature sensor functional block diagram . . . 608
Figure 131. Method for low REF_CLK frequencies . . . 610
Figure 132. Method for high REF_CLK frequencies . . . 610
Figure 133. Temperature sensor sequence . . . 613
Figure 134. Dual-channel DAC block diagram . . . 626
Figure 135. Data registers in single DAC channel mode . . . 629
Figure 136. Data registers in dual DAC channel mode . . . 629
Figure 137. Timing diagram for conversion with trigger disabled TEN = 0 . . . 631
Figure 138. DAC LFSR register calculation algorithm . . . 633

Figure 139. DAC conversion (SW trigger enabled) with LFSR wave generation . . . 634
Figure 140. DAC triangle wave generation . . . 634
Figure 141. DAC conversion (SW trigger enabled) with triangle wave generation . . . 635
Figure 142. DAC sample and hold mode phase diagram . . . 637
Figure 143. Comparator functional block diagram . . . 664
Figure 144. Comparator hysteresis . . . 666
Figure 145. Comparator output blanking . . . 667
Figure 146. Scaler block diagram . . . 670
Figure 147. Standalone mode: external gain setting mode . . . 677
Figure 148. Follower configuration . . . 678
Figure 149. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input not used . . . 679
Figure 150. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input used for filtering . . . 680
Figure 151. PGA mode, non-inverting gain setting (x2/x4/x8/x16) or inverting gain setting (x-1/x-3/x-7/x-15) . . . 681
Figure 152. Example configuration . . . 681
Figure 153. PGA mode, non-inverting gain setting (x2/x4/x8/x16) or inverting gain setting (x-1/x-3/x-7/x-15) with filtering . . . 682
Figure 154. Example configuration . . . 682
Figure 155. RNG block diagram . . . 690
Figure 156. NIST SP800-90B entropy source model . . . 691
Figure 157. RNG initialization overview . . . 694
Figure 158. HASH block diagram . . . 707
Figure 159. Message data swapping feature . . . 709
Figure 160. HASH suspend/resume mechanism . . . 714
Figure 161. Advanced-control timer block diagram . . . 729
Figure 162. Counter timing diagram with prescaler division change from 1 to 2 . . . 735
Figure 163. Counter timing diagram with prescaler division change from 1 to 4 . . . 735
Figure 164. Counter timing diagram, internal clock divided by 1 . . . 737
Figure 165. Counter timing diagram, internal clock divided by 2 . . . 737
Figure 166. Counter timing diagram, internal clock divided by 4 . . . 738
Figure 167. Counter timing diagram, internal clock divided by N . . . 738
Figure 168. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded) . . . 739
Figure 169. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded) . . . 740
Figure 170. Counter timing diagram, internal clock divided by 1 . . . 741
Figure 171. Counter timing diagram, internal clock divided by 2 . . . 742
Figure 172. Counter timing diagram, internal clock divided by 4 . . . 742
Figure 173. Counter timing diagram, internal clock divided by N . . . 743
Figure 174. Counter timing diagram, update event when repetition counter is not used . . . 743
Figure 175. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 . . . 745
Figure 176. Counter timing diagram, internal clock divided by 2 . . . 745
Figure 177. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36 . . . 746
Figure 178. Counter timing diagram, internal clock divided by N . . . 746
Figure 179. Counter timing diagram, update event with ARPE = 1 (counter underflow) . . . 747
Figure 180. Counter timing diagram, Update event with ARPE = 1 (counter overflow) . . . 748
Figure 181. Update rate examples depending on mode and TIMx_RCR register settings . . . 749
Figure 183. Control circuit in normal mode, internal clock divided by 1 . . . 751
Figure 184. tim_ti2 external clock connection example . . . 751
Figure 185. Control circuit in external clock mode 1 . . . 752
Figure 186. External trigger input block . . . 753

Figure 187. Control circuit in external clock mode 2 . . . 754
Figure 188. Capture/compare channel (example: channel 1 input stage) . . . 754
Figure 189. Capture/compare channel 1 main circuit . . . 755
Figure 190. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4) . . . 756
Figure 191. Output stage of capture/compare channel (channel 5, idem ch. 6) . . . 756
Figure 192. PWM input mode timing . . . 759
Figure 193. Output compare mode, toggle on tim_oc1 . . . 761
Figure 194. Edge-aligned PWM waveforms (ARR = 8) . . . 762
Figure 195. Center-aligned PWM waveforms (ARR = 8) . . . 763
Figure 196. Dithering principle . . . 764
Figure 197. Data format and register coding in dithering mode . . . 765
Figure 198. PWM resolution vs frequency . . . 766
Figure 199. PWM dithering pattern . . . 767
Figure 200. Dithering effect on duty cycle in center-aligned PWM mode . . . 768
Figure 201. Generation of 2 phase-shifted PWM signals with 50% duty cycle . . . 770
Figure 202. Combined PWM mode on channel 1 and 3 . . . 771
Figure 203. 3-phase combined PWM signals with multiple trigger pulses per period . . . 772
Figure 204. Complementary output with symmetrical dead-time insertion . . . 773
Figure 205. Asymmetrical deadtime . . . 774
Figure 206. Dead-time waveforms with delay greater than the negative pulse . . . 774
Figure 207. Dead-time waveforms with delay greater than the positive pulse . . . 774
Figure 208. Break and Break2 circuitry overview . . . 777
Figure 209. Various output behavior in response to a break event on tim_brk (OSSI = 1) . . . 779
Figure 210. PWM output state following tim_brk and tim_brk2 assertion (OSSI = 1) . . . 780
Figure 211. PWM output state following tim_brk assertion (OSSI = 0) . . . 781
Figure 212. Output redirection (tim_brk2 request not represented) . . . 782
Figure 213. tim_ocref_clr input selection multiplexer . . . 783
Figure 214. Clearing TIMx tim_ocxref . . . 784
Figure 215. 6-step generation, COM example (OSSR = 1) . . . 785
Figure 216. Example of one pulse mode. . . . 786
Figure 217. Retriggerable one-pulse mode . . . 787
Figure 218. Pulse generator circuitry . . . 788
Figure 219. Pulse generation on compare event, for edge-aligned and encoder modes . . . 789
Figure 220. Extended pulsewidth in case of concurrent triggers . . . 790
Figure 221. Example of counter operation in encoder interface mode. . . . 792
Figure 222. Example of encoder interface mode with tim_ti1fp1 polarity inverted. . . . 792
Figure 223. Quadrature encoder counting modes . . . 793
Figure 224. Direction plus clock encoder mode . . . 794
Figure 225. Directional clock encoder mode (CC1P = CC2P = 0) . . . 794
Figure 226. Directional clock encoder mode (CC1P = CC2P = 1) . . . 795
Figure 227. Index gating options . . . 796
Figure 228. Jittered Index signals . . . 796
Figure 229. Index generation for IPOS[1:0] = 11 . . . 797
Figure 230. Counter reading with index gated on channel A (IPOS[1:0] = 11) . . . 797
Figure 231. Counter reading with index ungated (IPOS[1:0] = 00) . . . 798
Figure 232. Counter reading with index gated on channel A and B . . . 798
Figure 233. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11) . . . 799
Figure 234. Counter reset Narrow index pulse (closer view, ARR = 0x07) . . . 800
Figure 235. Index behavior in x1 and x2 mode (IPOS[1:0] = 01) . . . 801
Figure 236. Directional index sensitivity . . . 801
Figure 237. Counter reset as function of FIDX bit setting . . . 802
Figure 238. Index blanking . . . 802

Figure 239. Index behavior in clock + direction mode, IPOS[0] = 1 . . . 803
Figure 240. Index behavior in directional clock mode, IPOS[0] = 1 . . . 803
Figure 241. State diagram for quadrature encoded signals . . . 804
Figure 242. Up-counting encoder error detection . . . 805
Figure 243. Down-counting encode error detection . . . 806
Figure 244. Encoder mode change with preload transferred on update (SMSPS = 0) . . . 807
Figure 245. Measuring time interval between edges on three signals . . . 808
Figure 246. Example of Hall sensor interface . . . 810
Figure 247. Control circuit in reset mode . . . 811
Figure 248. Control circuit in Gated mode . . . 812
Figure 249. Control circuit in trigger mode . . . 813
Figure 250. Control circuit in external clock mode 2 + trigger mode . . . 814
Figure 251. General-purpose timer block diagram . . . 872
Figure 252. Counter timing diagram with prescaler division change from 1 to 2 . . . 877
Figure 253. Counter timing diagram with prescaler division change from 1 to 4 . . . 877
Figure 254. Counter timing diagram, internal clock divided by 1 . . . 878
Figure 255. Counter timing diagram, internal clock divided by 2 . . . 879
Figure 256. Counter timing diagram, internal clock divided by 4 . . . 879
Figure 257. Counter timing diagram, internal clock divided by N . . . 880
Figure 258. Counter timing diagram, Update event when ARPE = 0 (TIMx_ARR not preloaded) . . . 880
Figure 259. Counter timing diagram, Update event when ARPE = 1 (TIMx_ARR preloaded) . . . 881
Figure 260. Counter timing diagram, internal clock divided by 1 . . . 882
Figure 261. Counter timing diagram, internal clock divided by 2 . . . 883
Figure 262. Counter timing diagram, internal clock divided by 4 . . . 883
Figure 263. Counter timing diagram, internal clock divided by N . . . 884
Figure 264. Counter timing diagram, Update event . . . 884
Figure 265. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 . . . 886
Figure 266. Counter timing diagram, internal clock divided by 2 . . . 886
Figure 267. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36 . . . 887
Figure 268. Counter timing diagram, internal clock divided by N . . . 887
Figure 269. Counter timing diagram, Update event with ARPE = 1 (counter underflow) . . . 888
Figure 270. Counter timing diagram, Update event with ARPE = 1 (counter overflow) . . . 889
Figure 271. Control circuit in normal mode, internal clock divided by 1 . . . 890
Figure 272. tim_ti2 external clock connection example . . . 890
Figure 273. Control circuit in external clock mode 1 . . . 891
Figure 274. External trigger input block . . . 892
Figure 275. Control circuit in external clock mode 2 . . . 893
Figure 276. Capture/compare channel (example: channel 1 input stage) . . . 893
Figure 277. Capture/compare channel 1 main circuit . . . 894
Figure 278. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4) . . . 894
Figure 279. PWM input mode timing . . . 897
Figure 280. Output compare mode, toggle on tim_oc1 . . . 899
Figure 281. Edge-aligned PWM waveforms (ARR = 8) . . . 900
Figure 282. Center-aligned PWM waveforms (ARR = 8) . . . 901
Figure 283. Dithering principle . . . 902
Figure 284. Data format and register coding in dithering mode . . . 903
Figure 285. PWM resolution vs frequency (16-bit mode) . . . 904
Figure 286. PWM resolution vs frequency (32-bit mode) . . . 904
Figure 287. PWM dithering pattern . . . 905
Figure 288. Dithering effect on duty cycle in center-aligned PWM mode . . . 906
Figure 289. Generation of two phase-shifted PWM signals with 50% duty cycle . . . 908
Figure 290. Combined PWM mode on channels 1 and 3 . . . 909

Figure 291. OCREF_CLR input selection multiplexer . . . 910
Figure 292. Clearing TIMx tim_ocxref . . . 910
Figure 293. Example of One-pulse mode . . . 911
Figure 294. Retriggerable one-pulse mode . . . 913
Figure 295. Pulse generator circuitry . . . 914
Figure 296. Pulse generation on compare event, for edge-aligned and encoder modes . . . 914
Figure 297. Extended pulse width in case of concurrent triggers . . . 915
Figure 298. Example of counter operation in encoder interface mode . . . 917
Figure 299. Example of encoder interface mode with tim_ti1fp1 polarity inverted . . . 917
Figure 300. Quadrature encoder counting modes . . . 918
Figure 301. Direction plus clock encoder mode . . . 919
Figure 302. Directional clock encoder mode (CC1P = CC2P = 0) . . . 920
Figure 303. Directional clock encoder mode (CC1P = CC2P = 1) . . . 920
Figure 304. Index gating options . . . 922
Figure 305. Jittered Index signals . . . 922
Figure 306. Index generation for IPOS[1:0] = 11 . . . 923
Figure 307. Counter reading with index gated on channel A (IPOS[1:0] = 11) . . . 923
Figure 308. Counter reading with index ungated (IPOS[1:0] = 00) . . . 924
Figure 309. Counter reading with index gated on channel A and B . . . 924
Figure 310. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11) . . . 925
Figure 311. Counter reset Narrow index pulse (closer view, ARR = 0x07) . . . 926
Figure 312. Index behavior in x1 and x2 mode (IPOS[1:0] = 01) . . . 927
Figure 313. Directional index sensitivity . . . 927
Figure 314. Counter reset as function of FIDX bit setting . . . 928
Figure 315. Index blanking . . . 928
Figure 316. Index behavior in clock + direction mode, IPOS[0] = 1 . . . 929
Figure 317. Index behavior in directional clock mode, IPOS[0] = 1 . . . 929
Figure 318. State diagram for quadrature encoded signals . . . 930
Figure 319. Up-counting encoder error detection . . . 931
Figure 320. Down-counting encode error detection . . . 932
Figure 321. Encoder mode change with preload transferred on update (SMSPS = 0) . . . 933
Figure 322. Control circuit in reset mode . . . 935
Figure 323. Control circuit in gated mode . . . 936
Figure 324. Control circuit in trigger mode . . . 936
Figure 325. Control circuit in external clock mode 2 + trigger mode . . . 938
Figure 326. Master/Slave timer example . . . 938
Figure 327. Master/slave connection example with 1 channel only timers . . . 939
Figure 328. Gating TIM_slv with tim_oc1ref of TIM_mstr . . . 940
Figure 329. Gating TIM_slv with Enable of TIM_mstr . . . 941
Figure 330. Triggering TIM_slv with update of TIM_mstr . . . 942
Figure 331. Triggering TIM_slv with Enable of TIM_mstr . . . 942
Figure 332. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input . . . 943
Figure 333. Basic timer block diagram . . . 987
Figure 334. Control circuit in normal mode, internal clock divided by 1 . . . 988
Figure 335. Counter timing diagram with prescaler division change from 1 to 2 . . . 989
Figure 336. Counter timing diagram with prescaler division change from 1 to 4 . . . 990
Figure 337. Counter timing diagram, internal clock divided by 1 . . . 991
Figure 338. Counter timing diagram, internal clock divided by 2 . . . 991
Figure 339. Counter timing diagram, internal clock divided by 4 . . . 992
Figure 340. Counter timing diagram, internal clock divided by N . . . 992
Figure 341. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded) . . . 993

Figure 342. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded). . . . 994
Figure 343. Dithering principle . . . 995
Figure 344. Data format and register coding in dithering mode . . . 995
Figure 345. FCnt resolution vs frequency . . . 996
Figure 346. PWM dithering pattern . . . 996
Figure 347. LPTIM1/2 block diagram(1) . . . 1007
Figure 348. Glitch filter timing diagram . . . 1011
Figure 349. LPTIM output waveform, single-counting mode configuration when repetition register content is different than zero (with PRELOAD = 1) . . . 1013
Figure 350. LPTIM output waveform, single-counting mode configuration and Set-once mode activated (WAVE bit is set) . . . 1013
Figure 351. LPTIM output waveform, Continuous counting mode configuration . . . 1014
Figure 352. Waveform generation . . . 1015
Figure 353. Encoder mode counting sequence . . . 1019
Figure 354. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1). . . . 1020
Figure 355. Capture/compare input stage (channel 1) . . . 1021
Figure 356. Capture/compare output stage (channel 1) . . . 1021
Figure 357. Edge-aligned PWM mode (PRELOAD = 1) . . . 1023
Figure 358. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0) . . . 1024
Figure 359. PWM mode with immediate update versus preloaded update . . . 1025
Figure 360. Independent watchdog block diagram . . . 1051
Figure 361. Reset timing due to timeout . . . 1053
Figure 362. Reset timing due to refresh in the not allowed area . . . 1054
Figure 363. Changing PR, RL, and performing a refresh(1) . . . 1055
Figure 364. Window comparator update(1) . . . 1056
Figure 365. Independent watchdog interrupt timing diagram . . . 1058
Figure 366. Early wake-up comparator update(1) . . . 1059
Figure 367. Watchdog block diagram . . . 1067
Figure 368. Window watchdog timing diagram . . . 1069
Figure 369. RTC block diagram . . . 1074
Figure 370. TAMP block diagram . . . 1123
Figure 371. Backup registers protection zones . . . 1127
Figure 372. Tamper sampling with precharge pulse . . . 1131
Figure 373. Low level detection with precharge and filtering . . . 1132
Figure 374. Active tamper filtering . . . 1134
Figure 375. Block diagram . . . 1163
Figure 376. I²C-bus protocol . . . 1165
Figure 377. Setup and hold timings . . . 1167
Figure 378. I2C initialization flow . . . 1169
Figure 379. Data reception . . . 1170
Figure 380. Data transmission . . . 1171
Figure 381. Target initialization flow . . . 1174
Figure 382. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0 . . . 1176
Figure 383. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1 . . . 1177
Figure 384. Transfer bus diagrams for I2C target transmitter (mandatory events only) . . . 1178
Figure 385. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0 . . . 1179
Figure 386. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1 . . . 1180
Figure 387. Transfer bus diagrams for I2C target receiver (mandatory events only) . . . 1180
Figure 388. Controller clock generation . . . 1182

Figure 389. Controller initialization flow . . . 1184
Figure 390. 10-bit address read access with HEAD10R = 0 . . . 1184
Figure 391. 10-bit address read access with HEAD10R = 1 . . . 1185
Figure 392. Transfer sequence flow for I2C controller transmitter, N ≤ 255 bytes . . . 1186
Figure 393. Transfer sequence flow for I2C controller transmitter, N > 255 bytes . . . 1187
Figure 394. Transfer bus diagrams for I2C controller transmitter (mandatory events only) . . . 1188
Figure 395. Transfer sequence flow for I2C controller receiver, N ≤ 255 bytes . . . 1190
Figure 396. Transfer sequence flow for I2C controller receiver, N > 255 bytes . . . 1191
Figure 397. Transfer bus diagrams for I2C controller receiver (mandatory events only) . . . 1192
Figure 398. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$ . . . 1196
Figure 399. Transfer sequence flow for SMBus target transmitter N bytes + PEC . . . 1199
Figure 400. Transfer bus diagram for SMBus target transmitter (SBC = 1) . . . 1200
Figure 401. Transfer sequence flow for SMBus target receiver N bytes + PEC . . . 1201
Figure 402. Bus transfer diagrams for SMBus target receiver (SBC = 1) . . . 1202
Figure 403. Bus transfer diagrams for SMBus controller transmitter . . . 1203
Figure 404. Bus transfer diagrams for SMBus controller receiver . . . 1205
Figure 405. I3C block diagram . . . 1229
Figure 406. I3C (primary) controller state and programming sequence diagram . . . 1233
Figure 407. I3C target state and programing sequence diagram . . . 1238
Figure 408. I3C CCC messages, as controller . . . 1252
Figure 409. I3C broadcast ENTDAA CCC, as controller . . . 1253
Figure 410. I3C broadcast, direct read and direct write RSTACT CCC, as controller . . . 1254
Figure 411. I3C CCC messages, as target . . . 1256
Figure 412. I3C broadcast ENTDAA CCC, as target . . . 1257
Figure 413. I3C broadcast DEFTGTS CCC, as target . . . 1258
Figure 414. I3C broadcast DEFGRPA CCC, as target . . . 1259
Figure 415. I3C private read/write messages, as controller . . . 1261
Figure 416. I3C private read/write messages, as target . . . 1262
Figure 417. Legacy I2C read/write messages, as controller . . . 1263
Figure 418. IBI transfer, as controller/target . . . 1264
Figure 419. Hot-join request transfer, as controller/target . . . 1265
Figure 420. Controller-role request transfer, as controller/target . . . 1266
Figure 421. C-FIFO management, as controller . . . 1267
Figure 422. TX-FIFO management, as controller . . . 1269
Figure 423. RX-FIFO management, as controller . . . 1271
Figure 424. S-FIFO management, as controller . . . 1274
Figure 425. RX-FIFO management, as target . . . 1275
Figure 426. TX-FIFO management with I3C_TGTTDR, as target . . . 1277
Figure 427. TX-FIFO management by software without I3C_TGTTDR if reading less bytes than TX-FIFO size, as target . . . 1279
Figure 428. USART block diagram . . . 1337
Figure 429. Word length programming . . . 1341
Figure 430. Configurable stop bits . . . 1343
Figure 431. TC/TXE behavior when transmitting . . . 1345
Figure 432. Start bit detection when oversampling by 16 or 8 . . . 1346
Figure 433. usart_ker_ck clock divider block diagram . . . 1349
Figure 434. Data sampling when oversampling by 16 . . . 1350
Figure 435. Data sampling when oversampling by 8 . . . 1351
Figure 436. Mute mode using Idle line detection . . . 1358
Figure 437. Mute mode using address mark detection . . . 1359

Figure 438. Break detection in LIN mode (11-bit break length - LBDL bit is set) . . . 1362
Figure 439. Break detection in LIN mode vs. Framing error detection. . . . 1363
Figure 440. USART example of synchronous master transmission. . . . 1364
Figure 441. USART data clock timing diagram in synchronous master mode (M bits = 00) . . . 1364
Figure 442. USART data clock timing diagram in synchronous master mode (M bits = 01) . . . 1365
Figure 443. USART data clock timing diagram in synchronous slave mode (M bits = 00) . . . 1366
Figure 444. ISO 7816-3 asynchronous protocol . . . 1368
Figure 445. Parity error detection using the 1.5 stop bits . . . 1370
Figure 446. IrDA SIR ENDEC block diagram . . . 1374
Figure 447. IrDA data modulation (3/16) - normal mode . . . 1374
Figure 448. Transmission using DMA . . . 1376
Figure 449. Reception using DMA . . . 1377
Figure 450. Hardware flow control between two USARTs . . . 1377
Figure 451. RS232 RTS flow control . . . 1378
Figure 452. RS232 CTS flow control . . . 1379
Figure 453. Wake-up event verified (wake-up event = address match, FIFO disabled) . . . 1382
Figure 454. Wake-up event not verified (wake-up event = address match, FIFO disabled) . . . 1382
Figure 455. LPUART block diagram . . . 1426
Figure 456. LPUART word length programming . . . 1429
Figure 457. Configurable stop bits . . . 1431
Figure 458. TC/TXE behavior when transmitting . . . 1433
Figure 459. lpuart_ker_ck clock divider block diagram . . . 1437
Figure 460. Mute mode using Idle line detection . . . 1441
Figure 461. Mute mode using address mark detection . . . 1442
Figure 462. Transmission using DMA . . . 1444
Figure 463. Reception using DMA . . . 1445
Figure 464. Hardware flow control between two LPUARTs . . . 1446
Figure 465. RS232 RTS flow control . . . 1446
Figure 466. RS232 CTS flow control . . . 1447
Figure 467. Wake-up event verified (wake-up event = address match, FIFO disabled) . . . 1450
Figure 468. Wake-up event not verified (wake-up event = address match, FIFO disabled) . . . 1450
Figure 469. SPI/I2S block diagram . . . 1482
Figure 470. Full-duplex single master/ single slave application . . . 1485
Figure 471. Half-duplex single master/ single slave application . . . 1486
Figure 472. Simplex single master / single slave application (master in transmit-only / slave in receive-only mode) . . . 1487
Figure 473. Master and three independent slaves connected in star topology . . . 1488
Figure 474. Master and three slaves connected in circular (daisy chain) topology . . . 1489
Figure 475. Multimaster application . . . 1490
Figure 476. Scheme of SS control logic . . . 1492
Figure 477. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0) . . . 1492
Figure 478. SS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0) . . . 1493
Figure 479. Data clock timing diagram . . . 1495
Figure 480. Data alignment when data size is not equal to 8, 16 or 32 bits . . . 1496
Figure 481. TI mode transfer . . . 1505
Figure 482. Optional configurations of the slave behavior when an underrun condition is detected . 1507

Figure 483. Waveform examples . . . 1515
Figure 484. Master I2S Philips protocol waveforms (16/32-bit full accuracy) . . . 1516
Figure 485. I2S Philips standard waveforms . . . 1516
Figure 486. Master MSB-justified 16- or 32-bit full-accuracy length . . . 1517
Figure 487. Master MSB-justified 16- or 24-bit data length . . . 1517
Figure 488. Slave MSB-justified 16-, 24- or 32-bit data length . . . 1518
Figure 489. LSB-justified 16 or 24-bit data length . . . 1518
Figure 490. Master PCM when the frame length is equal the data length . . . 1519
Figure 491. Master PCM standard waveforms (16 or 24-bit data length) . . . 1519
Figure 492. Slave PCM waveforms . . . 1520
Figure 493. Startup sequence, I2S Philips standard, master . . . 1523
Figure 494. Startup sequence, I2S Philips standard, slave . . . 1524
Figure 495. Stop sequence, I2S Philips standard, master . . . 1524
Figure 496. I^2S clock generator architecture . . . 1525
Figure 497. Data Format . . . 1527
Figure 498. Handling of underrun situation . . . 1529
Figure 499. Handling of overrun situation . . . 1530
Figure 500. Frame error detection, with FIXCH = 0 . . . 1531
Figure 501. Frame error detection, with FIXCH = 1 . . . 1531
Figure 502. CAN subsystem . . . 1556
Figure 503. FDCAN block diagram . . . 1558
Figure 504. Bit timing . . . 1560
Figure 505. Transceiver delay measurement . . . 1565
Figure 506. Pin control in bus monitoring mode . . . 1566
Figure 507. Pin control in loop-back mode . . . 1569
Figure 508. CAN error state diagram . . . 1570
Figure 509. Message RAM configuration . . . 1571
Figure 510. Standard message ID filter path . . . 1574
Figure 511. Extended message ID filter path . . . 1575
Figure 512. USB peripheral block diagram . . . 1622
Figure 513. Packet buffer areas with examples of buffer description table locations . . . 1628
Figure 514. Block diagram of debug support infrastructure . . . 1669
Figure 515. Product life cycle states and debug authentication . . . 1672
Figure 516. JTAG TAP state machine . . . 1674
Figure 517. CoreSight topology . . . 1695
Figure 518. Trace port interface unit (TPIU) . . . 1778
Figure 519. Embedded cross trigger . . . 1791

1 Documentation conventions

1.1 General information

The STM32H503xx devices have an Arm®[a] Cortex-M33 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.

1.3 Register reset value

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

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.

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.

2 Memory and bus architecture

2.1 System architecture

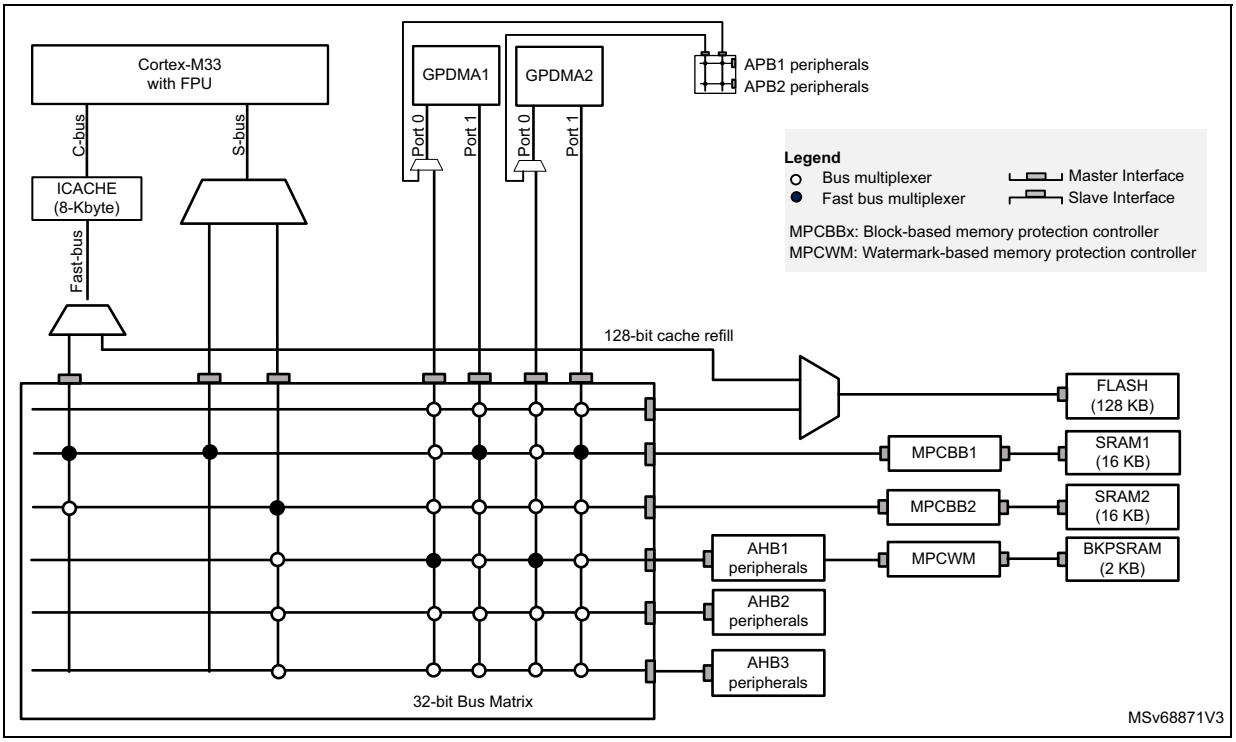
The STM32H503xx architecture relies on an Arm Cortex-M33 core optimized for execution thanks to an instruction cache having a direct access to the embedded Flash memory.

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

- up to 7 masters:
  - Fast C-bus, connecting Cortex-M33 with FPU core C-bus to the internal SRAMs through the instruction cache
  - Cortex-M33 with FPU core S-bus (two masters connected to two internal SRAMs without latency)
  - GPDMA1 (general purpose DMA featuring two master ports)
  - GPDMA2 (general purpose DMA featuring two master ports)
- up to 6 slaves:
  - internal flash memory (128 Kbytes)
  - internal SRAM1 (16 Kbytes)
  - internal SRAM2 (16 Kbytes)
  - AHB1 peripherals and backup RAM (2-Kbyte BKPSRAM) including AHB to APB bridges and APB peripherals (connected to APB1 and APB2)
  - AHB2 peripherals
  - AHB3 peripherals, including AHB to APB bridge and APB peripherals (connected to APB3)

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. This architecture is shown in the figure below.

Figure 1. System architecture



2.1.1 Fast C-bus

This bus connects the C-bus of the Cortex-M33 core to the internal Flash memory and to the bus matrix via the instruction cache. This bus is used for instruction fetch and data access to the internal memories mapped in code region. This bus targets the internal Flash memory and the internal SRAMs (SRAM1 and SRAM2).

SRAM1 and SRAM2 are accessible on this bus with a continuous mapping.

2.1.2 S-bus

This bus connects the system bus of the Cortex-M33 core to the bus matrix. This bus is used by the core to access data located in a peripheral or SRAM area. This bus targets the internal SRAMs (SRAM1, SRAM2, and BKPSRAM), the AHB1 peripherals (including the APB1, APB2), AHB2, and AHB3 peripherals.

SRAM1 and SRAM2 are accessible on this bus with a continuous mapping.

Note: The bus matrix has a zero latency when accessing SRAM1 and SRAM2.

2.1.3 GPDMA1 and GPDMA2 buses

These buses connect the four AHB master interfaces of the GPDMA1 and GPDMA2 to the bus matrix. These buses target the internal Flash memory, the internal SRAMs (SRAM1, SRAM2, and BKPSRAM), the AHB1 peripherals (including the APB1, APB2), AHB2, and AHB3 peripherals.

2.1.4 Bus matrix

The bus matrix manages the access arbitration between masters. The arbitration uses a Round-Robin algorithm. This bus matrix features a fast bus multiplexer used to connect each master to a given slave without latency (see *Figure 1*). For the same master, other slaves undergo a latency of at least one cycle at each new access.

2.1.5 AHB/APB bridges

The three AHB/APB bridges provide full synchronous connections between the AHB and the APB buses, allowing flexible selection of the peripheral frequency.

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the address mapping of the peripherals connected to these bridges.

After each device reset, all peripheral clocks are disabled (except for the internal SRAMs and Flash memory interface). Before using a peripheral, its clock must be enabled in the RCC_AHBxENR and RCC_APBxENR registers.

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

2.2 Memory organization

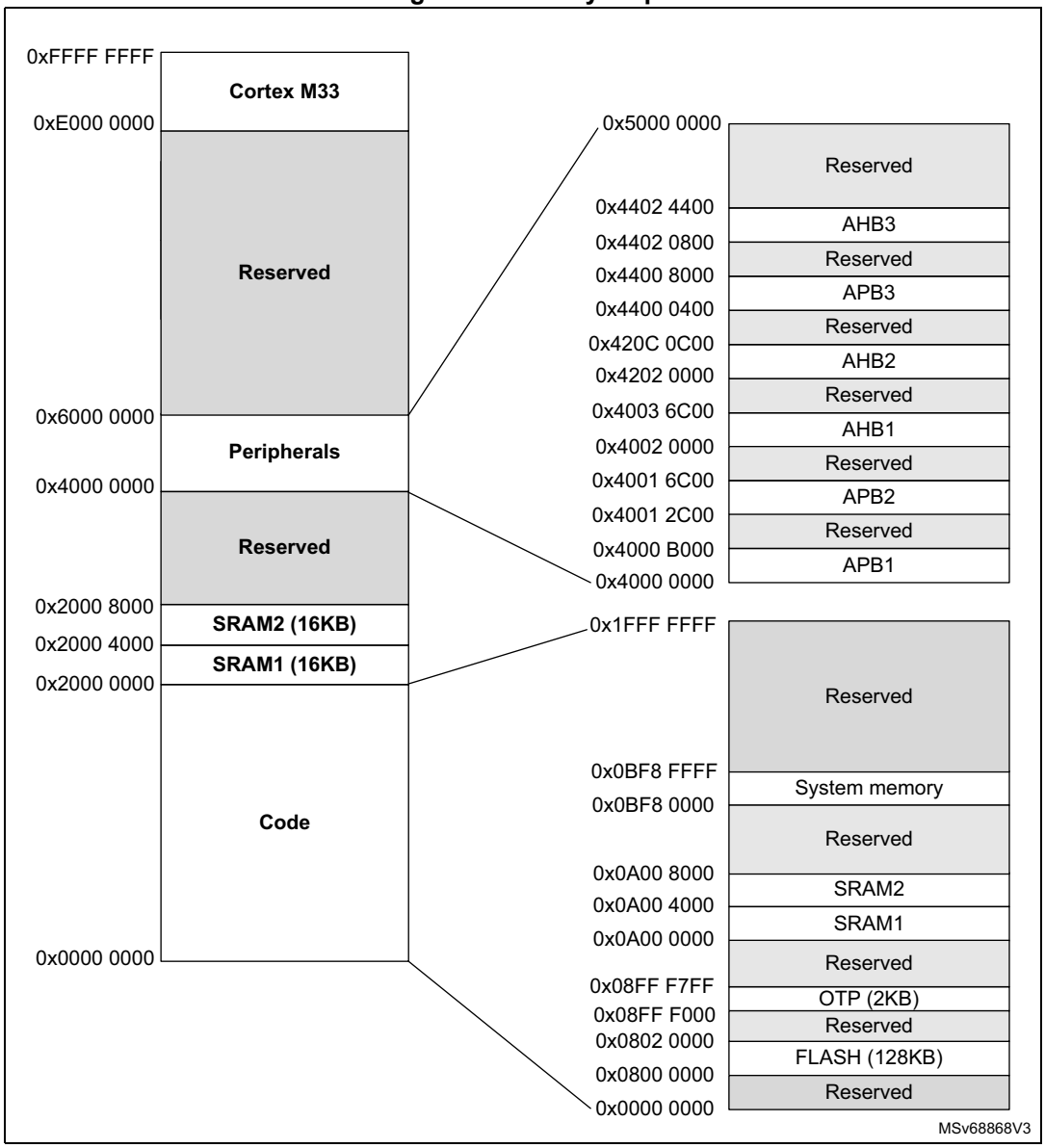
2.2.1 Introduction

Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte 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.

2.2.2 Memory map and register boundary addresses

Figure 2. Memory map



All the memory map areas that are not allocated to on-chip memories and peripherals are considered “Reserved”. The following table gives the boundary addresses of the peripherals available in the devices.

Table 1. Memory map and peripheral register boundary addresses

Bus	Boundary address	Peripheral	Peripheral register map
AHB3	0x4402 4400 - 0x44FF FFFF	Reserved	-
	0x4402 4000 - 0x4402 43FF	DEBUG	DEBUG register map
	0x4402 3000 - 0x4402 3FFF	Reserved	-
	0x4402 2000 - 0x4402 23FF	EXTI	EXTI register map
	0x4402 1000 - 0x4402 1FFF	Reserved	-
	0x4402 0C00 - 0x4402 0FFF	RCC	RCC register map
	0x4402 0800 - 0x4402 0BFF	PWR	PWR register map
	0x4402 0000 - 0x4402 07FF	Reserved	-
APB3	0x4400 8000 - 0x4400 FFFF	Reserved	-
	0x4400 7C00 - 0x4400 7FFF	TAMP	TAMP register map
	0x4400 7800 - 0x4400 7BFF	RTC	RTC register map
	0x4400 4800 - 0x4400 77FF	Reserved	-
	0x4400 4400 - 0x4400 47FF	LPTIM1	LPTIM1 register map
	0x4400 3400 - 0x4400 43FF	Reserved	-
	0x4400 3000 - 0x4400 33FF	I3C2	I3C2 register map
	0x4400 2800 - 0x4400 2FFF	Reserved	-
	0x4400 2400 - 0x4400 27FF	LPUART1	LPUART register map
	0x4400 0800 - 0x4400 23FF	Reserved	-
	0x4400 0400 - 0x4400 07FF	SBS	SBS register map
	0x4400 0000 - 0x4400 03FF	Reserved	-

Table 1. Memory map and peripheral register boundary addresses (continued)

Bus	Boundary address	Peripheral	Peripheral register map
AHB2	0x420C 0C00 - 0x43FF FFFF	Reserved	-
	0x420C 0800 - 0x420C 0BFF	RNG	RNG register map
	0x420C 0400 - 0x420C 07FF	HASH	HASH register map
	0x4202 8800 - 0x420C 03FF	Reserved	-
	0x4202 8400 - 0x4202 87FF	DAC1	DAC register map
	0x4202 8000 - 0x4202 83FF	ADC1	ADC register map
	0x4202 2000 - 0x4202 7FFF	Reserved	-
	0x4202 1C00 - 0x4202 1FFF	GPIOH	GPIO register map
	0x4202 1000 - 0x4202 1BFF	Reserved	-
	0x4202 0C00 - 0x4202 0FFF	GPIOD	GPIO register map
	0x4202 0800 - 0x4202 0BFF	GPIOC	
	0x4202 0400 - 0x4202 07FF	GPIOB	
	0x4202 0000 - 0x4202 03FF	GPIOA	
AHB1	0x4003 6C00 - 0x41FF FFFF	Reserved	-
	0x4003 6400 - 0x4003 6BFF	BKPSRAM	-
	0x4003 3400 - 0x4003 63FF	Reserved	-
	0x4003 3000 - 0x4003 33FF	GTZC1_MPCBB2	GTZC1 MPCBBz registers (z = 1 to 2)
	0x4003 2C00 - 0x4003 2FFF	GTZC1_MPCBB1	GTZC1 MPCBBz registers (z = 1 to 2)
	0x4003 2800 - 0x4003 2BFF	Reserved	-
	0x4003 2400 - 0x4003 27FF	GTZC1_TZSC	GTZC1 TZSC register map
	0x4003 0800 - 0x4003 23FF	Reserved	-
	0x4003 0400 - 0x4003 07FF	ICACHE	ICACHE register map
	0x4002 7000 - 0x4003 03FF	Reserved	-
	0x4002 6000 - 0x4002 6FFF	RAMCFG	RAMCFG register map
	0x4002 3400 - 0x4002 5FFF	Reserved	-
	0x4002 3000 - 0x4002 33FF	CRC	CRC register map
	0x4002 2400 - 0x4002 2FFF	Reserved	-
	0x4002 2000 - 0x4002 23FF	FLASH	FLASH register map
	0X4002 1000 - 0x4002 1FFF	GPDMA2	GPDMA register map
	0x4002 0000 - 0x4002 0FFF	GPDMA1	

Table 1. Memory map and peripheral register boundary addresses (continued)

Bus	Boundary address	Peripheral	Peripheral register map
APB2	0x4001 6C00 - 0x4001 FFFF	Reserved	-
	0x4001 6400 - 0x4001 6BFF	USB RAM	USB register map
	0x4001 6000 - 0x4001 63FF	USB	
	0x4001 3C00 - 0x4001 5FFF	Reserved	-
	0x4001 3800 - 0x4001 3BFF	USART1	USART register map
	0x4001 3400 - 0x4001 37FF	Reserved	-
	0x4001 3000 - 0x4001 33FF	SPI1 / I2S1	SPI register map
	0x4001 2C00 - 0x4001 2FFF	TIM1	TIMx register map
	0x4001 0000 - 0x4001 2BFF	Reserved	-

Table 1. Memory map and peripheral register boundary addresses (continued)

Bus	Boundary address	Peripheral	Peripheral register map
APB1	0x4000 B000 - 0x4000 FFFF	Reserved	-
	0x4000 AC00 - 0x4000 AFFF	FDCAN SRAM	FDCAN register map
	0x4000 A800 - 0x4000 ABFF	Reserved	
	0x4000 A400 - 0x4000 A7FF	FDCAN1	FDCAN register map
	0x4000 9800 - 0x4000 A3FF	Reserved	-
	0x4000 9400 - 0x4000 97FF	LPTIM2	LPTIM register map
	0x4000 9000 - 0x4000 93FF	Reserved	-
	0x4000 8C00 - 0x4000 8FFF	DTS	DTS register map
	0x4000 6400 - 0x4000 8BFF	Reserved	-
	0x4000 6000 - 0x4000 63FF	CRS	CRS register map
	0x4000 5C00 - 0x4000 5FFF	I3C1	I3C register map
	0x4000 5800 - 0x4000 5BFF	I2C2	I2C register map
	0x4000 5400 - 0x4000 57FF	I2C1	
	0x4000 4C00 - 0x4000 53FF	Reserved	-
	0x4000 4800 - 0x4000 4BFF	USART3	USART register map
	0x4000 4400 - 0x4000 47FF	USART2	
	0x4000 4000 - 0x4000 43FF	COMP	COMP register map
	0x4000 3C00 - 0x4000 3FFF	SPI3 / I2S3	SPI register map
	0x4000 3800 - 0x4000 3BFF	SPI2 / I2S2	
	0x4000 3400 - 0x4000 37FF	OPAMP	OPAMP register map
	0x4000 3000 - 0x4000 33FF	IWDG	IWDG hardware configuration register (IWDG_HWCFGR)
	0x4000 2C00 - 0x4000 2FFF	WWDG	WWDG register map
	0x4000 1800 - 0x4000 2BFF	Reserved	-
	0x4000 1400 - 0x4000 17FF	TIM7	TIM6/TIM7 register map
	0x4000 1000 - 0x4000 13FF	TIM6	
	0x4000 0800 - 0x4000 0FFF	Reserved	-
	0x4000 0400 - 0x4000 07FF	TIM3	TIM2/TIM3 register
	0x4000 0000 - 0x4000 03FF	TIM2	

2.2.3 Embedded SRAM

The devices feature up to 32-Kbyte SRAMs:

- 16-Kbyte SRAM1
- 16-Kbyte SRAM2
- 2-Kbyte BKPSRAM

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

The CPU can access the SRAM1 and SRAM2 through the system bus or through the C-bus depending on the selected address. The CPU can access the BKPSRAM through the system bus only.

SRAM features are detailed in *Section 5.3.1: Internal SRAMs features*.

2.2.4 Flash memory overview

The flash memory is composed of two distinct physical areas:

- The main flash memory block that contains the application program and user data.
- The information block that is composed of the following parts:
  - option bytes for hardware and memory protection user configuration
  - system memory that contains ST proprietary code
  - OTP (one-time programmable) area

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

3 System security

The STM32H503xx is designed with a comprehensive set of security features.

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 reuse, improving the interoperability, and minimizing the API fragmentation.

This section explains the different security features available on the STM32H503xx devices.

3.1 Key security features

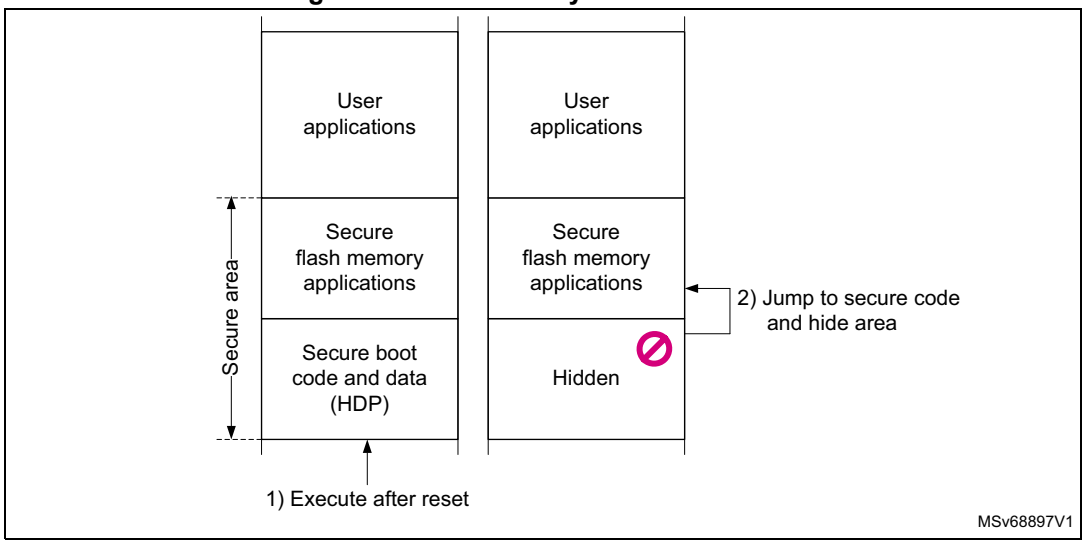
- Resource isolation using privilege mode of Cortex-M33, extended to privileged, memories, and peripherals
- Secure debug control (regression control). The secure debug control can be personalized in OTP by OEMs thanks to provisioning tools
- Temporal isolation: boot levels are isolated thanks to the HDPL (hide protect level) monotonic counter
- Battery-powered volatile secure storage, automatically erased in case of tamper
- Device 96-bit unique ID
- HASH processor, supporting SHA-1 checksums and SHA-2 secure hash (SHA2-224, SHA2-256)
- True random number generator (RNG), NIST SP800-90B precertified
- New flexible lifecycle scheme that enables in-the-field product maintenance
- Active tamper and protection against temperature, voltage, and frequency attacks
  - Active tamper available in different power modes (refer to TAMP low-power modes)

3.2 Resource isolation

3.2.1 Temporal isolation using secure hide protection (HDP)

The embedded flash memory allows an HDP area per watermarked-secure area of each bank (8-Kbyte sector 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 pictured on the figure below.

Figure 3. Flash memory secure HDP area



Activation of HDP area in user flash memory is related to HDPL1. It means that as soon as HDPL becomes equal or greater to 2, then data read, write, and instruction fetch (on the area defined by HDPx_PSTRT and HDPx_PEND in FLASH_HDPxR_PRG option bytes), are denied until the next device reset.

The END of the HDPx areas can be extended (dynamically by the application) thanks to FLASH_HDPEXTR flash memory register. This extension prevents access to protected sectors under HDPL3.

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

In the STM32H503xx devices, the hardware and software resources used to boot can be isolated. This is called temporal isolation.

It is based on a monotonic counter taking care of only incrementing the levels. When the counter is incremented, the resources of the previous levels become hidden (code and data) For more information, see *Section 7: Embedded flash memory (FLASH)*.

3.2.2 Resource isolation using Cortex privileged mode

The hardware and software resources of STM32H503xx devices can be partitioned so that they are restricted to software running in Cortex privileged mode.

Thanks to this hardware isolation technology, 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 region can be programmed to generate faults when accessed inappropriately by unprivileged software.

3.2.3 Resource isolation using GTZC privilege control

In the STM32H503 devices, the hardware and software resources can be partitioned so that they exist either in the privileged world or in the unprivileged world. Refer to *Section 6: Global privilege controller (GTZC)*.

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.

For masters other than the Cortex-M33, the GTZC can assign them as unprivileged initiators, automatically protecting resources defined as privileged against this master.

- Privileging peripherals with TZSC (privileged-only)

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

  – A 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 privilege-only.

  When such peripheral is made privileged-only with GTZC, if it is master on the interconnect (GPDMA), it automatically issues privileged transactions. Privilege-aware masters like GPDMA1 and GPDMA2, drive the privileged signal in the AHB interconnect according to their internal privileged mode, independently to the GTZC.

  The list of privilege-aware and privilege-only peripherals can be found in *Section 6: Global privilege controller (GTZC)*.

- Privileging memories with TZSC and MPCBB (privileged-only)

  The TZSC logic in GTZC provides the capability to manage the privilege level for BKPSRAM, programming the MPCWM resources defined in *Section 3.2.2*.

  Similarly, the TZSC logic in GTZC provides the capability to configure the privilege level of embedded SRAM blocks, programming the MPCBB resources defined in *Section 3.2.2*.

- 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 privilege in privileged-aware peripherals

Privileged-aware peripherals also implement privileged-only access mode:

- *Embedded flash memory*

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

  When a privileged bit NSPRIV is set in FLASH_PRIVCFGR, reading and writing the flash memory 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 both in privileged and unprivileged modes.
  – Each HDP area, is accessible in the privileged and unprivileged mode, if applicable.
  – Each 8-Kbyte sector of the embedded flash memory can be programmed on-the-fly as privileged only, using the block-based privileged configuration registers FLASH_PRIVBB1R1 and FLASH_PRIVBB2R1. An unprivileged sector is accessible by a privileged or unprivileged access.

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

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

- *Direct memory access controllers (GPDMAx)*

  When a DMA channel x is defined as privileged (PRIVx = 1 in GPDMA_PRIVCFGR), special rules apply when accessing the privileged/unprivileged source or destination. Those rules are summarized on the table below.

Table 2. 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.

- *Power control (PWR)*

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

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

  See *Section 9: Power control (PWR)* for details.

- *Reset and control block (RCC)*

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

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

  See *Section 10: Reset and clock control (RCC)* for details.

- *Real time clock (RTC)*

  By default after any 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_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 PRIV bit is cleared in 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 32: Real-time clock (RTC)* chapter for details.

- *Tamper and backup registers (TAMP)*

  By default after any 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_CFGR or TAMP_PRIVCFGR is always possible in privilege and unprivilege modes. All the other TAMP registers can be read only in privilege 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.

- *Extended interrupts and event controller (EXTI)*

  The EXTI peripheral is able to 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_PRIVCFGRx registers. When an input event is configured as privileged, only a privileged application can change the configuration, change the masking or clear the status of this input event.

  See *Section 17: Extended interrupts and event controller (EXTI)* for more details.

- *System configuration boot and security (SBS)*

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

  - SBS registers for CPU configuration: SBS _FPUIMR, SBS _CNSLCKR, and SBS_ECCNMIR

  See *Section 13: System configuration, boot, and security (SBS)* for more details.

3.3 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.

3.3.1 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 MPU split memory into up to eight regions.

3.3.2 Embedded flash memory write protection

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

Write-protected sectors can be modified through option byte changes only whern PRODUCT_STATE = open.

Note: Bank erase aborts when it contains a write-protected area (WRP or HDP area) For more information, see Section 7.5.6: Write protection.

3.3.3 Tamper detection and response

Principle

The devices include active protection of critical security assets against temperature, voltage and frequency attacks, with the following features:

- erasure of device secrets upon tamper detection
- improved guarantee of safe execution for the CPU and its associated security peripherals, including:
  - out-of-range voltage (example: V_{BAT}, V_{DDA}), temperature and clocking (LSE) detection
  - security watchdog IWDG clocked by the internal oscillator LSI
  - possible selection of internal oscillator HSI as system clock
- power supply protection
  - RTC/TAMP domain powered automatically with V_{DD} or V_{BAT}

See *Section 33: Tamper and backup registers (TAMP)* for more details.

Tamper detection sources

The devices support two active input/output pins, which allow one independent active-tamper mesh.

The active pins are clocked by the LSE, and are functional in different system operating modes (run, sleep, stop, or standby), and in VBAT mode. Refer to TAMP pins functionality over modes for list of tamper pins and their availability across power modes.

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 the table below.

Table 3. Internal tampers in TAMP

Tamper input	NOER bit number in TAMP_CR3	Tamper source
itamp1	0	Backup domain voltage continuous monitoring, functional in VBAT mode
itamp2	1	Temperature monitoring, functional in VBAT mode
itamp3	2	LSE monitoring(1), functional in VBAT mode
itamp4	3	HSE monitoring
itamp5	4	RTC calendar overflow (rtc_calovf)
itamp6	5	JTAG/SWD access
itamp7, 12, 13	6, 11, 12	Voltage monitoring (V_{CORE}), through ADC analog watchdog
itamp8	7	Monotonic counter overflow (generated internally)

Table 3. Internal tampers in TAMP (continued)

Tamper input	NOER bit number in TAMP_CR3	Tamper source
itamp9	8	Fault generation for RNG peripheral
itamp11	10	IWDG timeout and potential tamper (IWDG reset when at least one enabled tamper flag is set)
itamp15	14	System fault detection

1. LSE missing or over frequency detection (> 2 MHz), glitch filter (> 2 MHz).

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 mode and Standby modes (see TAMPxMSK bits in TAMP_CR2 register).
- Generate a hardware trigger for the low-power timers.
- Erase device secrets if the corresponding TAMPxNOER bit is cleared in TAMP_CR2 (for tamper pins) or TAMP_CR3 (for internal tamper). These erasable secrets are:
  - symmetric keys stored in backup registers (x32), and in HASH
  - other secrets stored in SRAM2 and CPU instruction cache memory (SRAM2 erased when V_{DD} is present)
  - 2-Kbyte backup SRAM (depending on configuration bit), erased when V_{DD} is present

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.

If V_{DD} is not present, the secrets that are erased when V_{DD} is present, are only erased at the next V_{DD} power on.

Note: Device secret erase is also triggered by setting the BKERASE bit in TAMP_CR2, or by performing a PRODUCT_STATE regression as defined in Section 3.5.1.

Device secrets are not reset by system reset or when the device wakes up from Standby mode.

Software filtering mechanism

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

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

- Backup registers, backup SRAM, SRAM2: read-as-zero, write-ignored
- HASH peripherals: automatically reset by RCC

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 erase of secrets.

Tamper detection and low-power modes.

The effect of low-power modes on a tamper detection are summarized on the table below.

Table 4. Effect of low-power modes on TAMP

Mode	Description
Sleep	No effect on tamper detection features TAMP interrupts cause the device to exit the Sleep mode.
Stop	No effect on tamper detection features, except for level detection with filtering and active tamper modes that remain active only when the clock source is LSE or LSI. Tamper events cause the device to exit the Stop mode.
Standby	No effect on tamper detection features, except for level detection with filtering and active tamper modes. They remain active only when the clock source is LSE or LSI. Tamper events cause the device to exit the Standby mode.

3.4 Unique ID

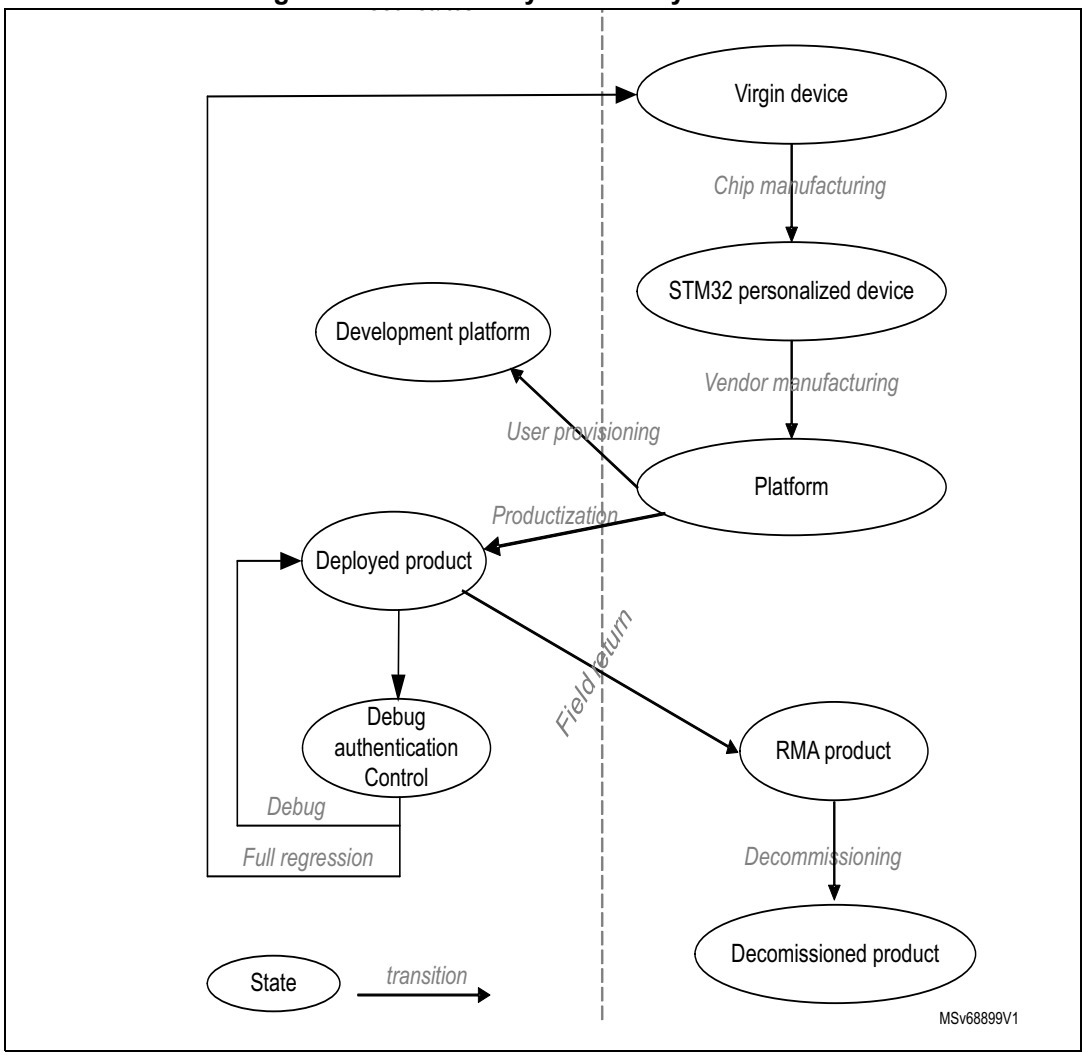
The devices store a 96-bit ID that is unique to each device (see *Section 42.1: Unique device ID register (96 bits)*).

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

3.5 Product life-cycle

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

Figure 4. 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 the table below.

Table 5. 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.
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 (“production” transition) or he/she can 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.

Table 5. Main product life-cycle transitions (continued)

Transitions	Description
User provisioning	Platform vendor prepares an individual platform for development, not to be connected to a production cloud network.
Field return	The product is returned for analysis to field return centers. Analysis is possible by launching a full regression. Such access requires provisioning the hash of the password provided by the vendor.

3.5.1 Life-cycle management

The product life-cycle allows to control access to different assets (code and data) of the product, including during development, manufacturing, and after sales.

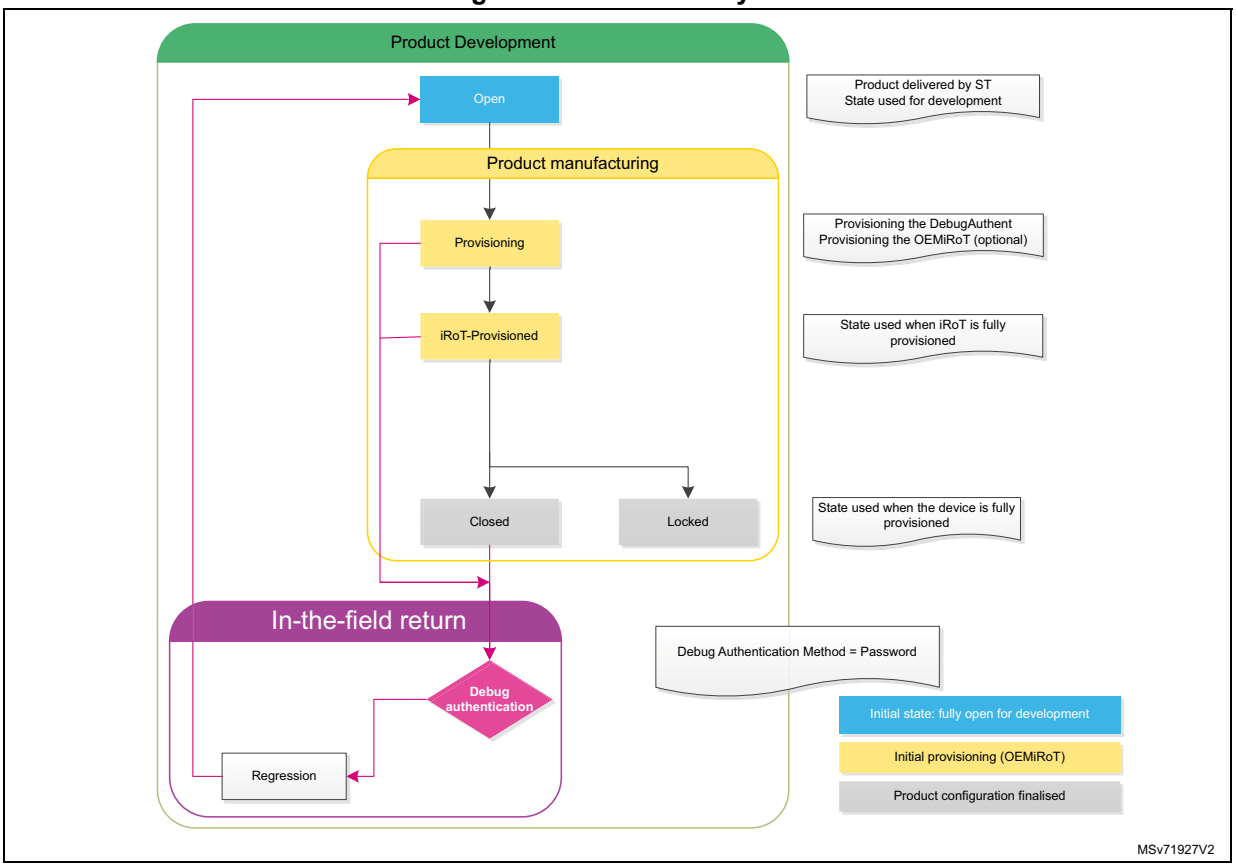
It allows the provisioning of the product with different distribution models taking care on the code and data provisioned.

Table 6. Typical product life-cycle phases

PRODUCT_STATE + DebugState		Debug (default configuration)	Comments
Open	Device open	Full available	This state allows the development of the product, as it provides the debug of the code. Using the boot pin enables the launch of the bootloader.
Provisioning	Debug partially opened	HDPL3	This state allows the management of the provisioning of the product (partial or full). It permits the launch, or bootloader to provision the product.
iROT-Provisioned	Debug partially opened	HDPL3	This state considers that an immutable root of trust is installed, including its configuration (code, option bytes).
Closed	Debug closed	NoDebug + debug authentication control	This state considers the product configuration finalized. It allows the support of a debug authentication for in-the-field repair (read the dedicated application note).
Locked	Debug locked	None	This state considers the product configuration finalized. The debug authentication is not permitted. The product is definitively in this state.
Regression	Debug closed	NoDebug	This state is a temporal state (but nonvolatile) to manage the full regression to an open state. It removes all user flash-memory code and data including the hide protected areas (HDP).

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

Figure 5. Product life-cycle



3.5.2 Recommended product settings

To ease the product maintenance (in-the-field), it is recommended to take benefit of the feature called debug authentication control. This enables the maintenance of product activating the debug and makes possible to manage regressions while considering the security of the sensible information. This implies the following actions:

- the first important step is to provision the HASH of the authentication password.
- set the PRODUCT_STATE in closed state when the product has been configured.
- the HASH of the authentication password (assume already provisioned by user in OTP) should be used for debug authentication in order to launch a full regression of the product.

Caution: The user must ensure to provision the HASH of the authentication password before changing the product state from OPEN to any other one (Provisioning, iROT-Provisioned or Closed).
The user must take care that changing product state to Locked is irreversible so once changed debugging the device is definitively lost.

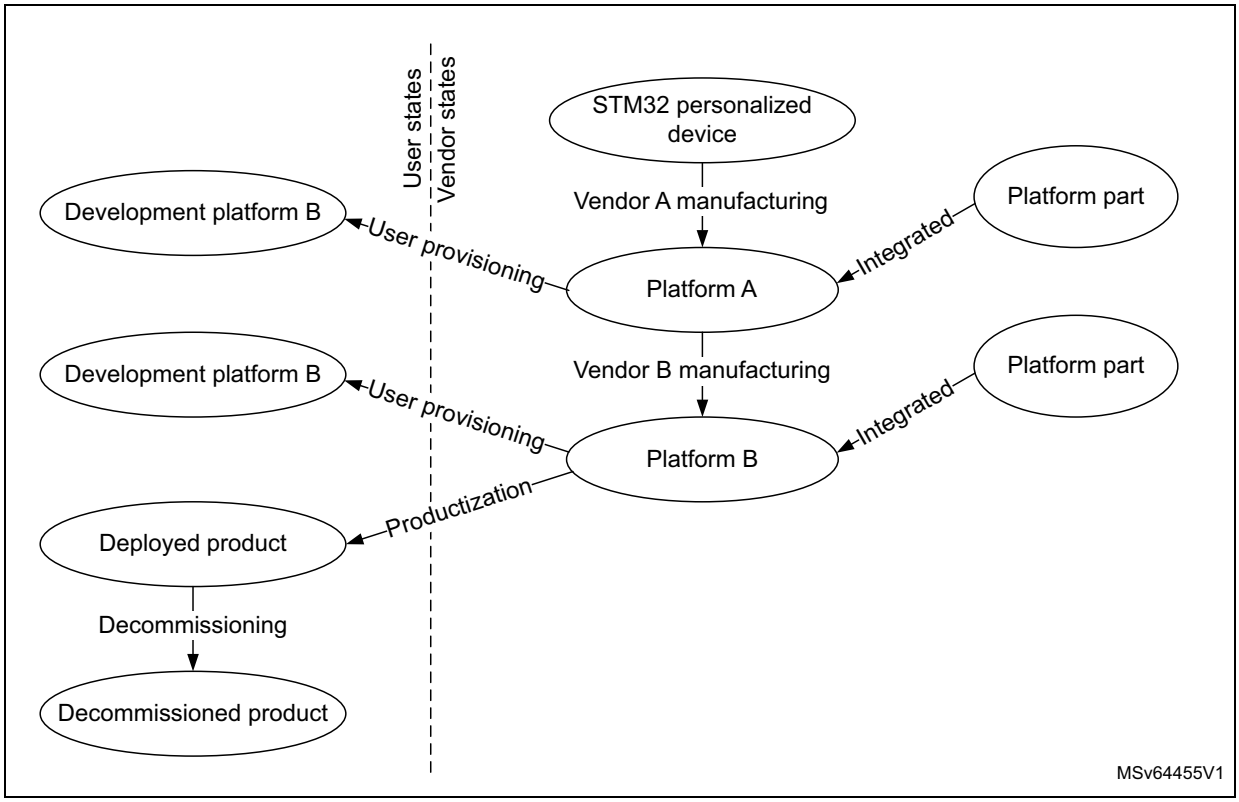
3.6 Software intellectual property protection and collaborative development

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 on the figure below. Starting from a personalized device sold by STMicroelectronics, a vendor A can integrate a portion of hardware and software on a platform A. Platform A can 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 6. Collaborative development principle



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

3.6.1 Software intellectual property protection

As described in *Section 3.5.1,* the hardware PRODUCT_STATE mechanism automatically controls the accesses to secrets provisioned in the device. The protection of these secrets is defined in the table below.

Table 7. Software intellectual property protection with PRODUCT_STATE

PRODUCT_STATE protection level		Secrets protection
Open	Device open	No special protections.
Provisioning	Debug partially opened	All areas protected with HDPL1 and HDPL2 cannot be dumped, debugged, or traced. The iROT can set up a higher level of protection.
iROT-Provisioned	Debug partially opened	All areas protected with HDPL1 and HDPL2 cannot be dumped, debugged, or traced. The iROT can set up a higher level of protection.
Closed	No debug	No debug possible except through debug authentication after a full regression.
Locked	No debug	All data and code stored in the device or encrypted in external flash memory cannot be dumped clear-text, debugged or traced.

3.6.2 Other software intellectual property protections

The device additional protections to software intellectual property are:

- Invasive attacks such as physical tampering or perturbation are countered by detection. Then, they are decommissioned 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.

4 Boot modes

At startup, a BOOT0 pin and NSBOOTADD[31:8] option bytes are used to select the boot memory address that includes:

- Boot from any address in user Flash memory
- Boot from system memory
  - Bootloader
  - Debug authentication library (ST-DA)

When boot from user Flash is selected, the boot address is defined by NSBOOTADD. This address can be locked thanks to NSBOOT_LOCK.

Embedded bootloader

The embedded bootloader is located in the system memory, programmed by ST during production. It is used to reprogram the Flash memory by using USART, I2C, I3C, SPI, FDCAN, or USB in device mode through the DFU (device firmware upgrade).

Refer to the application note *STM32 microcontroller system memory boot mode* (AN2606).

Embedded debug authentication (ST-DA)

The embedded ST-DA is located in the system memory, programmed by ST during production. ST-DA is the library that manages the debug authentication protocol by allowing to launch in a secure way regressions on secured products in the field.

The ST-DA can be launched from ST tools (STM32CubeProgrammer or IDEs). An authentication password should be used for debug authentication in order to launch a full regression of the product.

4.1 STM32H503 boot modes

The table below provides the detail of the boot mode for the STM32H503 devices.

Table 8. STM32H503 boot modes

PRODUCT_STATE	BOOT0 pin	Boot address option-byte selection	Boot area	ST programmed default value
Open	0	NSBOOTADD[31:8]	Boot address defined by user option byte NSBOOTADD[31:8]	Flash: 0x0800 0000
	1	NA	Bootloader	Bootloader
Provisioning	x	NA	Bootloader	Bootloader
Provisioned, closed, locked	x	NSBOOTADD[31:8]	Boot address defined by user option byte NSBOOTADD[31:8]	Flash: 0x0800 0000

5 RAMs configuration controller (RAMCFG)

5.1 Introduction

The RAMCFG configures the features of the internal SRAMs (SRAM1, SRAM2, and BKPSRAM).

5.2 RAMCFG main features

The internal SRAM supports some of the features listed hereafter, configured in RAMCFG:

- Error code correction (ECC):
  - Single error detection and correction with interrupt generation
  - Double error detection with interrupt or NMI generation
  - Status with failing address
- Write protection (1-Kbyte granularity)
- SRAM software erase

5.3 RAMCFG functional description

5.3.1 Internal SRAMs features

Three SRAMs are embedded in the devices, each with specific features:

- SRAM1 and SRAM2 are the main SRAMs.

  These SRAMs are made of blocks that can be powered down in Stop mode to reduce consumption:
  - SRAM1: one 16-Kbyte block
  - SRAM2: one 16-Kbyte block

  The backup SRAM (BKPSRAM) can be retained in all low-power modes and when V_{DD} is off in VBAT mode.

  Refer to *Section 9: Power control (PWR)* for more details.
- SRAM2 is erased when a system reset occurs if the SRAM2_RST option bit is selected in the Flash memory user option bytes. SRAM1 is erased when a system reset occurs if the SRAM1_RST option bit is selected in the Flash memory user option bytes. Refer to *Section 7: Embedded flash memory (FLASH)* for more details.
- SRAM2 and optionally backup SRAM are protected by the tamper detection circuit, and are erased by hardware in case of tamper detection. They are also erased by hardware in case of a Backup domain reset. Refer to *Section 33: Tamper and backup registers (TAMP)* for more details.
- The RAMCFG embeds the registers related to the internal SRAMs ECC, write protection, and software erase.

The table below summarizes the features supported by each internal SRAM

Table 9. Internal SRAMs features

SRAM feature	SRAM1 (16 Kbytes)	SRAM2 (16 Kbytes)	BKPSRAM (2 Kbytes)
Optional retention in Standby mode	-	-	X
Optional retention in VBAT mode	-	-	X
Optional retention in Stop mode	X	X	X
Erased with product state regression	X	X	X
Optionally erased with tamper detection	-	X	X
Backup domain reset	-	-	X
Optionally erased with system reset	X	X	-
Software erase	X	X	X
ECC	X	X	X
Write protection	-	X	X(1)

1. BKPSRAM write protection can be enabled/disabled by using disable backup domain write protection (DBP) bit in backup domain.

5.3.2 Error code correction (SRAM1, SRAM2, BKPSRAM)

The ECC is supported by SRAM1, SRAM2 and BKPSRAM when enabled with the SRAM1_ECC, SRAM2_ECC and BKPRAM_ECC user option bits. Refer to *Section 7: Embedded flash memory (FLASH)* for more details.

Seven ECC bits are added per 32 bits of SRAM, allowing two bits error detection and one bit error correction on memory read access.

As the ECC is calculated and checked for a 32-bit word, the byte and half-word write accesses are managed by the SRAM interface by first reading the whole word, then write the word again with the new byte/half-word value. ECC double errors are also detected during these byte or half-word AHB write accesses (read/modify/write done by interface). The byte or half-word write access latency is 2 AHB clock cycles.

Caution: In case of a byte or half-word write on SRAM with ECC, the read/modify/write operation is done in a buffer. The buffer content is written into the SRAM two AHB clock cycles after the SRAM AHB is released (when SRAM is no more accessed).

Single and double ECC errors

When a single error is detected, it is automatically corrected and the SEDC/CSEDC bits are set in the *RAMCFG memory interrupt status register (RAMCFG_MxISR)* and *RAMCFG memory x interrupt clear register x (RAMCFG_MxICR)* respectively. The associated ECC single error address is updated only if the single error interrupt is enabled (SEIE bit of RAMCFG_MxIER is set). An interrupt is generated if enabled by the SEIE bit in the *RAMCFG memory x interrupt enable register (RAMCFG_MxIER)*. The failing address is stored in the *RAMCFG memory x ECC single error address register (RAMCFG_MxSEAR)* if the ALE bit is set in the *RAMCFG memory x control register (RAMCFG_MxCR)*.

Caution: Single errors cannot be detected when the SEDC bit is set.

When a double error is detected, the DED and CDED bits are set in the *RAMCFG memory interrupt status register (RAMCFG_MxISR)* and *RAMCFG memory x interrupt clear register x (RAMCFG_MxICR)* respectively. The associated ECC double error address is updated only if double error interrupt or NMI is enabled (DEIE bit or ECCNMI bit of RAMCFG_MxIER is set). An interrupt or NMI is generated if enabled by the DEIE or ECCNMI bit in the *RAMCFG memory x interrupt enable register (RAMCFG_MxIER)*. The failing address is stored in the *RAMCFG memory x ECC double error address register (RAMCFG_MxDEAR)* if the ALE bit is set in the *RAMCFG memory x control register (RAMCFG_MxCR)*.

Caution: Double errors cannot be detected when the DED bit is set.

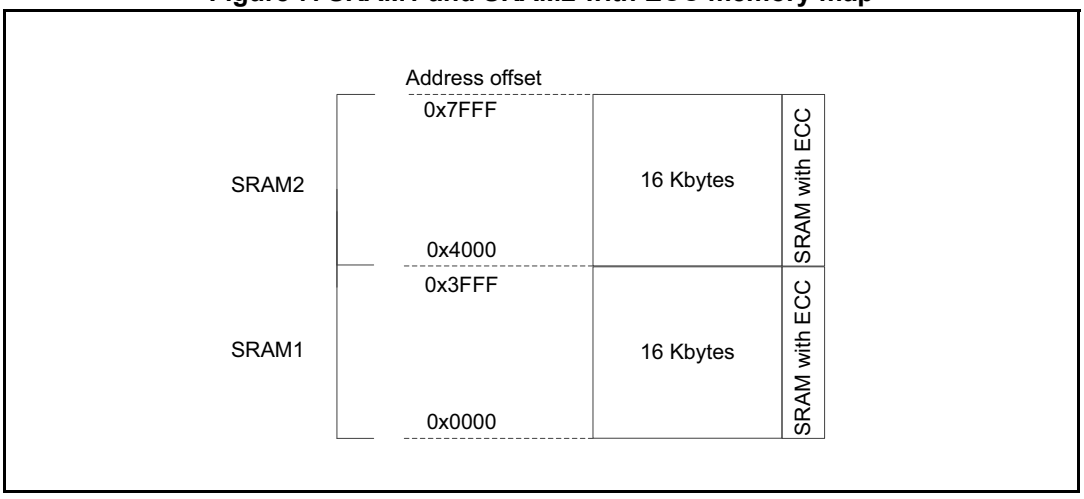
SRAM1/SRAM2 with ECC memory map

When the ECC is enabled for SRAM1, the full 16 Kbytes of SRAM1 have ECC.

When the ECC is enabled for SRAM2, the full 16 Kbytes of SRAM2 have ECC.

The figure below shows the SRAM areas, when ECC is enabled for both SRAM1 and SRAM2.

Figure 7. SRAM1 and SRAM2 with ECC memory map



When ECC is enabled by user option bits, the ECCE bit is automatically set after system reset in the related *RAMCFG memory x control register (RAMCFG_MxCR)*.

The ECC can be deactivated by executing the following software sequence:

1. Write 0xAE in the *RAMCFG memory x ECC key register (RAMCFG_MxECCKEYR)*.
2. Write 0x75 in the *RAMCFG memory x ECC key register (RAMCFG_MxECCKEYR)*.
3. Write 0 in the ECCE bit of the *RAMCFG memory x control register (RAMCFG_MxCR)*.

Testing ECC mechanism

In order to test the ECC mechanism, 1 or 2 bits by word for single or double error test respectively.

The procedure to check ECC is the following:

1. On an erased memory, write data with ECC ON.
2. Disable ECC.
3. Write same data with 1- or 2-bit modification (for single or double error test respectively).
4. Enable ECC.
5. Read data. Enabled interrupt is generated because of single or double error.

5.3.3 Write protection (SRAM2)

The SRAM2 is made of 16 1-Kbyte pages. Each 1-Kbyte page can be write-protected by setting its corresponding PxWP (x = 0 to 15) bit in the *RAMCFG memory 2 write protection register 1 (RAMCFG_M2WPR1)*.

5.3.4 Software erase

SRAM erase can be requested by executing this software sequence:

1. Write 0xCA in the *RAMCFG memory x erase key register (RAMCFG_MxERKEYR)*.
2. Write 0x53 in the *RAMCFG memory x erase key register (RAMCFG_MxERKEYR)*.
3. Write 1 in the SRAMER bit of the *RAMCFG memory x control register (RAMCFG_MxCR)*.

SRAMBUSY flag is set in the related SRAM interrupt status register as long as the erase is on going.

The total duration of each SRAM erase is N AHB clock cycles, where N is the size of the SRAM in 32-bit words.

If the SRAM is read or written while an erase is on going, wait states are inserted on the AHB bus until the end of the erase operation.

5.4 RAMCFG low-power modes

Table 10. Effect of low-power modes on RAMCFG

Mode	Description
Sleep	No effect. RAMCFG interrupts cause the device to exit the Sleep mode.
Stop	The content of RAMCFG registers is kept.
Standby	The RAMCFG peripheral is powered down and must be reinitialized after exiting Standby.

5.5 RAMCFG interrupts

The table below gives the list of RAMCFG interrupt requests.

Table 11. RAMCFG 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 modes
RAMCFG	ECC single error detection and correction	SEDC	SEIE	Write 1 in CSEDC	Yes	No	No
	ECC double error detection	DED	DEIE = 1 and ECCNMI = 0	Write 1 in CDED	Yes	No	No
NMI	ECC double error detection	DED	ECCNMI	Write 1 in CDED	Yes	No	No

5.6 RAMCFG registers

In the registers described below, x refers to:

- SRAM1/2 when x = 1/2 respectively
- BKPSRAM when x = 5

5.6.1 RAMCFG memory x control register (RAMCFG_MxCR)

Address offset: 0x000 + 0x040 * (x - 1), (x = 1, 2, 5)

Reset value: 0x0000 000X

ECCE reset value depends on ECC enable user option 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.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
							rs				rw				rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMER**: SRAM erase

This bit can be set by software only after writing the unlock sequence in the ERASEKEY field of the RAMCFG_MxERKEYR register. Setting this bit starts the SRAM erase. This bit is automatically cleared by hardware at the end of the erase operation.

0: No erase operation on going

1: Erase operation on going

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **ALE**: Address latch enable
0: Failing address not stored in the SRAMx ECC single/double error address registers
1: Failing address stored in the SRAMx ECC single/double error address registers
Note: This bit is reserved and must be kept at reset value in SRAM1 control register.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **ECCE**: ECC enable.
This bit reset value is defined by the user option bit configuration. When set, it can be cleared by software only after writing the unlock sequence in the RAMCFG_MxECCKEYR register.
0: ECC disabled
1: ECC enabled
Note: This bit is reserved and must be kept at reset value in SRAM1 control register.

5.6.2 RAMCFG memory x interrupt enable register (RAMCFG_MxIER)

Address offset: 0x004 + 0x040 * (x - 1), (x = 2, 3, 5)

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.	ECCNMI	Res.	DEIE	SEIE
												rs		rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ECCNMI**: Double error NMI
This bit is set by software and cleared only by a global RAMCFG reset.
0: NMI not generated in case of ECC double error
1: NMI generated in case of ECC double error
Note: if ECCNMI is set, the RAMCFG maskable interrupt is not generated whatever DEIE bit value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DEIE**: ECC double error interrupt enable
0: Double error interrupt disabled
1: Double error interrupt enabled

Bit 0 **SEIE**: ECC single error interrupt enable
0: Single error interrupt disabled
1: Single error interrupt enabled

5.6.3 RAMCFG memory interrupt status register (RAMCFG_MxISR)

Address offset: 0x008 + 0x040 * (x - 1), (x = 1, 2, 3, 5)

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.	SRAM BUSY	Res.	Res.	Res.	Res.	Res.	Res.	DED	SEDC
							r							r	r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMBUSY**: SRAM busy with erase operation

0: No erase operation on going

1: Erase operation on going

Note: Depending on the SRAM, the erase operation can be performed due to software request, system reset if the option bit is enabled, tamper detection or product state regression. Refer to Table 9: Internal SRAMs features.

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

Bit 1 **DED**: ECC double error detected

0: No double error

1: Double error detected

Note: This bit is reserved and must be kept at reset value in SRAM1 interrupt status register.

Bit 0 **SEDC**: ECC single error detected and corrected

0: No single error

1: Single error detected and corrected

Note: This bit is reserved and must be kept at reset value in SRAM1 interrupt status register.

5.6.4 RAMCFG memory x ECC single error address register (RAMCFG_MxSEAR)

Address offset: 0x00C + 0x040 * (x - 1), (x = 2, 3, 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ESEA[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
ESEA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ESEA[31:0]**: ECC single error address

When the ALE bit is set in the RAMCFG_MxCR register, this field is updated with the address corresponding to the ECC single error.

5.6.5 RAMCFG memory x ECC double error address register (RAMCFG_MxDEAR)

Address offset: 0x010 + 0x040 * (x - 1), (x = 2, 3, 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EDEA[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
EDEA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EDEA[31:0]**: ECC double error address

When the ALE bit is set in the RAMCFG_MxCR register, this field is updated with the address corresponding to the ECC double error.

5.6.6 RAMCFG memory x interrupt clear register x (RAMCFG_MxICR)

Address offset: 0x014 + 0x040 * (x - 1), (x = 2, 3, 5)

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.	CDED	CSEDC
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **CDED**: Clear ECC double error detected

Writing 1 to this flag clears the DED bit in the RAMCFG_MxISR register. Reading this flag returns the DED value.

Bit 0 **CSEDC**: Clear ECC single error detected and corrected

Writing 1 to this flag clears the SEDC bit in the RAMCFG_MxISR register. Reading this flag returns the SEDC value.

5.6.7 RAMCFG memory 2 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PyWP**: SRAM2 1-Kbyte page y write protection (y = 15 to 0)
These bits are set by software and cleared only by a global RAMCFG reset.
0: Write protection of SRAM2 1-Kbyte page y is disabled.
1: Write protection of SRAM2 1-Kbyte page y is enabled.

5.6.8 RAMCFG memory x ECC key register (RAMCFG_MxECCKEYR)

Address offset: 0x024 + 0x040 * (x - 1), (x = 2, 3, 5)

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.	ECCKEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **ECCKEY[7:0]**: ECC write protection key
The following steps are required to unlock the write protection of the ECCE bit in the RAMCFG_MxCR register.
1) Write 0xAE into ECCKEY[7:0].
2) Write 0x75 into ECCKEY[7:0].
Note: Writing a wrong key reactivates the write protection.

5.6.9 RAMCFG memory x erase key register (RAMCFG_MxERKEYR)

Address offset: 0x028 + 0x040 * (x - 1), (x = 1 to 5)

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_MxCR register.

1) Write 0xCA into ERASEKEY[7:0].

2) Write 0x53 into ERASEKEY[7:0].

Note: Writing a wrong key reactivates the write protection.

5.6.10 RAMCFG register map

Table 12. RAMCFG register map and reset values

Offset	Register name reset value	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	**RAMCFG_M1CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
	Reset value																								0				0				x
0x40	**RAMCFG_M1IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCNMI	Res.	DEIE	SEIE
	Reset value																													0		0	0
0x08	**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.	DED	SEDC
	Reset value																								0							0	0
0x0C	**RAMCFG_ M1SEAR**	ESEA[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	**RAMCFG_ M1DEAR**	EDEA[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	**RAMCFG_M1ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDED	CSEDC
	Reset value																															0	0
0x18 to 0x20	Reserved	Reserved																															
0x024	**RAMCFG_ M1ECCKEYR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 12. RAMCFG register map and reset values (continued)

Offset	Register name reset value	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
0x28	**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
0x2C to 0x3C	Reserved	Reserved																															
0x40	**RAMCFG_M2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
	Reset value																								0				0				x
0x44	**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.	ECCNMI	Res.	DEIE	SEIE
	Reset value																													0		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.	DED	SEDC
	Reset value																								0							0	0
0x04C	**RAMCFG_ M2SEAR**	ESEA[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	**RAMCFG_ M2DEAR**	EDEA[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	**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.	CDED	CSEDC
	Reset value																															0	0
0x058	**RAMCFG_ M2WPR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x05C to 0x060	Reserved	Reserved																															
0x064	**RAMCFG_ M2ECCKEYR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
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 0x0FC	Reserved	Reserved																															
0x100	**RAMCFG_M5CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
	Reset value																								0				0				x
0x104	**RAMCFG_M5IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCNMI	Res.	DEIE	SEIE
	Reset value																													0		0	0
0x108	**RAMCFG_M5ISR**	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.	DED	SEDC
	Reset value																								0							0	0
0x10C	**RAMCFG_ M5SEAR**	ESEA[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 12. RAMCFG register map and reset values (continued)

Offset	Register name reset value	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
0x110	**RAMCFG_ M5DEAR**	EDEA[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	**RAMCFG_M5ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDED	CSEDC
	Reset value																															0	0
0x118 to 0x120	Reserved	Reserved																															
0x124	**RAMCFG_ M5ECCKEYR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x128	**RAMCFG_ M5ERKEYR**	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

Refer to *Section 2.2* for the register boundary addresses.

6 Global privilege controller (GTZC)

6.1 Introduction

This section describes the global privilege controller (GTZC) block that contains the following subblocks:

- **TZSC**: privilege controller

  This subblock defines the privileged state of slave peripherals. It also controls the subregion area size and properties for the watermark memory peripheral controller (MPCWM).

- **MPCBB**: memory protection controller - block based

  This subblock configures the internal SRAMs (SRAM1 and SRAM2) in a system having segmented SRAM (pages of 512 bytes) with privileged attributes.

These subblocks are used to configure privilege in a product having bus agents with privileged attributes such as:

- on-chip RAM with programmable privileged blocks (pages)
- AHB and APB peripherals with programmable privileged access
- AHB master granted as privilege

6.2 GTZC main features

The GTZC main features are listed below:

- 2 independent 32-bit AHB interfaces for TZSC and MPCBB
- Privileged and unprivileged access to TZSC and MPCBB
- Set of registers to define product privileged settings:
  - Privileged blocks for internal SRAMs (with MPCBB)
  - Privileged regions for internal backup SRAM (with MPCWM)
  - Privileged access mode for privileged and privilege-aware peripherals
  - Privileged access mode for privilege-aware masters

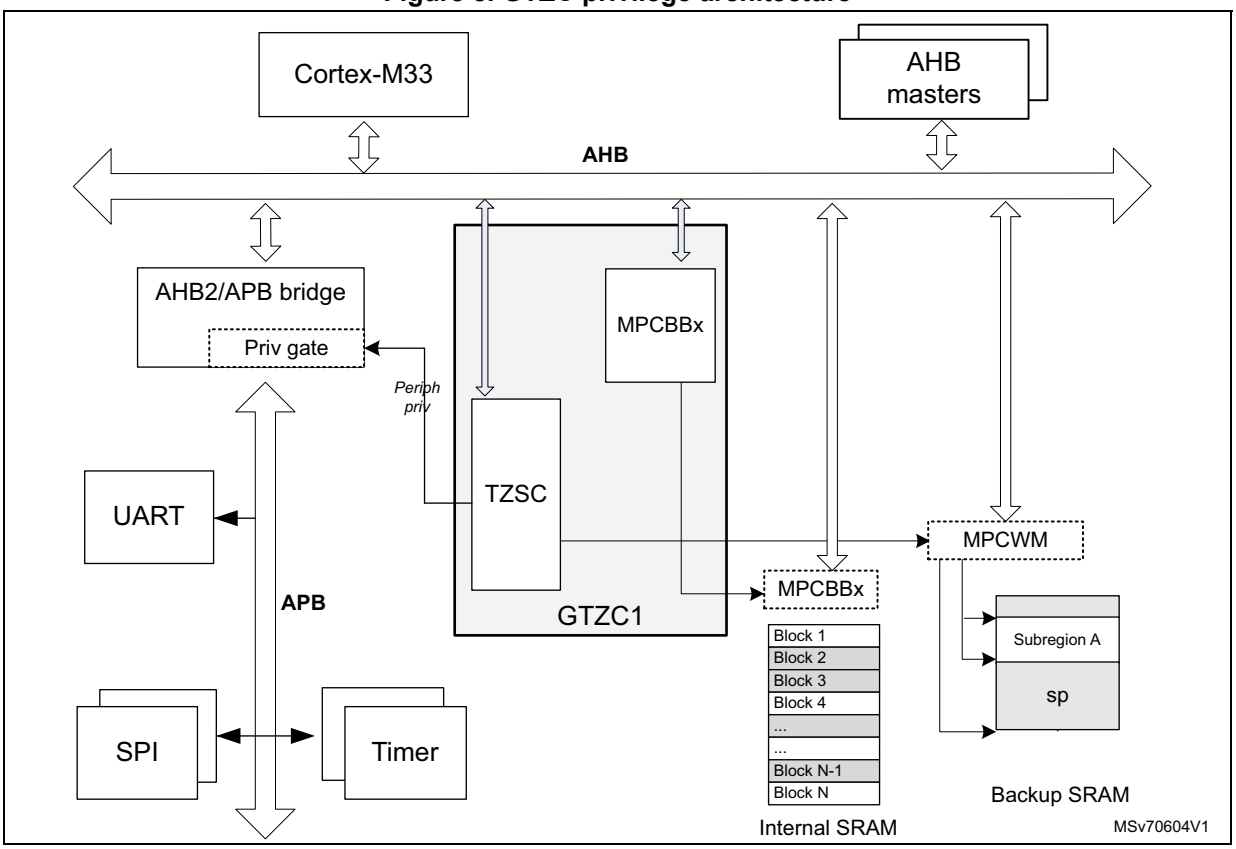
GTZC privilege system architecture

AHB and APB Peripherals can be categorized as:

- **privileged**: peripherals protected by AHB/APB firewall stub that is controlled from TZSC to define privileged properties
- **unprivileged:** peripherals connected directly to AHB/APB interconnect without any privileged gate
- **privilege-aware**: peripherals connected directly to AHB or APB bus and implementing a specific privileged behavior (such as a subset of registers being privileged). Privilege-aware AHB masters always drive HPROT[1] signal according to their privilege mode (such as CPU or DMA)

The privilege protection architecture with privilege and privilege-aware peripherals is shown in the figure below.

Figure 8. GTZC privilege architecture



6.3 GTZC implementation

The STM32H503xx devices embed one instance of GTZC.

Table 13. GTZC implementation

GTZC subblocks	GTZC1
TZSC	X
TZIC	-
Number of MPCBB subblock	2

Table 14. GTZC features

Feature	GTZC
Arm® TrustZone®	-
Privilege	X

The table below shows the address offset of GTZC subblocks versus GTZC base address (refer to *Section 2.2* for GTZC1 base address).

Table 15. GTZC1 subblock address offset

GTZC1 subblock	Address offset
GTZC1_TZSC	0x0
GTZC1_MPCBB1	0x800
GTZC1_MPCBB2	0xC00

The table below describes the characteristics of the available MPCWM.

Table 16. MPCWM resource assignment

GTZC	MPC	Target memory interface	Number of priv/unpriv regions	Watermark granularity (bytes)
GTZC1	MPCWM	BKPSRAM	1	32

The table below describes the characteristics of the available MPCBB.

Table 17. MPCBB resource assignment

GTZC	MPC	Resource	Memory size (Kbytes)	Block size (bytes)	Number of blocks	Number of super-blocks
GTZC1	MPCBB1	SRAM1	16	512	32	1
	MPCBB2	SRAM2				

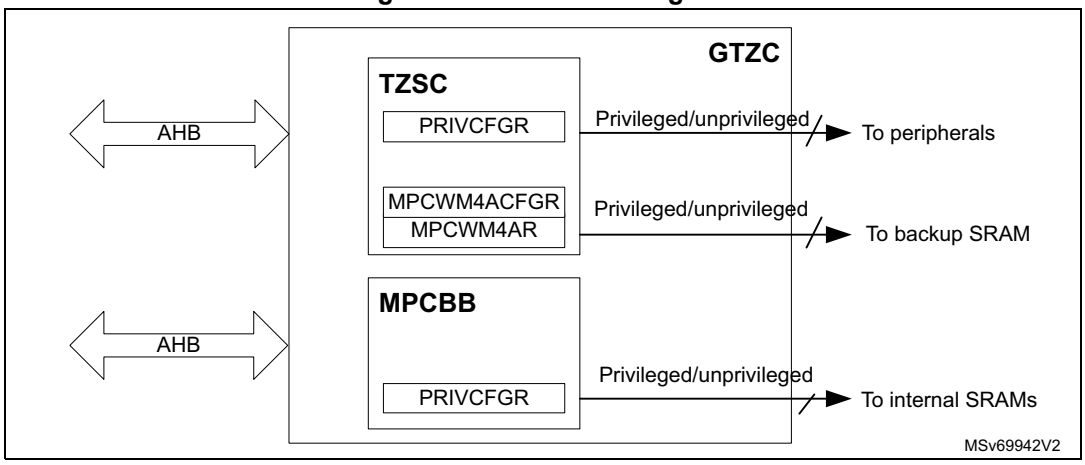
6.4 GTZC functional description

6.4.1 GTZC block diagram

The figure below describes the combined feature of TZSC and MPCBB. Each sub-block is controlled by its own AHB configuration port.

The TZSC defines which peripheral is privileged.The privileged configuration bit of a peripheral can be modified by a privileged transaction.

Figure 9. GTZC block diagram



6.4.2 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.

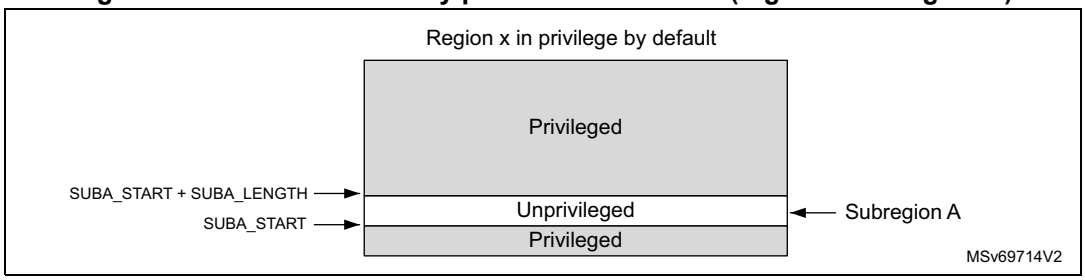
6.4.3 Privilege controller (TZSC)

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

- Control of privileged state for all peripherals, done through GTZC1_TZSC_PRIVCFGRx registers to control AHB/APB firewall stubs for the privileged peripherals
- For watermark memory protection controller (backup SRAM), one region can be defined and the following fields are used to program:
  - the start of the protected subregion on backup SRAM: SUBA_START[10:0]
  - the length of the protected subregion on backup SRAM: SUBA_LENGTH[11:0]

The control register for the subregion can be used to enable/disable the watermark memory protection controller as well as defining the right attribute.

Figure 10. Watermark memory protection controller (region x/subregion A)

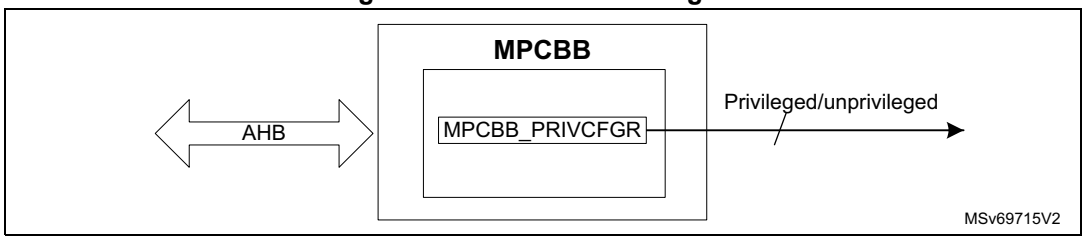


In the figure above, region x represents the backup SRAM region. The privileged attribute of subregion A is configurable. When no subregion is defined or enabled on the region x, then the default attribute of the region x is set as privileged.

6.4.4 Memory protection controller - block based (MPCBB)

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

Figure 11. MPCBB block diagram



In order to setup the MPCBB, the privileged firmware must define which memory blocks are privileged by setting the correct bits in GTZC1_MPCBBz_PRIVCFGRx.
A MPCBB super-block is made of 32 consecutive blocks.

*Note: The block size is 512 bytes.The super-block size is 512 * 32 = 16 Kbytes.*

6.4.5 Power-on/reset state

The power-on and reset state of the TZSC clear to 0 all bits of GTZC1_TZSC_PRIVCFGRx, meaning that all peripherals are set to unprivileged.

For internal SRAMx (x = 1 to 2), all GTZC1_MPCBBz_PRIVCFGRx are set to 0x0000 0000, making these internal memory block unprivileged by default.

For backup SRAM, GTZC1_TZSC_MPCWM4AR is set to 0x0800 0000, making the memory unprivileged by default.

6.5 GTZC1 TZSC registers

All registers are accessed only by words (32-bit).

6.5.1 GTZC1 TZSC privilege configuration register 1 (GTZC1_TZSC_PRIVCFGR1)

Address offset: 0x020

Reset value: 0x0000 0000

Write-privileged access only.

This register can be read or written only by privileged transaction.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM2PRIV	DTSPRIV	Res.	Res.	Res.	Res.	DAC1PRIV	Res.	Res.	Res.	Res.	CRSPRIV	I3C1PRIV	I2C2PRIV	I2C1PRIV	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.	USART3PRIV	USART2PRIV	SPI3PRIV	SPI2PRIV	IWDGPRIV	WWDGPRIV	Res.	Res.	Res.	TIM7PRIV	TIM6PRIV	Res.	Res.	TIM3PRIV	TIM2PRIV
	rw	rw	rw	rw	rw	rw				rw	rw			rw	rw

Bit 31 **LPTIM2PRIV**: privileged access mode for LPTIM2
0: unprivileged
1: privileged

Bit 30 **DTSPRIV**: privileged access mode for DTS
0: unprivileged
1: privileged

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **DAC1PRIV**: privileged access mode for DAC1
0: unprivileged
1: privileged

Bits 24:21 Reserved, must be kept at reset value.

Bit 20 **CRSPRIV**: privileged access mode for CRS
0: unprivileged
1: privileged

Bit 19 **I3C1PRIV**: privileged access mode for I3C1
0: unprivileged
1: privileged

Bit 18 **I2C2PRIV**: privileged access mode for I2C2
0: unprivileged
1: privileged

Bit 17 **I2C1PRIV**: privileged access mode for I2C1
0: unprivileged
1: privileged

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART3PRIV**: privileged access mode for USART3
0: unprivileged
1: privileged

Bit 13 **USART2PRIV**: privileged access mode for USART2
0: unprivileged
1: privileged

Bit 12 **SPI3PRIV**: privileged access mode for SPI3
0: unprivileged
1: privileged

Bit 11 **SPI2PRIV**: privileged access mode for SPI2
0: unprivileged
1: privileged

Bit 10 **IWDGPRIV**: privileged access mode for IWDG
0: unprivileged
1: privileged

Bit 9 **WWDGPRIV**: privileged access mode for WWDG
0: unprivileged
1: privileged

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **TIM7PRIV**: privileged access mode for TIM7
0: unprivileged
1: privileged

Bit 4 **TIM6PRIV**: privileged access mode for TIM6
0: unprivileged
1: privileged

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM3PRIV**: privileged access mode for TIM3
0: unprivileged
1: privileged

Bit 0 **TIM2PRIV**: privileged access mode for TIM2
0: unprivileged
1: privileged

6.5.2 GTZC1 TZSC privilege configuration register 2 (GTZC1_TZSC_PRIVCFGR2)

Address offset: 0x024

Reset value: 0x0000 0000

Write-privileged access only.

This register can be read or written only by privileged transaction.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	LPTIM1PRIV	Res.	Res.	LPUART1PRIV	Res.	Res.	Res.	Res.	Res.	USBPRIV	Res.	Res.	Res.
			rw			rw						rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	USART1PRIV	Res.	SPI1PRIV	TIM1PRIV	Res.	Res.	Res.	COMPPRIV	OPAMPPRIV	Res.	Res.	FDCAN1PRIV
				rw		rw	rw				rw	rw			rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **LPTIM1PRIV**: privileged access mode for LPTIM1
0: unprivileged
1: privileged

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **LPUART1PRIV**: privileged access mode for LPUART
0: unprivileged
1: privileged

Bits 24:20 Reserved, must be kept at reset value.

Bit 19 **USBPRIV**: privileged access mode for USBSF
0: unprivileged
1: privileged

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 **USART1PRIV**: privileged access mode for USART1
0: unprivileged
1: privileged

Bit 10 Reserved, must be kept at reset value.

Bit 9 **SPI1PRIV**: privileged access mode for SPI1
0: unprivileged
1: privileged

Bit 8 **TIM1PRIV**: privileged access mode for TIM1
0: unprivileged
1: privileged

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **COMPPRIV**: privileged access mode for COMP
0: unprivileged
1: privileged

Bit 3 **OPAMPPRIV**: privileged access mode for OPAMP
0: unprivileged
1: privileged

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **FDCAN1PRIV**: privileged access mode for FDCAN1
0: unprivileged
1: privileged

6.5.3 GTZC1 TZSC privilege configuration register 3 (GTZC1_TZSC_PRIVCFGR3)

Address offset: 0x028

Reset value: 0x0000 0000

Write-privileged access only.

This register can be read or written only by privileged transaction.

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

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ADC1PRIV	Res.	ICACHEPRIV	Res.	Res.	Res.	CRCPRIV	Res.	Res.	Res.	Res.	Res.	I3C2PRIV	Res.	Res.
	rw		rw				rw						rw		

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **RAMCFGPRIV**: privileged access mode for RAMSCFG
0: unprivileged
1: privileged

Bits 25:19 Reserved, must be kept at reset value.

Bit 18 **RNGPRIV**: privileged access mode for RNG
0: unprivileged
1: privileged

Bit 17 **HASHPRIV**: privileged access mode for HASH
0: unprivileged
1: privileged

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **ADC1PRIV**: privileged access mode for ADC1
0: unprivileged
1: privileged

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ICACHEPRIV**: privileged access mode for ICACHE
0: unprivileged
1: privileged

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **CRCPRIV**: privileged access mode for CRC
0: unprivileged
1: privileged

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **I3C2PRIV**: privileged access mode for I3C2
0: unprivileged
1: privileged

Bits 1:0 Reserved, must be kept at reset value.

6.5.4 GTZC1 TZSC BKPSRAM subregion A watermark configuration register (GTZC1_TZSC_MPCWM4ACFGR)

Address offset: 0x070
Reset value: 0x0000 0000

Privilege 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.	PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
						rw								rs	rw

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

Bit 9 **PRIV**: Privileged subregion A
This bit is taken into account only if SREN is set.
0: Privileged and unprivileged accesses are granted in subregion A.
1: Only privileged accesses are granted in subregion A.

Bits 8:2 Reserved, must be kept at reset value.

Bit 1 **SRLOCK**: Subregion A lock

This bit, once set, can be cleared only by a system reset.
0: GTZC1_TZSC_MPCWM4ACFGR and GTZC1_TZSC_MPCWM4AR can be written.
1: Writes to GTZC1_TZSC_MPCWM4ACFGR and GTZC1_TZSC_MPCWM4AR are ignored.

Bit 0 **SREN**: Subregion A enable

0: Subregion A is disabled. Access control of base region applies to any access between this subregion start- and end-addresses.
1: Subregion A is enabled. Access control defined in GTZC1_TZSC_MPCWM4ACFGR applies to any access between this subregion start- and end-address, both defined in GTZC1_TZSC_MPCWM4AR.

6.5.5 GTZC1 TZSC BKPSRAM subregion A watermark register (GTZC1_TZSC_MPCWM4AR)

Address offset: 0x074
Reset value: 0x0800 0000

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.

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 subregion A

This field defines the length of the subregion A, to be multiplied by the granularity defined in *Table 16*.
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 subregion A is disabled
(SREN bit in GTZC1_TZSC_MPCMW4ACFGR is cleared).

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

Bits 10:0 **SUBA_START[10:0]**: Start of subregion A

This field defines the address offset of the subregion A, to be multiplied by the granularity defined in *Table 16*.

6.5.6 GTZC1 TZSC register map

Table 18. 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
0x000-0x01C	Reserved	Reserved																															
0x020	GTZC1_TZSC_PRIVCFGR1	LPTIM2PRIV	DTSPRIV	Res.	Res.	Res.	Res.	DAC1PRIV	Res.	Res.	Res.	Res.	CRSPRIV	I3C1PRIV	I2C2PRIV	I2C1PRIV	Res.	Res.	USART3PRIV	USART2PRIV	SPI3PRIV	SPI2PRIV	IWDGPRIV	WWDGPRIV	Res.	Res.	Res.	TIM7PRIV	TIM6PRIV	Res.	Res.	TIM3PRIV	TIM2PRIV
	Reset value	0	0					0					0	0	0	0			0	0	0	0	0	0				0	0			0	0
0x024	GTZC1_TZSC_PRIVCFGR2	Res.	Res.	Res.	LPTIM1PRIV	Res.	Res.	LPUART1PRIV	Res.	Res.	Res.	Res.	Res.	USBPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART1PRIV	Res.	SPI1PRIV	TIM1PRIV	Res.	Res.	Res.	COMPPRIV	OPAMPPRIV	Res.	Res.	FDCAN1PRIV
	Reset value				0			0						0								0		0	0				0	0			0
0x028	GTZC1_TZSC_PRIVFGR3	Res.	Res.	Res.	Res.	Res.	RAMCFGPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGPRIV	HASHPRIV	Res.	Res.	ADC1PRIV	Res.	ICACHEPRIV	Res.	Res.	Res.	CRCPRIV	Res.	Res.	Res.	Res.	Res.	I3C2PRIV	Res.	Res.
	Reset value						0								0	0			0		0				0						0		
0x02C-0x06C	Reserved	Reserved																															
0x070	GTZC1_TZSC_MPCWM4ACFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
	Reset value																							0								0	0
0x74	GTZC1_TZSC_MPCWM4AR	Res.	Res.	Res.	Res.	SUBA_LENGTH[11:0]												Res.	Res.	Res.	Res.	Res.	SUBA_START[10:0]										
	Reset value					1	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 15: GTZC1 subblock address offset*.

6.6 GTZC1 MPCBBz registers (z = 1 to 2)

All registers are accessed only by words (32-bit).

6.6.1 GTZC1 SRAMz MPCBB privileged configuration for super-block x register (GTZC1_MPCBBz_PRIVCFGRx) (z = 1 to 2)

Address offset: Block 1: 0x200 + 0x04 * x, (x = 0 to 31)

Address offset: Block 2: 0x600 + 0x04 * x, (x = 0 to 31)

Reset value: 0x0000 0000

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 **PRIVy**: Privileged configuration for block y, belonging to super-block x (y = 31 to 0).
0: Privileged and unprivileged access to block y, belonging to super-block x
1: Only privileged access to block y, belonging to super-block x

6.6.2 GTZC1 MPCBBz register map (z = 1 to 2)

Table 19. GTZC1 MPCBBz register map and reset values (z = 1 to 2)

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
0x200 + 0x04 *x (x = 0 to 31)	**GTZC1_MPCBB1_ PRIVCFGRx**	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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x600 + 0x04 *x (x = 0 to 31)	**GTZC1_MPCBB2_ PRIVCFGRx**	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	0	0	0	0	0	0	0	0	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 *Table 15: GTZC1 subblock address offset*.

7 Embedded flash memory (FLASH)

7.1 Introduction

The embedded flash memory (FLASH) manages the accesses of any master to the 128 Kbytes of embedded non-volatile memory. It implements the read, program and erase operations, error corrections, as well as various integrity and confidentiality protection mechanisms.

The embedded flash memory manages the automatic loading of non-volatile user option bytes at power-on reset, and implements the dynamic update of these options.

The embedded flash memory also features a one-time-programmable (OTP) area and a read-only area provisioned by STMicroelectronics during the product manufacturing.

7.2 FLASH main features

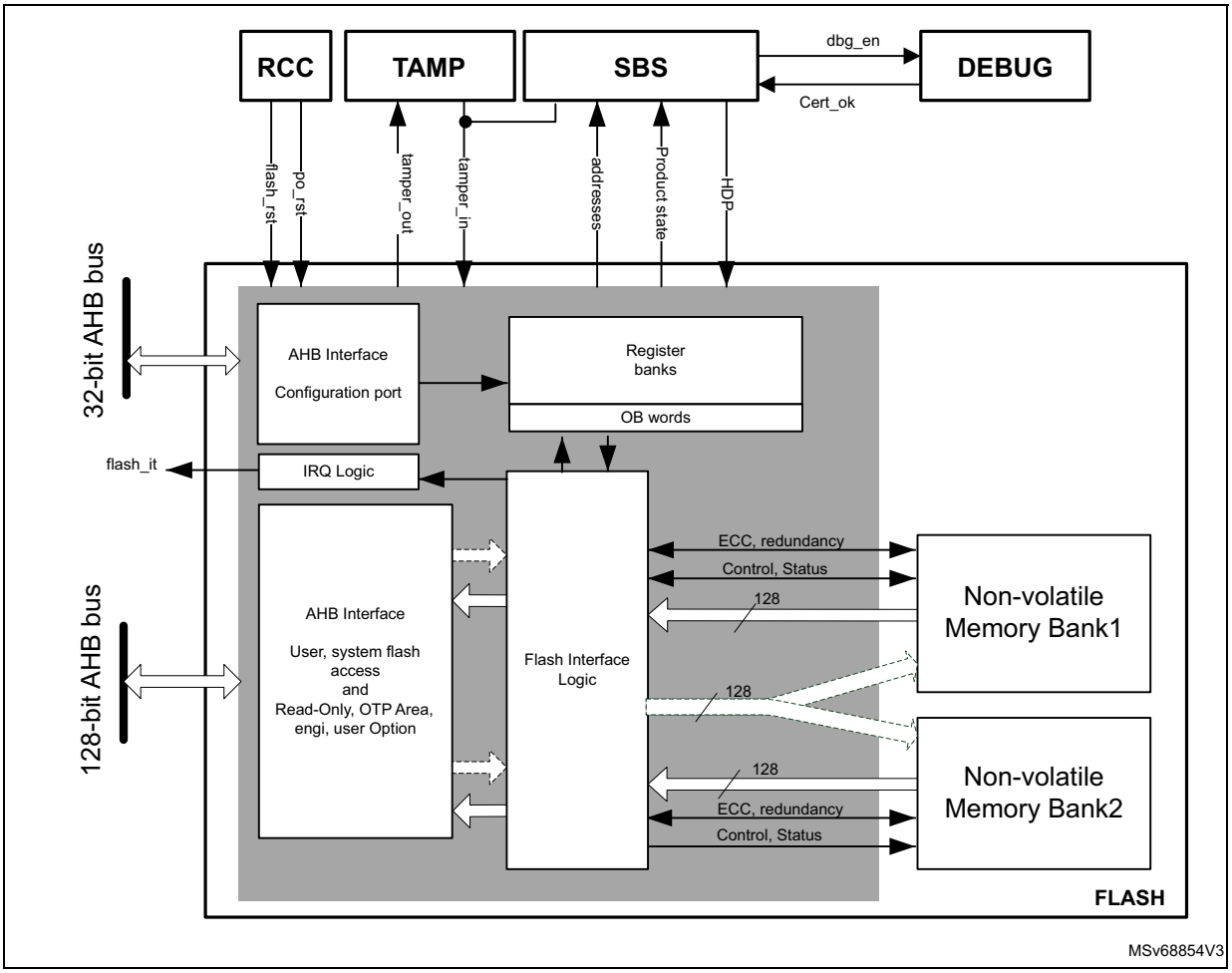
- 128 Kbytes of non-volatile memory divided into two banks of 64 Kbytes
- Flash memory read operations supporting multiple lengths: 128 bits, 64 bits, 32 bits, 16 bits or one byte
- Flash memory programming by 128 bits (user area) and 16 bits (OTP)
- 8 Kbytes sector erase, bank erase and dual-bank mass erase
- Dual-bank organization supporting:
  - simultaneous operations: read-while-write (program and erase) is supported.
  - the two banks share the same interface, write and erase can not be performed in parallel (write-while-write is not supported).
  - bank swapping: the address mapping of the user flash memory of each bank can be swapped, along with the corresponding registers. Security flags remain valid for the physical bank, so the data are not reveled by swapping to a bank with lower security configuration.
- Error code correction (ECC): one error detection/correction or two error detections per 128-bit Flash word using 9 ECC bits
- User configurable non-volatile option bytes
- Flash memory enhanced protections, activated by option bytes
  - different product states for protecting memory content from debug access
  - sector write-protection (WRPSG), configurable by sector
  - HDP protection providing temporal isolation for startup code
- 2 Kbytes one-time programmable (OTP) area
- Read-only area provisioned by STMicroelectronics
- Prefetch is reading the next sequential instruction from flash memory

7.3 FLASH functional description

7.3.1 FLASH block diagram

The following figure presents the embedded flash memory block diagram.

Figure 12. Flash block diagram (simplified)



7.3.2 FLASH signals

The flash memory has two AHB connections: the flash AHB register interface and the main AHB interface.

Flash AHB register interface

- Data size is 32 bits
- Except for some registers (FLASH_NSKEYR and FLASH_OPTKEYR registers that are used to insert unlock sequences for control and option registers and can be wrote by 32 bits), it is possible to read and write all registers by 8, 16 and 32 bits
- When unlock sequence for control and option registers is wrong, a bus error is raised, otherwise no read or write errors are generated on the bus

Main AHB interface

The AHB data bus size is 128 bits.

This interface is used to handle three different targets:

- Code placed in user and system memory. It is protected by 9 bits of ECC
- OTP, read only protected by 6 bits of ECC.

The main AHB interface is implemented as follows:

- User and system memory:
  - Supports multiple length: 128-, 64-, 32-, 16- and 8-bit data width
  - There is a read buffer of 128 bits for each bank where the last data read is stored. If the data is available in the read buffer, no read access is done to the flash memory. Buffer is flushed when write access, OTP access, user option change request or erase operation occurs
  - There is a prefetch of the same size as the read buffer
  - A 9-bit ECC is associated to each 128-bit data flash memory word
- OTP, read only
  - Two dedicated data buffer of 137 bits are used to manage 16-bit data with 6-bit ECC
  - Reading two times the same address triggers two Flash read access
  - During a read access, two wait states are added in addition of the memory wait states. These wait states are necessary to parse the data buffer
  - Each write access triggers a write in the flash memory
  - 6-bit ECC is associated to each 16-bit data.

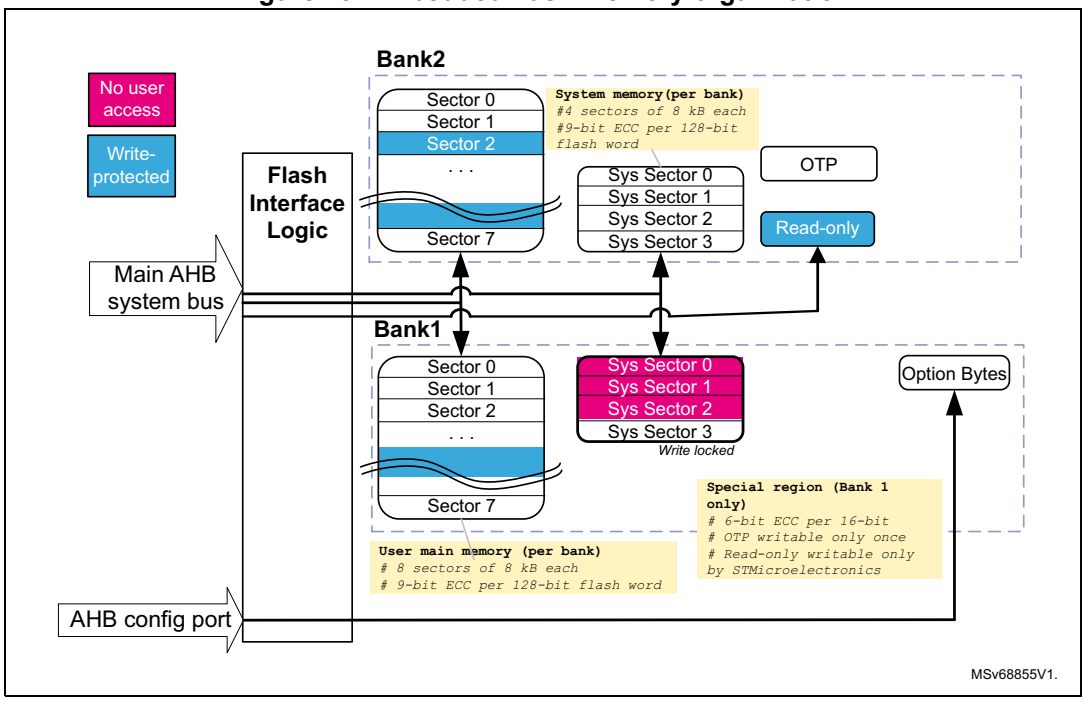
By default, all the AHB memory range is cacheable. For regions where caching is not practical (OTP, RO), MPU has to be used to disable local cacheability.

7.3.3 Flash memory architecture and usage

Flash memory architecture

The following figure shows the non-volatile memory organization supported by the embedded flash memory.

Figure 13. Embedded flash memory organization



The embedded Flash non-volatile memory is composed of:

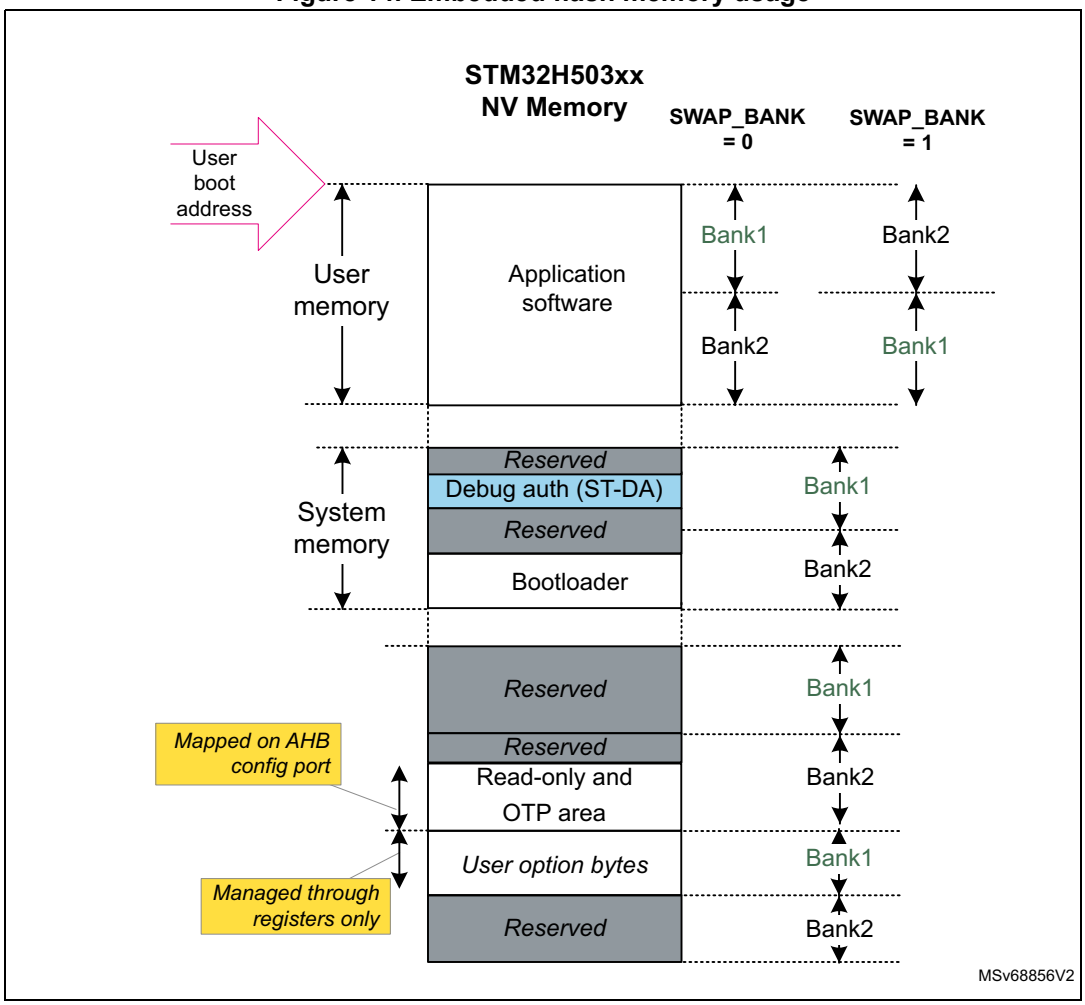
- A 128-Kbyte main memory block, organized in two banks, 64 Kbyte each. Each bank is divided in 8 sectors of 8 Kbytes each, and features Flash-word rows of 128 bits + 9 bits of ECC per word
- A system memory block of 64 Kbytes, divided into two 32-Kbyte banks. Each bank is divided in four 8 Kbytes sectors. The system flash memory is ECC protected (9-bit ECC per 128-bit word)
- A set of non-volatile option bytes loaded at reset by the embedded flash memory and accessible by the application software only through the AHB configuration register interface.
- A 2-Kbyte one-time-programmable (OTP) area that can be written only once by the application software
- A 2-Kbyte read-only area. It contains a unique device ID and product information

The overall flash memory architecture and its corresponding access interface is summarized in *Table 24*.

Partition usage

The following figure shows how the embedded flash memory is used both by STMicroelectronics and the application software.

Figure 14. Embedded flash memory usage



User and system memories are used differently according to product state and other option bytes settings:

- The user memory contains the application code and data, the root secure services (RSS), the debug authentication code (ST-DA), and the STM32 bootloader. When a reset occurs, the core jumps to the boot address configured through the BOOT pin, the BOOT_ADDR option bytes and the product state
- If the debugger is attached to the product, the entry point is the debug authentication policy, used to unlock the device via the SBS when attached to debugger. A digital signature must be provided to perform a regression to product state, where debug is allowed

Note: For more information on option byte setup for boot, refer to Section 7.4.7.

7.3.4 FLASH read operations

Read operation overview

Read access to main flash memory and system flash memory operates as follows:

- There is a 128-bit read data buffer associated to each bank, which stores the last data read. If several consecutive read accesses request data belonging to the same Flash data word (128 bits), the data are read directly from the current data read buffer, without triggering additional Flash read operations. This mechanism occurs each time a read access is granted. When a read access is rejected for security reasons, the corresponding read error response is issued by the embedded flash memory and no read operation to flash memory is triggered
- The read data buffer is disabled when write access or OTP access, user option change request or other erase operation occurs

Read access to OTP, RO operates as follows:

1. Flash data word of 137 bits is read and stored in a temporary buffer
2. The interface parses the 137 bits data word and selects the 16-bit or 32-bit data requested
3. While parsing the 137-bit data word, two wait states are added and the AHB bus is stalled
4. If the application reads an OTP data not previously written, a double ECC error is reported and only word full of set bits is returned (see *Section 7.3.9: OTP and RO memory access* for details). The read data (in 16 bits) is stored in FLASH_ECCDR register so that the user can identify if the double ECC error is due to a virgin data or a real ECC error
5. Reading two times the same address triggers two reads in the flash memory
6. For 8-bit accesses, an AHB bus error is generated.

Note: The embedded flash memory can perform single-error correction and double-error detection while read operations are being executed (see Section 7.3.8: Flash memory error protections).

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 when 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 filled in instruction cache and executed by the CPU.

Prefetch is enabled by setting the PRFTEN bit in the flash memory access control register (FLASH_ACR). PRFTEN must be set only if at least one wait state is needed to access the Flash memory.

Adjusting read timing constraints

The embedded flash memory clock must be enabled and running before reading data from a non-volatile memory.

To correctly read data from the flash memory, the number of wait states (LATENCY) must be correctly programmed in the Flash access control register (FLASH_ACR) according to the embedded flash memory main AHB interface clock frequency, and the internal voltage range of the device (V_{core}).

The table below shows the correspondence between the number of wait states (LATENCY), the programming delay parameter (WRHIGHFREQ), the embedded flash memory clock frequency and the supply voltage ranges.

Table 20. FLASH recommended number of wait states and programming delay

Number of wait states (LATENCY)	Programming delay (WRHIGH FREQ)	Interface clock frequency v.s. V_{CORE} range			
		VOS3 range 0.95 V - 1.05 V	VOS2 range 1.05 V - 1.15 V	VOS1 range 1.15 V - 1.26 V	VOS0 range 1.30 V - 1.40 V
0 WS (1 FLASH clock cycle)	00	[0 MHz; 20 MHz]	[0 MHZ; 30 MHz]	[0 MHz; 34 MHz]	[0 MHz; 42 MHz]
1 WS (2 FLASH clock cycles)		[20 MHz; 40 MHz]	[30 MHz; 60 MHz]	[34 MHz; 68 MHz]	[42 MHz; 84 MHz]
2 WS (3 FLASH clock cycles)	01	[40 MHz; 60 MHz]	[60 MHz; 90 MHz]	[68 MHz; 102 MHz]	[84 MHz; 126 MHz]
3 WS (4 FLASH clock cycles)		[60 MHz; 80 MHz]	[90 MHz; 120 MHz]	[102 MHz; 136 MHz]	[126 MHZ; 168 MHz]
4 WS (5 FLASH clock cycles)	10	[80 MHz; 100 MHz]	[120 MHz; 150 MHz]	[136 MHz; 170 MHz]	[168 MHz; 210 MHz]
5 WS (6 FLASH clock cycles)		N/A	N/A	[170 MHz; 200 MHz]	[210 MHz; 250 MHz]

Note: The voltage range from 1.26 to 1.30 V is not supported.

Adjusting system frequency

After power-on, the embedded flash memory is clocked by the 32 MHz high-speed internal oscillator (HSI), with a voltage range set at a voltage scaling value of VOS3. As a result, a conservative 3 wait-state latency is specified in FLASH_ACR register (see the previous table).

When changing the bus frequency, the application software must follow the sequence described below, in order to tune the number of wait states required to access the non-volatile memory.

To increase the CPU frequency:

1. If necessary, program the LATENCY and WRHIGHFREQ bits to the right value in the FLASH_ACR register, as described in *Table 20.*
2. Check that the new number of wait states is taken into account by reading back the FLASH_ACR register.
3. Modify the embedded flash memory clock source and/or the clock prescaler in the RCC_CFGR register of the reset and clock controller (RCC).
4. Check that the new embedded flash memory clock source and/or the new AHB clock prescaler value are taken in account by reading back the embedded flash memory

clock source status and/or the prescaler value in the RCC_CFGR register of the reset and clock controller (RCC).

To decrease the CPU frequency:

1. Modify the embedded flash memory clock source and/or the clock prescaler in the RCC_CFGR register of reset and clock controller (RCC).
2. Check that the embedded flash memory new clock source and/or the new clock prescaler value are taken in account by reading back the embedded flash memory clock source status and/or the AHB interface prescaler value in the RCC_CFGR register of reset and clock controller (RCC).
3. If necessary, program the LATENCY and WRHIGHFREQ bits to the right value in FLASH_ACR register, as described in *Table 20.*
4. Check that the new number of wait states has been taken into account by reading back the FLASH_ACR register.

Error code correction (ECC)

The embedded flash memory embeds an error correction mechanism. Single-error correction and double-error detection are performed for each read operation. For more details, refer to *Section 7.3.8: Flash memory error protections*.

Read errors

When the ECC mechanism is not able to correct the read operation, the embedded flash memory reports read errors as described in *Section 7.8.6: Error correction code error (ECCC/ECCD)*

Read interrupts

See *Section 7.9: FLASH interrupts* for details.

7.3.5 FLASH program operations

Program operation overview

Program operation consists in issuing write commands. The embedded flash memory supports the execution of only one write-command at a time. Write-while-write are not supported. Nothing prevents overwriting a non-virgin flash word but this is not recommended. The result may lead to invalid data and inconsistent ECC code.

User Flash and system Flash memories sectors:

For the user and system flash memories, 9-bits ECC is associated to each 128-bits data Flash word. In this case, the embedded flash memory must always perform write operations to non-volatile memory with a 128-bit word granularity. Once the write buffer is full (128 bits), the Busy flag is set and a programming operation is triggered.

There is a write buffer common to Bank1 and 2 which support multiple write-access types (128, 64, 32, 16 or 8 bits). The application can decide to write as little as 8 bits to a 128 Flash word. In this case, a force-write mechanism to the 128 bits + ECC is used (see NSFW bit of FLASH_NSCR register).

When the write request is issued to the memory any new write request stalls the main AHB bus.

Moreover, while a write operation is ongoing, any new read request to the same bank stalls the main AHB bus.

OTP and RO:

When the target memory is OTP, RO sectors, 6-bits ECC code is associated to each 16-bit data Flash word. The embedded flash memory supports 16-bit or 32-bit write operations (8-bit write operations are not supported).For 8-bit accesses, write accesses are ignored. There is no write data buffer. Each write access triggers a write in the flash memory.

Note: The OTP area is typically write-protected on the final product, as described in Section 7.3.9: OTP and RO memory access.

The write protection check is performed at the receiption of the write request (during address phase). Write protection is not checked anymore at the output of the write buffer. If a write protection violation is detected, the write operation is canceled and write protection error (WRPERR) is raised in FLASH_NSSR register.

For write protections of main Flash, ICP, and OTP see Section 7.5.

Monitoring ongoing write operations

The application software can use a status flags located in FLASH_NSSR to monitor ongoing write operations. Since only one operation is possible at a time, this flag indicates if any operation (write, erase, option change) is ongoing; whatever the banks.

- **BSY**: this bits indicate that an effective write, erase, option byte change is ongoing in the non-volatile memory. This flag is not dedicated to a specific bank. It is set when an operation is starting on the memory, whatever the bank. An operation is triggered by:
  - An erase (FLASH_NSCR.STRT)
  - A write (FLASH_NSCR.PG + AHB write)
  - An option modification (FLASH_OPTCR.OPTSTRT)

They are cleared when the current operation is ending or in case of error.

- **NSWBNE**: this bit indicates that the embedded flash memory is waiting for new data to complete the 128-bit write buffer. In this state the write buffer is not empty. It is reset as soon as the application software fills the write buffer, or forces the writes by using NSFW bit in FLASH_NSCR, or an error is detected. When NSWBNE is high, it is not possible to launch an erase, an option modification on flash memory.
- **DBNE**: this bit indicates that the data buffer for parsing 16-bits data is not empty:
  - 16-bits data write access is received and that the data buffer is being filled. It is set at the receipt of the a valid write access and it is reset as soon as the write request preparation has been processed.

Note: If the flash memory is busy at the receipt of the AHB write request, the CPU execution is stalled.

Enabling write operations

Before programming the user flash memory in Bank1 or in Bank2, the application software must make sure that the PG bit is set to 1 in FLASH_NSCR. If it is not the case, an unlock sequence must be used (see *Section 7.5.5: Flash memory configuration protection*) and the PG bit must be set.

When the option bytes must be modified, the application software must make sure that FLASH_OPTCR is unlocked. If this is not the case, an unlock sequence must be used (see *Section 7.5.5: Flash memory configuration protection*).

Note: The application software must not unlock an already unlocked register, otherwise this register remains locked until the next system reset.

If needed, the application software can update the programming delay, as described in *Adjusting programming timing constraints*.

Writing to the FLASH control register FLASH_NSCR and FLASH_OPTCR

The FLASH_NSCR, FLASH_OPTCR registers are not accessible in write mode when the BSY bit is set. Any attempt to write these registers while the BSY bit is set causes the AHB bus to stall until BSY bit is cleared.

Single-write sequence

The recommended single-write sequence is the following:

1. Make sure protection mechanism does not prevent to attempt programming
2. Check that no flash memory operation is ongoing by checking the BSY bit in the FLASH_NSSR register and DBNE bits in the FLASH_NSSR register. Check that the write buffer is empty by checking the NSWBNE bit in the FLASH_NSSR register
3. Check and clear all the error flags due to previous programming/erase operation
4. Unlock the FLASH_NSCR register, as described in *Section 7.5.5: Flash memory configuration protection* (only if register is not already unlocked)
5. Enable write operations by setting PG bit in the FLASH_NSCR register
6. Write one Flash-word at aligned address

Note: NSWBNE flag indicates if the 128-bit write buffer is waiting for new data.

Note: No erase request, options change request is allowed between the first write and the completion of the Flash write operation.

7. Wait for the BSY bit to be cleared in the corresponding FLASH_NSSR register
8. Clear PG bit in FLASH_NSCR register if there are not any more programming requests

If step 6 is executed incrementally (for example byte per byte), the write buffer can become partially filled. In this case the application software can decide to force-write what is stored in the write buffer by using NSFW bit in FLASH_NSCR register. In this particular case, the unwritten bits are automatically set to 1. If no bit in the write buffer is cleared to 0, the NSFW bit has no effect.

Note: Using a force-write operation prevents the application from updating later the missing bits with a value different from 1, which is likely to lead to a unexpected or inconsistent data or ECC.

Adjusting programming timing constraints

Program operation timing constraints depend of the embedded flash memory clock frequency, which directly impacts the performance. If timing constraints are too tight, the non-volatile memory does not operate correctly, if they are too lax, the programming speed is not optimal.

The user must therefore trim the optimal programming delay through the WRHIGHFREQ parameter in the FLASH_ACR register. Refer to *Table 20* in *Section 7.3.4: FLASH read operations* for the recommended programming delay depending on the embedded flash memory clock frequency.

FLASH_ACR configuration register is common to both banks.

The application software must check that no program/erase operation is ongoing before modifying WRHIGHFREQ parameter.

Caution: Modifying WRHIGHFREQ while programming/erasing the flash memory can corrupt the flash memory content.

Programming errors

When a program operation fails, an error can be reported as described in *Section 7.8: FLASH error management*.

Programming interrupts

See *Section 7.9: FLASH interrupts* for details.

7.3.6 FLASH erase operations

Erase operation overview

The embedded flash memory can perform erase operations on 8-Kbyte user sectors, on one user flash memory bank or on two user flash memory banks (for example mass erase).

For more details in user flash memory, user options and OTP erase protection see Section 7.5: FLASH security and protections.

Erase commands are issued through the AHB configuration interface. Since the embedded flash memory supports one operation at a time, if it receives simultaneously a write and an erase request an error flag is raised and both operation are canceled. See *Section 7.8: FLASH error management* for details.

Erase and WRP

If the application software attempts to erase a user sector that is write protected, the sector erase operation is aborted and the WRPERR flag is raised in the FLASH_NSSR register, as described in *Section 7.8.2: Write protection error (WRPERR)*.

Flash Busy

Busy signals is described in *Section : Monitoring ongoing write operations* in *Section 7.3.5: FLASH program operations*

Writing to the FLASH control register FLASH_NSCR and FLASH_OPTCR

Refer to *Writing to the FLASH control register FLASH_NSCR and FLASH_OPTCR* in *Section 7.3.5: FLASH program operations*

Enabling erase operations

Before erasing a sector, the application software must make sure that FLASH_NSCR is unlocked. If this is not the case, an unlock sequence must be used (see *Section 7.5.5: Flash memory configuration protection*).

Note: The application software must not unlock a register that is already unlocked, otherwise this register remains locked until next system reset. This can be used to deliberately lock-out a register from further acceses.

Similar constraints apply to bank erase requests.

Flash sector erase sequence

To erase a 8-Kbyte user sector , proceed as follows:

1. Make sure protection mechanism does not prevent to attempt sector erase (WRP, HDP).
2. Check that no flash memory operation is ongoing by checking the BSY and DBNE bits in the FLASH_NSSR register and that the write buffer is empty by checking the WBNE bit in the FLASH_NSSR register.
3. Check and clear all the error flags due to previous programming/erase operation. Refer to *Section 7.8: FLASH error management* for details
4. Unlock the FLASH_NSCR register, as described in *Section 7.5.5: Flash memory configuration protection* (only if register is not already unlocked)
5. Set the BKSEL bit, the SER bit and SNB bitfield in the FLASH_NSCR register. BKSEL indicates in which physical bank sector has to be erased, then SER indicates a sector erase operation, while SNB contains the target sector number.
6. Set the STRT bit in the FLASH_NSCR register.
7. Wait for the BSY bit to be cleared in the FLASH_NSSR register.
8. STRT bit is automatically cleared at the end of the sector erase or in case of error.
9. Clear SER in FLASH_NSCR register if there are not anymore sector erase request to be issued.

Note: If another erase flag is requested simultaneously to the sector erase, an PGSERR error is generated.

Standard flash memory bank erase sequence

To erase bank :

1. Make sure protection mechanism does not prevent to attempt sector erase.
2. Check that no flash memory operation is ongoing by checking the BSY and DBNE bits in the FLASH_NSSR register and that the write buffer is empty by checking the WBNE bit in the same register.
3. Check and clear all the error flags due to previous programming/erase operation. Refer to *Section 7.8: FLASH error management* for details.
4. Unlock the FLASH_NSCR register, as described in *Section 7.5.5: Flash memory configuration protection* (only if register is not already unlocked).
5. Set the BKSEL bit and the BER bit in the FLASH_NSCR register to the targeted physical bank (swap setting is ignored).
6. Set the STRT bit in the FLASH_NSCR register to start the bank erase operation. Then wait until the BSY bit is cleared in the FLASH_NSSR register.
7. STRT bit is automatically cleared at the end of erase sequence or in case of error.
8. Clear BER in FLASH_NSCR register if there is not other bank erase request to be issued.

Flash mass erase sequence

The application software can set the MER bit to 1 in FLASH_NSCR register, as described below:

1. Make sure protection mechanisms do not prevent to attempt mass erase (*Chapter 7.5*)
2. Check that no flash memory operation is ongoing by checking the BSY and DBNE bits in the FLASH_NSSR register and that the write buffer is empty by checking the WBNE bit in the FLASH_NSSR register
3. Check and clear all the error flags due to previous programming/erase operation. Refer to *Section 7.8: FLASH error management* for details
4. Unlock the FLASH_NSCR register as described in *Section 7.5.5: Flash memory configuration protection* (only if the registers are not already unlocked)
5. Set the MER bit to 1 in FLASH_NSCR register
6. Set the STRT bit in the FLASH_NSCR register. Then wait until BSY bit is cleared in the FLASH_NSSR register
7. STRT bit is cleared automatically at the end of the erase sequence or in case of error
8. Clear MER in FLASH_NSCR register

7.3.7 FLASH parallel operations

Since the non-volatile memory is divided into two independent banks but encapsulated in the same macro, the embedded flash memory interface only supports reading in one bank while writing or erasing is executed in the other bank. This feature is called read-while-write capability (RWW). It does not support write-while-write or read-while-read.

In all cases, the sequences are described in *Section 7.3.4: FLASH read operations*, *Section 7.3.5: FLASH program operations* and *Section 7.3.6: FLASH erase operations* apply.

7.3.8 Flash memory error protections

Error correction codes (ECC)

The embedded flash memory supports an error correction code (ECC) mechanism. It is based on the SECDED algorithm in order to correct single errors and detect double errors.

This mechanism uses 9 ECC bits per 128-bit Flash word, and applies to user and system memory.For read-only, OTP, a stronger 6 ECC bits per 16-bit word is used. A double ECC error is generated for an OTP virgin word (for example a word with 22 bits at 1). When this OTP word is no more virgin, the ECC error disappears.

More specifically, during each read operation from a 128-bit Flash word, the embedded flash memory retrieves the 9-bit ECC information, computes the ECC of the Flash word, and compares the result with the reference value. If they do not match, the corresponding ECC error is raised as described in *Section 7.8.6: Error correction code error (ECCC/ECCD)*

During each program operation, a 9-bit ECC code is associated to each 128-bit data Flash word, and the resulting 137-bit Flash word information is written in non-volatile memory.

A similar mechanism applies to read-only and OTP areas, but with 6-bit ECC for 16-bit data.

7.3.9 OTP and RO memory access

OTP and RO memory are accessed through main AHB interface. The OTP is accessible at the address 0x08FF_F000 to 0x08FF F7FF and the read only section is accessible from 0x08FF F800 to 0x08FF_FFFF.

FLASH one-time programmable area

The embedded flash memory offers a 2048-byte memory area dedicated to application non-confidential, one-time programmable data (OTP). This area is composed of 1024 words of 16 bits (plus 6 bits of ECC). It cannot be erased, and can be written only once. The OTP area can be accessed through the main AHB interface from address 0x08FF F000 to 0x08FF F7FE (last whole readable 16-bit word).

OTP data can be programmed by the application software by chunks of 16 bits. Overwriting an already programmed 16-bit half-word can lead to data and ECC errors and is therefore not supported.

Note: The OTP area is virgin when the device is delivered by STMicroelectronics.

When reading OTP data with a single error corrected or a double error detected, the embedded flash memory reports read errors as described in *Section 7.8.6: Error correction code error (ECCC/ECCD)*.

When reading OTP data not written by the application software (such as virgin OTP), the ECC correction reports a double-error detection (ECCD), and the data are to be found if FLASH_ECCDR register. ECCD implies a NMI raised, unless setting in SBS prevents that.

OTP write protection

OTP data are organized as 32 blocks of 32 OTP words, as shown in *Table 21*. An entire OTP block can be protected (locked) from write accesses by setting the LOCKBLi (i = 0 to 31) bit corresponding to each OTP block (i = 0 to 31) in the FLASH_OTPBLR option byte register. A block can be write-protected whether or not it has been programmed (even partially).

The OTP block locking operation is irreversible and independent from the product life state.

Note: The OTP area can only be accessed in read mode.

Table 21. Flash memory OTP organization

OTP block	AHB address	AHB word		Lock bit
		[31:16]	[15:0]	
Block 0	0x08FF F000	OTP001	OTP000	LOCKBL0
	0x08FF F004	OTP003	OTP002	
	...			
	0x08FF F03C	OTP031	OTP030	
Block 1	0x08FF F040	OTP033	OTP032	LOCKBL1
	0x08FF F044	OTP035	OTP034	
	...			
	0x08FF F07C	OTP063	OTP062	
Block 2	0x08FF F080	OTP065	OTP064	LOCKBL2
	0x08FF F084	OTP067	OTP066	
	...			
	0x08FF F0BC	OTP95	OTP94	
...				
Block 31	0x08FF F7C0	OTP993	OTP992	LOCKBL31
	0x08FF F7C4	OTP995	OTP994	
	...			
	0x08FF F7FC	OTP1023	OTP1022	

OTP write sequence

Follow the sequence below to write an OTP word:

1. Check that no flash memory operation is ongoing by checking the BSY bit in the FLASH_NSSR register and that the data buffer is empty by checking the DBNE bit in the FLASH_NSSR register
2. Check and clear all the error flags due to previous programming/erase operation.
3. Set PG bit in the FLASH_NSCR register
4. Check the protection status of the target OTP word (see *Table 21*). The corresponding LOCKBLi bit must not be set to 1
5. Write two OTP words (32 bits) corresponding to the 4-byte aligned address shown in *Table 21*. Alternatively, the application software can program separately the 16-bit MSB

or 16-bit LSB. In this case the first 16-bit write operation starts immediately without waiting for the second one

6. Wait for the BSY bit to be cleared in the FLASH_NSSR register
7. Clear PG bit in FLASH_NSCR register if there is not any programming request anymore in the bank
8. Optionally, lock the OTP block using LOCKBLi to prevent further data changes.

Note: Do not write twice an OTP 16-bit word, otherwise an ECC error can be generated.

Writing OTP data at byte level is not supported and generates a bus error.

To avoid data corruption, it is important to complete the OTP write process (for example by reading back the OTP value), before starting an option change.

Flash read-only area

The embedded flash memory offers a 2-Kbyte memory area to store read-only data. This area is mapped as described in *Table 22: Read-only public data organization*. It can be accessed through the AHB main port. This read-only area is protected by a robust ECC scheme, as explained in *Section 7.3.8: Flash memory error protections*.

The read-only information that can be used by the application software are described in the table below. This information is programmed by STMicroelectronics.

Table 22. Read-only public data organization

Read-only data name	Address	Comment
Unique device ID	0x08FF F800	U_ID[31:0]
	0x08FF F804	U_ID[63:32]
	0x08FF F808	U_ID[96:64]
Flash memory size/ package	0x08FF F80C	Flash memory size[15:0] || Package code[15:0]
Reserved	0x08FF F810 to 0x08FF FFFF	Reserved information

7.3.10 Flash bank swapping

The embedded flash memory Bank1 and Bank2 can be swapped for the user Flash. This feature can be used after a firmware upgrade to restart the device on the new firmware. Bank swapping is an User Option byte flag controlled by the SWAP_BANK bit of the FLASH_OPTCR register.

Bank specific settings for data area and security attributes follow the original bank and its contents. Control bit always refers to physical bank, not the SWAP_BANK setting.

The following table shows the memory map that can be accessed depending on the SWAP_BANK bit configuration.

Table 24. memory map and the swapping option

Flash memory area	Flash memory corresponding bank		Start address	End address	Size (bytes)	Region name
	SWAP_ BANK=0	SWAP_ BANK=1				
User main memory	Bank1	Bank2	0x0800 0000	0x0800 1FFF	8 K	Sector 0
			0x0800 2000	0x0800 3FFF	8 K	Sector 1
			...	...	...	...
			0x0800 E000	0x0800 FFFF	8 K	Sector 7
	Bank2	Bank1	0x0801 0000	0x0801 1FFF	8 K	Sector 0
			0x0801 2000	0x0801 3FFF	8 K	Sector 1
			...	...	...	...
			0x0801 E000	0x0801 FFFF	8 K	Sector 7
System memory	Bank1		0x0BF8 0000	0x0BF8 1FFF	8 K	System 1 Sector 0
			0x0BF8 2000	0x0BF8 3FFF	8 K	System 1 Sector 1
			...	...	...	...
			0x0BF8 6000	0x0BF8 7FFF	8 K	System 1 Sector 3
	Bank2		0x0BF9 8000	0x0BF9 9FFF	8 K	System 2 Sector 0
			0x0BF9 A000	0x0BF9 BFFF	8 K	System 2 Sector 1
			...	...	...	...
			0x0BF8 E000	0x0BF9 FFFF	8 K	System 2 Sector 3

The SWAP_BANK bit in FLASH_OPTCR register is loaded from the SWAP_BANK option bit only after system reset or POR.

To change the SWAP_BANK bit (for example to apply a new firmware update), respect the sequence below:

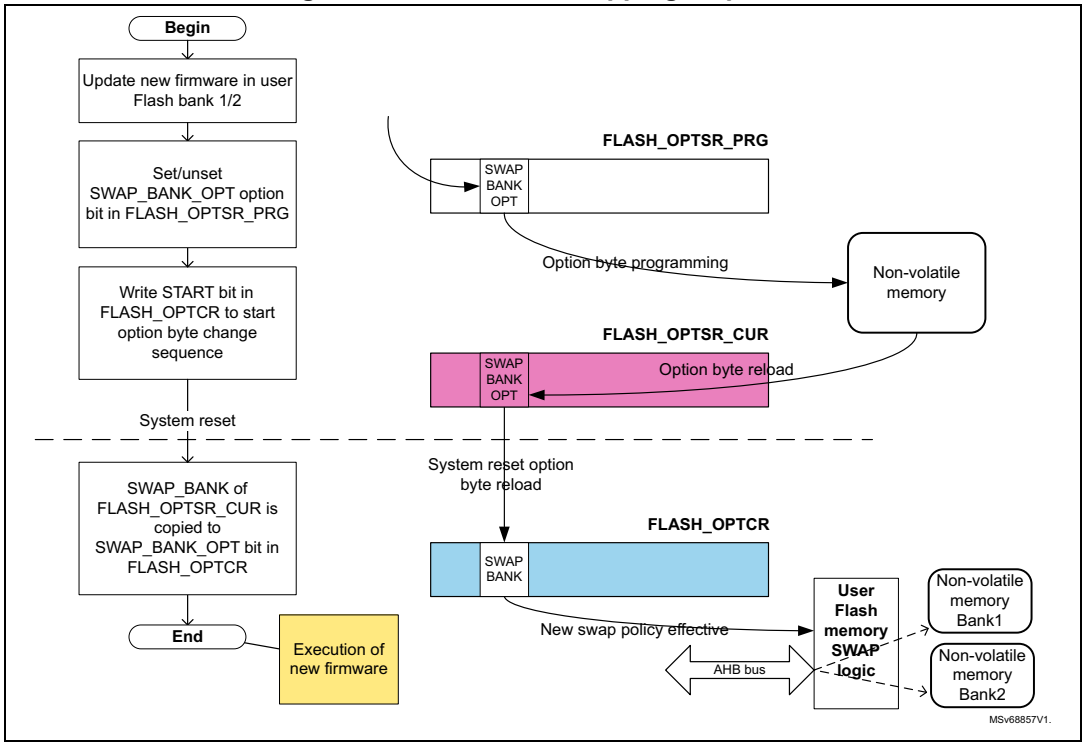
1. Check that no flash memory operation is ongoing by checking the BSY and DBNE bits in the FLASH_NSSR register and that the write buffer is empty by checking the NSWBNE bit in the FLASH_NSSR register
2. Clear all error flags due to a previous operation
3. Unlock OPTLOCK bit, if not already unlocked
4. Set the new desired SWAP_BANK value in the FLASH_OPTSR_PRG register
5. Start the option byte change sequence by setting the OPTSTRT bit in the FLASH_OPTCR register
6. Once the option byte change has completed, FLASH_OPTSR_CUR contains the expected SWAP_BANK value, but SWAP_BANK bit in FLASH_OPTCR has not yet been modified and the bank swapping is not yet effective
7. Force a system reset or a POR. When the reset rises up, the bank swapping is effective (SWAP_BANK value updated in FLASH_OPTCR) and the new firmware shall be executed.

Note: The SWAP_BANK bit in FLASH_OPTCR is read-only and cannot be modified by the application software.

The SWAP_BANK option bit in FLASH_OPTSR_PRG can be modified whatever the product state. Instead of being locked by PRODUCT_STATE, it is locked by NSBOOT_LOCK User OB.

The following figure gives an overview of the bank swapping sequence.

Figure 15. Flash bank swapping sequence



7.3.11 FLASH reset and clocks

Reset management

The embedded flash memory can be reset by a core domain reset, driven by the reset and clock control (RCC). The main effects of this reset are the following:

- All registers, except for option byte registers, are cleared, including read and write latencies. If the bank swapping option is changed, it is applied.
- Most control registers are automatically protected against write operations. To unprotect them, new unlock sequences must be used as described in *Section 7.5.5: Flash memory configuration protection*.

The embedded flash memory can be reset by a power-on core domain reset, driven by the reset and clock control (RCC). When the reset falls, all option byte registers are reset. When the reset rises up, the option bytes are loaded, potentially applying new features. During this loading sequence, the device remains under reset and the embedded flash memory is not accessible.

Reset occurring during Flash operation

If a reset occurs during a Flash operation (programing, erase or option change), the content of the flash memory is not guaranteed. It is mandatory for flash memory integrity than user restarts operation. The status register FLASH_OPSR gives information if any Flash operation has been interrupted by a reset.

FLASH_OPSR.CODE_OP gives opcode of operation. The following table indicates how to use FLASH_OPSR and which operation is required.

Table 25. Recommended reactions to FLASH_OPSR contents

CODE_OP	Operation interrupted	OTP_OP	SYSF_OP	BK_OP	Flash area	ADDR_OP min	ADDR_OP max	Recommended action
0x000	No Flash operation ongoing while reset	0	0	0	-[1]	-	-	No extra action
0x001	Write operation	0	0	0/1	User Flash	0x0000	0xFFFF	Erase sector and rewrite
		1	0	0	OTP	0x0600	0x07FF	
0x011	Sector erase[2]	0	0	0/1	User Flash	0x0000	0xFFFF	Relaunch sector erase
0x100	Bank erase	0	0	0/1	User Flash	-	-	Relaunch bank erase
0x101	Mass erase	0	0	0	User Flash	-	-	Relaunch mass erase
0x110	Option change	0	0	0	User configuration	-	-	New attempt on option change

1. Dash represents “does not matter”.
2. Addresses indicated are aligned to sector start, data area sectors are erased by erase request to corresponding user flash memory sector.

Clock management

The embedded flash memory uses the microcontroller system clock (sys_ck), here the AHB interface clock.

7.4 FLASH option bytes

7.4.1 About option bytes

The embedded flash memory includes a set of non-volatile option bytes. They are loaded at power-on reset and can be read and modified only through configuration registers.This section documents:

- When option bytes are loaded
- How application software can modify them
- The detailed list of option bytes, together with their initial values (before the first option byte change, user default configuration).

7.4.2 Option-byte loading

There are multiple ways of loading the option bytes into embedded flash memory:

1. **Power-on wakeup**

   When the device is first powered, the embedded flash memory automatically loads all the option bytes. During the option byte loading sequence, the device remains under reset and the embedded flash memory cannot be accessed.

2. **Wakeup from system Standby**

   When the core power domain, which contains the embedded flash memory, is switched from Standby mode to Run mode, the embedded flash memory behaves as during a power-on sequence. **During loading time the device is not under reset, unlike power-on sequence.**

3. **Dedicated option byte reloading by the application**

   When the user application successfully modifies the option byte content through the embedded flash memory registers, the non-volatile option bytes are programmed and the embedded flash memory automatically reloads all option bytes to update the option registers.

Note: The option-byte read sequence is protected by error correction code.

In case of error, the option bytes are loaded with default values (see Section 7.4.3: Option-byte modification). This value differs from the initial values (user default configuration) and tends to be more restrictive.

7.4.3 Option-byte modification

Changing user option bytes

A user option-byte change operation can be used to modify the configuration and the protection settings saved in the non-volatile option byte area.

There are numerous rules enforced when attempting to change a user option byte, they are all summarized in *Section 7.4.8: Specific rules for modifying option bytes*. Failing to stick to those rules usually results in error, described in *Section 7.8.7: Option byte change error (OPTCHANGEERR)*.

The embedded flash memory features two sets of option byte registers:

- The first register set contains the current values of the option bytes. Their names have the _CUR extension. All “_CUR” registers are read-only. Their values are automatically loaded from the non-volatile memory after power-on reset, wakeup from system standby or after an option byte change operation.
- The second register set allows the modification of the option bytes. Their names contain the _PRG extension. All “_PRG” registers can be accessed in read/write mode.

When the OPTLOCK bit in FLASH_OPTCR register is set, modifying the FLASH_XXX_PRG registers is not possible.

When OPTSTRT bit is set to 1, the embedded flash memory checks the programming sequence (PGSERR) and the conditions described in *Section 7.4.8: Specific rules for modifying option bytes* (OPTCHANGEERR). If no error has been detected (PGSERR/OPTCHANGEERR), the embedded flash memory launches the option byte modification in its non-volatile memory and updates the option byte registers with _CUR extension.

If one of the condition described in *Section 7.4.8: Specific rules for modifying option bytes*, *Section 7.8.7: Option byte change error (OPTCHANGEERR)* or *Section 7.8.3: Programming sequence error (PGSERR)* is not respected, the embedded flash memory aborts the option byte change operation. In this case, the FLASH_XXX_PRG registers are not overwritten by current option value. The user application can check what was wrong in their configuration.

Unlocking the option-byte modification

After reset, the OPTLOCK bit is set to 1 and the FLASH_OPTCR is locked. As a result, the application software must unlock the option configuration register before attempting to change the option bytes. The FLASH_OPTCR unlock sequence is described in *Section 7.5.5: Flash memory configuration protection*.

Option-byte modification sequence

To modify user option bytes, follow the sequence below:

1. Check that no flash memory operation is ongoing by checking the BSY bit in the FLASH_NSSR register and that the write buffer is empty by checking the NSWBNE bit in the FLASH_NSSR register
2. Check the data buffer is empty (DBNE=0) in FLASH_NSSR register.
3. Clear all error flags due to a previous operation
4. Unlock FLASH_OPTCR register as described in *Section 7.5.5: Flash memory configuration protection*, unless the register is already unlocked.
5. Write the desired new option byte values in the corresponding option registers (FLASH_XXX_PRG).
6. Set the option byte start change OPTSTRT bit to 1 in the FLASH_OPTCR register.
7. Wait that BSY bit is cleared in FLASH_NSSR register
8. OPTSTRT bit is cleared automatically at the end of the sequence (or in case of error)
9. Reset the device. This step is mandatory in case any of the following OB were included in the modification: Product state and NSBOOTADDR. Reset is recommended for other option-byte, but it is not strictly required.

Note: If a reset or a power-down occurs while the option byte modification is ongoing, the original option byte value is kept. A new option byte modification sequence is required to program the new value

7.4.4 Option-byte overview

The following table lists all the user option bytes managed through the embedded flash memory registers, as well as initial values before the first option byte change (user default configuration).

Table 26. Option-byte organization

Register	Bitfield															
FLASH_OPTSR[31:16]	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SWAP_BANK	Res	Res.								IWDG_STDBY	IWDG_STOP	Res.		IO_VDDIO2_HSLV	IO_VDD_HSLV
Default factory value	*0*	*0*	*1*	*0*	*1*	*1*	*0*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*0*	*0*
FLASH_OPTSR[15:0]	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PRODUCT_STATE								NRST_STDBY	NRST_STOP	Res.	WWDG_SW	IWDG_SW	BORH_EN	BOR_LEV	
Default factory value	*1*	*1*	*1*	*0*	*1*	*1*	*0*	*1*	*1*	*1*	*0*	*1*	*1*	*0*	*0*	*0*

Table 26. Option-byte organization (continued)

Register	Bitfield															
FLASH_OPTSR2[31:16]	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
Default factory value	*1*	*1*	*0*	*0*	*0*	*0*	*1*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
FLASH_OPTSR2[15:0]	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Res.	Res.	Res.	Res.	Res.	SRAM1_ECC	SRAM1_RST	Res.	Res.	SRAM2_ECC	Res.	BKPRAM_ECC	SRAM2_RST	Res.	Res.	Res.
Default factory value	*0*	*0*	*0*	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*0*	*1*	*1*	*0*	*0*	*0*
FLASH_NSBOOTR[31:16]	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NSBOOTADD[23:8]															
Default factory value	*0x0800*															
FLASH_NSBOOTR[15:0]	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NSBOOTADD[7:0]								NSBOOT_LOCK							
Default factory value	*0x00*								*0xC3*							
FLASH_WRP1R[31:16]	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
Default factory value	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*
FLASH_WRP1R[15:0]	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP[7]	WRP[6]	WRP[5]	WRP[4]	WRP[3]	WRP[2]	WRP[1]	WRP[0]
Default factory value	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*
FLASH_WRP2R[31:16]	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
Default factory value	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*
FLASH_WRP2R[15:0]	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP[7]	WRP[6]	WRP[5]	WRP[4]	WRP[3]	WRP[2]	WRP[1]	WRP[0]
Default factory value	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*
FLASH_OTPBLR[31:16]	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	LOCKBL															
Default factory value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26. Option-byte organization (continued)

Register	Bitfield															
FLASH_OTPBLR[15:0]	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LOCKBL															
Default factory value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FLASH_HDP1R[31:16]	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_END		
Default factory value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
FLASH_HDP1R[15:0]	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.	HDP1_STRT		
Default factory value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FLASH_HDP2R[31:16]	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_END		
Default factory value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
FLASH_HDP2R[15:0]	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.	HDP2_STRT		
Default factory value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

7.4.5 Description of user and system option bytes

The general-purpose option bytes that can be used by the application are listed below:

- Watchdog management
  - IWDG_STOP: independent watchdog IWDG counter active in Stop mode if 1 (stop counting or freeze if 0)
  - IWDG_STDBY: independent watchdog IWDG counter active in Standby mode if 1 (stop counting or freeze if 0)
  - IWDG_SW: hardware (0) or software (1) IWDG watchdog control selection

Note: If the hardware watchdog "control selection" feature is enabled (set to 0), the watchdog is automatically enabled at power-on, thus generating a reset unless the watchdog key register is written to or the down-counter is reloaded before the end-of-count is reached.

Depending on the configuration of IWDG_STOP and IWDG_STDBY options, the IWDG can continue counting (1) or not (0) when the device is in Stop or Standby mode, respectively.

When the IWDG is kept running during Stop or Standby mode, it can wake up the device from these modes.

- Reset management
  - BOR_LEV: Brownout reset (BOR) level, indicating the supply level threshold that activates/releases the reset. BOR can be enabled by setting BORH_EN option bit. Refer to *Section 9: Power control (PWR)*
  - NRST_STDBY: generates a reset when entering Standby mode if cleared to 0
  - NRST_STOP: generates a reset when entering Stop or Stop2 mode if cleared to 0
  - BORH_EN: enables brownout reset (BOR). Refer to *Section 9: Power control (PWR)*

Note: Whenever a Standby (respectively Stop) mode entry sequence is successfully executed, the device is reset instead of entering Standby (respectively Stop) mode if NRST_STDBY (respectively NRST_STOP) is cleared to 0.

- Device options
  - IO_VDDIO2_HSLV: enables the configuration of pads below 2.7 V for VDDIO2 power rail if set to 1
  - IO_VDD_HSLV: enables the configuration of pads below 2.7 V for VDD power rail if set to 1

When STMicroelectronics delivers the device, the values programmed in the general-purpose option bytes are the following:

- Watchdog management
  - IWDG active in Standby and Stop modes (option value = 0x1)
  - IWDG not automatically enabled at power-on (option byte value = 0x1)
- Reset management:
  - BOR: brownout level option (reset level) equals 2.1 V (option byte value = 0x0). A reset is not generated when the device enters Standby, Stop or Stop2 low-power mode (option byte value = 0x1)
- Device working in the full voltage range with I/O speed optimization at low-voltage disabled (IO_VDDIO2_HSLV=IO_VDD_HSLV=0)

Refer to *Section 7.10: FLASH registers* for details.

7.4.6 Description of data protection option bytes

The option bytes that can be used to enhance data protection are listed below:

- PRODUCT_STATE[7:0]: A product life cycle state (see *Section 7.5.8: Product state transitions* for details).
- WRP1/2: write protection option of sectors in Bank1 (respectively Bank2). It is active low. Refer to *Section 7.5.6: Write protection* for details.
  - Bit N: sector N write protected.
- HDPx: HDPL exclusive area control in the user flash memory.

When factory programmed, values of the data protection option bytes are the following:

- Product state depends on sales type
- Write protection disabled (all option byte bits set to 1)

Refer to *Section 7.10: FLASH registers* for details.

7.4.7 Description of boot address option bytes

Below the list of option bytes that can be used to configure the appropriate boot address for an application:

- PRODUCT_STATE
  Influences the boot sequence indirectly.
- NSBOOTADD
  Selects default boot address.
- BOOT_LOCK
  Protects the boot configuration from further modification attempts.
- SWAP_BANK: bank swapping option, set to 1 to swap user Flash banks after boot (see *Section 7.3.10: Flash bank swapping*). If BOOT_LOCK is active, the value in SWAP_BANK is fixed, read-only.

When STMicroelectronics delivers the device, the PRODUCT_STATE is Open, BOOT_LOCK is not set (0xC3). Address NSBOOTADD = 0x0800 0000.

Refer to *Section 7.10: FLASH registers* for details.

7.4.8 Specific rules for modifying option bytes

On top of OPTLOCK bit and register access rules, there are also other protection means for selected security-sensitive option byte fields.

More option bytes can be modified simultaneously, but if they rely on each other for protection, both states are checked.

With few exceptions listed in this chapter, failing to uphold the rules results in raising OPTCHANGEERR flag (*Section 7.8.7: Option byte change error (OPTCHANGEERR)* for additional details).

Table 27. Specific OB modifying rules overview

OB	HDPL	Value	Product state
PRODUCT_STATE	0,1(1)	Set of possible transitions	Set of possible transitions (*Table 37*)
HDP	0,1(1)	-	-
NSBOOTADD	-	-	Open or Provisioning, NSBOOT_LOCK disabled
LOCKBL	-	One way switch(1)	-
SWAP_BANK	-	-	-
NSBOOT_LOCK	-	-	Open, Regression or Provisioning to unlock

1. Most transitions are possible regardless of HDPL. Only selected few require specific HDPL, see text below.

Even in the closed PRODUCT_STATE progression, some OB can still be modified, if all the other constraints are satisfied. An overview is presented in the following table.

Table 28. OB modifiable in closed product

PRODUCT_STATE	OBs that may be modified
Closed	SWAP_BANK, LOCKBL, PRODUCT_STATE
Locked	SWAP_BANK, LOCKBL

Specific rules must be respected to update the following OB:

- **PRODUCT_STATE**

  PRODUCT_STATE transitions follow a state machine with two types of transition. Locking down the product is allowed without restriction. Opening the product is only possible using a debug interface and following a digital signature verification. The regression to “Open PRODUCT_STATE” results in the erasing of the protected content. More details are given in *Section 7.5.8: Product state transitions*.

  Summary: Selected changes only possible in HDPL1, regression in HDPL0.
- **HDP**

  Can only be modified in HDPL0 and HDPL1.
- **NSBOOTADD**

  Can only be changed in Open and Provisioning. Locked by NSBOOT_LOCK.
- **LOCKBL**

  Can only be changed freely in one direction. A permanent irreversible switch.
- **SWAP_BANK**

  Not modifiable when NSBOOT_LOCK is active (0xB4).
- **NSBOOT_LOCK**

  Can only be changed freely in locking direction. Unlock is possible in Open, Provisioning and Regression.

Note: For all user option bytes above: default values are loaded and Tamper is signaled when a double ECC error occurs during OBL.

7.5 FLASH security and protections

Since sensitive information are stored in the flash memory, it is important to protect it against unwanted operations such as reading confidential areas, illegal programming of immutable sectors, or malicious flash memory erasing.

For that purpose flash memory implements the following protection mechanisms:

- Temporal isolation protection (HDP)
- Configuration protection
- User Flash write protection
- Device boot state management
- OTP locking

This section provides a detailed description of all these security mechanisms.

Two access modes are possible:

- Unprivilege
- Privilege.

The flash memory interface evaluates the access restrictions in the following order:

1. Write protection (for write access)
2. HDP
3. Privilege

7.5.1 Hide protection (HDP)

One non-volatile secure hide protection (HDP) area per bank can be defined with a sector granularity. Access to the hide protection area can be denied by progressing the HDPL level in the SBS.

When the HDPL = 1, no user Flash is protected by the HDP. With HDPL ≥ 2 the user OB defined HDP area in each bank is closed. No read, write, fetch or erase is allowed in the HDP area.

The HDP area can be extended, the extension setting is only a register value, not stored in OB. With HDPL = 3 the HDP area remains activated and the extension is added. If extension is added while the base user OB defined area is not active, the extension covers one more sector, because HDPx_END sector is also protected.

The HDPL level can be only cleared by a system reset, there is no means to deactivate the HDP area.

The protected HDP area is defined by setting its size using start and end sectors .

The size of the inaccessible area can be extended using register FLASH_HDPEXTR values HDPx_EXT. The value HDPx_EXT is a number of sectors added to the HDP area (past the HDPx_END sector). The volatile HDPx_EXT value in the register can be increased, but cannot decrement.

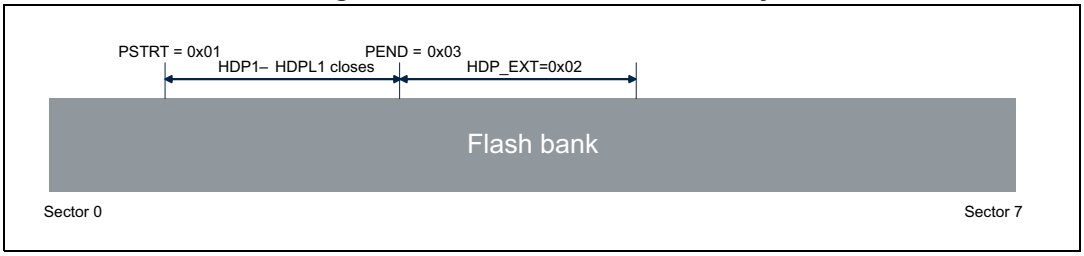
In case of HDPx_STRT > HDPx_END the extension will cover (HDPx_EXT + 1) size area between sectors marked by HDPx_END and HDPx_END + HDPx_EXT.

By default both the HDPx_END is set to zero and HDPx_STRT is set to one. This means that the HDP area in user Flash is zero size and extends from the Flash start address (first sector).

For example, to protect area by HDP from the address 0x0800 2000 (included) to the address 0x0800 5FFF (included):

- For physical Bank1, the option-byte registers must be programmed with:
  - HDP1_STRT = 0x01,
  - HDP1_END = 0x03.

Figure 16. HDP in user flash memory



- Once the HDPL is incremented to 2, the marked area becomes inaccessible. The HDPL = 2 code then sets the extension size to 2. Once the HDPL is incremented to 3, the extension area becomes also inaccessible.

  Alternatively, if:

  HDP1_END = 0

  HDP1_START = 1

  HDPx_EXT = 1

  Sector 0 and Sector 1 are HDP protected.

  Or a rather unusual example:

  HDP1_END = 2

  HDP1_START = 7

  HDPx_EXT = 3
- Sectors 2 until 5 are HDP protected. Here the user should know that if his application needs sectors 2 until 4 only as HDP area, they should program HDPx_EXT at 2 instead of 3.
- If the two banks are swapped, the protection defined to physical Bank1 remains on the physical Bank1, unaffected by swapping. Separate protection applies to physical Bank2 and the option bytes registers must also be programmed with:
  – HDP2_STRT = 0x01
  – HDP2_END = 0x03

Note: For more details on the bank swapping mechanism, refer to Section 7.5.4: Flash memory banks attributes in case of bank swap.

Table 29. HDP protected definition

User Flash	OB HDP Area enabled	HDP extension enabled
HDPL = 0,1	HDP protected = 0	HDP protected = 0
HDPL = 2	HDP protected = 1	HDP protected = 0
HDPL = 3	HDP protected = 1	HDP protected = 1

Table 30. Secure hide protection

HDPx watermark option bytes values (x = 1,2)		Hide protection area
HDPL ≤ 1	-	No inaccessible HDP area in user Flash

Table 30. Secure hide protection

HDPx watermark option bytes values (x = 1,2)		Hide protection area
HDPL = 2	HDPx_END < HDPx_STRT	No inaccessible HDP area in user flash memory
	HDPx_END = HDPx_STRT	Exactly one sector is protected by HDP
	HDPx_END > HDPx_STRT	The area between HDPx_STRT and HDPx_END is HDP protected
HDPL = 3	HDPx_END < HDPx_STRT and HDPx_EXT=0	No inaccessible HDP area in user flash memory
	HDPx_END ≥ HDPx_STRT or HDPx_EXT>0	The area between min(HDPx_STRT, HDPx_END) and (HDPx_END + HDPx_EXT) is HDP protected

Table 31. HDP protections summary

HDP protection	User Flash			
	Access to OB HDP area		Access to EXT HDP area	
	HDPL 2 or 3	HDPL0 or HDPL1	HDPL3	HDPL0, 1 or 2
Fetch	BUS ERROR	OK	BUS ERROR	OK
Read	RAZ		RAZ	
Write	WI, WRPERR	If WRP disabled: OK else: WI, WRPERR	WI, WRPERR	If WRP disabled: OK else: WI, WRPERR
Erase (mass erase)				

7.5.2 Privileged flash memory area protection

Any sector can be programmed on-the-fly as privileged or unprivileged using the configuration registers. FLASH_PRIVBB1x (respectively FLASH_PRIVBB2x) registers are used to configure the privilege attribute for sectors in Bank1 (respectively Bank2).

When the sector privilege attribute PRIVBByx[i] bit is set, the sector is only accessible by a privileged access. An unprivileged sector is accessible by a privileged or unprivileged access.

To modify a block-based privilege attribution, it is recommended to:

- Check that no Flash operation is ongoing on the related sector
- Add an ISB instruction after modifying the sector security attribute PRIVBByx[i]

Caution: Switching a sector from privileged to unprivileged does not erase the content of the associated sector.

Table 32. Privilege protection summary

Access		Main Flash	
		Unprivilege sector	Privilege sector
Privilege	Fetch	OK	
	Read		
	Write		
	Erase		
Unprivilege	Fetch	OK	RAZ
	Read		
	Write		WI, WRPERR
	Erase		

Table 33. Privilege and mass or bank erase

Access	Main Flash		
	Unprivilege FLASH	Privilege FLASH	Mix unprivilege and privilege FLASH
Privilege mass erase	OK		
Unprivilege mass erase	OK	WI, WRPERR	

Note: For PRIVBByx[i] access control, refer to Table 34: Privilege configuration register access conditions.

Table 34. Privilege configuration register access conditions

Access		Sector setting access in PRIVBBRxy
X	P/NP	Bus error
R	P/NP	OK
W	P	OK
W	NP	WI

7.5.3 Flash memory registers privileged and unprivileged modes

The Flash registers can be read and written by privileged and unprivileged accesses depending on the NSPRIV bits in *FLASH privilege configuration register (FLASH_PRIVCFGR)*.

- When the NSPRIV bit is reset, all flash memory registers could be read and written by both privileged or unprivileged access.
- When the NSPRIV bit is set, all flash memory registers could be read and written by privileged access only. Unprivileged access to a privileged registers is RAZ/WI.

7.5.4 Flash memory banks attributes in case of bank swap

The SWAP_BANK option bit modifies the address of each bank in the memory map. When SWAP_BANK is reset, Flash Bank1 is mapped at the lower address range. When SWAP_BANK is set, Flash Bank1 is mapped at the higher address range. Flash bank attributes follow their bank contents so there is no need to modify their setting registers when swapping banks:

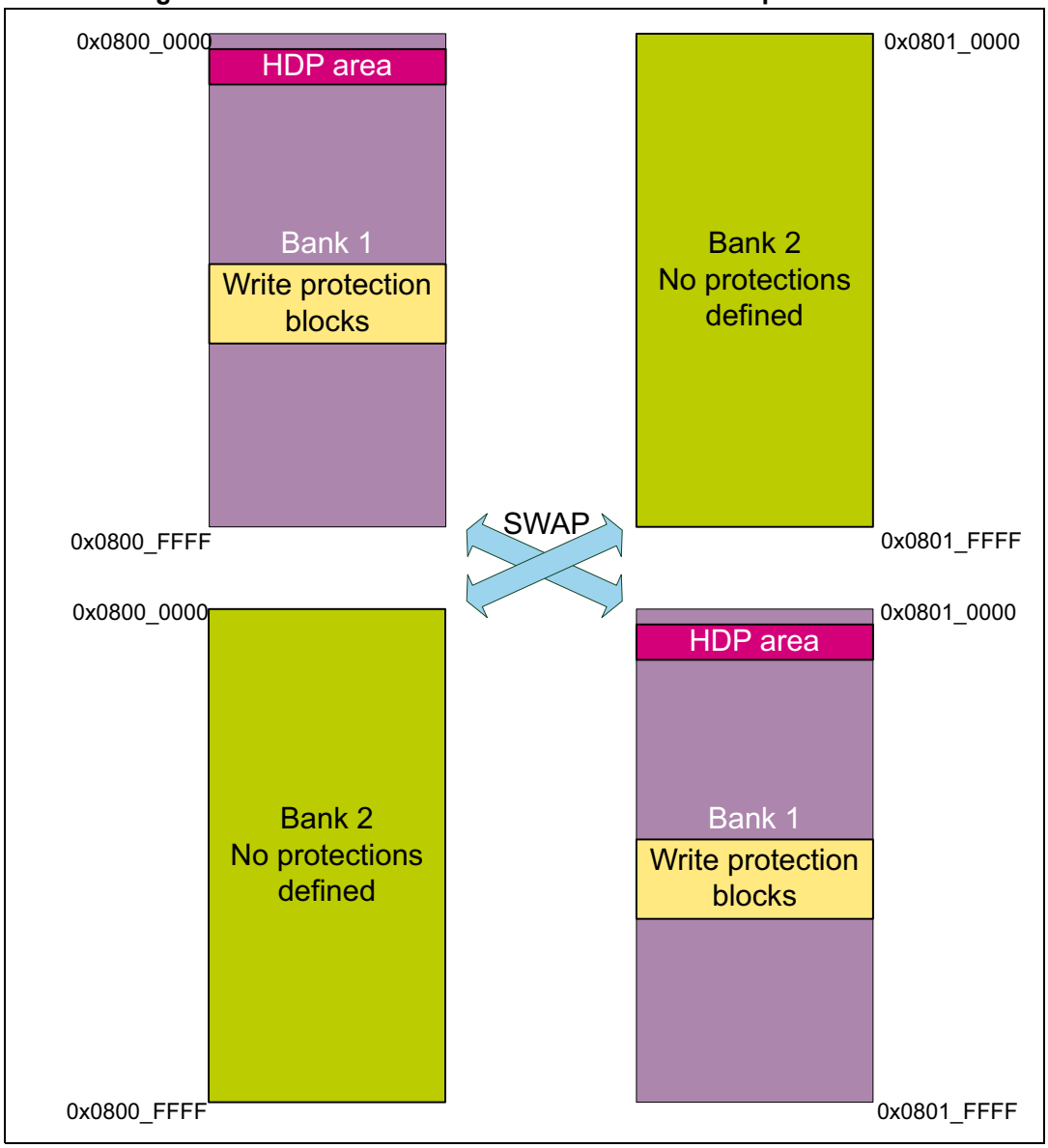
- Flash write protection sector FLASH_WRPSG (refer to *Section 7.5.6: Write protection*)
- Hide Protection in FLASH_HDPx registers.
- Flash privilege block based bank x register y FLASH_PRIVBBxy

The SWAP_BANK is rendered immutable by setting NSBOOT_LOCK.

Note: The BK_ECC bit in the FLASH ECC correction register (FLASH_ECCCORR) and FLASH ECC detection register (FLASH_ECCDETR), BKSEL bit in FLASH control register (FLASH_NSCR) always refers to Bank1 (respectively Bank2) when it is low (respectively high), regardless of the SWAP_BANK value.

The figure below shows how security attributes and protections behave in case of bank swap.

Figure 17. Protection attributes in case of bank swap illustration



7.5.5 Flash memory configuration protection

The embedded flash memory uses hardware mechanisms to protect the following assets against unwanted or spurious modifications (such as software bugs):

- Option bytes change
- Write operations
- Erase commands
- Interrupt masking

The embedded flash memory configuration registers protection is summarized in the next table. Registers not present in this table are not protected by a key.

Table 35. Flash interface register protection summary

Register name	Unlocking register	Protected asset
FLASH_NSCR	FLASH_NSKEYR	Write/erase control
FLASH_OPTCR	FLASH_OPTKEYR	Flash bank option byte words change

7.5.6 Write protection

The purpose of embedded flash memory write protection is to prevent unwanted modifications to embedded non-volatile code and/or data.

Any 8 Kbytes sector can be independently write-protected or unprotected by clearing/setting the corresponding WRP1/2 bit in the FLASH_WRP1/2 register (see *FLASH write sector protection for Bank1 (FLASH_WRP1R_CUR)* and *FLASH write sector protection for Bank2 (FLASH_WRP2R_CUR)*).

A write-protected sectors can neither be erased nor programmed. As a result, a bank erase cannot be performed if some sector is write-protected.

Note: Write protection errors are documented in Section 7.8: FLASH error management.

7.5.7 Life cycle management

Non-volatile states, or debug states, are determined by the product state set by user option bytes. Debug possibility and possibility to change selected security settings are related to this state. While the state is stored in the OB, it is the SBS that controls the debug policy.

HDP works equally regardless of the debug state. Even in debug the HDP area is hidden when HDPL increments.

The following states are defined:

Table 36. Product states, debug states and debug policy

PRODUCT_STATE	Code in OB	Description
Open	0xED	User Flash open (~RDP0)
Provisioning	0x17	Provisioning - OEM immutable root of trust is being installed.
Provisioned	0x2E	OEM complete - the OEM immutable root of trust is installed. (~RDP 0.5)
Closed	0x72	State for running application. Debug disabled, regression is possible.
Locked	0x5C	Transition to other state, policy change or debug is not permitted.(1)
Regression	0x9A	The temporary state initiated by the debug authentication system in transition to Open

1. Effectively the same situation is Closed state with incorrect or not defined debug certificates.

1. **no debug protection**

Read, program and erase operations into the Flash main memory area are possible. Least restriction on the option bytes.

2. **full product protection**

All debug features are disabled:

- User mode: code executing in user mode (boot Flash) can access Flash main memory and selected option bytes with all operations (read, erase, program).
- Boot RAM mode: boot from SRAM is not possible.
- Transition to Open state is possible with the consequence of Flash mass erase. Depends on debug unlock policy (see below).

7.5.8 Product state transitions

Progressing to more closed state is the normal product life cycle, that does not require security measures. Transition in direction to open state is a regression, controlled by the debug authentication control. If debug unlock policy is set to “locked”, no regression is accepted. Locked state may be reinforced by invalidating the debug authentication certificates.

All transitions not listed in this chapter are invalid.

There is no restriction on reading the current product state.

Transitions in progress direction

Transition from Open to any of the closed states is matter of correct product configuration and provisioning. The transitions must be done in correct order.

The normal progression is:

– Open to Provisioning or Provisioned
– Provisioning to Provisioned
– Provisioned to Locked or Closed

Transition is managed either by a software running on the device, or directly, using a debug interface. By software it is assumed that transitions are triggered by bootloader, or root of trust services, but generally any software running on the device can do a progress transition.

Transition to open debug state

This transition is a full regression. The starting state is any except Open and Locked. The debug tools are used to authenticate debug regression access rights with the debug authentication library, running on the device in HDPL=1 (refer to *Section 13.3.6: SBS debug control*). After verifying the credentials, it puts the device into intermediate state Regression. From this state the device regresses securely in HDPL=0 to Open.

The transition has the following consequence:

– Product state updated
– User options updated (security settings reset)
– Flash memory mass erase (even write protected sectors)
– Backup RAM erase (even if backup domain write protection is enabled)

Table 37. PRODUCT_STATE transitions

From	To[1]					
	Open	Provisioning	Provisioned	Closed	Locked	(Full) regression
Open	-	OK	OK	X	X	X
Provisioning	X	-	OK	OK	OK	OK (HDPL1)
Provisioned	X	X	-	OK	OK	OK (HDPL1)
Closed	X	X	X	-	X	OK (HDPL1)
Locked	X	X	X	X	-	X
(Full) regression	OK (HDPL0)	X	X	X	X	-

1. X = transitions that are not permitted (OPTCHANGEERR raised)
- = no change
OK = valid transitions.

In the *Table 37: PRODUCT_STATE transitions* the X marks transitions that are not permitted, dash marks no change, OK marks valid transitions. Some transitions are only possible in correct HDPL provided in parenthesis. Attempt to do a illegal transition results on OPTCHANGEERR.

Incorrect PRODUCT_STATE (on OBL) is interpreted as Locked.

7.5.9 One-time-programmable and read-only memory protections

Sections of OTP/RO in the flash memory are described in details in *Section 7.3.9: OTP and RO memory access*. There is no protection provided by the Flash interface other than the dedicated write protection.

No write is possible to the RO area, once it was established in manufacturing. The *OTP write protection* section describes a method of locking out the OTP area.

The access conditions are summarized in the following table.

Table 38. OTP/RO access constraints

-	8 Kbytes sector dedicated to RO/OTP			
	0x08FF_E000 - E7FF	**0x08FF_E800 - EFFF Reserved**	**0x08FF_F000 - F7FF OTP**	**0x08FF_F800 - FFFF RO**
X	BUS ERROR	BUS ERROR		
R		If correct size(1) RAZ, else Bus Error	If correct size OK, else Bus Error	
W		If correct size WI, else Bus Error	Bus Error if incorrect size. WRPERR if block is protected by LOCKBL, else OK.	If correct size WI, else Bus Error
erase	WI			

1. Word size is 16-bits, 32b access is possible. Other access attempt leads to bus error.

7.6 System memory

7.6.1 System memory introduction

System memory stores RSS (root secure services) firmware that is programmed by ST during STM32H503xx production. The RSS provides runtime services to user firmware. These services are described hereafter in this section.

7.6.2 RSS user 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 STM32CubeH5 firmware package. Please refer to UM2656 to get more details regarding STM32CubeH5 firmware package. RSS library provides runtime services through prefixed NSSLIB functions.

Table 39. C defined macro for RSS services

C defined macro	Location in flash memory
NSSLIB_PFUNC	0x0BF8FE6C

NSSLIB

The user firmware calls NSSLIB functions using NSSLIB_PFUNC C defined macro, that points to a location within system memory.

Table 40. NSSlib interface function list

Library	Function
NSSLIB_PFUNC	JumpHDPLvl2
	JumpHDPLvl3

The NSSLIB functions are described within section hereafter.

a) JumpHDPLvl2

Prototype:

```
uint32_t JumpHDPLvl2(uint32_t VectorTableAddr, uint32_t MPUIndex)
```

User code function call example:

```
NSSLIB_PFUNC->JumpHDPLvl2((uint32_t)NextVectorTableAddr, 1U );
```

Arguments:

- VectorTableAddr:

Input parameter, address of the next vector table to apply.

The vector table format is the one used by the Cortex-M33 core.

- MPUIndex:

Input parameter, MPU region index. Caller function shall define but keep disable the corresponding MPU region before calling JumpHDPLvl2. The function enables the MPU region before jumping to the Reset Handler of the vector table. The vector table Reset Handler function shall belong to the MPU region.

Descriptions:

User calls JumpHDPLvl2 to:

- Close user Flash HDPL1 area by incrementing HDPL to 2
- Jump to the reset handler embedded within the Vector Table which address is passed as input parameter

After closing HDPL1, JumpHDPLvl2 enables the MPU region provided as input parameter. Once the MPU is enabled, the function sets the SP to the address provided by the passed Vector Table and jumps to the Reset Handler function supported by the Vector Table too. JumpHDPLvl2 does not set the new vector table.

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

Errors:

In case of failure (bad input parameter value), RSSLIB_Sec_JumpHDPLvl2 returns 0xF5F5F5F5.

b) JumpHDPLvl3

Prototype

```
uint32_t JumpHDPLvl3(uint32_t VectorTableAddr, uint32_t MPUIndex)
```

User code function call example:

```
NSSLIB_PFUNC->JumpHDPLvl3((uint32_t)NextVectorTableAddr, 1U );
```

Arguments:

- VectorTableAddr:
  Input parameter, address of the next vector table to apply.
  The vector table format is the one used by the Cortex-M33 core.
- MPUIndex:
  Input parameter, MPU region index. Caller function shall define but keep disable the corresponding MPU region before calling JumpHDPLvl3. The function enables the MPU region before jumping to the Reset Handler of the vector table. The vector table Reset Handler function shall belong to the MPU region.

Descriptions:

User calls JumpHDPLvl3 to:

- Close user Flash HDPL1 and HDPL2 areas by incrementing HDPL up to 3
- Then jump to the reset handler embedded within the Vector Table which address is passed as input parameter

After closing HDPL1/2, JumpHDPLvl3 enables the MPU region provided as input parameter. Once the MPU is enabled, the function sets the SP to the address provided by the passed Vector Table and jumps to the Reset Handler function supported by the Vector Table too. JumpHDPLvl3 does not set the new vector table.

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

Errors:

In case of failure (bad input parameter value), JumpHDPLvl3 returns 0xF5F5F5F5.

7.7 FLASH low-power modes

Table 41 summarizes the behavior of embedded flash memory in STM32 low-power modes. Embedded flash memory belongs to the core domain.

Table 41. Effect of low-power modes on the embedded flash memory

Power mode	Core domain voltage range	Allowed if FLASH Busy	FLASH power mode
Run/Sleep	VOS0/1/2/3	Yes	Run
Stop (clock stopped)	SVOS3/4/5	No	Clock gated (in normal or Low-power mode)(1)
Standby	Off	No	Off

1. Flash is forced in Low-power mode when SVOS5 is selected.

When the system state changes or within a given system state, the embedded flash memory might get a different voltage supply range (VOS) according to the application. The procedure to switch the embedded flash memory into various power mode (run, clock gated, stopped, off) is described hereafter.

Note: For more information in the microcontroller power states, refer to the Power control section (PWR).

Managing the FLASH domain switching to Stop or Standby

As explain in *Table 41*, if the embedded flash memory informs the reset and clock controller (RCC) that it is busy (BSY, DBNE, NS/WBNE is set), the microcontroller cannot switch the core domain to Stop or Standby mode.

There are two ways to release the embedded flash memory:

- Reset the NSWBNE busy flag in FLASH_NSSR register by any of the following actions:
  a) Complete the write buffer with missing data.
  b) Force the write operation without filling the missing data by activating the NSFW bit in FLASH_NSCR register. This forces all missing data “high”.
- Poll BSY busy bits in FLASH_NSSR register until they are cleared. This indicates that all recorded write, erase and option change operations are complete.

The microcontroller can then switch the domain to Stop or Standby mode.

7.8 FLASH error management

7.8.1 Introduction

The embedded flash memory automatically reports when an error occurs during a read, program or erase operation. A wide range of errors are reported:

- *Write protection error (WRPERR)*
- *Programming sequence error (PGSERR)*
- *Error correction code error (ECCC/ECCD)*
- *Option byte change error (OPTCHANGEERR)*

The application software can individually enable the interrupt for each error, as detailed in *Section 7.9: FLASH interrupts*.

Note: For all errors, the application software must clear the error flag before attempting a new modify operation.

Since there is just one write buffer and only one operation is allowed at a time, the write control bits and status bits are shared by both banks:

- Write control bits (PG, NSFW, EOPIE, WRPERRIE, PGSERRIE, STRBERRIE, INCERRIE, CLR_EOP, CLR_WRPERR, CLR_STRBERR, CLR_INCERR, CLR_PGSERR) controls bank1 and 2 at the same time.
- Write status flag reports (NSWBNE, DBNE, EOP, WRPERR, PGSERR, STRBERR, INCERR, BSY) error for Bank1 and 2 at the same time.

7.8.2 Write protection error (WRPERR)

When an illegal erase/program operation is attempted to the non-volatile memory, the embedded flash memory sets the write protection error flag WRPERR in FLASH_NSSR register.

An erase operation is rejected and flagged as illegal if it targets one of the following memory areas:

- A sector write-protected with WRP.
- An HDP area while the HDPL has made it inaccessible.
- Attempt to erase privilege sector within unprivileged mode.

A program operation is ignored and flagged as illegal if it targets one of the following memory areas:

- The system flash memory
- A user sector write-protected with WRP
- An OTP block, locked with LOCKBL
- A read-only section
- A reserved area
- Privileged area from within unprivileged mode

When WRPERR flag is raised, the operation is rejected and nothing is changed in the corresponding bank.

If this error is detected the write buffer is invalidated.

Note: WRPERR flag has to be cleared before any erase/program operation.

WRPERR flag is cleared by setting CLR_WRPERR bit to 1 in FLASH_NSCCR register.

If WRPERRIE bit in FLASH_NSCR register is set to 1, an interrupt is generated when WRPERR flag is raised (see *Section 7.9: FLASH interrupts* for details).

7.8.3 Programming sequence error (PGSERR)

When the programming sequence is incorrect, the Flash interface sets the programming sequence error flag PGSERR in FLASH_NSSR register.

More specifically, PGSERR flag is set when one of below conditions is met:

- An error like INCERR, WRPERR, PGSERR, STRBERR, OPTCHANGEERR have not been cleared before requesting a new write or erase operation or options change.
- For erase, PGSERR is set if:
  - missing erase operation (FLASH_NSCR): STRT=1 with (MER=0, SER=0 and BER=0)
  - sector and bank erase requested at the same time (STRT and more than one of MER, SER and BER)
  - PG set during erase operation (it ensures that no erase req and write req happen at the same time): (STRT=1) with (PG=1)
  - erase operation is started while write buffer is waiting for next data: (STRT=1) with WBNE=1.
  - erase operation is started while DBNE=1
- For programming, PGSERR is set if:
  - missing write flag: A write operation is requested but the program enable bit PG has not been set in FLASH_NSCR register prior to the request.
  - AHB write request is received and (SER=1, BER=1 or MER=1)
  - 16 bit data access requested while WBNE=1
- For options change, PGSERR is set if:
  - option change is started while write buffer is waiting for next data: OPTSTRT=1 with NSWBNE=1
  - OPTSTRT set with DBNE=1.

When PGSERR flag is raised as consequence of failed write attempt (Flash programming), the current program operation is aborted and nothing is changed in the corresponding bank. The write data buffer is also invalidated.

Note: When PGSERR flag is raised, there is a risk that the last write operation performed by the application has been lost because of the above protection mechanism. Hence it is recommended to generate interrupts on PGSERR and verify in the interrupt handler if the last write operation has been successful by reading back the value in the flash memory.

The PGSERR flag also blocks any new program operation. This means that PGSERR must be cleared before starting a new program operation.

PGSERR flag is cleared by setting CLR_PGSERR bit to 1 in FLASH_NSCCR register.

If PGSERRIE bit in FLASH_NSCR register is set to 1, an interrupt is generated when PGSERR flag is raised. See *Section 7.9: FLASH interrupts* for details.

7.8.4 Strobe error (STRBERR)

When the application software writes several times to the same byte in the write buffer, the Flash interface sets the strobe error flag STRBERR (FLASH_NSSR) whatever the target bank of the write access.

When STRBERR flag is raised, the current program operation is aborted and the write buffer is invalidated.

STRBERR flag is cleared by setting CLR_STRBERR bit to 1 in FLASH_NSCCR register.

If STRBERRIE bit in FLASH_NSCR register is set to 1, an interrupt is generated when STRBERR flag is raised. See *Section 7.9: FLASH interrupts* for details.

7.8.5 Inconsistency error (INCERR)

When a programming inconsistency in access is detected, the Flash interface sets the inconsistency error flag INCERR in register FLASH_NSSR.

More specifically, INCERR flag is set when one of the following conditions is met:

- A write operation is attempted before completion of the previous write operation, for example:
  - The application software starts a write operation to fill the 128-bit write buffer, but sends a new write burst request to a different flash memory address before the buffer is full.
  - One master starts a write operation, but before the buffer is full, another master starts a new write operation to the same address or to a different address.

Note: INCERR flag must be cleared before starting a new write operation, otherwise a sequence error (PGSERR) is raised.

It is recommended to follow the sequence below to avoid losing data when an inconsistency error occurs:

1. Execute a handler routine when INCERR flag is raised.
2. Stop all write requests to embedded flash memory.
3. Clear INCERR bit.
4. Restart the write operations where they have been interrupted.

INCERR flag is cleared by setting CLR_INCERR bit to 1 in FLASH_NSCCR register.

If INCERRIE bit in FLASH_NSCR register is set to 1, an interrupt is generated when INCERR flag is raised (see *Section 7.9: FLASH interrupts* for details).

7.8.6 Error correction code error (ECCC/ECCD)

When a single-error correction is detected during a read, the Flash interface sets the single-error correction flag ECCC in FLASH_ECCCORR register.

When two ECC errors are detected during a read, the Flash interface sets the double error detection flag ECCD in FLASH_ECCDETR register.

When ECCC flag is raised, the corrected read data are returned. Hence the application can ignore the error and request new read operations. When ECCD flag is raised a NMI is generated. The software must invalidate the instruction cache (CACHEINV=1) in the NMI interrupt service routine when the ECCD flag is set. This NMI can be masked in SBS registers *SBS flift ECC NMI mask register (SBS_ECCNMIR)* for data access (OTP, data area, RO data).

When ECCC or ECCD flag is raised, the address of the Flash word that generated the error is saved in the FLASH_ECCCORR (FLASH_ECCDETR) register. If the address corresponds to a read-only area or to an OTP area, the OTP_ECC bit is also set to 1 in the FLASH_ECCCORRR (FLASH_ECCDETR) register. This register is automatically cleared when the associated flag that generated the error is reset.

A BK_ECC flag indicates in which Flash bank the error occurred.

A SYSF_ECC flag indicates error detected in the system Flash area.

Table 42. Locating ECC failure

OTP_ECC	SYSF_ECC	BK_ECC	Flash area	ADDR_ECC **min**	ADDR_ECC **max**
0	0	0/1	User Flash	0x0000	0xFFFF
0	1	0/1	System Flash	0x0000	0x0FFF
1	0	0	OTP	0x0600	0x07FF

Note: In case of successive single correction or double detection errors, only the address corresponding to the first error is stored in FLASH_ECCCORR (FLASH_ECCDETR) *register.*

Note: It is mandatory to clear ECCC or ECCD flags before starting a new read operation.

Note: As the ECC interface is shared by the two banks, if the same error is registered simultaneously, a physical Bank1 error is registered. Both errors are registered if one bank reports ECCD and the other bank ECCC.

ECCC (respectively ECCD) flag is cleared by setting to 1 ECCC bit (respectively ECCD bit) in FLASH_ECCCORR (FLASH_ECCDETR) register.

If ECCC bit in FLASH_ECCCORR register is set to 1, an interrupt is generated when ECCC flag is raised. Only NMI is generated for the ECCD. See *Section 7.9: FLASH interrupts* for details.

7.8.7 Option byte change error (OPTCHANGEERR)

When the Flash interface finds an error during an option change operation, it aborts the operation and sets the option byte change error flag OPTCHANGEERR in FLASH_NSSR register.

The error is raised after OPTSTRT bit set if:

- Any forbidden PRODUCT_STATE transition (*Table 37*).
- SWAP_BANK is locked out by NSBOOT_LOCK.
- OB modification is attempted in wrong HDPL (selected PRODUCT_STATE transitions)
- Attempt to modify OB with invalid value not recognized by the specification (enumerated magic numbers only)
- Attempt to modify OB in product state where it is not allowed (usually a state open for debug is required for change).

Note: Exceptions and details provided in the OB description (see Section 7.4.8: Specific rules for modifying option bytes) and Table 27.

OPTCHANGEERR flag is cleared by setting CLR_OPTCHANGEERR bit to 1 in FLASH_CCR register.

If OPTCHANGEERRIE bit in FLASH_NSCR register is set to 1, an interrupt is generated when OPTCHANGEERR flag is raised (see *Section 7.9: FLASH interrupts* for details).

It is mandatory to clean the OPTCHANGEERR flag before starting a new operation (option change, erase or write).

7.8.8 Miscellaneous HardFault errors

The following events generate a bus error on the corresponding bus interface:

- On main AHB system bus for access targeting code and data with 9 bits ECC.
  - Access to invalid address (including data addresses forbidden for code use).
  - Fetching from HDP area with incorrect HDPL value.
  - Fetching from privilege area in unprivilege mode.
- On AHB configuration or system bus for accesses targeting OTP/RO (all addresses using 6-bits ECC):
  - wrong key input to FLASH_KEYR or FLASH_OPTKEYR.
  - 8-bit accesses to system AHB interface.
  - Wrong unlock sequence on a register.

7.9 FLASH interrupts

The Flash interface can generate a maskable interrupt to signal the following events on a given bank:

- Read and write errors (see *Section 7.8: FLASH error management*)
  - Single ECC error correction during read operation
  - Write inconsistency error
  - Bad programming sequence
  - Strobe error during write operations
  - option change operation error
- Security errors (see *Section 7.8: FLASH error management*)
  - Write protection error
- Miscellaneous events (described below)
  - End of programming

A NMI is raised on double ECC error detection during read operation.

The user can individually enable or disable Flash interface interrupt sources by changing the mask bits in the FLASH_CR, FLASH_ECCCORR and FLASH_ECCDETR register. Setting the appropriate mask bit to 1 enables the interrupt.

Note: Prior to writing, FLASH_CR register must be unlocked as explained in Section 7.5.5: Flash memory configuration protection.

Table 43 gives a summary of the available Flash interface interrupt features. As mentioned in the table below, some flags need to be cleared before a new operation is triggered.

Table 43. Flash interrupt request

Interrupt event Event flag	Error flag label in register	Enable control bit	Clear flag to resume operation
End of operation event	EOP	EOPIE	N/A
Write protection error	WRPERR	WRPERRIE	Yes
Programming sequence error	PGSERR	PGSERRIE	Yes
Strobe error	STRBERR	STRBERRIE	Yes
Inconsistency error	INCERR	INCERRIE	Yes
Option byte error	OPTCHANGEERR	-	Yes
ECC single error correction event	ECCC	ECCCIE	No
ECC double error detection event(1)	ECCD	Disabled	No

1. NMI.

The status of the individual maskable interrupt sources described in *Table 43* (except for option byte error and ECC) can be read from the FLASH_NSSR register. They can be cleared by setting to 1 the adequate bit in FLASH_NSCCR register.

Note: No unlocking mechanism is required to clear an interrupt.

End of operation event

Setting the end of operation interrupt enable bit (EOPIE) in the FLASH_NSCR register enables the generation of an interrupt at the end of an erase operation, a program operation or an option byte change.

Setting CLR_EOP bit to 1 in FLASH_NSCCR register clears EOP flag.

7.10 FLASH registers

Each register is assigned a offset address and a reset value. In case of registers representing option byte value, the reset value is determined by the OBL process. In case of success the reset value is loaded from OB. In case of OBL failure, a highly restrictive default value is set.

7.10.1 FLASH access control register (FLASH_ACR)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

For more details, refer to *Section 7.3.4: FLASH read operations* and *Section 7.3.5: FLASH program operations*.

Address offset: 0x000

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.	PRFTEN	Res.	Res.	WRHIGHFREQ[1:0]		LATENCY[3:0]			
							rw			rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **PRFTEN**: Prefetch enable. When bit value is modified, user must read back ACR register to be sure PRFTEN has been taken into account.
Bits used to control the prefetch.
0: prefetch disabled.
1: prefetch enabled when latency is at least one wait state.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **WRHIGHFREQ[1:0]**: Flash signal delay
These bits are used to control the delay between non-volatile memory signals during programming operations. Application software has to program them to the correct value depending on the embedded flash memory interface frequency. Please refer to *Table 20* for details.
Note: No check is performed to verify that the configuration is correct.
Two WRHIGHFREQ values can be selected for some frequencies.

Bits 3:0 **LATENCY[3:0]**: Read latency
These bits are used to control the number of wait states used during read operations on both non-volatile memory banks. The application software has to program them to the correct value depending on the embedded flash memory interface frequency and voltage conditions.
0000: zero wait state used to read a word from non-volatile memory
0001: one wait state used to read a word from non-volatile memory
0010: two wait states used to read a word from non-volatile memory
...
0111: seven wait states used to read a word from non-volatile memory
1111: 15 wait states used to read from non-volatile memory
Note: No check is performed by hardware to verify that the configuration is correct.

7.10.2 FLASH key register (FLASH_NSKEYR)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

FLASH_NSKEYR is a write-only register. The following values must be programmed consecutively to unlock FLASH_NSCR register and allow programming/erasing it. A wrong sequence locks the FLASH_NSCR register until next system reset.

1st key = 0x4567 0123

2nd key = 0xCDEF 89AB

Address offset: 0x004

Reset value: 0x0000 0000

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]**: Non-volatile memory configuration access unlock key

7.10.3 FLASH option key register (FLASH_OPTKEYR)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

FLASH_OPTKEYR is a write-only register. The following values must be programmed consecutively to unlock FLASH_OPTCR register. A wrong sequence locks FLASH_OPTCR register until next system reset.

1st key = 0x0819 2A3B

2nd key = 0x4C5D 6E7F

Address offset: 0x00C

Reset value: 0x0000 0000

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]**: FLASH option bytes control access unlock key

7.10.4 FLASH operation status register (FLASH_OPSR)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x0018

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CODE_OP[2:0]			Res.	Res.	Res.	Res.	OTP_OP	SYSF_OP	BK_OP	Res.	Res.	ADDR_OP[19:16]			
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
- 000: No Flash operation on going during previous reset
- 001: Single write operation interrupted
- 010: reserved
- 011: Sector erase operation interrupted
- 100: Bank erase operation interrupted
- 101: Mass erase operation interrupted
- 110: Option change operation interrupted
- 111: reserved

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **OTP_OP**: OTP operation interrupted
Indicates that reset interrupted an ongoing operation in OTP area.

Bit 23 **SYSF_OP**: Operation in system flash memory interrupted
Indicates that reset interrupted an ongoing operation in System Flash.

Bit 22 **BK_OP**: Interrupted operation bank
It indicates which bank was concerned by operation.

Bits 21:20 Reserved, must be kept at reset value.

Bits 19:0 **ADDR_OP[19:0]**: Interrupted operation address.

7.10.5 FLASH option control register (FLASH_OPTCR)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Access: No wait state when no flash memory operation is ongoing. The FLASH_OPTCR register is not accessible in write mode when the BSY bit is set. Any attempt to write to it while the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

Address offset: 0x01C

Reset value: 0xX000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWAP_BANK	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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTSTRT	OPTLOCK
														rw	rs

Bit 31 **SWAP_BANK**: Bank swapping option configuration bit

SWAP_BANK controls whether Bank1 and Bank2 are swapped or not. This bit is loaded with the SWAP_BANK bit of FLASH_OPTSR_CUR register only after reset or POR.

0: Bank1 and Bank2 not swapped

1: Bank1 and Bank2 swapped

Bits 30:2 Reserved, must be kept at reset value.

Bit 1 **OPTSTRT**: Option byte start change option configuration bit

OPTSTRT triggers an option byte change operation. The user can set OPTSTRT only when the OPTLOCK bit is cleared to 0. It's set only by Software and cleared when the option byte change is completed or an error occurs (PGSERR or OPTCHANGEERR). It's reseted at the same time as BSY bit.

The user application cannot modify any FLASH_XXX_PRG flash memory register until the option change operation has been completed.

Before setting this bit, the user has to write the required values in the FLASH_XXX_PRG registers. The FLASH_XXX_PRG registers are locked until the option byte change operation has been executed in non-volatile memory.

Bit 0 **OPTLOCK**: FLASH_OPTCR lock option configuration bit

The OPTLOCK bit locks the FLASH_OPTCR register as well as all _PRG registers. The correct write sequence to FLASH_OPTKEYR register unlocks this bit. If a wrong sequence is executed, or the unlock sequence to FLASH_OPTKEYR is performed twice, this bit remains locked until next system reset.

It is possible to set OPTLOCK by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When OPTLOCK changes from 0 to 1, the others bits of FLASH_OPTCR register do not change.

0: FLASH_OPTCR register unlocked

1: FLASH_OPTCR register locked.

7.10.6 FLASH status register (FLASH_NSSR)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x020

Reset value: 0x0000 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTCHANGEERR	Res.	Res.	INCERR	STRBERR	PGSERR	WRPERR	EOP
								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.	DBNE	Res.	WBNE	BSY
												r		r	r

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **OPTCHANGEERR**: Option byte change error flag

OPTCHANGEERR flag indicates that an error occurred during an option byte change operation. When OPTCHANGEERR is set to 1, the option byte change operation did not successfully complete. An interrupt is generated when this flag is raised if the OPTCHANGEERRIE bit of FLASH_NSCR register is set to 1.

Writing 1 to CLR_OPTCHANGEERR of register FLASH_CCR clears OPTCHANGEERR.

0: no option byte change errors occurred

1: one or more errors occurred during an option byte change operation.

Note: The OPTSTRT bit in FLASH_OPTCR cannot be set while OPTCHANGEERR is set.

Bits 22:21 Reserved, must be kept at reset value.

Bit 20 **INCERR:** inconsistency error flag

INCERR flag is raised when a inconsistency error occurs. An interrupt is generated if INCERRIE is set to 1. Writing 1 to CLR_INCERR bit in the FLASH_NSCCR register clears INCERR.

0: no inconsistency error occurs

1: a inconsistency error occurs

Bit 19 **STRBERR:** strobe error flag

STRBERR flag is raised when a strobe error occurs (when the master attempts to write several times the same byte in the write buffer). An interrupt is generated if the STRBERRIE bit is set to 1. Writing 1 to CLR_STRBERR bit in FLASH_NSCCR register clears STRBERR.

0: no strobe error occurred

1: a strobe error occurred

Bit 18 **PGSERR:** programming sequence error flag

PGSERR flag is raised when a sequence error occurs. An interrupt is generated if the PGSERRIE bit is set to 1. Writing 1 to CLR_PGSERR bit in FLASH_NSCCR register clears PGSERR.

0: no sequence error occurred

1: a sequence error occurred

Bit 17 **WRPERR**: write protection error flag

WRPERR flag is raised when a protection error occurs during a program operation. An interrupt is also generated if the WRPERRIE is set to 1. Writing 1 to CLR_WRPERR bit in FLASH_NSCCR register clears WRPERR.

0: no write protection error occurred

1: a write protection error occurred

Bit 16 **EOP**: end of operation flag

EOP flag is set when a operation (program/erase) completes. An interrupt is generated if the EOPIE is set to 1. It is not necessary to reset EOP before starting a new operation. EOP bit is cleared by writing 1 to CLR_EOP bit in FLASH_NSCCR register.

0: no operation completed

1: a operation completed

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **DBNE**: data buffer not empty flag

DBNE flag is set when the flash memory interface is processing 6-bits ECC data in dedicated buffer. This bit cannot be set to 0 by software. The hardware resets it once the buffer is free.

0: data buffer not used

1: data buffer used, wait

Bit 2 Reserved, must be kept at reset value.

Bit 1 **WBNE**: write buffer not empty flag

WBNE flag is set when the embedded flash memory is waiting for new data to complete the write buffer. In this state, the write buffer is not empty. WBNE is reset by hardware each time the write buffer is complete or the write buffer is emptied following one of the event below:

- the application software forces the write operation using FW bit in FLASH_NSCR
- the embedded flash memory detects an error that involves data loss

This bit cannot be reset by software writing 0 directly. To reset it, clear the write buffer by performing any of the above listed actions, or send the missing data.

0: write buffer empty or full

1: write buffer waiting data to complete

Bit 0 **BSY**: busy flag

BSY flag indicates that a flash memory is busy by an operation (write, erase, option byte change). It is set at the beginning of a flash memory operation and cleared when the operation finishes or an error occurs.

0: no programming, erase or option byte change operation being executed

1: programming, erase or option byte change operation being executed

7.10.7 FLASH control register (FLASH_NSCR)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register..

Access: No wait state when no flash memory operation is ongoing. The FLASH_NSCR register is not accessible in write mode when the BSY bit is set. Any attempt to write to it with the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

Address offset: 0x028

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTCHANGEERRIE	Res.	Res.	INCERRIE	STRBERRIE	PGSERRIE	WRPERRIE	EOPIE
rw								rw			rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MER	Res.	Res.	Res.	Res.	Res.	Res.	SNB[2:0]			STRT	FW	BER	SER	PG	LOCK
rw							rw	rw	rw	rs	rw	rw	rw	rw	rs

Bit 31 **BKSEL**: Bank selector bit

BKSEL can only be programmed when LOCK is cleared to 0. The bit selects physical bank, SWAP_BANK setting is ignored.

0: Bank1 is selected for Bank erase / sector erase / interrupt enable

1: Bank2 is selected for BER / SER

Bits 30:24 Reserved, must be kept at reset value.

Bit 23 **OPTCHANGEERRIE**: Option byte change error interrupt enable bit

OPTCHANGEERRIE bit controls if an interrupt has to be generated when an error occurs during an option byte change. This bit can be programmed only when LOCK bit is cleared to 0.

0: no interrupt is generated when an error occurs during an option byte change

1: an interrupt is generated when and error occurs during an option byte change.

Bits 22:21 Reserved, must be kept at reset value.

Bit 20 **INCERRIE**: inconsistency error interrupt enable bit

When INCERRIE bit is set to 1, an interrupt is generated when an inconsistency error occurs during a write operation. INCERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated when a inconsistency error occurs

1: interrupt generated when a inconsistency error occurs.

Bit 19 **STRBERRIE**: strobe error interrupt enable bit

When STRBERRIE bit is set to 1, an interrupt is generated when a strobe error occurs (the master programs several times the same byte in the write buffer) during a write operation. STRBERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated when a strobe error occurs

1: interrupt generated when strobe error occurs.

Bit 18 **PGSERRIE**: programming sequence error interrupt enable bit

When PGSERRIE bit is set to 1, an interrupt is generated when a sequence error occurs during a program operation. PGSERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated when a sequence error occurs

1: interrupt generated when sequence error occurs

Bit 17 **WRPERRIE**: write protection error interrupt enable bit

When WRPERRIE bit is set to 1, an interrupt is generated when a protection error occurs during a program operation. WRPERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated when a protection error occurs

1: interrupt generated when a protection error occurs

Bit 16 **EOPIE**: end of operation interrupt control bit

Setting EOPIE bit to 1 enables the generation of an interrupt at the end of a program or erase operation. EOPIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated at the end of operation.

1: interrupt enabled when at the end of operation

Bit 15 **MER**: Mass erase request

Setting MER bit to 1 requests a mass erase operation (user flash memory only). MER can be programmed only when LOCK is cleared to 0.

If BER or SER are both set, a PGSERR is raised.

0: mass erase not requested

1: mass erase requested

Error is triggered when a mass erase is required and some sectors are protected.

Bits 14:9 Reserved, must be kept at reset value.

Bits 8:6 **SNB[2:0]**: sector erase selection number

These bits are used to select the target sector for an erase operation (they are unused otherwise). SNB can be programmed only when LOCK is cleared to 0.

0x00: Sector 0 selected

0x01: Sector 1 selected

...

0x07: Sector 7 selected

Bit 5 **STRT**: erase start control bit

STRT bit is used to start a sector erase or a bank erase operation. STRT can be programmed only when LOCK is cleared to 0.

STRT is reset at the end of the operation or when an error occurs. It cannot be reseted by software.

Bit 4 **FW:** write forcing control bit

FW forces a write operation even if the write buffer is not full. In this case all bits not written are set to 1 by hardware. FW can be programmed only when LOCK is cleared to 0.

The embedded flash memory resets FW when the corresponding operation has been acknowledged.

Note: Using a force-write operation prevents the application from updating later the missing bits with something else than 1, because it is likely that it leads to permanent ECC error.

Write forcing is effective only if the write buffer is not empty (in particular, FW does not start several write operations when the force-write operations are performed consecutively).

Since there is just one write buffer, FW can force a write in bank1 or bank2.

Bit 3 **BER**: erase request

Setting BER bit to 1 requests a bank erase operation (user flash memory only). BER can be programmed only when LOCK is cleared to 0.

If MER and SER are also set, a PGSERR is raised.

0: bank erase not requested

1: bank erase requested

Note: Write protection error is triggered when a bank erase is required and some sectors are protected.

Bit 2 **SER**: sector erase request

Setting SER bit to 1 requests a sector erase. SER can be programmed only when LOCK is cleared to 0.

If MER and SER are also set, a PGSERR is raised.

0: sector erase not requested

1: sector erase requested

Bit 1 **PG**: programming control bit

PG can be programmed only when LOCK is cleared to 0.

PG allows programming in Bank1 and Bank2.

0: programming disabled

1: programming enabled

Bit 0 **LOCK**: configuration lock bit

This bit locks the FLASH_NSCR register. The correct write sequence to FLASH_NSKEYR register unlocks this bit. If a wrong sequence is executed, or if the unlock sequence to FLASH_NSKEYR is performed twice, this bit remains locked until the next system reset. LOCK can be set by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When LOCK changes from 0 to 1, the other bits of FLASH_NSCR register do not change.

0: FLASH_NSCR register unlocked

1: FLASH_NSCR register locked

7.10.8 FLASH clear control register (FLASH_NSCCR)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x030

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.	CLR_OPTCHANGEERR	Res.	Res.	CLR_INCERR	CLR_STRBERR	CLR_PGSERR	CLR_WRPERR	CLR_EOP
								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.	Res.	Res.	Res.	Res.

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **CLR_OPTCHANGEERR:** Clear the flag corresponding flag in FLASH_NSSR by writing this bit.

Bits 22:21 Reserved, must be kept at reset value.

Bit 20 **CLR_INCERR**: INCERR flag clear bit

Setting this bit to 1 resets to 0 INCERR flag in FLASH_NSSR register.

Bit 19 **CLR_STRBERR**: STRBERR flag clear bit

Setting this bit to 1 resets to 0 STRBERR flag in FLASH_NSSR register.

Bit 18 **CLR_PGSERR**: PGSERR flag clear bit

Setting this bit to 1 resets to 0 PGSERR flag in FLASH_NSSR register.

Bit 17 **CLR_WRPERR**: WRPERR flag clear bit
Setting this bit to 1 resets to 0 WRPERR flag in FLASH_NSSR register.

Bit 16 **CLR_EOP**: EOP flag clear bit
Setting this bit to 1 resets to 0 EOP flag in FLASH_NSSR register.

Bits 15:0 Reserved, must be kept at reset value.

7.10.9 FLASH privilege configuration register (FLASH_PRIVCFGR)

Address offset: 0x03C

Reset value: 0x0000 0000

This register can be read by both privileged and unprivileged access. It can only be written by privilege 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.	Res.	Res.	Res.	Res.	Res.	Res.	NSPRIV	Res.
														rw	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **NSPRIV**: privilege attribute
0: access to registers is always granted
1: access to registers is denied in case of non privileged access.

Bit 0 Reserved, must be kept at reset value.

7.10.10 FLASH HDP extension register (FLASH_HDPEXTR)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

The register can only be modified in HDPL<=2.

The values in this register cannot be decremented. Attempt to write a lower value that current is ignored.

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.	HDP2_EXT[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.	HDP1_EXT[2:0]		
													rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **HDP2_EXT[2:0]:** HDP area extension in 8 Kbytes sectors in Bank2. Extension is added after the HDP2_END sector (included).

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:0 **HDP1_EXT[2:0]**: HDP area extension in 8 Kbytes sectors in Bank1. Extension is added after the HDP1_END sector (included).

7.10.11 FLASH option status register (FLASH_OPTSR_CUR)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Default value: 0x0030 5CD8 (value in case of double ECC issue during OBL).

This read-only register reflects the current values of corresponding option bits.

Address offset: 0x050

Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWAP_BANK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG_STDBY	IWDG_STOP	Res.	Res.	IO_VDDIO2_HSLV	IO_VDD_HSLV
r										r	r			r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRODUCT_STATE[7:0]								NRST_STDBY	NRST_STOP	Res.	WWDG_SW	IWDG_SW	BORH_EN	BOR_LEV[1:0]	
r	r	r	r	r	r	r	r	r	r		r	r	r	r	r

Bit 31 **SWAP_BANK**: Bank swapping option status bit
SWAP_BANK reflects whether Bank1 and Bank2 are swapped or not.
SWAP_BANK is loaded to SWAP_BANK of FLASH_OPTCR after a reset.
0: Bank1 and Bank2 not swapped
1: Bank1 and Bank2 swapped

Bits 30:22 Reserved, must be kept at reset value.

Bit 21 **IWDG_STDBY**: IWDG Standby mode freeze option status bit
When set to 0 the independent watchdog IWDG is frozen in system Standby mode.
0: Independent watchdog frozen in Standby mode
1: Independent watchdog keep running in Standby mode.

Bit 20 **IWDG_STOP**: IWDG Stop mode freeze option status bit
When set to 0 the independent watchdog IWDG is frozen in system Stop mode.
0: Independent watchdog frozen in system Stop mode
1: Independent watchdog keep running in system Stop mode.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **IO_VDDIO2_HSLV**: High-speed IO at low VDDIO2 voltage status bit. This bit can be set only with VDDIO2 below 2.7 V.
0: High-speed IO at low VDDIO2 voltage feature disabled (VDDIO2 can exceed 2.7 V)
1: High-speed IO at low VDDIO2 voltage feature enabled (VDDIO2 remains below 2.7 V)

Bit 16 **IO_VDD_HSLV**: High-speed IO at low VDD voltage status bit. This bit can be set only with VDD below 2.7 V.
0: High-speed IO at low VDD voltage feature disabled (VDD can exceed 2.7 V)
1: High-speed IO at low VDD voltage feature enabled (VDD remains below 2.7 V)

Bits 15:8 **PRODUCT_STATE[7:0]**: Life state code (based on Hamming 8,4). More information in *Section 7.5.8: Product state transitions*.

Bit 7 **NRST_STDBY**: Core domain Standby entry reset option status bit
0: a reset is generated when entering Standby mode on core domain
1: no reset generated when entering Standby mode on core domain.

Bit 6 **NRST_STOP**: Core domain Stop entry reset option status bit
0: a reset is generated when entering Stop mode on core domain
1: no reset generated when entering Stop mode on core domain.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **WWDG_SW**: WWDG control mode option status bit
0: WWDG watchdog is controlled by hardware
1: WWDG watchdog is controlled by software

Bit 3 **IWDG_SW**: IWDG control mode option status bit
0: IWDG watchdog is controlled by hardware
1: IWDG watchdog is controlled by software

Bit 2 **BORH_EN**: Brownout high enable status bit
0: brownout reset (BOR) disabled
1: brownout reset (BOR) enabled

Bits 1:0 **BOR_LEV[1:0]**: Brownout level option status bit
These bits reflects the power level that generates a system reset.
00 or 11: BOR Level 1, the threshold level is low (VBOR1 around 2.1 V)
01: BOR Level 2, the threshold level is medium (VBOR2 around 2.4 V)
10: BOR Level 3, the threshold level is high (VBOR3 around 2.7 V)

7.10.12 FLASH option status register (FLASH_OPTSR_PRG)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits. Values after reset reflects the current values of the corresponding option bits.

Address offset: 0x054

Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWAP_BANK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG_STDBY	IWDG_STOP	Res.	Res.	IO_VDDIO2_HSLV	IO_VDD_HSLV
rw										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRODUCT_STATE[7:0]								NRST_STDBY	NRST_STOP	Res.	WWDG_SW	IWDG_SW	BORH_EN	BOR_LEV[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bit 31 **SWAP_BANK**: Bank swapping option configuration bit

SWAP_BANK option bit is used to configure whether the Bank1 and Bank2 are swapped or not. This bit is loaded with the SWAP_BANK bit of FLASH_OPTSR_CUR register after a reset.

0: Bank1 and Bank2 not swapped

1: Bank1 and Bank2 swapped

Bits 30:22 Reserved, must be kept at reset value.

Bit 21 **IWDG_STDBY**: IWDG Standby mode freeze option configuration bit

When set the independent watchdog IWDG is frozen in system Standby mode.

0: Independent watchdog frozen in Standby mode

1: Independent watchdog keep running in Standby mode.

Bit 20 **IWDG_STOP**: IWDG Stop mode freeze option configuration bit

When set the independent watchdog IWDG is in system Stop mode.

0: Independent watchdog frozen in system Stop mode

1: Independent watchdog keep running in system Stop mode.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **IO_VDDIO2_HSLV**: High-speed IO at low VDDIO2 voltage configuration bit. This bit can be set only with VDDIO2 below 2.7 V.

0: High-speed IO at low VDDIO2 voltage feature disabled (VDDIO2 can exceed 2.7 V)

1: High-speed IO at low VDDIO2 voltage feature enabled (VDDIO2 remains below 2.7 V)

Bit 16 **IO_VDD_HSLV**: High-speed IO at low VDD voltage configuration bit. This bit can be set only with VDD below 2.7 V.

0: High-speed IO at low VDD voltage feature disabled (VDD can exceed 2.7 V)

1: High-speed IO at low VDD voltage feature enabled (VDD remains below 2.7 V)

Bits 15:8 **PRODUCT_STATE[7:0]**: Life state code (based on Hamming 8,4). More information in *Section 7.5.8: Product state transitions*.

Bit 7 **NRST_STDBY**: Core domain Standby entry reset option configuration bit
0: a reset is generated when entering Standby mode on core domain
1: no reset generated when entering Standby mode on core domain.

Bit 6 **NRST_STOP**: Core domain Stop entry reset option configuration bit
0: a reset is generated when entering Stop mode on core domain
1: no reset generated when entering Stop mode on core domain.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **WWDG_SW**: WWDG control mode option configuration bit
0: WWDG watchdog is controlled by hardware
1: WWDG watchdog is controlled by software

Bit 3 **IWDG_SW**: IWDG control mode option configuration bit
0: IWDG watchdog is controlled by hardware
1: IWDG watchdog is controlled by software

Bit 2 **BORH_EN**: Brownout high enable configuration bit
0: brownout reset (BOR) disabled
1: brownout reset (BOR) enabled

Bits 1:0 **BOR_LEV[1:0]**: Brownout level option configuration bit
These bits reflects the power level that generates a system reset.
00 or 11: BOR Level 1, the threshold level is low (VBOR1 around 2.1 V)
01: BOR Level 2, the threshold level is medium (VBOR2 around 2.4 V)
10: BOR Level 3, the threshold level is high (VBOR3 around 2.7 V)

7.10.13 FLASH option status register 2 (FLASH_OPTSR2_CUR)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Default value: 0xC300 0658

This read-only register reflects the current values of corresponding option bits.

Address offset: 0x070

Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	SRAM1_ECC	SRAM1_RST	Res.	Res.	SRAM2_ECC	Res.	BKPRAM_ECC	SRAM2_RST	Res.	Res.	Res.
					r	r			r		r	r			

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **SRAM1_ECC**: SRAM1 ECC detection and correction disable
0: SRAM1 ECC check enabled
1: SRAM1 ECC check disabled

Bit 9 **SRAM1_RST**: SRAM1 erase upon system reset
0: SRAM1 erased when a system reset occurs
1: SRAM1 not erased when a system reset occurs

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **SRAM2_ECC**: SRAM2 ECC detection and correction disable
0: SRAM2 ECC check enabled
1: SRAM2 ECC check disabled

Bit 5 Reserved, must be kept at reset value.

Bit 4 **BKPRAM_ECC**: Backup RAM ECC detection and correction disable
0: BKPRAM ECC check enabled
1: BKPRAM ECC check disabled

Bit 3 **SRAM2_RST**: SRAM2 erase when system reset
0: SRAM2 erased when a system reset occurs
1: SRAM2 not erased when a system reset occurs.

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

7.10.14 FLASH option status register 2 (FLASH_OPTSR2_PRG)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits. Values after reset reflects the current values of the corresponding option bits.

Address offset: 0x074

Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	SRAM1_ECC	SRAM1_RST	Res.	Res.	SRAM2_ECC	Res.	BKPRAM_ECC	SRAM2_RST	Res.	Res.	Res.
					rw	rw			rw		rw	rw			

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **SRAM1_ECC**: SRAM1 ECC detection and correction disable
0: SRAM1 ECC check enabled
1: SRAM1 ECC check disabled

Bit 9 **SRAM1_RST**: SRAM1 erase upon system reset
0: SRAM1 erased when a system reset occurs
1: SRAM1 not erased when a system reset occurs

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **SRAM2_ECC**: SRAM2 ECC detection and correction disable
0: SRAM2 ECC check enabled
1: SRAM2 ECC check disabled

Bit 5 Reserved, must be kept at reset value.

Bit 4 **BKPRAM_ECC**: Backup RAM ECC detection and correction disable
0: BKPRAM ECC check enabled
1: BKPRAM ECC check disabled

Bit 3 **SRAM2_RST**: SRAM2 erase when system reset
0: SRAM2 erased when a system reset occurs
1: SRAM2 not erased when a system reset occurs.

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

7.10.15 FLASH unique boot entry register (FLASH_NSBOOTR_CUR)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register reflects the current values of corresponding option bits.

Address offset: 0x080

Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSBOOTADD[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
NSBOOTADD[7:0]								NSBOOT_LOCK[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:8 **NSBOOTADD[23:0]**: unique boot entry address
These bits reflect the UBE address

Bits 7:0 **NSBOOT_LOCK[7:0]:** A field locking the values of SWAP_BANK, and NSBOOTADD settings.
0xC3: The SWAP_BANK and NSBOOTADD can still be modified following their individual rules.
0xB4: The NSBOOTADD and SWAP_BANK are frozen.

7.10.16 FLASH unique boot entry address (FLASH_NSBOOTR_PRG)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits.

Address offset: 0x084

Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSBOOTADD[23:8]															
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
NSBOOTADD[7:0]								NSBOOT_LOCK[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 **NSBOOTADD[23:0]**: Unique boot entry address
These bits allow configuring the BOOT address

Bits 7:0 **NSBOOT_LOCK[7:0]:** A field locking the values of SWAP_BANK, and NSBOOTADD settings.
0xC3: The SWAP_BANK and NSBOOTADD can still be modified following their individual rules.
0xB4: The NSBOOTADD and SWAP_BANK are frozen.

7.10.17 FLASH OTP block lock (FLASH_OTPBLR_CUR)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register reflects the current values of corresponding option bits.

Address offset: 0x090

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCKBL[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
LOCKBL[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **LOCKBL[31:0]**: OTP block lock

Block *n* corresponds to OTP 16-bit word 32 x *n* to 32 x *n* + 31.
LOCKBL[n] = 1 indicates that all OTP 16-bit words in OTP Block *n* are locked and attempt to program them results in WRPERR.
LOCKBL[n] = 0 indicates that all OTP 16-bit words in OTP Block *n* are not locked.
When one block is locked, it's not possible to remove the write protection.
Also if not locked, it is not possible to erase OTP words.

7.10.18 FLASH OTP block lock (FLASH_OTPBLR_PRG)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits.

Address offset: 0x094

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCKBL[31:16]															
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
LOCKBL[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **LOCKBL[31:0]**: OTP block lock

Block *n* corresponds to OTP 16-bit word 32 x *n* to 32 x *n* + 31.
LOCKBL[n] = 1 indicates that all OTP 16-bit words in OTP Block *n* are locked and attempt to program them results in WRPERR.
LOCKBL[n] = 0 indicates that all OTP 16-bit words in OTP Block *n* are not locked.
When one block is locked, it is not possible to remove the write protection.
LOCKBL bits can be set if the corresponding bit in FLASH_OTPBLR_CUR is cleared.

7.10.19 FLASH privilege register for bank 1 (FLASH_PRIVBB1R1)

This register is privileged. It can be read written only by a privileged access.

Address offset: 0x0C0

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.	PRIVBB1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRIVBB1[7:0]**: Privileged / unprivileged 8 Kbytes Flash Bank1 sector attribute (y = 0 to 7)
0: sectors y in bank 1 is non privileged
1: sector y in bank 1 is privileged

7.10.20 FLASH write sector protection for Bank1 (FLASH_WRP1R_CUR)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This read-only register reflects the current values of corresponding option bits.

Address offset: 0x0E8

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.	WRPSG1[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **WRPSG1[7:0]**: Bank1 sector protection option status byte
Setting WRPSG1 bits to 0 write protects the corresponding sectors in bank 1 (0: write protected; 1: not write protected)

7.10.21 FLASH write sector protection for Bank1 (FLASH_WRP1R_PRG)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits.

Address offset: 0x0EC

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.	WRPSG1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **WRPSG1[7:0]**: Bank1 sector protection option status byte
Setting WRPSG1 bits to 0 write protects the corresponding sectors in bank 1 (0: write protected; 1: not write protected)

7.10.22 FLASH HDP Bank1 register (FLASH_HDP1R_CUR)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is initially loaded with the values of corresponding option bits.

Address offset: 0x0F8

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.	HDP1_END[2:0]		
													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.	HDP1_STRT[2:0]		
													w	w	w

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **HDP1_END[2:0]:** HDPL barrier end set in number of 8 Kbytes sectors

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:0 **HDP1_STRT[2:0]:** HDPL barrier start set in number of 8 Kbytes sectors

7.10.23 FLASH HDP Bank1 register (FLASH_HDP1R_PRG)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Register is accessible only in HDPL0 and HDPL1. In HDPL=2 or 3 it is WI, RAZ.

This register is used to program values in corresponding option bits.

Address offset: 0x0FC

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.	HDP1_END[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.	Res.	Res.	Res.	Res.	HDP1_STRT[2:0]		
													r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **HDP1_END[2:0]:** Bank 1 HDPL barrier end set in number of 8 Kbytes sectors

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:0 **HDP1_STRT[2:0]:** Bank 1 HDPL barrier start set in number of 8 Kbytes sectors

7.10.24 FLASH ECC correction register (FLASH_ECCCORR)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ECCC	Res.	Res.	Res.	Res.	ECCCIE	OTP_ECC	SYSF_ECC	BK_ECC	Res.	Res.	Res.	Res.	Res.	Res.
	rw					rw	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 Reserved, must be kept at reset value.

Bit 30 **ECCC:** ECC correction set by hardware when single ECC error has been detected and corrected.

Cleared by writing 1.

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **ECCCIE**: ECC single correction error interrupt enable bit When ECCCIE bit is set to 1, an interrupt is generated when an ECC single correction error occurs during a read operation.

0: no interrupt generated when an ECC single correction error occurs
1: interrupt generated when an ECC single correction error occurs

Bit 24 **OTP_ECC**: OTP ECC error bit

This bit is set to 1 when one single ECC correction occurred during the last successful read operation from the read-only/ OTP area. The address of the ECC error is available in ADDR_ECC bitfield.

Bit 23 **SYSF_ECC**: ECC flag for corrected ECC error in system FLASH

It indicates if system flash memory is concerned by ECC error.

Bit 22 **BK_ECC**: ECC bank flag for corrected ECC error

It indicates which bank is concerned by ECC error

Bits 21:16 Reserved, must be kept at reset value.

Bits 15:0 **ADDR_ECC[15:0]**: ECC error address

When an ECC error occurs (for single correction) during a read operation, the ADDR_ECC contains the address that generated the error.

ADDR_ECC is reset when the flag error is reset.

The embedded flash memory programs the address in this register only when no ECC error flags are set. This means that only the first address that generated an ECC error is saved.

The address in ADDR_ECC is relative to the flash memory area where the error occurred (user flash memory, system flash memory, data area, read-only/OTP area).

7.10.25 FLASH ECC detection register (FLASH_ECCDETR)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECCD	Res.	Res.	Res.	Res.	Res.	Res.	OTP_ECC	SYSF_ECC	BK_ECC	Res.	Res.	Res.	Res.	Res.	Res.
rw							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 set by hardware when two ECC error has been detected.
When this bit is set, a NMI is generated.
Cleared by writing 1. Needs to be cleared in order to detect subsequent double ECC errors.

Bits 30:25 Reserved, must be kept at reset value.

Bit 24 **OTP_ECC**: OTP ECC error bit
This bit is set to 1 when double ECC detection occurred during the last read operation from the read-only/ OTP area. The address of the ECC error is available in ADDR_ECC bit field.

Bit 23 **SYSF_ECC**: ECC fail for double ECC error in system flash memory
It indicates if system flash memory is concerned by ECC error.

Bit 22 **BK_ECC**: ECC fail bank for double ECC Error
It indicates which bank is concerned by ECC error

Bits 21:16 Reserved, must be kept at reset value.

Bits 15:0 **ADDR_ECC[15:0]**: ECC error address
When an ECC error occurs (double detection) during a read operation, the ADDR_ECC contains the address that generated the error.
ADDR_ECC is reset when the flag error is reset.
The embedded flash memory programs the address in this register only when no ECC error flags are set. This means that only the first address that generated an double ECC error is saved.
The address in ADDR_ECC is relative to the flash memory area where the error occurred (user flash memory, system flash memory, data area, read-only/OTP area).

7.10.26 FLASH ECC data (FLASH_ECCDR)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x108

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_ECC[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_ECC[15:0]**: ECC error data

When an double detection ECC error occurs on special areas with 6-bit ECC on 16-bit of data (data area, read-only/OTP area), the failing data is read to this register.

By checking if it is possible to determine whether the failure was on a real data, or due to access to uninitialized memory.

7.10.27 FLASH privilege register for Bank2 (FLASH_PRIVBB2R1)

This register is privileged. It can be read written only by a privileged access.

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.	PRIVBB2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRIVBB2[7:0]**: Privileged / unprivileged 8 Kbytes Flash Bank2 sector attribute (y = 0 to 7)

0: sectors y in bank 2 is non privileged

1: sector y in bank 2 is privileged

7.10.28 FLASH write sector protection for Bank2 (FLASH_WRP2R_CUR)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This read-only register reflects the current values of corresponding option bits.

Address offset: 0x1E8

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.	WRPSG2[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **WRPSG2[7:0]**: Bank2 sector protection option status byte

Setting WRPSG2 bits to 0 write protects the corresponding sectors in bank 2 (0: write protected; 1: not write protected)

7.10.29 FLASH write sector protection for Bank2 (FLASH_WRP2R_PRG)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits.

Address offset: 0x1EC

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.	WRPSG2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **WRPSG2[7:0]**: Bank2 sector protection option status byte

Setting WRPSG2 bits to 0 write protects the corresponding sectors in bank 2 (0: write protected; 1: not write protected)

7.10.30 FLASH HDP Bank2 register (FLASH_HDP2R_CUR)

This register is initially loaded with the values of corresponding option bits.

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x1F8

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.	HDP2_END[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.	Res.	Res.	Res.	Res.	HDP2_STRT[2:0]		
													r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **HDP2_END[2:0]:** Bank 2 HDPL barrier end set in number of 8 Kbytes sectors

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:0 **HDP2_STRT[2:0]:** Bank 2 HDPL barrier start set in number of 8 Kbytes sectors

7.10.31 FLASH HDP Bank2 register (FLASH_HDP2R_PRG)

This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Register is accessible only in HDPL0 and HDPL1. In HDPL=2 or 3 it is WI, RAZ.

This register is used to program values in corresponding option bits.

Address offset: 0x1FC

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.	HDP2_END[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.	HDP2_STRT[2:0]		
													rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **HDP2_END[2:0]:** Bank 2 HDPL barrier end set in number of 8 Kbytes sectors

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:0 **HDP2_STRT[2:0]:** Bank 2 HDPL barrier start set in number of 8 Kbytes sectors

7.10.32 FLASH register map

Table 44. Register map and reset value table

Offset	Register name reset value	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.	Res.	Res.	Res.	Res.	Res.	Res.	PRFTEN	Res.	Res.	WRHIGHFREQ[1:0]		LATENCY[3:0]			
	0x0000 0013																								0			0	1	0	0	1	1
0x004	**FLASH_NSKEYR**	NSKEY[31:0]																															
	0x0000 0000	0	0	0	0	0	0	0	0	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_OPTKEYR**	OPTKEY																															
	0x0000 0000	0	0	0	0	0	0	0	0	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	**FLASH_OPSR**	CODE_OP[2:0]			Res.	Res.	Res.	Res.	OTP_OP	SYSF_OP	BK_OP	Res.	Res.	ADDR_OP[19:0]																			
	0xXXXX XXXX	X	X	X					X	X	X			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x01C	**FLASH_OPTCR**	SWAP_BANK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTSTRT	OPTLOCK
	0xX000 0001	0																														0	1
0x020	**FLASH_NSSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTCHANGERR	Res.	Res.	INCERR	STRBERR	PGSERR	WRPERR	EOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBNE	Res.	WBNE	BSY
	0x0000 000X									0			0	0	0	0	0													0		0	0
0x028	**FLASH_NSCR**	BKSEL.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTCHANGEERRIE	Res.	Res.	INCERRIE.	STRBERRIE	PGSERRIE	WRPERRIE	EOPIE	MER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNB[2:0]			STRT	FW	BER	SER	PG	LOCK
	0x0000 0001	0											0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	1
0x030	**FLASH_NSCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR_OPTCHANGEERR.	Res.	Res.	CLR_INCERR	CLR_STRBERR	CLR_PGSERR	CLR_WRPERR	CLR_EOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	0x0000 0000									0			0	0	0	0	0																

Table 44. Register map and reset value table (continued)

Offset	Register name reset value	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
0x03C	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	Res.
	0x0000 0000																															0	
0x048	FLASH_HDPEXTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_EXT[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_EXTT[2:0]		
	0x0000 0000														0	0	0														0	0	0
0x050	FLASH_OPTSR_ CUR	SWAP_BANK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG_STDBY	IWDG_STOP	Res.	Res.	IO_VDDIO2_HSLV	IO_VDD_HSLV	PRODUCT_STATE[7:0]								NRST_STDBY	NRST_STOP	Res.	WWDG_SW	IWDG_SW	BORH_EN	BOR_LEV[1:0]	
	0xXXXX XXXX	X										X	X			X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X
0x054	FLASH_OPTSR_ PRG	SWAP_BANK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG_STDBY	IWDG_STOP	Res.	Res.	IO_VDDIO2_HSLV	IO_VDD_HSLV	PRODUCT_STATE[7:0]								NRST_STDBY	NRST_STOP	Res.	WWDG_SW	IWDG_SW	BORH_EN	BOR_LEV[1:0]	
	0xXXXX XXXX	0										1	1					1	1	1	0	1	1	1	1	1	1		1	1	1	1	0
0x070	FLASH_OPTSR2_ CUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM1_ECC	SRAM1_RST	Res.	Res.	SRAM2_ECC	Res.	BKPRAM_ECC	SRAM2_RST	Res.	Res.
	0xXXXX XXXX																							X	X			X		X	X		
0x074	FLASH_OPTSR2_ PRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM1_ECC	SRAM1_RST	Res.	Res.	SRAM2_ECC	Res.	BKPRAM_ECC	SRAM2_RST	Res.	Res.
	0xXXXX XXXX																							X	X			X		X	X		
0x080	FLASH_NSBOOTR _CUR	NSBOOTADD[23:0]																								NSBOOT_LOCK[7:0]							
	0xXXXX XXXX	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x084	FLASH_NSBOOTR _PRG	NSBOOTADD[23:0]																								NSBOOT_LOCK[7:0]							
	0xXXXX XXXX	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x090	FLASH_OTPBLR_ CUR	LOCKBL[31:0]																															
	0xXXXX XXXX	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x094	FLASH_OTPBLR_ PRG	LOCKBL[31:0]																															
	0xXXXX XXXX	X	X	X	X	X	X	X	X	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 44. Register map and reset value table (continued)

Offset	Register name reset value	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
0x0C0	**FLASH_ PRIVBB1R1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVBB1[7:0]							
	0x0000 0000																									0	0	0	0	0	0	0	0
0x0E8	**FLASH_ WRP1RCUR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1[7:0]							
	0xXXXX XXXX																									X	X	X	X	X	X	X	X
0x0EC	**FLASH_ WRP1R_PRG**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1[7:0]							
	0xXXXX XXXX																									X	X	X	X	X	X	X	X
0x0F8	**FLASH_HDP1R_ CUR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_END[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_STRT[2:0]		
	0xXXXX XXXX														X	X	X														X	X	X
0x0FC	**FLASH_HDP1R_ PRG**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_END[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_STRT[2:0]		
	0xXXXX XXXX														X	X	X														X	X	X
0x100	**FLASH_ECCCORR**	Res.	ECCR	Res.	Res.	Res.	Res.	ECCCIE	OTP_ECC	SYSF_ECC	BK_ECC	Res.	Res.	Res.	Res.	Res.	Res.	ADDR_ECC[15:0]															
	0x0000 0000		0					0	0	0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104	**FLASH_ECCDETR**	ECCR	Res.	Res.	Res.	Res.	Res.	Res.	OTP_ECC	SYSF_ECC	BK_ECC	Res.	Res.	Res.	Res.	Res.	Res.	ADDR_ECC[15:0]															
	0x0000 0000	0							0	0	0																						
0x108	**FLASH_ECCDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATA_ECC[15:0]															
	0x0000 0000																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C0	**FLASH_ PRIVBB2R1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVBB2[7:0]							
	0x0000 0000																									0	0	0	0	0	0	0	0
0x1E8	**FLASH_WRP2R_ CUR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPSG2[7:0]							
	0xXXXX XXXX																									X	X	X	X	X	X	X	X
0x1EC	**FLASH_WRP2R_ PRG**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPSG2[7:0]							
	0xXXXX XXXX																									X	X	X	X	X	X	X	X

Table 44. Register map and reset value table (continued)

Offset	Register name reset value	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
0x1F8	**FLASH_HDP2R_CUR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_END[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_STRT[2:0]		
	0xXXXX XXXX														X	X	X														X	X	X
0x1FC	**FLASH_HDP2R_PRG**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_END[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_STRT[2:0]		
	0xXXXX XXXX														X	X	X														X	X	X

Refer to *Section 2.2* for the register boundary addresses.

8 Instruction cache (ICACHE)

8.1 ICACHE introduction

The instruction cache (ICACHE) is introduced on the C-AHB code bus of the Cortex®-M33 processor, to improve performance when fetching instructions and data from internal memories.

Some specific features like hit-under-miss, and critical-word-first refill policy, allow 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)
  - One 128-bit AHB master port: master1 port (output to Fast bus of the main AHB bus matrix)
  - 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
  - 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
- Replacement and refill
  - pLRU-t replacement policy (pseudo-least-recently-used, based on binary tree), algorithm with best complexity/performance balance
  - Critical-word-first refill policy, minimizing processor stalls
- 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
- Maintenance operation
  - Cache invalidate: full cache invalidation, fast command, noninterruptible

8.3 ICACHE implementation

Table 45. ICACHE features

Feature	ICACHE
Number of ways	2
Cache size	8 Kbytes
Cache line width	16 bytes
Number of regions to remap	0
Data size of AHB slave interface	32 bits
Data size of AHB fast master1 interface	128 bits
Data size of AHB slow master2 interface	0

8.4 ICACHE functional description

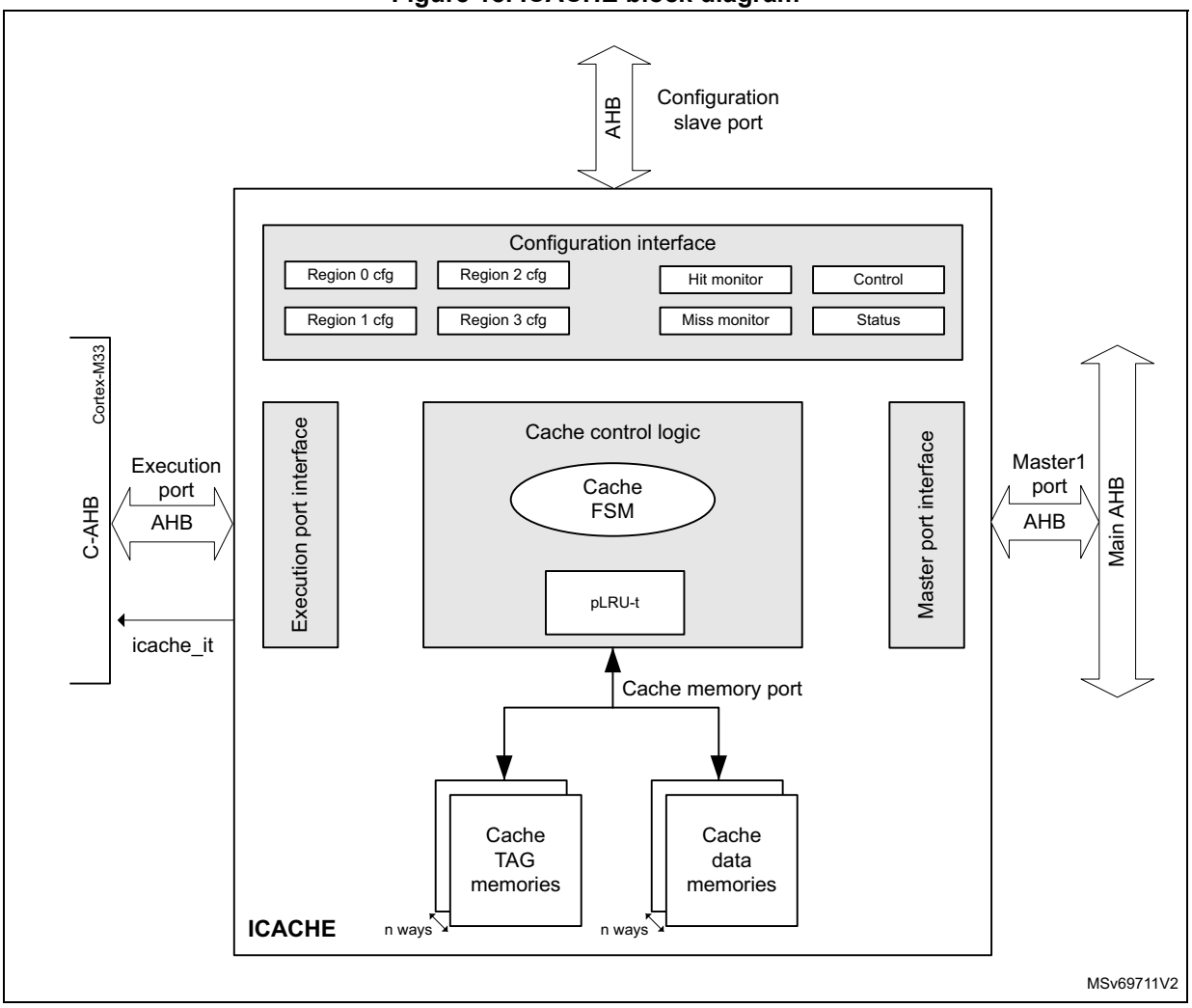
The purpose of the instruction cache is to cache instruction fetches or instruction memories loads, coming from the processor. As such, the ICACHE only manages cacheable read transactions and does not manage cacheable write transactions.

The noncacheable transactions (both read and write ones) bypass the ICACHE.

For the error management purpose, in case a write cacheable transaction is presented (this only happens 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 18. 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: slave input requests are forwarded to the master port.

8.4.3 ICACHE TAG memory

The ICACHE TAG memory contains:

- address tags that indicate 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 46* for B and W definitions):

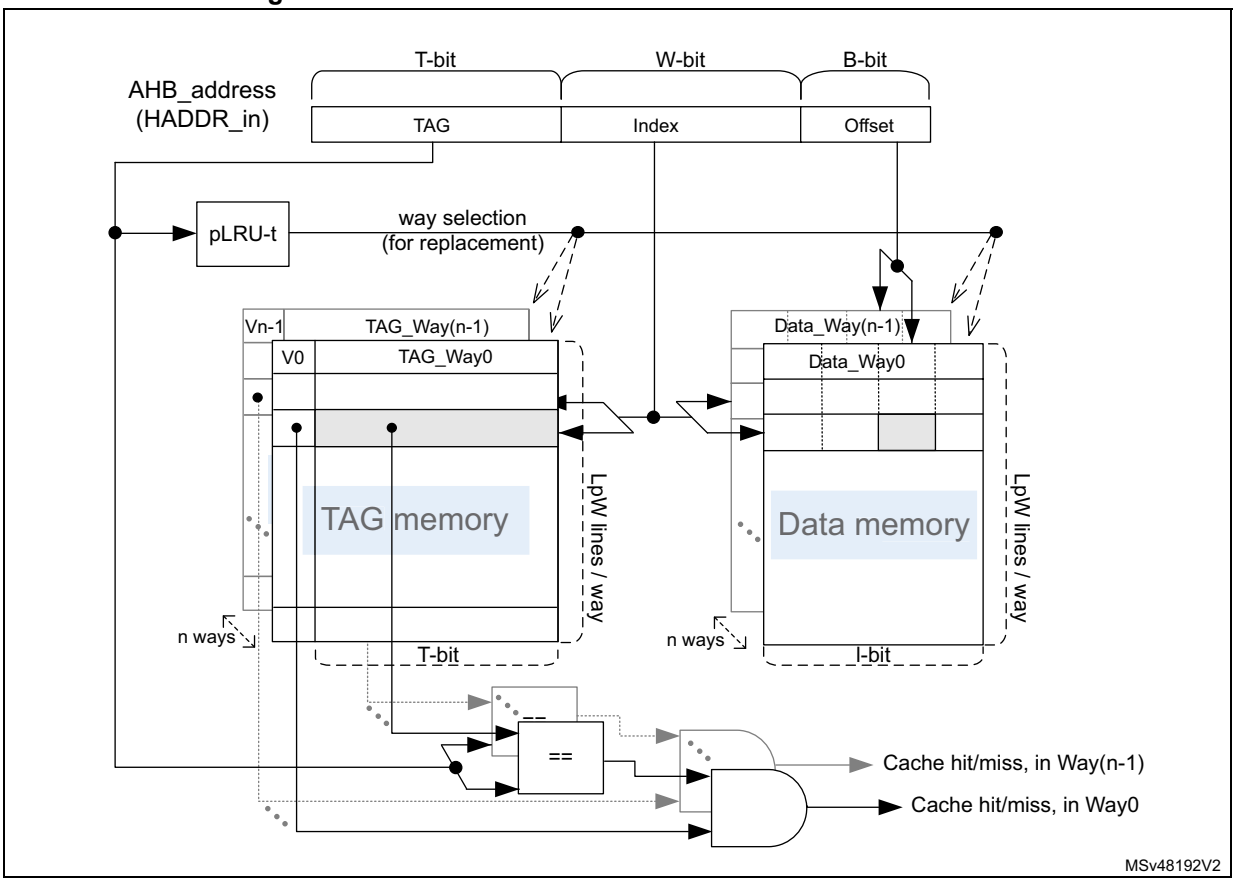
- HADDR_in[B-1:0]: address byte offset, indicates which byte to select inside a cache line.
- HADDR_in[B+W-1:B]: address way index, indicates which cache line to select inside each way.
- HADDR_in[31:B+W]: tag address, to be compared to the TAG memory address to check if the requested data is already available (meaning valid) inside the ICACHE.

The following table gives a summary of the ICACHE main parameters for TAG memory dimensioning. *Figure 19* shows the functional view of TAG and data memories, for an n-way set associative ICACHE.

Table 46. 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 (per way)	LpW = s / (n x L) lines / way	256 lines / way
Address byte offset size	B = log2(L) bit	4-bit
Address way index size	W = log2(LpW) bit	8-bit
TAG address size	T = (32 - W - B) bit	20-bit

Figure 19. 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 47. 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 = log2(L) bit	4-bit
Address way index size	W = log2(LpW) bit	9-bit
TAG address size	T = (32 - W - B) bit	19-bit

All cache operations (such as read, refill, 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 8.4.7*): only one way (the unique one) is possible for any data refill.

8.4.5 ICACHE enable

To activate the ICACHE, the EN bit must be set to 1 in ICACHE_CR.

When the ICACHE is disabled, it is bypassed and all transactions are copied from the slave port to the master port 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 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.

An incoming memory request to the ICACHE is defined as cacheable according to its AHB transaction memory lookup attribute, as shown in *Table 48*. This AHB attribute depends on the MPU (memory protection unit) programming for the addressed region.

Table 48. ICACHE cacheability for AHB transaction

AHB lookup attribute	Cacheability
1	Cacheable
0	Noncacheable

In the case of a noncacheable access (either a noncacheable read or a noncacheable write), the ICACHE is bypassed, meaning that the AHB transaction is propagated unchanged to the master output port.

The bypass 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 8.4.7*.

Cacheable memory regions are defined and programmed by the user in the MPU that is responsible for the generation of the AHB attribute signals for any transaction addressing a given region.

8.4.7 Cacheable accesses

When the ICACHE receives a cacheable transaction from the Cortex®-M33, the ICACHE 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).

  The AHB transaction attributes are also propagated to the main AHB bus matrix on the master port.
- 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 the data is 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 ever 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.

8.4.8 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.

8.4.9 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 8.4.7*).
- 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.

8.4.10 ICACHE boot

The ICACHE is disabled (EN = 0 in ICACHE_CR) at boot.

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.

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 19*). 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 49. 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 address decoding issues) from the master1 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.	MISSMRST	HITMRST	MISSMEN	HITMEN
												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.	WAYSEL	CACHEINV	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 register map

Table 50. 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
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

Refer to *Section 2.2* for the register boundary addresses.

9 Power control (PWR)

9.1 Introduction

The power controller manages all device power supplies and power modes transitions.

9.2 PWR main features

The power controller (PWR) main features are:

- Power supplies and supply domains
  - Core domain (V_{CORE})
  - V_{DD} domain
  - Backup domain (V_{SW})
  - Analog domain (V_{DDA})
  - V_{DDIO2} supply for nine I/Os (PA8, PA9, PA15, PB3:8)
- System supply voltage regulation
  - Linear voltage regulator (LDO)
- Power supply supervision
  - POR/PDR monitor
  - BOR monitor
  - PVD monitor
  - AVD monitor
  - Out of functional range temperature monitor
  - Out of functional range backup domain voltage monitor
- Power management
  - Operating modes
  - Voltage scaling control
  - Low-power modes
- V_{BAT} battery charging
- Privileged protection

9.3 PWR pins and internal signals

Table 51. PWR input/output pins

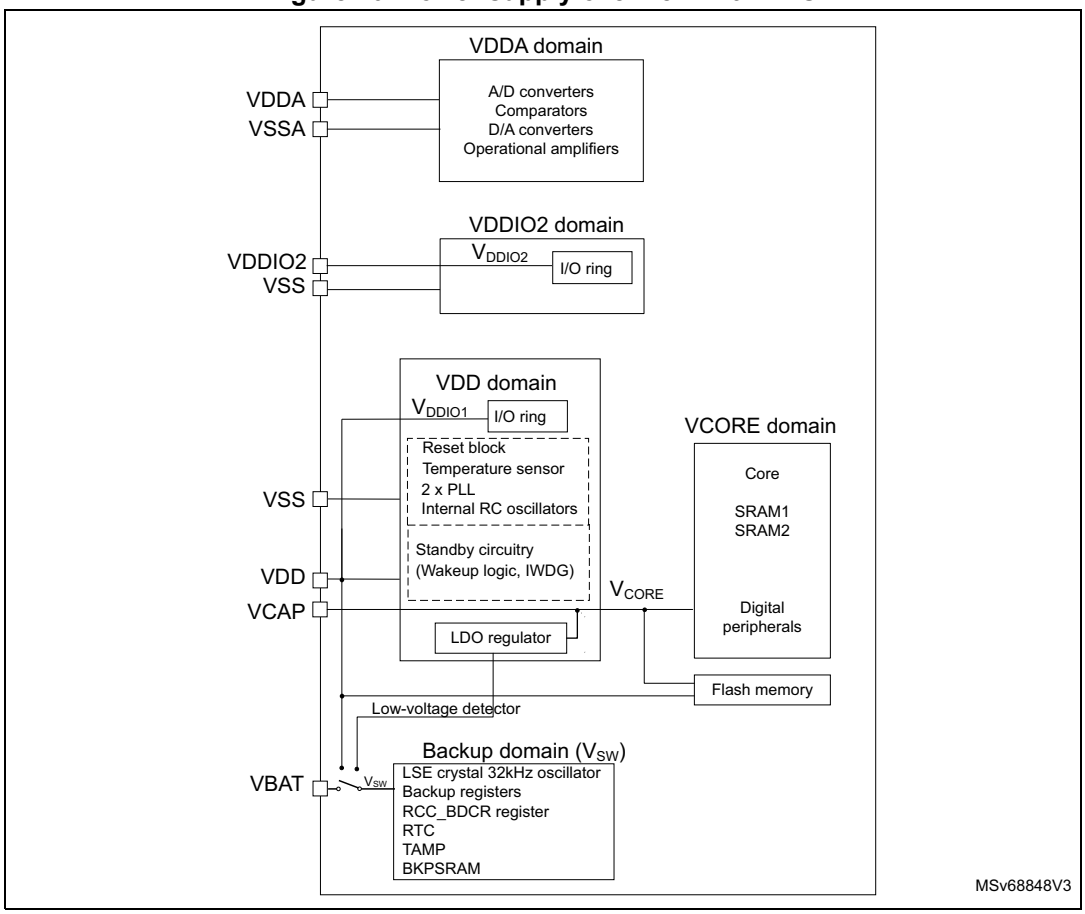
Pin name	Signal type	Description
VDD	Supply	Main supply
GND	Supply	Main ground
VDDA	Supply	Analog peripherals supply
VSSA	Supply	Analog peripherals ground
VDDIO2	Supply	Independent I/O supply
VCAP	Supply	Logic supply (V_{CORE})
VBAT	Supply	Backup domain supply
VREF+	Supply	ADC/DAC high reference voltage
VREF-	Supply	ADC/DAC low reference voltage
WKUPx (x = 1 to 5)	Input	Wakeup pins
CSLEEP	Output	MCU in Sleep mode
CSTOP	Output	CPU in Stop modes

Table 52. PWR internal input/output signals

Internal signal name	Signal type	Description
WKUPx (x = 1 to 5)	Input	Wakeup event source

9.4 PWR power supplies and supply domains

Figure 20. Power supply overview with LDO



9.4.1 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_{DDA} = 1.62 V (ADC, COMP) or 1.8 V (DAC) or 2.0 V (OPAMP) to 3.6 V

  V_{DDA} is the external analog power supply for A/D converters, D/A converters. The V_{DDA} voltage level is independent from the V_{DD} voltage and must preferably be connected to V_{DD} when these peripherals are not used.

- V_{DDIO2} = 1.08 V to 3.6 V

  V_{DDIO2} is the external power supply for nine I/Os (PA8, PA9, PA15, PB3:8). The V_{DDIO2} voltage level is independent from the V_{DD} voltage and must preferably be connected to V_{DD} when those pins are not used.

Note: VDDIO2 pin is available only on some packages, for more information refer to the device datasheet.

- V_{BAT} = 1.2 V to 3.6 V

  V_{BAT} is the power supply when V_{DD} is not present (through power switch) for RTC, external clock 32 kHz oscillator, backup registers, and optionally backup SRAM.
- V_{CAP}: V_{CORE} supply, which value depends on voltage scaling.
  It is configured through VOS bits in the PWR_VOSCR register and SVOS bits in the PWR_PMCR register.
- V_{REF-}, V_{REF+}

  V_{REF+} is the input reference voltage for ADCs and DACs.

  Independent VREF- and VREF+ pins are not available. They are bonded to VSSA and VDDA, respectively.

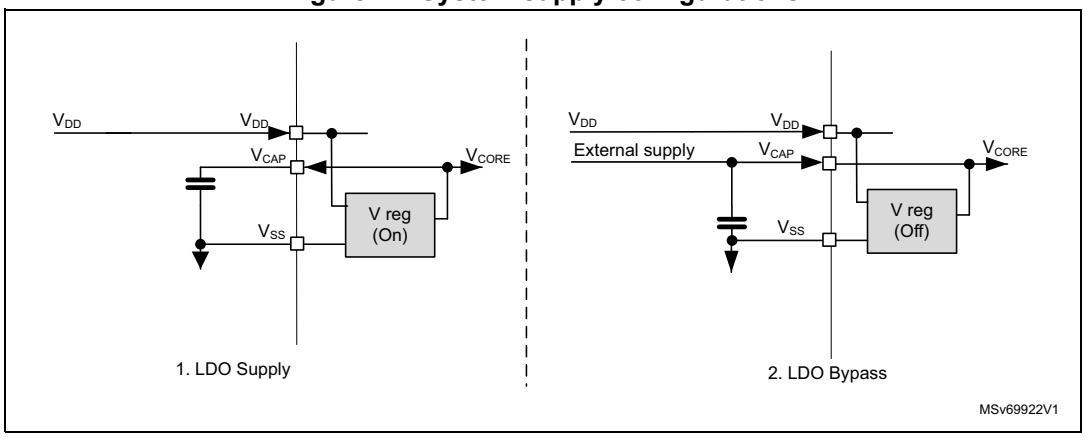
9.4.2 Internal regulator

The devices embed an LDO regulator that aims to provide the V_{CORE} supply for digital peripherals, SRAMs (except BKPSRAM) and embedded flash memory.

The LDO generates this voltage on VCAP (one or two pins depending on packages) with a total of the external capacitor of 4.7 µF typical. It can provide four different voltages (voltage scaling) and can operate in Stop modes.

The supply configurations shown in the figure below are supported for the VCORE core domain supply.

Figure 21. System supply configurations



Startup with VCORE provided from an external supply (Bypass)

When VCORE is supplied in Bypass mode, the VCORE voltage must first settle at a default level higher than 1.1 V. Due to the LDO default state after power-up (enabled by default), the external VCORE voltage must remain higher than 1.1 V until the LDO is disabled by software.

When the LDO is disabled, the external VCORE voltage can be adjusted according to the user application needs (refer to section *General operating conditions* of the datasheet for details on VCORE level versus the maximum operating frequency).

When operating in Bypass mode, the application must adjust VOS level in the power voltage scaling selection bits VOS[1:0] in the PWR_VOSCR register. VOS[1:0] must be set according to the external provided core voltage level and related performance.

To adjust the VOS level, the software must select sequentially the intermediate levels.

When increasing the performance:

a) The voltage scaling must be incremented first (for example, when changing from VOS3 to VOS0, the below levels must be selected in the VOS[1:0] bits: VOS2, VOS1, and finally VOS0).
b) The external voltage can be increased.
c) The system frequency can be increased.

When decreasing the performance:

a) The system frequency must be decreased.
b) The external voltage must be decreased.
c) Then the voltage scaling can be decremented (for example when changing from VOS1 to VOS3, bellow levels must be selected in the VOS[1:0] bits: VOS2 and then VOS3)

9.4.3 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_{DDIO2}) 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 to be discharged with different time constants during the power-down transient phase.

9.4.4 Independent analog peripherals supply

To improve ADC and DAC conversion accuracy and to extend the supply flexibility, the analog peripherals have an independent power supply that can be separately filtered and shielded from noise on the PCB:

- The analog peripherals voltage supply input is available on a separate VDDA pin.
- An isolated supply ground connection is provided on the VSSA pin.

The V_{DDA} supply voltage can be different from V_{DD}. The presence of V_{DDA} must be checked before enabling any of the analog peripherals supplied by V_{DDA} (A/D converter, D/A converter).

Power supply level monitoring, is available on VDDA via AVDO bit in the PWR_VMSR register (see *Section 9.6.4: Analog voltage detector (AVD)*).

When a single supply is used, V_{DDA} can be externally connected to V_{DD} through the external filtering circuit in order to ensure a noise-free V_{DDA} reference voltage.

Analog switch voltage booster

By setting BOOSTE and AVD_READY bits in the PWR_PMCR register, the application can reduce the total harmonic distortion (THD) of the I/O analog switches when the supply voltage is below 2.7V to guarantee the same performance as with the full voltage range.

To avoid current consumption due to the booster activation when V_{DDA} < 2.7 V and V_{DD} > 2.7 V, V_{DD} can be selected as supply voltage for the analog switches, by keeping AVD_READY bit as 0 in the PWR_PMCR register.

Table 53. Analog switched recommended configuration

VDDA	VDD	BOOSTE	AVD_READY	Analog switch supply voltage
≤ 2.7 V	≤ 2.7 V	1	1	Booster
	> 2.7 V	0	0	VDD
> 2.7 V	Don't care	0	1	VDDA

9.4.5 USB transceivers supply

The USB transceivers are supplied from the V_{DD} power supply pin. To operate properly, USB requires a V_{DD} range between 3.0 V and 3.6 V.

9.4.6 Independent I/O supply rail

Some I/Os (PA8, PA9, PA15, PB3:8) are supplied from a separate supply rail. The power supply for this rail can range from 1.08 V to 3.6 V and is provided externally through the VDDIO2 pin. The V_{DDIO2} voltage level is completely independent from V_{DD} or V_{DDA}. The VDDIO2 pin is available only for some packages. Refer to the pinout diagrams or tables in the related device datasheet(s) for the I/O list(s).

Power supply level monitoring (IO2VM) is available on VDDIO2 via the VDDIO2RDY bit in the PWR_VMSR register.

9.4.7 Backup domain

To retain the content of the backup registers and supply the RTC function when V_{DD} is turned off, the VBAT pin can be connected to an optional backup voltage supplied by a battery or by another source.

The VBAT pin powers the RTC unit, the LSE oscillator and the PC13 to PC15 I/Os, allowing the RTC to operate even when the main power supply is turned off. The backup SRAM is optionally powered by the VBAT pin when the BREN bit is set in the *PWR backup domain control register (PWR_BDCR)*. The switch to the V_{BAT} supply is controlled by the power-down reset embedded in the reset block.

Warning: **During $t_{RSTTEMPO}$ (temporization at V_{DD} startup) or after a PDR has been detected, the power switch between V_{BAT} and V_{DD} remains connected to V_{BAT}.**
During the startup phase, if V_{DD} is established in less than $t_{RSTTEMPO}$ (refer to the datasheet for the value of $t_{RSTTEMPO}$) and V_{DD} > V_{BAT} + 0.6 V, a current may be injected into V_{BAT} through an internal diode connected between V_{DD} and the

power switch (V_{BAT}).
If the power supply/battery connected to the VBAT pin cannot support this current injection, it is strongly recommended to connect an external low-drop diode between this power supply and the VBAT pin.

If no external battery is used in the application, it is recommended to connect V_{BAT} to V_{DD} supply and add a 100 nF ceramic decoupling capacitor on VBAT pin

When the backup domain is supplied by V_{BAT} (analog switch connected to V_{BAT}), the following pins are available:

- PC13, PC14, and PC15 that can be configured by RTC or LSE (refer to *Section 32.3: RTC functional description*)
- PC13, PA0 when they are configured by TAMP peripheral as tamper pins.

Note: Due to the fact that the analog switch can transfer only a limited amount of current, the use of GPIO PC13 to PC15 in output mode is restricted: the speed must be limited to 2 MHz with a maximum load of 30 pF and these I/Os must not be used as a current source (for example to drive a LED).

Backup domain access

After a system reset, the backup domain (RCC backup domain control register RCC_BDCR, RTC registers, TAMP registers, backup registers, and backup SRAM) is protected against possible unwanted write accesses. To enable access to the backup domain, set the DBP bit in the *PWR disable backup protection control register (PWR_DBPCR)* to enable access to the backup domain.

Backup RAM

The backup domain includes 2 Kbytes of backup RAM accessible in 32-, 16-, or 8-bit data mode. The backup RAM is supplied from the backup regulator in the backup domain. When the backup regulator is enabled through BREN bit in the PWR_BDCR, the backup RAM content is retained even in Standby, and/or VBAT mode (it can be considered as an internal EEPROM if VBAT is always present).

The backup regulator can be ON or OFF depending whether the application needs the backup RAM function in Standby or VBAT modes.

The backup RAM is read protected and mass erased when a tamper event occurs, this is to prevent confidential data such as cryptographic private key, from being accessed.

The backup RAM can be erased in the following ways:

- through the flash memory interface after a full product state regression
- after a tamper event
- after backup domain reset

V_{BAT} battery charging

When V_{DD} is present, it is possible to charge the external battery on VBAT through an internal resistance.

The V_{BAT} charging is done either through a 5 kΩ resistor or through a 1.5 kΩ resistor depending on the VBRS bit value in the PWR_BDCR register.

The battery charging is enabled by setting the VBE bit in the PWR_BDCR register. It is automatically disabled in V_{BAT} mode.

9.5 PWR system supply voltage regulation

9.5.1 LDO embedded regulator

The regulator is enabled on power-on reset. To supply the V_{CORE} from the external source that it is possible to disable the regulator by setting the BYPASS bit in the PWR_SCCR register.

The BYPASS bit is written once after power-on reset. Written-once mechanism locks the register and any further write access is ignored. The system must be power cycled before writing a new value.

When V_{CORE} is supplied from an external source the externally applied voltage level must be reflected in the VOSx bits in the PWR_VOSCR register.

The LDO regulator can provide four different voltages (voltage scaling) and can operate in Stop modes.

9.5.2 V_{CORE} supply versus reset, voltage scaling, and low-power modes

After reset, the V_{CORE} is in VOS3.

When exiting the Stop or Standby mode, the voltage range is the VOS3.

9.5.3 Embedded voltage regulator operating modes

There are three different power modes: Run, Stop and Standby modes.

Run mode

The voltage regulator (LDO) provides full power to the V_{CORE} domain (core, memories, and digital peripherals). The regulator output voltage can be scaled by software to different voltage levels (VOS0, VOS1, VOS2, and VOS3) that are configured through the VOS bits in the PWR voltage scaling control register (PWR_VOSCR).

The VOS voltage scaling allows optimization of the power consumption when the system is clocked below the maximum frequency. By default, VOS3 is selected after system reset.

VOSx bits can be changed on-the-fly to adapt to the required system performance.

Stop mode

The voltage regulator (LDO) supplies the V_{CORE} domain to retain the content of registers and internal memories. The regulator mode is selected through the SVOS bits in the PWR power mode control register (PWR_PMCR).

Stop mode power consumption can be further reduced using SVO4 (lower voltage level than VOS3) and even further with SVOS5.

Standby mode

The regulator (LDO) is OFF and the V_{CORE} domains are powered down. The content of the registers and memories are lost except for the Standby circuitry and the backup domain.

9.6 PWR power supply and temperature supervision

Power supply level monitoring is available on the following supplies:

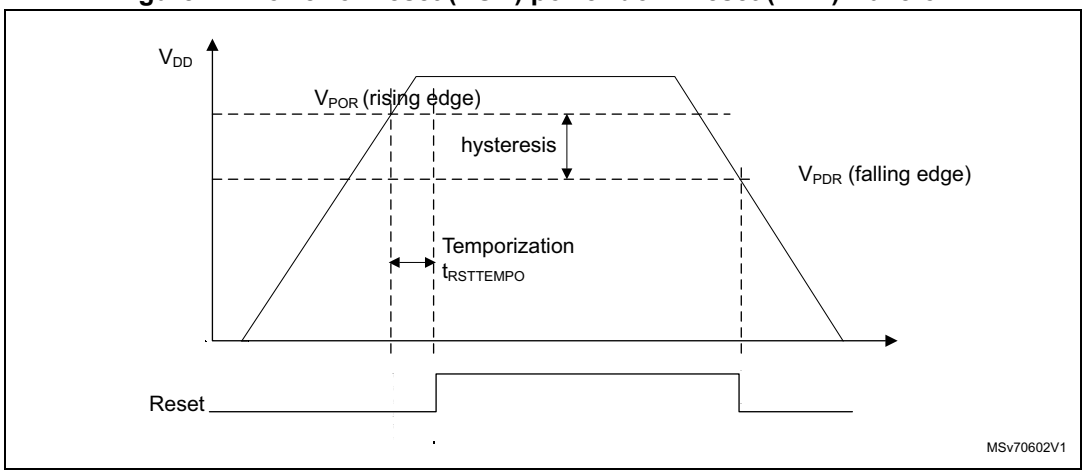
- VDD via POR/PDR (see *Section 9.6.1*), BOR (see *Section 9.6.2*), and PVD monitor (see *Section 9.6.3*)
- VDDA via AVD monitor (see *Section 9.6.4*)
- VDDIO2 via VDDIO2RDY bit (see *Section 9.6.5*)
- VBAT via VBAT threshold (see *Section 9.6.6*)
- Temperature monitoring (see *Section 9.6.7*)

9.6.1 Power-on reset (POR)/power-down reset (PDR)/

The system has an integrated POR/PDR circuitry that ensures proper startup operation.

The system remains in reset mode when VDD is below a specified VPOR threshold, without the need for an external reset circuit. Once the VDD supply level is above the VPOR threshold, the system is taken out of reset (see *Figure 22*). For more details concerning the power-on/power-down reset threshold, refer to the electrical characteristics section of the datasheets.

Figure 22. Power-on reset (POR)/power-down reset (PDR) waveform



1. For thresholds and hysteresis values, refer to the datasheets.

9.6.2 Brownout reset (BOR)

During power-on, the brownout reset (BOR) keeps the system under reset until the VDD supply voltage reaches the specified VBOR threshold.

The VBOR threshold is configured through system option bytes. By default, BOR is OFF and it can be enabled by setting the BORH_EN option bit.

The following programmable VBOR thresholds can be selected:

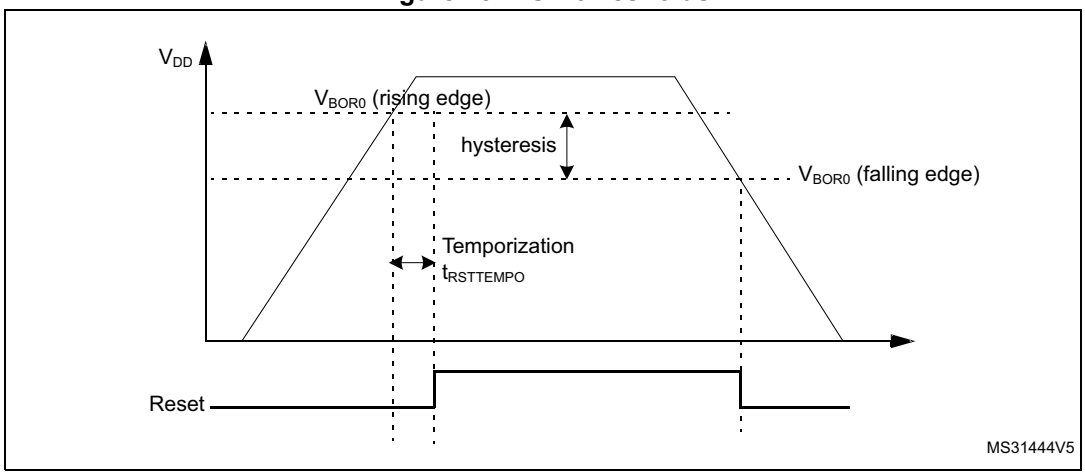
- BOR OFF (BORH_EN=0)
- BOR level 1 (VBOR1)
- BOR level 2 (VBOR2)
- BOR level 3 (VBOR3)

For more details on the brownout reset thresholds, refer to the section "*Electrical characteristics*" of the product datasheets.

A system reset is generated when the BOR is enabled and the V_{DD} supply voltage drops below the selected VBOR threshold.

BOR can be disabled by programming the BORH_EN option bit to 0. To disable the BOR function, V_{DD} must have been higher than the POR threshold to start the system option byte programming sequence. Once BOR is disabled, the power-down is then monitored by the PDR (see *Section 9.6.6*).

Figure 23. BOR thresholds



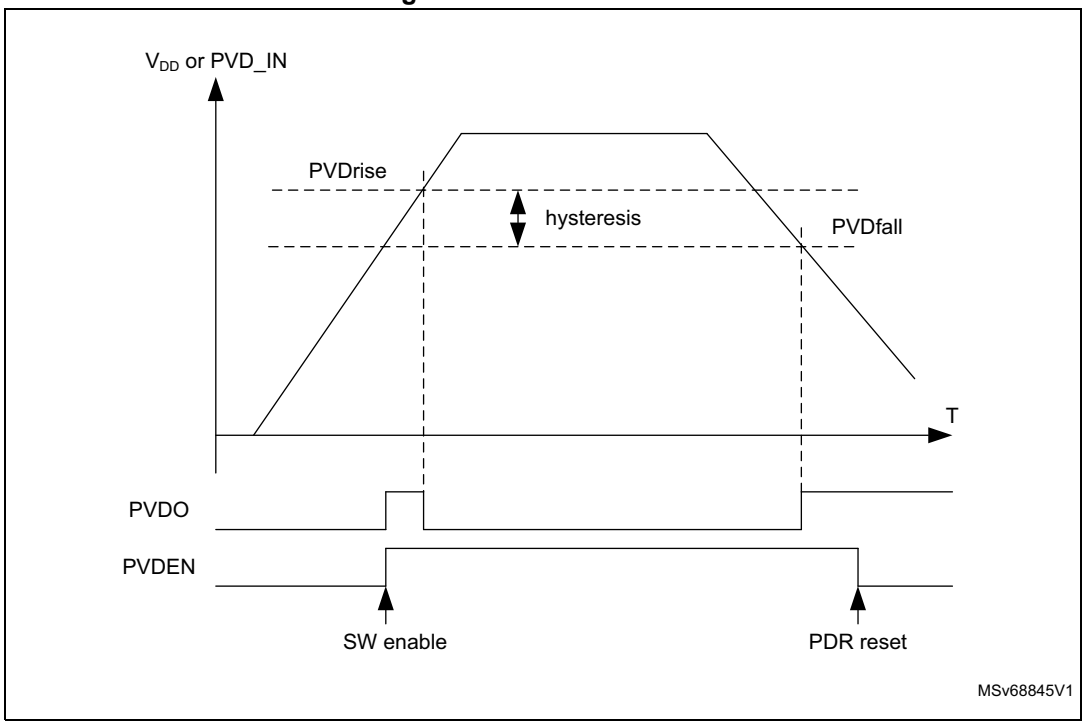
9.6.3 Programmable voltage detector (PVD)

The PVD can be used to monitor the V_{DD} power supply by comparing it to a threshold selected by the PLS[2:0] bits in the *PWR voltage monitor control register (PWR_VMCR)*. The PVD can also be used to monitor a voltage level on the PVD_IN pin. In this case PVD_IN voltage is compared to the internal VREFINT level.

The PVD is enabled by setting the PVDE bit in *PWR voltage monitor control register (PWR_VMCR)*.

A PVDO flag is available in the *PWR voltage monitor status register (PWR_VMSR)* to indicate if the V_{DD} or PVD_IN voltage is higher or lower than the PVD threshold. This event is internally connected to the EXTI and can generate an interrupt, provided it has been enabled through the EXTI registers. The rising/falling edge sensitivity of the EXTI line must be configured according to PVD output behavior, that is, if the EXTI line is configured to rising edge sensitivity, the interrupt is generated when the V_{DD} or PVD_IN voltage drops below the PVD threshold. As an example, the service routine could perform the emergency shutdown.

Figure 24. PVD thresholds



1. For thresholds and hysteresis values, refer to the datasheets.

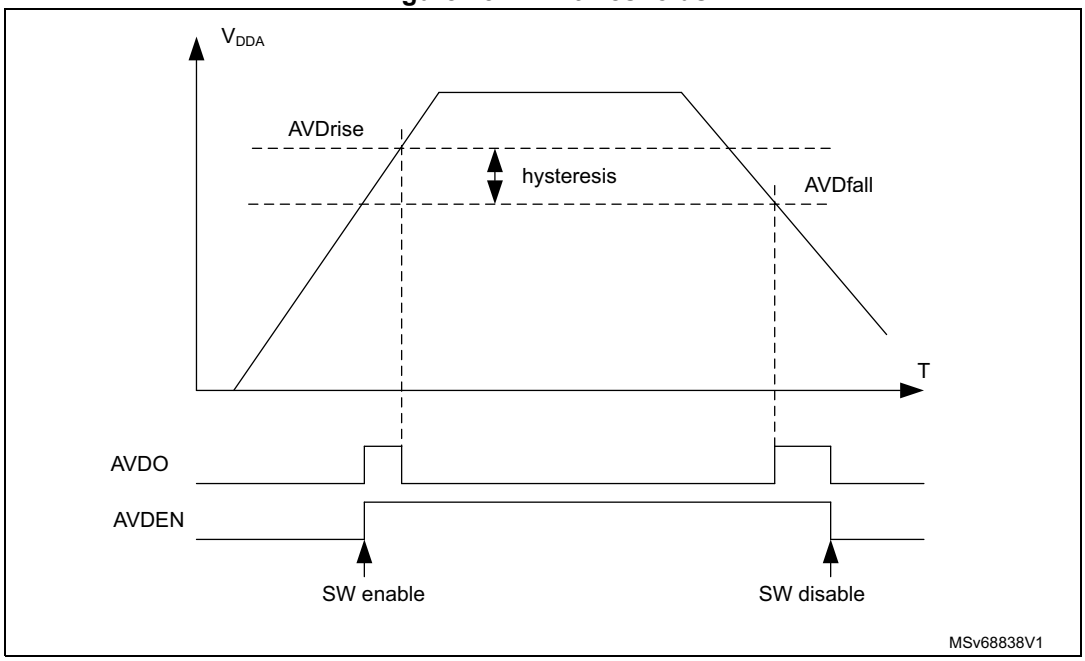
9.6.4 Analog voltage detector (AVD)

The AVD can be used to monitor the V_{DDA} supply by comparing it to a threshold selected by the ALS[1:0] bits in the *PWR voltage monitor control register (PWR_VMCR)*.

The AVD is enabled by setting the AVDEN bit in *PWR voltage monitor control register (PWR_VMCR)*.

An AVDO flag is available in the *PWR voltage monitor status register (PWR_VMSR)* to indicate whether V_{DDA} is higher or lower than the AVD threshold. This event is internally connected to the EXTI and can generate an interrupt if enabled through the EXTI registers. The AVDO interrupt can be generated when V_{DDA} drops below the AVD threshold and/or when V_{DDA} rises above the AVD threshold depending on EXTI rising/falling edge configuration. As an example the service routine could indicate when the V_{DDA} supply drops below a minimum level.

Figure 25. AVD thresholds



1. For thresholds and hysteresis values, refer to the datasheets.

9.6.5 VDDIO2 voltage monitor (IO2VM)

The IO2VM monitors the independent supply voltage VDDIO2 to ensure that the peripheral is in its functional supply range. The VDDIO2RDY flag (available in *Section 9.11.10: PWR voltage monitor status register (PWR_VMSR)*) indicates whether a valid VDDIO2 supply is present or not.

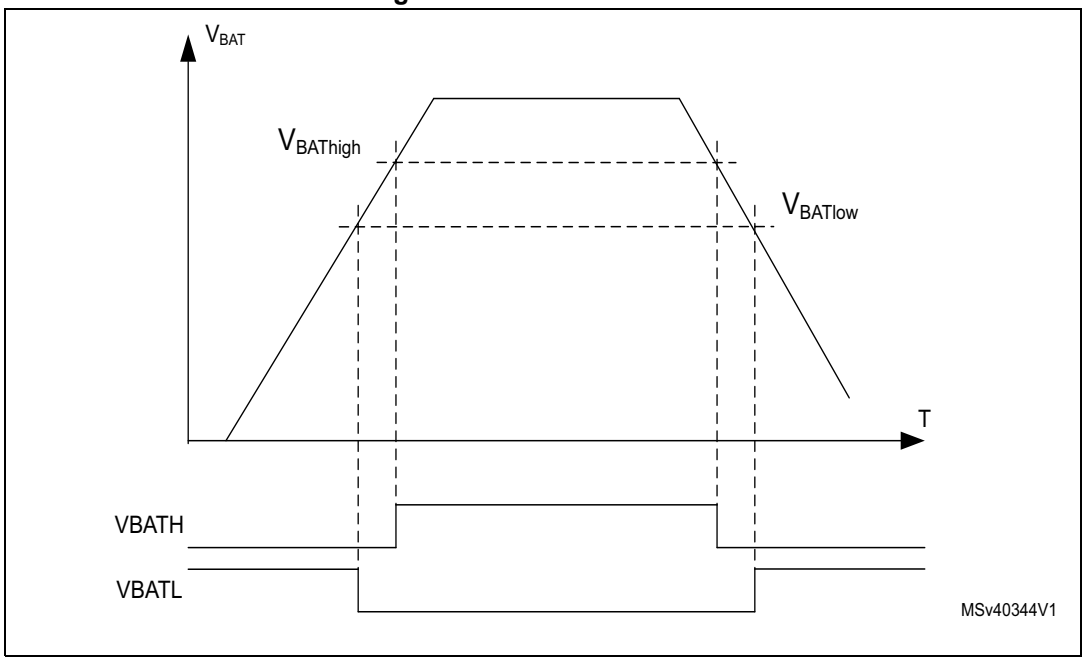
9.6.6 Backup domain voltage monitoring

In VBAT mode, the battery voltage supply (backup domain) can be monitored by comparing it with two threshold levels: VBAThigh and VBATlow. The VBAT supply monitoring can be enabled/disabled via MONEN bit in *PWR backup domain control register (PWR_BDCR)*. When it is enabled, the battery voltage thresholds increase power consumption.

If the backup domain voltage monitoring internal tamper is enabled in the TAMP peripheral (ITAMP1E = 1 in the TAMP_CR1 register), a tamper event is generated when the battery voltage is above the functional range.

Note: The backup domain voltage is VDD when present, VBAT otherwise.

Figure 26. VBAT thresholds



1. For thresholds and hysteresis values, refer to the datasheets.

9.6.7 Temperature monitoring

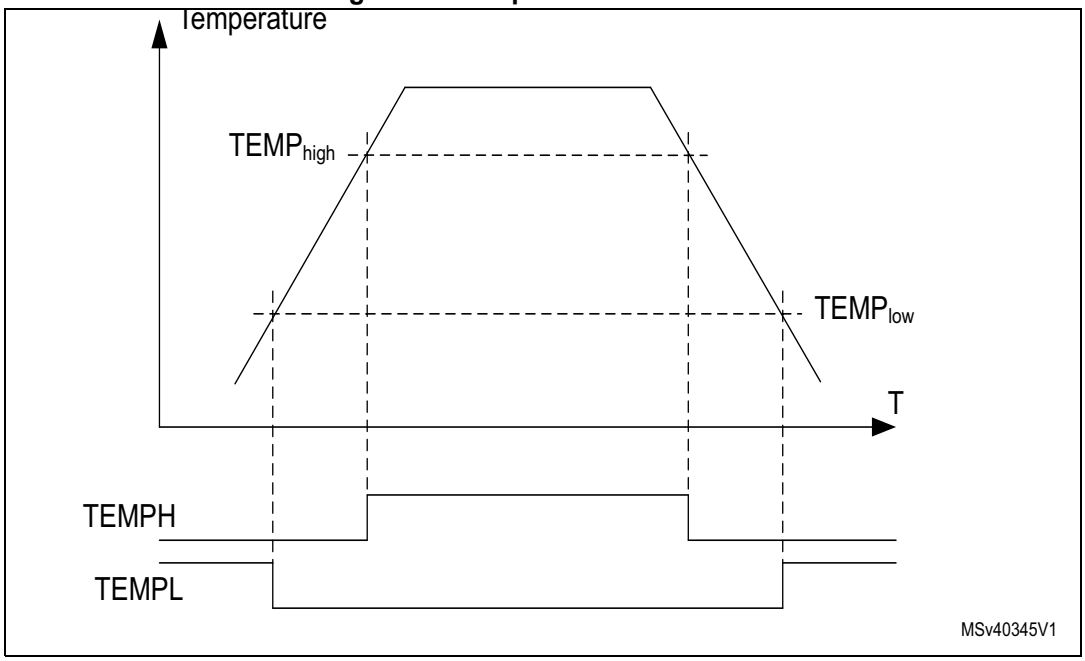
A dedicated temperature sensor cell is embedded in the power control. The junction temperature can be monitored by comparing it with two threshold levels, TEMPhigh and TEMPlow. TEMPH and TEMPL flags in the *PWR backup domain status register (PWR_BDSR)*, which indicates whether the device temperature is higher or lower than the threshold. The temperature monitoring can be enabled/disabled via the MONEN bit in *PWR backup domain control register (PWR_BDCR)*.

When enabled, the temperature thresholds increase power consumption. As an example the levels may be used to trigger a routine to perform temperature control tasks.

If the temperature monitoring internal tamper is enabled in the TAMP peripheral (ITAMP2E = 1 in the TAMP_CR1 register), a tamper event is generated when the temperature is above or below the functional range.

TEMPH and TEMPL wakeup interrupts are available on the RTC tamper signals (see *Section 33: Tamper and backup registers (TAMP)*).

Figure 27. Temperature thresholds



1. For thresholds and hysteresis values, refer to the datasheets.

9.7 PWR power management

9.7.1 Voltage scaling

The voltage regulator supporting voltage scaling with the following features:

- Run mode voltage scaling
  - VOS0: scale 0
  - VOS1: scale 1
  - VOS2: scale 2
  - VOS3: scale 3
- Stop mode voltage scaling
  - SVOS3: scale 3
  - SVOS4: scale 4
  - SVOS5: scale 5

For more details on voltage scaling values, refer to the product datasheets.

After reset, the system starts on the lowest Run mode voltage scaling (VOS3). The voltage scaling can then be changed on-the-fly by software by programming VOS bits in *PWR voltage scaling control register (PWR_VOSCR)* according to the required system performance. When exiting from the Stop mode or Standby mode, the Run mode voltage scaling is reset to the default VOS3 value.

Before entering Stop mode, the software should preselect the SVOS level in *PWR power mode control register (PWR_PMCR)*. The Stop mode voltage scaling for SVOS4 and SVOS5 also sets the voltage regulator in Low-power mode to further reduce power consumption.

9.7.2 Power management examples

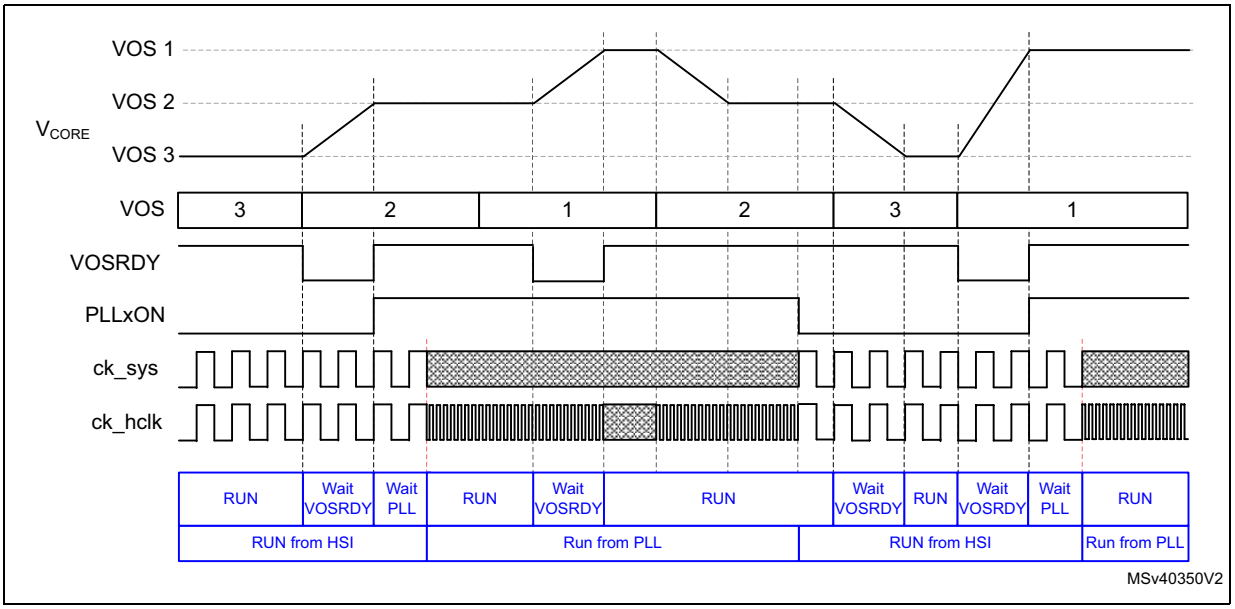
Example of VCORE voltage scaling behavior in Run mode.

Figure 28 illustrates the following system operation sequence example:

1. After reset, the system starts from HSI with VOS3.
2. The system performance is first increased to a medium-speed clock from the PLL with voltage scaling VOS2. To do this:
   a) Program the voltage scaling to VOS2.
   b) Once the VCORE supply has reached the required level indicated by VOSRDY, increase the clock frequency by enabling the PLL.
   c) Once the PLL is locked, switch the system clock.
3. The system performance is then increased to high-speed clock from the PLL with voltage scaling VOS1. To do this:
   a) Program the voltage scaling to VOS1.
   b) Once the VCORE supply has reached the required level indicated by VOSRDY, increase the clock frequency.
4. The system performance is then reduced to a medium-speed clock with voltage scaling VOS2. To do this:
   a) First decrease the system frequency.
   b) Then decrease the voltage scaling to VOS2.
5. The next step is to reduce the system performance to the HSI clock with voltage scaling VOS3. To do this:
   a) Switch the clock to HSI.
   b) Disable the PLL.
   c) Decrease the voltage scaling to VOS3.
6. The system performance can then be increased to high-speed clock from the PLL. To do this:
   a) Program the voltage scaling to VOS1.
   b) Once the VCORE supply has reached the required level indicated by VOSRDY, increase the clock frequency by enabling the PLL.
   c) Once the PLL is locked, switch the system clock.

When the system performance (clock frequency) is changed, VOS shall be set accordingly, otherwise the system might be unreliable.

Figure 28. Dynamic voltage scaling in Run mode



9.8 Power modes

By default, the microcontroller is in Run mode after a system or a power reset. Several low-power modes are available to save power when the CPU does 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 wakeup sources.

The device features these low-power modes:

- Sleep mode:

  CPU clock off, all peripherals including the Cortex-M33 core such as NVIC and SysTick can run and wake up the CPU when an interrupt or an event occurs. Refer to *Section 9.8.4: Sleep mode*.

- Stop mode:

  Stop mode achieves the lowest power consumption while retaining the content of SRAM and registers. All clocks in the core domain are stopped. The PLL, the HSE crystal oscillators, HSI (except if HSIKERON is set), HSI48, and CSI RC (except if CSIKERON is set) are disabled. The LSE or LSI is still running.

  The RTC can remain active (Stop mode with RTC, Stop mode without RTC).

  The system clock when exiting from Stop mode can be either HSI up to 64 MHz or CSI, depending on software configuration.

  Refer to *Section 9.8.5: Stop mode*.

- Standby mode:

  The Standby mode is used to achieve the lowest power consumption with BOR. The internal regulator is switched off so that the core domain is powered off. The PLL, the

HSI RC, HSI48, the CSI RC, and the HSE crystal oscillators are also switched off.

The RTC can remain active (Standby mode with RTC, Standby mode without RTC).

The Brownout reset (BOR) always remains active in Standby mode.

The state of the I/O (except I/Os used by Standby mode) during Standby mode can be retained.

After entering Standby mode, SRAMs and register contents are lost except for registers and backup SRAM in the backup domain and Standby circuitry.

The device exits Standby mode when an external reset (NRST pin), an IWDG reset, WKUP pin event (configurable rising or falling edge), an RTC event occurs (alarm, periodic wakeup, timestamp), or a tamper detection. The tamper detection can be raised either due to external pins or due to an internal failure detection.

The system clock after wakeup is HSI at 32 MHz.

Refer to *Section 9.8.6: Standby mode*.

In addition, 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 ranges.
- Gating the clocks to the APB and AHB peripherals when they are unused.

The table below shows the power modes overview.

Table 54. Low-power mode summary(1)

Mode name	Entry	Wakeup source(2)	Wakeup system clock	Effect on clocks	Voltage regulators
Sleep (Sleep-now or Sleep-on-exit)	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	VOS3, VOS2, VOS1 or VOS0
	WFE	Wakeup event			
Stop	LPMS = 0 + SLEEPDEEP bit + WFI or Return from ISR or WFE	Any EXTI line (configured in the EXTI registers) Specific peripherals events	CSI when STOPWUCK = 1 in RCC_CFGR HSI with the frequency before entering the Stop mode, at up to 64 MHz, when STOPWUCK = 0	All clocks OFF except LSI and LSE HSI or CSI can be enabled temporarily when requested by software	SVOS3, SVOS4, or SVOS5
Standby	LPMS = 1 + SLEEPDEEP bit + WFI or Return from ISR or WFE	WKUP pin edge, RTC event, external reset in NRST pin, IWDG reset	HSI clock at 32 MHz	All clocks OFF except LSI and LSE	OFF

1. Peripherals are able to wake-up the system from top mode this is only possible when SVOS3 is selected before entering stop mode. Refer to *Table 55*. Functionalities depend on the work mode.
2. Refer to the next table.

Table 55. Functionalities depending on the working mode(1)

Peripheral	Run	Sleep	Stop				Standby		VBAT
			Available	Wakeup capability			Available	Wakeup capability	
				SVOS3	SVOS4	SVOS5			
CPU	Y	-	-	-	-	-	-	-	-
Flash memory (128K Mbytes)	O	O	-(2)	-	-	-	-	-	-
SRAM1 (16 Kbytes)	Y	Y(3)	O(4)	-	-	-	-	-	-
SRAM2 (16 Kbytes)	Y(3)	Y(3)	O(4)	-	-	-	-	-	-
BKPSRAM	O	O	O	-	-	-	O	-	O
Backup registers	Y	Y	Y	-	-	-	Y	-	Y
Brownout reset (BOR)	Y	Y	Y	O	O	O	Y	O	-
Programmable voltage detector (PVD)	O	O	O	O	O	O	-	-	-
Analog voltage detector (AVD)	O	O	O	O	O	O	-	-	-
GPDMA	O	O	-	-	-	-	-	-	-
High-speed internal (HSI)	O	O	O	-	-	-	-	-	-
Oscillator HSI48	O	O	-	-	-	-	-	-	-
High-speed external (HSE)	O	O	-	-	-	-	-	-	-
Low-speed internal (LSI)	O	O	O	O	-	-	O	-	-
Low-speed external (LSE)	O	O	O	-	-	-	O	-	O
Low-power RC oscillator (CSI)	O	O	O	-	-	-	-	-	-
Clock security system (CSS) on HSE	O	O	-	-	-	-	-	-	-
Clock security system on LSE	O	O	O	O	O(5)	O(5)	O	O	O
Backup domain voltage monitoring, temperature monitoring	O	O	O	O	O(5)	O(5)	O	O	O
RTC/TAMP	O	O	O	O	O	O	O	O	O
Number of TAMP pins	2	2	2	2	2	2	2	2	2
USB	O	O	O	O	-	-	-	-	-
USARTx (x = 1,2,3)	O	O	O	O	-	-	-	-	-
Low-power UART (LPUART)	O	O	O	O	-	-	-	-	-
I2Cx (x = 1,2)	O	O	O	O	-	-	-	-	-
I3Cx (x = 1,2)	O	O	O	O	-	-	-	-	-
SPIx (x = 1,2,3)	O	O	O	O	-	-	-	-	-
FDCAN1	O	O	-	-	-	-	-	-	-
ADC1	O	O	-	-	-	-	-	-	-
DAC1 (2 converters)	O	O	O	-	-	-	-	-	-
COMP1	O	O	O	O	-	-	-	-	-
OPAMP1	O	O	-	-	-	-	-	-	-
Temperature sensor (DTS)	O	O	O	O	-	-	-	-	-

Table 55. Functionalities depending on the working mode(1) (continued)

Peripheral	Run	Sleep	Stop				Standby		VBAT
			Available	Wakeup capability			Available	Wakeup capability	
				SVOS3	SVOS4	SVOS5			
Timers (TIMx)	O	O	-	-	-	-	-	-	-
Low-power timer LPTIMx (x = 1,2)	O	O	O	O	-	-	-	-	-
Independent watchdog (IWDG)	O	O	O	O	O	O	O	O	-
Window watchdog (WWDG)	O	O	-	-	-	-	-	-	-
SysTick timer (SYSTICK)	O	O	-	-	-	-	-	-	-
Random number generator (RNG)	O	O	-	-	-	-	-	-	-
HASH accelerator	O	O	-	-	-	-	-	-	-
CRC calculation unit	O	O	-	-	-	-	-	-	-
GPIOs	O	O	-	-	-	-	-(6)	O(7)	-
EXTI	O	O	O	O	O	O	-	-	-

1. Y = yes (enable). O = optional (disable by default, can be enabled by software). - = not available.
2. The flash memory can be configured in low-power mode. By default, it is not in low-power mode during stop (SVOS3, SVOS4).
3. The SRAM clock can be gated ON or OFF independently. By default clock is enabled in RUN and sleep mode.
4. The SRAMs can be powered ON or OFF independently. By default, it is not in power OFF mode during stop.
5. Wakeup with internal tamper.
6. GPIOs state can be retained during Standby mode. By default GPIOs states are not retained.
7. 5-pin capable of wakeup from Standby mode PA0, PA2, PB7, PC1, and PC13.

Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop or Standby mode while the debug features are used. This is because the Cortex-M33 core is no longer clocked.

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: Debug and low-power modes*.

9.8.1 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 10: Reset and clock control (RCC)*.

9.8.2 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.

To further reduce the power consumption in Sleep mode, the peripheral clocks can be disabled before executing the WFI or WFE instructions.

The peripheral clock gating is controlled by the RCC_AHBxENR and RCC_APBxENR registers.

Disabling the peripherals clocks in Sleep mode can be performed automatically by resetting the corresponding bit in the RCC_AHBxLPENR and RCC_APBxLPENR registers.

9.8.3 Low-power modes

Entering into a low-power mode

The MCU enters in low-power modes by 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.

Exiting a low-power mode

The way the MCU exits the Sleep or Stop mode depends on the way the low-power mode was entered:

- If the WFI instruction or Return from ISR was 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 MCU exits the low-power mode as soon as an event occurs. The wakeup event can be generated either 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 MCU 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 MCU.

    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 MCU 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 MCU.
  - an event:
    Configuring an 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 may be necessary to clear the interrupt flag in the peripheral.

The MCU exits Standby mode through an external reset (NRST pin), an IWDG reset, a rising edge on one of the enabled WKUPx pins or a RTC/TAMP event (see *Figure 369: RTC block diagram*).

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 the an internal oscillator. The corresponding NVIC interrupt channel must be enabled to allow the interrupt to exit the device from Stop mode. It is not allowed to disable a peripheral interrupt by disabling only the NVIC channel while keeping the peripheral interrupt enable, as the device could remain in Stop mode with the clock ON.

9.8.4 Sleep mode

I/O states in Sleep mode

In Sleep mode, all I/O pins keep the same state as in Run mode.

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 clear (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 56. Sleep mode

Sleep mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP = 0 – No interrupt (for WFI) or event (for WFE) pending Refer to the Cortex-M33 system control register.
	On return from ISR while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 – No interrupt pending Refer to the Cortex-M33 system control register.
Mode exit	If WFI or Return from ISR was used for entry Interrupt (see *Table 97: STM32H503xx vector table*) If WFE was used for entry and SEVONPEND = 0: Wakeup event (see *Section 17.3: EXTI functional description*) If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC (see *Table 97: STM32H503xx vector table*) or Wakeup event (see *Section 17.3: EXTI functional description*)
Wakeup latency	None

9.8.5 Stop mode

The Stop mode is based on the Cortex-M33 Deepsleep mode combined with the peripheral clock gating. The voltage regulator is configured by SVOSx bits (the selected SVOS4 and SVOS5 levels add an additional startup delay when exiting from system Stop mode). In Stop mode, all clocks in the core domain are stopped. The PLL, HSI, HSI48, CSI, and HSE oscillators are disabled.

It is possible to keep the HSI or CSI clock enabled during Stop mode in order to be quickly available as kernel clock for peripherals.

All SRAMs and register contents are preserved, but the SRAMs can be totally switched off to further reduced consumption. The user can select which memory is discarded during Stop mode by means of xxSO bits in *PWR power mode control register (PWR_PMCR)*.

The BOR is always available in Stop mode.

I/O states in Stop mode

In the Stop mode, all I/O pins keep the same state as in the Run mode.

Entering the Stop mode

The MCU enters the Stop 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 57* for details on how to enter the Stop mode).

If the flash memory programming is ongoing, the Stop mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, the Stop mode entry is delayed until the APB access is finished.

In Stop 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 30.4: IWDG functional description*).
- The real-time clock (RTC) is configured by the RTCEN bit in the *RCC backup domain control register (RCC_BDCR)*.
- The internal RC oscillator LSI clock is configured by the LSION bit in RCC_BCDR.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in RCC_BCDR.

The AVD and the PVD can be used in Stop mode. If they are not needed, they must be disabled by software to save their power consumptions.

The ADC1, the DAC1 (two channels) and the temperature sensor can consume power during the Stop mode, unless they are disabled before entering this mode.

Exiting the Stop mode

The MCU exits Stop mode by enabling an EXTI interrupt or event depending on how the low-power mode was entered. Some peripherals are able to wakeup the system (refer to *Table 100: EXTI line connections*) from Stop mode, this is only possible when SVOS3 is selected before entering stop mode (refer to *Table 55: Functionalities depending on the*

working mode). GPIO pins configured to EXTI external interrupt or event are able to wakeup the system from stop mode in all voltage scales including SVOS4 and SVOS5.

Note: When wakeup from Stop with peripherals is needed, SVOS3 must be selected.

When exiting Stop mode by issuing an interrupt or a wakeup event, CSI is selected as system clock if the bit STOPWUCK is set in *RCC clock configuration register 1 (RCC_CFGR)*. The HSI oscillator is selected as system clock if STOPWUCK is cleared. The wakeup time is shorter when CSI is selected as the wakeup system clock. The HSI selection allows a wakeup at higher frequency (up to 64 MHz).

The MCU exits Stop mode by enabling an EXTI interrupt or event depending on how the low-power mode was entered.

When exiting the Stop mode, the MCU is in Run mode, VOS3.

Table 57. Stop mode

Stop 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 – LPMS = 000 in PWR_CR1
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending – LPMS = 0 in PWR_PMCR
	Note: To enter Stop mode, all EXTI line pending bits (in the EXTI rising edge pending register (EXTI_RPR1)), and the peripheral flags generating wakeup interrupts must be cleared. Otherwise, the Stop mode entry procedure is ignored and the program execution continues.
Mode exit	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 wakeup capability (see *Table 97: STM32H503xx vector table*). If WFE was used for entry and SEVONPEND = 0: - any EXTI line configured in event mode (see *Section 17.3: 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 wakeup capability (see *Table 97: STM32H503xx vector table*). - any EXTI line is configured in event mode (see *Section 17.3: EXTI functional description*)
	Note: All peripheral clocks must be enabled to allow this peripheral to generate a wakeup from Stop interrupt ([PERIPH]EN and [PERIPH]LPEN bits must be set in the RCC, and a functional independent clock must be selected).
Wakeup latency	Longest wakeup time between: HSI or CSI wakeup time and Flash wakeup time from Stop mode.

9.8.6 Standby mode

The lowest power mode in which the BOR is active is the Standby mode. It is based on the Cortex-M33 Deepsleep mode, with the voltage regulators disabled. The PLL, HSI, HSI48, CSI, and HSE oscillators are switched off.

The SRAMs and register contents are lost except for registers in the backup domain and Standby circuitry (see *Figure 20: Power supply overview with LDO*).

The BOR is always available in Standby mode.

The RTC outputs on PC13 are functional in Standby mode. PC14 and PC15 used for LSE are also functional. The five wakeup pins (WKUPx, x = 1 to 5) and four RTC tampers pins are available.

I/O states in Standby mode

In the Standby mode, the I/Os are by default in floating state. If the IORETEN bit in the PWR_IORETR register is set, the I/Os output state is retained. IO Retention mode is enabled for all IO except the IO support the standby functionality and JTAG IOs (PA13, PA14, PA15, and PB4). When entering into Standby mode, the state of the output is sampled, and the pull-up or pull-down resistor are set to maintain the IO output during Standby mode.

If the JTAGIORETEN bit in the PWR_IORETR register is set, the I/Os output state is retained. IO Retention mode is enabled for PA13, PA14, PA15, and PB4 (default JTAG pull-up/pull-down after wakeup are not enabled).

After wakeup from Standby mode, as long as IORETEN (or JTAGIORETEN for JTAG IOs) is set, the retained state (pull-up/pull-down) remains applied.

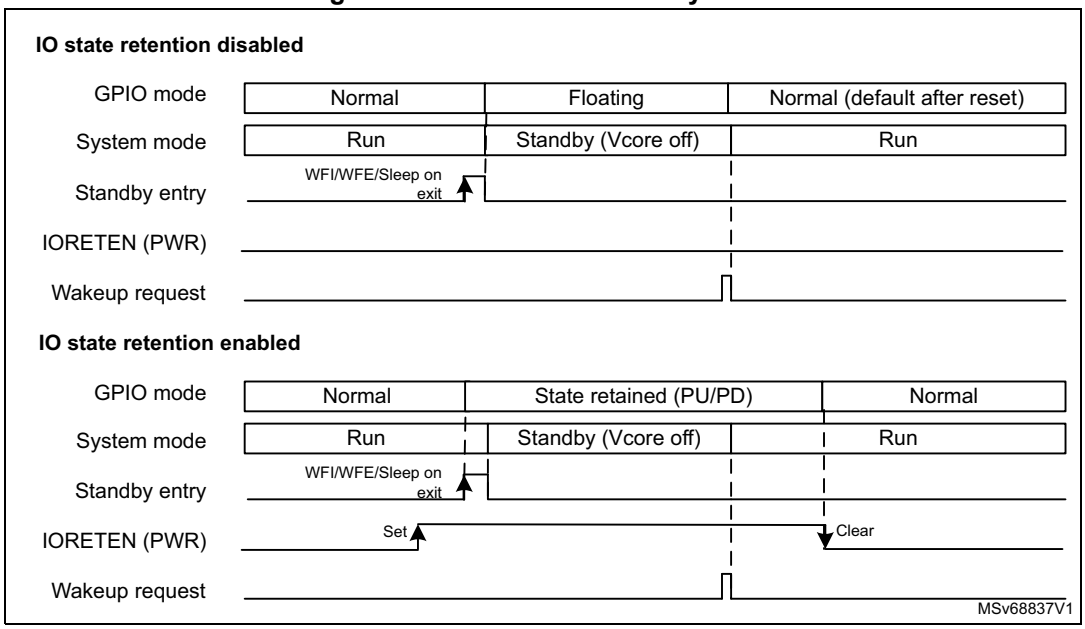
The GPIO pin state before Standby can be identified with the GPIO_IDR register (when both GPIO port clock and input buffer are enabled).

The application can release the IO state (clear the retained Pull-up/Pull-down) by clearing the IORETEN (or JTAGIORETEN for JTAG IOs) bit, this can be done before or after reconfiguring the GPIOs and related peripherals.

For example, this allows the application to configure the GPIO to a known state before releasing the retained state.

Note: Only the I/O digital output state is retained.

Figure 29. I/O states in Standby mode



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 58* for details on how to enter Standby mode).

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 30.4: IWDG functional description*).
- The real-time clock (RTC) is configured by the RTCEN bit in *RCC backup domain control register (RCC_BDCR)*.
- The internal RC oscillator LSI clock is configured by the LSION bit in RCC_BDCR.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in RCC_BDCR.
- The IOs retention is configured by the IORETEN bit in the PWR_IORETR register.

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_PMSR)* indicates that the MCU was in Standby mode. All registers are reset after wakeup from Standby except for *PWR backup domain control register (PWR_BDCR)* and *PWR I/O retention register (PWR_IORETR)* (see *Table 58* for more details on how to exit Standby mode).

When exiting Standby mode, I/Os output state that were retained during Standby through IORETEN bit, keep this configuration upon exiting Standby mode until the IORETEN bit in the PWR_IORETR register is cleared. Once IORETEN is cleared, the I/Os are either configured to their reset values or to the pull-up/pull-down state according to the GPIOx_PUPDR registers.

For I/Os, with a pull-up or pull-down predefined after reset (some JTAG/SWD I/Os), in case those pull-up or pull-down are different from the retained values during Standby, both a pull-down and pull-up are applied until IORETEN is cleared, releasing the retained value.

Also in case the GPIOx_PUPDR values programed after exiting from Standby are different from the retained values during Standby, both a pull-down and pull-up are applied until IORETEN is cleared, releasing the retained value.

Table 58. 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 – LPMS = 1 in PWR_PMCR – WUFx bits cleared in PWR_WUSR
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending – LPMS = 1 in PWR_PMCR – WUFx bits cleared in PWR_WUSR – RTC/TAMP flags corresponding to the chosen wakeup source, cleared
Mode exit	WKUPx pin edge, RTC event, external Reset in NRST pin, IWDG Reset, BOR reset
Wakeup latency	Reset phase

9.8.7 Power modes output pins

In order to help the debug, two signals are available as device pins alternate functions:

- **CSLEEP**

  When set, CSLEEP indicates that the system is in Sleep mode: WFI or WFE has been executed.

  When cleared, CSLEEP indicates that the system is in Run mode.

- **CSTOP**

  When set, CSTOP indicates that the system is in Stop mode, meaning that the following conditions are filled:

  – WFI or WFE has been executed with CPU SLEEPDEEP = 1.

  – No AHB/APB clock is running in the system.

  When cleared, CSTOP indicates that the system is not in Stop mode: AHB/APB clocks are running.

The table below explains the MCU power mode depending on these signals states.

Table 59. Power modes output states versus MCU power modes

CSLEEP	CSTOP	MCU power modes(1)
0	0	Run mode
1	0	Sleep mode
1	1	Stop mode

1. CSLEEP and CSTOP are generated in the core domain, therefore they are not driven in Standby mode.

9.9 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 privileged and unprivileged accesses.
The NSPRIV bit of PWR_PRIVCFGR can be written with privileged access only. This bit configures the privileged access of all PWR functions defined in *Table 60: PWR privilege configuration summary*.
When the NSPRIV bit is set in PWR_PRIVCFGR:

- The PWR privileged bits can be written only with privileged access.
- The PWR privileged bits can be read only with privileged access except the PWR_PRIVCFGR that can be read by privileged or unprivileged accesses.
- The VOSRDY bit in PWR_VOSSR, PWR_PMSR, PWR_VMSR, 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).

Table 60. PWR privilege configuration summary

PWR function	Register name	Privileged bits
Low-power mode	PWR_PMCR	All bits
Backup domain and VBAT mode configuration	PWR_BDCR	All bits
	PWR_DBPCR	All bits
I/Os retention configuration	PWR_IORETR	All bits
Wake-up pins WKUPx (x = 1 to 5)	PWR_WUCR	WUPENx
		WUPPx
		WUPPUPD
	PWR_WUSCR	CWUFx
Voltage scaling (VOS) configuration	PWR_VOSCR	VOS[1:0]
Voltage detection and monitoring	PWR_SCCR	All bits
	PWR_VMCR	All bits

9.10 PWR interrupts

The table below gives a summary of the interrupt sources and the way to control them.

Table 61. PWR interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep, Stop modes	Exit and Standby modes
PVD/AVD output	Programmable voltage detector through EXTI line 16	PVDO/AVDO	EXTI line 16 enabled	Write EXTI PIF16 = 1	Yes	No

9.11 PWR registers

The PWR registers can be accessed in word, half-word and byte format, unless otherwise specified.

9.11.1 PWR power mode control register (PWR_PMCR)

This register is protected against unprivileged access when NSPRIV = 1.

Address offset: 0x000

Reset value: 0x0000 000C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	SRAM1SO	SRAM2SO	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.	AVD_READY	BOOSTE	Res.	Res.	FLPS	Res.	CSSF	Res.	Res.	Res.	SVOS[1:0]		Res.	LPMS
		rw	rw			rw		rw				rw	rw		rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **SRAM1SO**: AHB SRAM1 shut-off in Stop mode
0: AHB RAM1 content is kept in Stop mode.
1: AHB RAM1 content is lost in Stop mode.

Bit 25 **SRAM2SO**: AHB SRAM2 shut-off in Stop mode.
0: AHB RAM2 content is kept in Stop mode.
1: AHB RAM2 content is lost in Stop mode.

Bits 24:14 Reserved, must be kept at reset value.

Bit 13 **AVD_READY**: analog voltage ready

This bit is only used when the analog switch boost needs to be enabled (see BOOSTE bit). It must be set by software when the expected V_{DDA} analog supply level is available. The correct analog supply level is indicated by the AVDO bit (PWR_VMSR register) after setting the AVDEN bit (PWR_VMCR register) and selecting the supply level to be monitored (ALS bits).

0: peripheral analog voltage V_{DDA} not ready (default)

1: peripheral analog voltage V_{DDA} ready.

Bit 12 **BOOSTE**: analog switch V_{BOOST} control

This bit enables the booster to guarantee the analog switch AC performance when the V_{DD} supply voltage is below 2.7 V (reduction of the total harmonic distortion to have the same switch performance over the full supply voltage range) The V_{DD} supply voltage can be monitored through the PVD and the PLS bits.

0: booster disabled (default)

1: booster enabled if analog voltage ready (AVD_READY = 1).

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **FLPS**: Flash memory low-power mode in Stop mode

This bit is used to obtain the best trade-off between low-power consumption and restart time when exiting from Stop mode.

When it is set, the Flash memory enters low-power mode when the system is in Stop mode.

0: Flash memory remains in normal mode when the system enters Stop mode (quick restart time).

1: Flash memory enters low-power mode when the system enters Stop mode (low-power consumption).

Note: When system enters stop mode with SVOS5 enabled, Flash memory is automatically forced in low-power mode.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **CSSF**: clear Standby and Stop flags (always read as 0)

This bit is cleared to 0 by hardware.

0: no effect

1: STOPF and SBF flags cleared.

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

Bits 3:2 **SVOS[1:0]**: system Stop mode voltage scaling selection

These bits control the V_{CORE} voltage level in system Stop mode, to obtain the best trade-off between power consumption and performance.

00: reserved

01: SVOS5 scale 5

10: SVOS4 scale 4

11: SVOS3 scale 3 (default).

Bit 1 Reserved, must be kept at reset value.

Bit 0 **LPMS**: low-power mode selection

This bit defines the Deepsleep mode.

0: Keeps Stop mode when entering DeepSleep.

1: Allows Standby mode when entering DeepSleep.

9.11.2 PWR status register (PWR_PMSR)

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.	SBF	STOPF	Res.	Res.	Res.	Res.	Res.
									r	r					

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SBF**: System standby flag

This bit is set by hardware and cleared only by a POR or by setting the CSSF bit.
0: system has not been in Standby mode.
1: system has been in Standby mode.

Bit 5 **STOPF**: Stop flag

This bit is set by hardware and cleared only by any reset or by setting the CSSF bit.
0: system has not been in Stop mode.
1: system has been in Stop mode.

Bits 4:0 Reserved, must be kept at reset value.

9.11.3 PWR voltage scaling control register (PWR_VOSCR)

These fields can be protected against unprivileged access depending on the PWR_PRIVCFGR register configuration.

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.	VOS[1:0]		Res.	Res.	Res.	Res.
										rw	rw				

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **VOS[1:0]**: voltage scaling selection according to performance

These bits control the V_{CORE} voltage level and allow to obtain the best trade-off between power consumption and performance:

- In bypass mode, these bits must also be set according to the external provided core voltage level and related performance.
- When increasing the performance, the voltage scaling must be changed before increasing the system frequency.
- When decreasing performance, the system frequency must first be decreased before changing the voltage scaling.

00: scale 3 (default)
01: scale 2
10: scale 1
11: scale 0

Bits 3:0 Reserved, must be kept at reset value.

9.11.4 PWR voltage scaling status register (PWR_VOSSR)

Address offset: 0x014

Reset value: 0x0000 2008

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **ACTVOS[1:0]**: voltage output scaling currently applied to V_{CORE}

This field provides the last VOS value.

00: VOS3 (lowest power)
01: VOS2
10: VOS1
11: VOS0 (highest frequency)

Bit 13 **ACTVOSRDY**: Voltage level ready for currently used VOS

0: V_{CORE} is above or below the current voltage scaling provided by ACTVOS[1:0].
1: V_{CORE} is equal to the current voltage scaling provided by ACTVOS[1:0]

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **VOSRDY**: Ready bit for V_{CORE} voltage scaling output selection.

0: Not ready, voltage level below VOS selected level.
1: Ready, voltage level at or above VOS selected level.

Note: The VOSRDY flag should be used only when switching from low to high-voltage scale (like switching from VOS3 to VOS0).

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

9.11.5 PWR backup domain control register (PWR_BDCR)

This register is protected against unprivileged access when NSPRIV = 1.

This register is not reset by wakeup from Standby mode, RESET signal, and VDD POR. It is only reset by VSW POR and VSWRST reset. This register must not be accessed when the VSWRST bit in the RCC_BDCR register resets the VSW domain.

MONEN and BREN bits must not be accessed when the VSWRST bit in the RCC_BDCR register resets the VSW domain. After reset, MONEN and BREN of this register are write-protected. Prior to modifying their content, the DBP bit in the PWR_DBPCR register must be set to disable the write protection.

Address offset: 0x020

Power-on 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.	VBRS	VBE	Res.	Res.	Res.	Res.	Res.	Res.	MONEN	BREN
						rw	rw							rw	rw

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

Bit 9 **VBRS**: V_{BAT} charging resistor selection
0: Charge V_{BAT} through a 5 kΩ resistor.
1: Charge V_{BAT} through a 1.5 kΩ resistor.

Bit 8 **VBE**: V_{BAT} charging enable
0: V_{BAT} battery charging disabled.
1: V_{BAT} battery charging enabled.
Note: Reset only by POR,.

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

Bit 1 **MONEN**: Backup domain voltage and temperature monitoring enable
0: Backup domain voltage and temperature monitoring disabled
1: Backup domain voltage and temperature monitoring enabled

Bit 0 **BREN**: Backup RAM retention in Standby and V_{BAT} modes
When this bit set, the backup regulator (used to maintain the backup RAM content in Standby and V_{BAT} modes) is enabled.
If BREN is cleared, the backup regulator is switched off. The backup RAM can still be used in Run and Stop modes. However its content is lost in Standby and V_{BAT} modes.
If BREN is set, the application must wait till the backup regulator ready flag (BRRDY) is set to indicate that the data written into the SRAM is maintained in Standby and V_{BAT} modes.
0: Backup RAM content lost in Standby and V_{BAT} modes.
1: Backup RAM content preserved in Standby and V_{BAT} modes

9.11.6 PWR disable backup protection control register (PWR_DBPCR)

This register is protected against unprivileged access when NSPRIV = 1.

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.	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 and SRAM in backup domain are protected against parasitic write access. This bit must be set to enable write access to these registers.

0: Write access to backup domain disabled

1: Write access to backup domain enabled

9.11.7 PWR backup domain status register (PWR_BDSR)

This register is not reset by wakeup from Standby mode, RESET signal, and VDD POR. It is only reset by VSW POR and VSWRST reset.

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.	TEMPH	TEMPL	VBATH	VBATL	Res.	Res.	Res.	BRRDY
								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 31:24 Reserved, must be kept at reset value.

Bit 23 **TEMPH**: temperature level monitoring versus high threshold

0: temperature below high threshold level

1: temperature equal or above high threshold level

Bit 22 **TEMPL**: temperature level monitoring versus low threshold

0: temperature above low threshold level

1: temperature equal or below low threshold level

Bit 21 **VBATH**: V_{BAT} level monitoring versus high threshold

0: V_{BAT} level below high threshold level

1: V_{BAT} level equal or above high threshold level

Bit 20 **VBATL**: V_{BAT} level monitoring versus low threshold
0: V_{BAT} level above low threshold level
1: V_{BAT} level equal or below low threshold level

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

Bit 16 **BRRDY**: backup regulator ready
This bit is set by hardware to indicate that the backup regulator is ready.
0: backup regulator not ready
1: backup regulator ready

Bits 15:0 Reserved, must be kept at reset value.

9.11.8 PWR supply configuration control register (PWR_SCCR)

This register is protected against unprivileged access when NSPRIV = 1.

Address offset: 0x030

Reset value: 0x0000 0100

Reset by POR only, not reset by wakeup from Standby mode and RESET pad. The BYPASS bit of this register is written once after POR. Written-once mechanism locks the register and any further write access is ignored. The system must be power cycled before writing a new value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	LDOEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS
							r								rwo

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **LDOEN**: LDO enable
The value is set by hardware when the package uses the LDO regulator.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **BYPASS**: power management unit bypass
0: Power management unit normal operation. Use the internal regulator.
1: Power management unit bypassed. Use the external power (voltage monitoring still active)

9.11.9 PWR voltage monitor control register (PWR_VMCR)

This register is protected against unprivileged access when NSPRIV = 1.

The PVDE and PLS bits are protected by the lock mechanism. The lock control is in the SBS module controlled by the PVDL bit in the SBS_CFGR2 register. By default, the PVDE, and PLS are unlocked.

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
Res.	Res.	Res.	Res.	Res.	ALS[1:0]		AVDEN	Res.	Res.	Res.	Res.	PLS[2:0]			PVDE
					rw	rw	rw					rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:9 **ALS[1:0]**: analog voltage detector (AVD) level selection
These bits select the analog voltage detector (AVD) thresholds.
00: AVD level0 (V_{AVD0} around 1.7 V)
01: AVD level1 (V_{AVD1} around 2.1 V)
10: AVD level2 (V_{AVD2} around 2.5 V)
11: AVD level3 (V_{AVD3} around 2.8 V)

Bit 8 **AVDEN**: peripheral voltage monitor on V_{DDA} enable
0: peripheral voltage monitor on V_{DDA} disabled
1: peripheral voltage monitor on V_{DDA} enabled

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:1 **PLS[2:0]**: programmable voltage detector (PVD) level selection
These bits select the programmable voltage detector (PVD) thresholds.
000: PVD level0 (V_{PVD0} around 1.95 V)
001: PVD level1 (V_{PVD1} around 2.1 V)
010: PVD level2 (V_{PVD2} around 2.25 V)
011: PVD level3 (V_{PVD3} around 2.4 V)
100: PVD level4 (V_{PVD4} around 2.55 V)
101: PVD level5 (V_{PVD5} around 2.7 V)
110: PVD level6 (V_{PVD6} around 2.85 V)
111: PVD_IN pin

Bit 0 **PVDE**: PVD enable
0: PVD disabled
1: PVD enabled

9.11.10 PWR voltage monitor status register (PWR_VMSR)

Address offset: 0x03C

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.	PVDO	Res.	VDDIO 2RDY	AVDO	Res.	Res.	Res.
									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 31:23 Reserved, must be kept at reset value.

Bit 22 **PVDO**: programmable voltage detect output

This bit is set and cleared by hardware. It is valid only if the PVD has been enabled by the PVDE bit.

0: V_{DD} is equal or higher than the PVD threshold selected through the PLS[2:0] bits.

1: V_{DD} is lower than the PVD threshold selected through the PLS[2:0] bits.

Note: Since the PVD is disabled in Standby mode, this bit is equal to 0 after Standby or reset until the PVDE bit is set.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **VDDIO2RDY**: voltage detector output on V_{DDIO2}

This bit is set and cleared by hardware.

0: V_{DDIO2} is below the threshold of the V_{DDIO2} voltage monitor.

1: V_{DDIO2} is equal or above the threshold of the V_{DDIO2} voltage monitor.

Bit 19 **AVDO**: analog voltage detector output on V_{DDA}

This bit is set and cleared by hardware. It is valid only if AVD on VDDA is enabled by the AVDEN bit.

0: V_{DDA} is equal or higher than the AVD threshold selected with the ALS[2:0] bits.

1: V_{DDA} is lower than the AVD threshold selected with the ALS[2:0] bits.

Note: Since the AVD is disabled in Standby mode, this bit is equal to 0 after standby or reset until the AVDEN bit is set.

Bits 18:0 Reserved, must be kept at reset value.

9.11.11 PWR wakeup status clear register (PWR_WUSCR)

Each bit CWUFx is protected against unprivileged access when NSPRIV = 1.

Address offset: 0x040

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.	CWUF5	CWUF4	CWUF3	CWUF2	CWUF1
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **CWUFx**: clear wakeup pin flag for WUFx (x = 5 to 1)

These bits are always read as 0.

0: no effect

1: writing 1 clears the WUFx wakeup pin flag (bit is cleared to 0 by hardware).

9.11.12 PWR wakeup status register (PWR_WUSR)

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.	WUF5	WUF4	WUF3	WUF2	WUF1
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **WUFx**: wakeup pin WUFx flag (x = 5 to 1)

This bit is set by hardware and cleared only by a RESET pin or by setting the CWUFx bit in PWR_WUSCR register.

0: no wakeup event occurred.

1: a wakeup event received from WUFx pin.

9.11.13 PWR wakeup configuration register (PWR_WUCR)

Each bit WUPENx is protected against unprivileged accesswhen NSPRIV=1.

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.	WUPPUPD5[1:0]		WUPPUPD4[1:0]		WUPPUPD3[1:0]		WUPPUPD2[1:0]		WUPPUPD1[1:0]	
						rw	w	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.	WUPP5	WUPP4	WUPP3	WUPP2	WUPP1	Res.	Res.	Res.	WUPEN5	WUPEN4	WUPEN3	WUPEN2	WUPEN1
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

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

Bits 25:16 **WUPPUPDx[1:0]**: wakeup pin pull configuration for WKUPx (x = 5 to 1)

These bits define the I/O pad pull configuration used when WUPENx = 1. The associated GPIO port pull configuration must be set to the same value or to 00. The wakeup pin pull configuration is kept in Standby mode.

00: no pull-up

01: pull-up

10: pull-down

11: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **WUPPx**: wakeup pin polarity bit for WUPx (x = 5 to 1)

These bits define the polarity used for event detection on WUPx external wakeup pin.

0: detection on high level (rising edge)

1: detection on low level (falling edge)

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **WUPENx**: enable wakeup pin WUPx (x = 5 to 1)

These bits are set and cleared by software.

0: an event on WUPx pin does not wakeup the system from Standby mode.

1: a rising or falling edge on WUPx pin wakes up the system from Standby mode.

Note: an additional wakeup event is detected if WUPx pin is enabled (by setting the WUPENx bit) when WUPx pin level is already high when WUPPx selects rising edge, or low when WUPPx selects falling edge.

9.11.14 PWR I/O retention register (PWR_IORETR)

This register is protected against unprivileged access when NSPRIV=1.

Address offset: 0x050

Reset value: 0x0001 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.	JTAGIORETEN
															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.	IORETEN
															rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **JTAGIORETEN**: IO retention enable for JTAG IOs

0: IO Retention mode is disabled.

1: IO Retention mode is enabled for PA13, PA14, PA15 and PB4.

when entering into Standby mode, the output is sampled, and applied to the output IO during the Standby power mode

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **IORETEN**: IO retention enable:

0: IO Retention mode is disabled.

1: IO Retention mode is enabled for all IOs except the ones supporting the standby functionality (PC13, PC14 and PC15) and JTAG I/Os (PA13, PA14, PA15, PB4).

When entering into Standby mode, the output is sampled, and applied to the output IO during the Standby power mode.

Note: the IO state is not retained if the DBG_STANDBY bit is set in DBGMCU_CR register.

9.11.15 PWR privilege configuration register (PWR_PRIVCFGR)

This register can be written only when the access is privileged. It can be read by privileged or unprivileged access.

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.	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	Res.
														rw	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **NSPRIV**: PWR functions privilege configuration

Set and reset by software. This bit can be written only by privileged access.

0: Read and write to PWR functions can be done by privileged or unprivileged access.

1: Read and write to PWR functions can be done by privileged access only.

Bit 0 Reserved, must be kept at reset value.

9.11.16 PWR register map

Table 62. 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_PMCR**	Res.	Res.	Res.	Res.	Res.	SRAM1SO	SRAM2SO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AVD_READY	BOOSTE	Res.	Res.	FLPS	Res.	CSSF	Res.	Res.	Res.	SVOS[1:0]		Res.	LPMS
	Reset value						0	0												0	0			0		0				1	1		0
0x004	**PWR_PMSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SBF	STOPF	Res.	Res.	Res.	Res.	Res.
	Reset value																										0	0					
0x008 to 0x00F	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.
0x010	**PWR_VOSCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VOS[1:0]		Res.	Res.	Res.	Res.
	Reset value																											0	0				
0x014	**PWR_VOSSR**	Res.	Res.	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.	VOSRDY	Res.	Res.	Res.
	Reset value																	0	0	1										1			
0x018 to 0x01F	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.
0x020	**PWR_BDCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBRS	VBE	Res.	Res.	Res.	Res.	Res.	Res.	MONEN	BREN
	Reset value																							0	0							0	0
0x024	**PWR_DBPCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x028	**PWR_BDSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TEMPH	TEMPL	VBATH	VBATL	Res.	Res.	Res.	BRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0	0	0	0				0																
0x02C	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.
0x030	**PWR_SCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LDOEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS
	Reset value																								1								0
0x034	**PWR_VMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALS		AVDEN	Res.	Res.	Res.	Res.	PLS			PVDE
	Reset value																						0	0	0					0	0	0	0

Table 62. 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
0x03C	**PWR_VMSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDD	Res.	VDDIO2RDY	AVDD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										0		0	0																			
0x040	**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.	Res.	Res.	Res.	CWUF5	CWUF4	CWUF3	CWUF2	CWUF1
	Reset value																												0	0	0	0	0
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.	Res.	Res.	Res.	WUF5	WUF4	WUF3	WUF2	WUF1
	Reset value																												0	0	0	0	0
0x048	**PWR_WUCR**	Res.	Res.	Res.	Res.	Res.	Res.	WUPPUPD5		WUPPUPD4		WUPPUPD3		WUPPUPD2		WUPPUPD1		Res.	Res.	Res.	WUPP5	WUPP4	WUPP3	WUPP2	WUPP1	Res.	Res.	Res.	WUPEN5	WUPEN4	WUPEN3	WUPEN2	WUPEN1
	Reset value							0	0	0	0	0	0	0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x04C	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.
0x050	**PWR_IORETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JTAGIORETEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IORETEN
	Reset value																1																1
0x054 to 0x0FF	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.
0x104	**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	Res.
	Reset value																															0	

Refer to *Section 2.2* for the register boundary addresses.

10 Reset and clock control (RCC)

10.1 Introduction

The reset and clock control (RCC) manages the different kind of reset, and generates all the clocks for the bus and peripherals.

10.2 RCC pins and internal signals

The table below lists the RCC inputs and output signals connected to package pins or balls.

Table 63. 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
MCO1	O	Output clock 1 for external devices
MCO2	O	Output clock 2 for external devices
LSCO	O	Low-speed output clock for external devices
AUDIOCLK	I	External kernel clock input for I2S1, I2S2, and I2S3

10.3 RCC reset functional description

There are three types of reset:

- a system reset
- a power reset
- a backup domain reset

10.3.1 Power reset

A power reset is generated when one of the following events occurs:

- a brownout reset (BOR)
- when exiting standby mode

A brownout reset, including power-on or power-down reset (POR/PDR), sets all registers to their reset values except the ones in the backup domain.

When exiting standby mode, all registers in the core domain are set to their reset value. Registers outside the core domain are not impacted. These registers are: RTC, WKUP, IWDG, and GPIO pullup/pulldown configuration during standby and standby mode exit).

10.3.2 System reset

A system reset sets all registers to their reset values except the reset flags in *RCC reset status register (RCC_RSR)* 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)
- a window watchdog event (WWDG 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*)
- a brownout reset

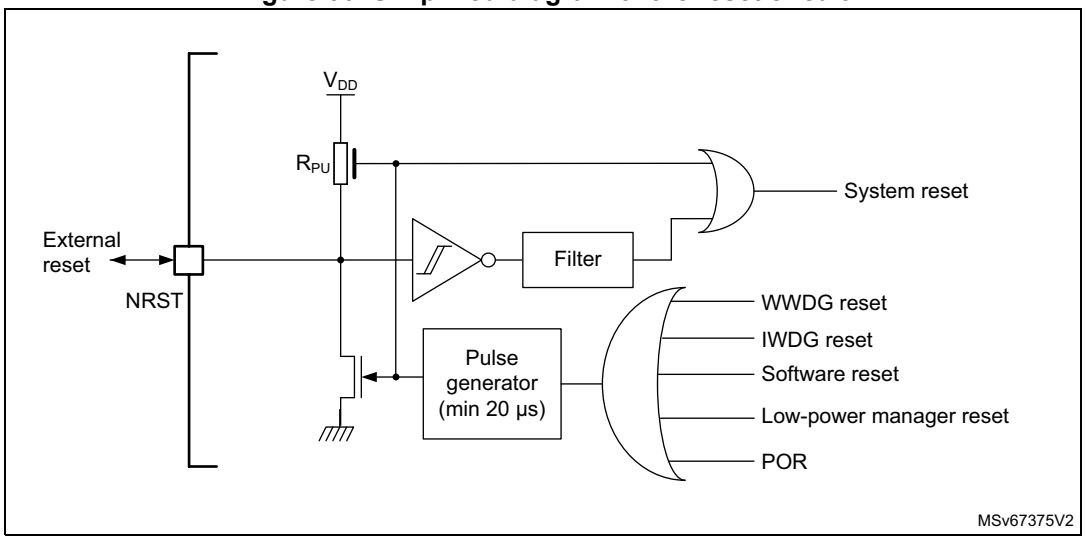
The reset source can be identified by checking the reset flags in *RCC reset status register (RCC_RSR)*.

These sources act on the NRST pin and this pin is always kept low during the delay phase. The reset service-routine vector is selected depending on product state, on boot option bytes or on both.

The system reset signal provided to the device is output on the NRST pin. The pulse generator guarantees a minimum reset-pulse duration of 20 µs for each internal reset source. In the case of an external reset, the reset pulse is generated while the NRST pin is asserted low.

In case on an internal reset, the internal pull-up R_{PU} is deactivated in order to save the power consumption through the pull-up resistor.

Figure 30. Simplified diagram of the reset circuit



Software reset

The SYSRESETREQ bit in the Cortex-M33 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, the following low-power mode security resets are available. If enabled in option bytes, the resets are generated in any of the following conditions:

- Entering standby mode: this type of reset is enabled by resetting the nRST_STDBY bit in user option bytes. In this case, whenever a standby mode entry sequence is successfully executed, the device is reset instead of entering standby mode.
- Entering stop mode: this type of reset is enabled by resetting the nRST_STOP bit in user option bytes. In this case, whenever a stop mode entry sequence is successfully executed, the device is reset instead of entering stop mode.

For further information on the user option bytes, refer to *Section 7.4.1: About option bytes*.

10.3.3 Backup domain reset

A backup domain reset is generated when one of the following events occurs:

- A software reset, triggered by setting the VSWRST bit in the *RCC backup domain control register (RCC_BDCR)*. Write access to backup domain must be enabled before setting VSWRST bit to perform the reset.
- A V_{DD} or V_{BAT} power on, if both supplies have previously been powered off.

The backup RAM is erased in these conditions:

- after a backup domain reset
- through the full product state regression (refer to *Section 7: Embedded flash memory (FLASH)*)
- when a tamper event occurs

Refer to *Section 9.6.5: VDDIO2 voltage monitor (IO2VM)* for additional information.

A backup domain reset affects the LSE oscillator, the RTC, the backup registers, the backup SRAM, and the RCC_BDCR register.

10.3.4 Reset source identification

The application can identify the reset source by checking the reset flags in the RCC_RSR register.

The software can reset the flags by setting RMVF bit.

The following table shows how the status bits of the RCC_RSR register behave according to the situation that generated the reset. For example, when an IWDG timeout occurs (line #6), if the CPU is reading the RCC_RSR register during the boot phase, both PINRSTF and IWDGRSTF bits are set, indicating that the IWDG also generated a pin reset.

Table 64. Reset source identification (RCC_RSR)(1)

#	Situations that generate a reset	LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	BORRSTF	PINRSTF
1	Power-on reset	0	0	0	0	1	1
2	Pin/pad reset	0	0	0	0	0	1
3	Brownout (low or high) reset	0	0	0	0	1	1
4	System reset generated by CPU	0	0	0	1	0	1
5	WWDG reset	0	1	0	0	0	1
6	IWDG reset	0	0	1	0	0	1
7	Illegal stop entry reset	1	0	0	0	0	1

1. Gray cells highlight the register bits that are set.

10.4 RCC clocks functional description

Four different clock sources can be used to drive the system clock (SYSCLK):

- HSI: high-speed internal 64 MHz RC oscillator clock
- CSI: low-power internal 4 MHz RC oscillator clock
- HSE: high-speed external crystal or clock, from 4 to 50 MHz
- PLL1 clock

The HSI is used as a system clock source after startup from reset, configured at 32 MHz.

The device has the following additional clock sources:

- LSI: 32 kHz low-speed internal RC that drives the independent watchdog and optionally the RTC used for auto-wakeup from stop and standby modes
- LSE: 32.768 kHz low-speed external crystal or clock that optionally drives the real-time clock (rtc_ck)
- HSI48: internal 48 MHz RC that potentially drives the USB and the RNG
- PLL2 clock

Each clock source can be switched on or off independently when it is not used, to optimize power consumption.

Several prescalers can be used to configure the AHB frequency, the APB1, and APB2 domains. The maximum frequency of the AHB and APB domains is 250 MHz.

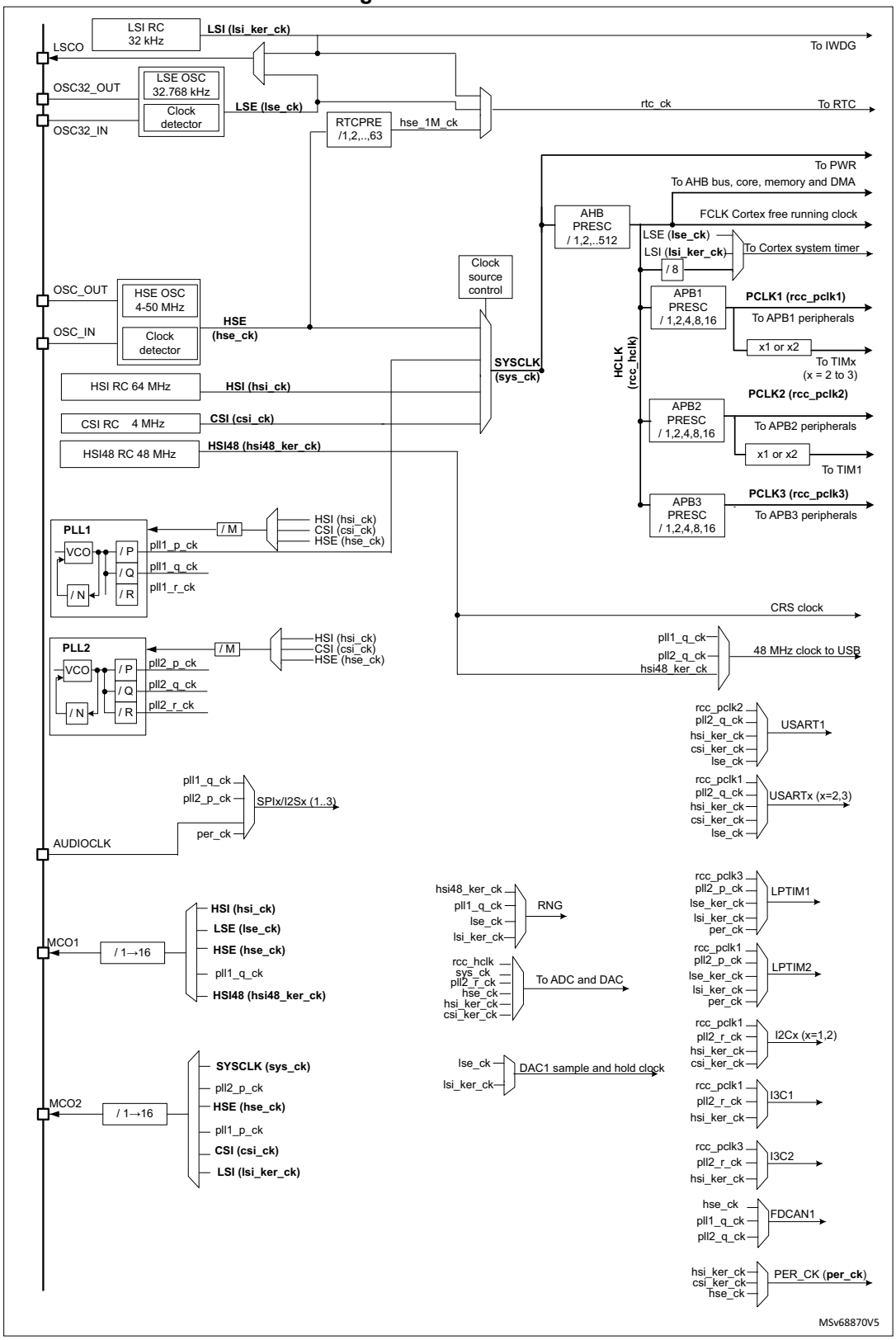
All the peripheral clocks are derived from their bus clock (HCLK, PCLK1, PCLK2, or PCLK3) except the following ones that receive an independent kernel clock. This kernel clock can be selected by software between several sources thanks to RCC_CCIPRx registers (x = 1, 2, 3, 4, 5): USB, RNG, ADC1, DAC1, U(S)ARTx (x = 1 to 3), LPUART1, I2Cx (x = 1 to 2), I3Cx (x = 1 to 2), SPIx (x = 1 to 3), FDCAN1, LPTIMx (x = 1 to 2).

In addition, the RTC kernel clock is selected by software in RCC_BDCR. The IWDG clock is always the LSI 32 kHz clock.

The RCC feeds the Cortex system timer (SysTick) external clock with the AHB clock (HCLK) divided by eight, or LSE or LSI. The SysTick can work either with this clock or directly with the Cortex clock (HCLK), configurable in the SysTick control and status register.

FCLK acts as Cortex-M33 free-running clock.

Figure 31. Clock tree



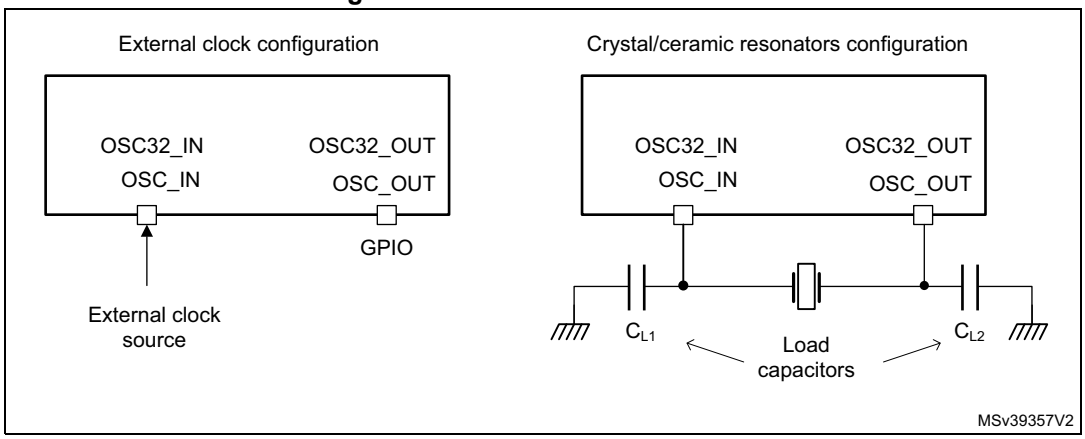
1. For details on the internal/external clock source, refer to the electrical characteristics in the datasheet.

10.4.1 HSE clock

The HSE block can generate a clock from two possible sources:

- external crystal/ceramic resonator
- external clock source

Figure 32. HSE/ LSE clock sources



External clock source (HSE bypass)

In this mode, an external clock source must be provided to the OSC_IN pin. The external clock can be low swing (analog) or digital.

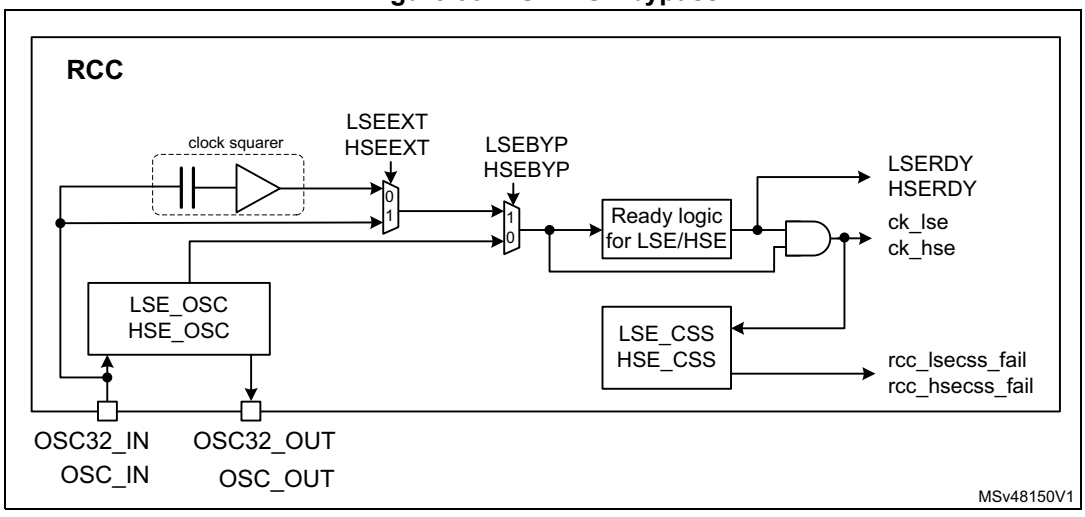
If this clock is directly used by a peripheral, the duty cycle requirement is defined by this peripheral, and the application (refer to the datasheet for more details).

This external clock is provided to the OSC_IN pin while the OSC_OUT pin can be used as a GPIO (see *Figure 32*).

In the case of an analog clock (low swing) the HSEBYP and HSEON bits must be set to 1 in the *RCC clock control register (RCC_CR)*.

In the case of a digital clock, the HSEBYP and the HSEEXT bits must be set to 1 followed by setting the HSEON bit to 1 in the *RCC clock control register (RCC_CR)*.

Figure 33. HSE/ LSE bypass



External crystal/ceramic resonator

The oscillator is enabled by setting the HSEBYP bit to 0 and the HSEON bit to 1.

The HSE can be used when the product requires a very accurate high-speed clock.

The associated hardware configuration is shown in *Figure 32*: the resonator and the load capacitors must be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected crystal or ceramic resonator. Refer to the electrical characteristics section of the datasheet for more details.

The HSERDY flag of the *RCC clock control register (RCC_CR)* indicates whether the HSE oscillator is stable or not. At startup, the **hse_ck** clock is not released until this bit is set by the hardware. An interrupt can be generated if enabled in the *RCC clock source interrupt enable register (RCC_CIER)*.

The HSE can be switched ON and OFF through the HSEON bit. Note that the HSE cannot be switched OFF if one of the two conditions is met:

- The HSE is used directly (via software mux) as a system clock.
- The HSE is selected as thereference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via software mux).

In that case the hardware does not allow programming the HSEON bit to 0.

The HSE is automatically disabled by hardware, when the system enters stop or standby mode.

In addition, the HSE clock can be driven to the MCO1 and MCO2 outputs and used as clock source for other application components.

10.4.2 HSI clock

The HSI block provides the default clock to the product.

The HSI is a high-speed internal RC oscillator that can be used directly as system clock, peripheral clock, or as PLL input. A predivider allows the application to select an HSI output frequency of 8, 16, 32 or 64 MHz. This predivider is controlled by the HSIDIV.

The HSI advantages are the following:

- low-cost clock source since no external crystal is required
- faster startup time than HSE (a few microseconds)

The HSI frequency, even with frequency calibration, is less accurate than an external crystal oscillator or ceramic resonator.

The HSI can be switched ON and OFF using the HSION bit. Note that the HSI cannot be switched OFF if one of the two conditions is met:

- the HSI is used directly (via software mux) as system clock
- the HSI is selected as the reference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via software mux).

In that case the hardware does not allow programming the HSION bit to 0. Note that the HSIDIV cannot be changed if the HSI is selected as reference clock for at least one enabled PLL (PLLxON bit set to 1). In that case the hardware does not update the HSIDIV with the new value. However, it is possible to change the HSIDIV if the HSI is used directly as the system clock.

The HSIRDY flag indicates if the HSI is stable or not. At startup, the HSI output clock is not released until this bit is set by hardware.

The HSI clock can also be used as a backup source (auxiliary clock) if the HSE fails (refer to *Section 10.4.10: Clock security system (CSS)*). The HSI can be disabled or not when the system enters stop mode.

In addition, the HSI clock can be driven to the MCO1 output and used as clock source for other application components.

Care must be taken when the HSI is used as kernel clock for communication peripherals, the application must take into account the following parameters:

- the time interval between the moment where the peripheral generates a kernel clock request and the moment where the clock is really available
- the frequency accuracy.

Note: The HSI can remain enabled when the system is in stop mode.

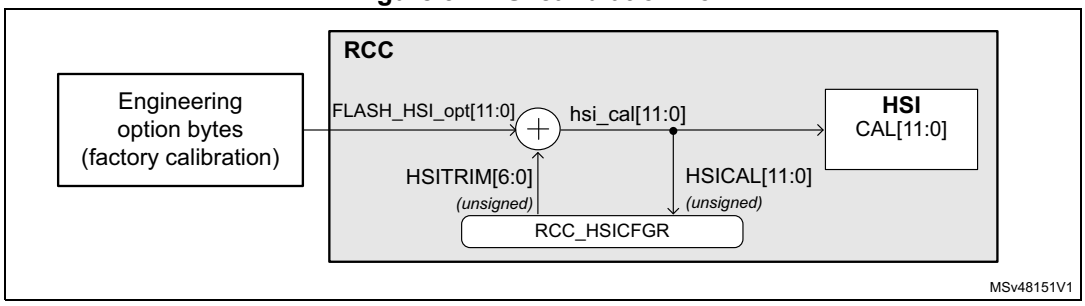
HSI calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations. That is why each device is factory calibrated by STMicroelectronics to achieve an accuracy of ACCHSI (refer to the product datasheet for more information).

After a power-on reset, the factory calibration value is loaded in the HSICAL[11:0] bits. If the application is subject to voltage or temperature variations, this may affect the RC oscillator frequency. The user application can trim the HSI frequency using the HSITRIM[6:0] bits.

Note: HSICAL[11:0] and HSITRIM[6:0] bits are located in the RCC HSI calibration register (RCC_HSICFGR).

Figure 34. HSI calibration flow



10.4.3 CSI oscillator

The CSI is a low-power RC oscillator that can be used directly as system clock, peripheral clock, or PLL input.

The CSI advantages are the following:

- low-cost clock source since no external crystal is required
- faster startup time than HSE (a few microseconds)
- very low-power consumption,

The CSI provides a clock frequency of about 4 MHz, while the HSI is able to provide a clock up to 64 MHz.

CSI frequency, even with frequency calibration, is less accurate than an external crystal oscillator or ceramic resonator.

The CSI can be switched ON and OFF through the CSION bit. The CSIRDY flag indicates whether the CSI is stable or not. At startup, the CSI output clock is not released until this bit is set by the hardware.

The CSI cannot be switched OFF if one of the two conditions is met:

- The CSI is used directly (via software mux) as the system clock.
- The CSI is selected as reference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via software mux).

In that case the hardware does not allow programming the CSION bit to 0.

The CSI can be disabled or not when the system enters stop mode.

In addition, the CSI clock can be driven to the MCO2 output and used as the clock source for other application components.

Even if the CSI settling time is faster than the HSI, care must be taken when the CSI is used as kernel clock for communication peripherals: the application must take into account the following parameters:

- the time interval between the moment where the peripheral generates a kernel clock request and the moment where the clock is really available
- the frequency precision

Note: CSION and CSIRDY bits are located in the RCC clock control register (RCC_CR).

CSI calibration

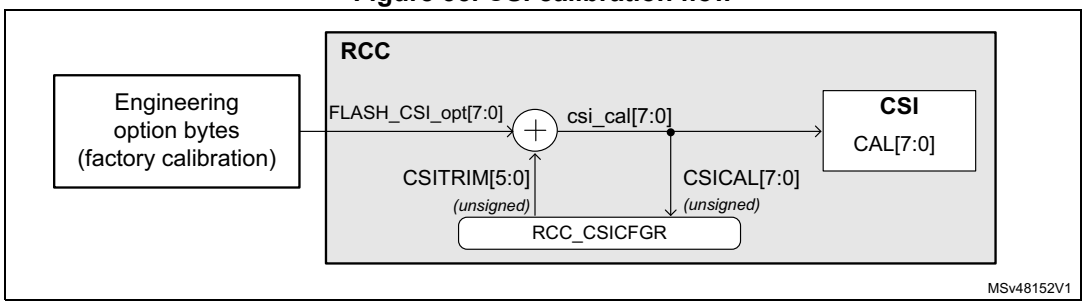
RC oscillator frequencies can vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated by STMicroelectronics to achieve the accuracy of ACC_{CSI} (refer to the product datasheet for more information).

After reset, the factory calibration value is loaded in the CSICAL[7:0] bits.

If the application is subject to voltage or temperature variations, this may affect the RC oscillator frequency. The user application can trim the CSI frequency using the CSITRIM[5:0] bits.

Note: Bits CSICAL[7:0] and CSITRIM[5:0] are located into the RCC CSI calibration register (RCC_CSICFGR).

Figure 35. CSI calibration flow



10.4.4 HSI48 clock

The HSI48 is an RC oscillator delivering a 48 MHz clock that can be used directly as kernel clock for some peripherals such as USB or RNG.

The HSI48 oscillator mainly aims at providing a high-precision clock to the USB peripheral by means of a special clock recovery system (CRS) circuitry that can use the USB SOF signal, the LSE or an external signal to automatically adjust the oscillator frequency on-the-fly, with a very small granularity.

The HSI48 oscillator is disabled as soon as the system enters stop or standby mode. When the CRS is not used, this oscillator is free running and thus subject to manufacturing process variations. That is why each device is factory calibrated by STMicroelectronics to achieve an accuracy of ACCHSI48 (refer to the product datasheet for more information).

For more details on how to configure and use the CRS, refer to *Section 11: Clock recovery system (CRS)*).

The HSI48RDY flag indicates whether the HSI48 oscillator is stable or not. At startup, the HSI48 output clock is not released until this bit is set by the hardware.

The HSI48 can be switched ON and OFF using the HSI48ON bit.

The HSI48 clock can also be driven to the MCO1 multiplexer and used as clock source for other application components.

Note: HSI48ON and HSI48RDY bits are located in the RCC clock control register (RCC_CR).

10.4.5 PLL description

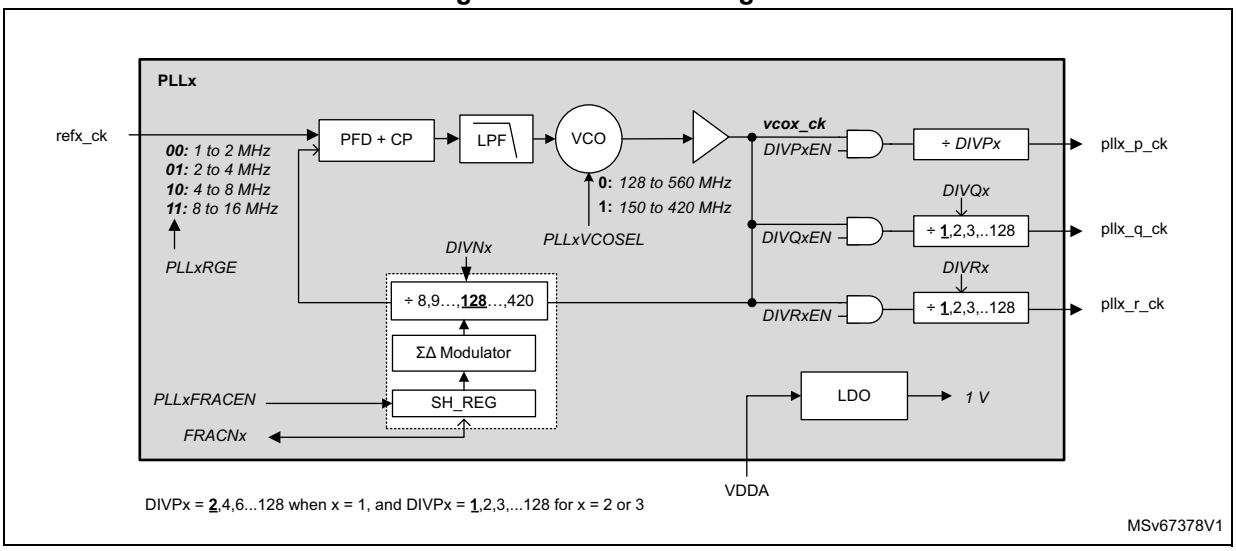
The RCC features two PLLs:

- a main PLL, PLL1, generally used to provide clocks to the CPU and some peripherals
- a dedicated PLL, PLL2 used to generate the kernel clock for peripherals

The PLLs integrated into the RCC are completely independent. They offer the following features:

- A VCO supporting two modes:
  - A wide-range
  - A low-range used for instance in audio application cases
- Input frequency range:
  - 2 to 16 MHz for the VCO in wide-range mode
  - 1 to 2 MHz for the VCO in low-range mode
- Capability to work either in integer or fractional mode
- 13-bit sigma-delta modulator, allowing to fine-tune the VCO frequency by steps of 11 to 0.3 ppm
- The sigma-delta modulator can be updated on-the-fly without generating frequency overshoots on PLLs outputs.
- Each PLL offers three outputs with post-dividers.

Figure 36. PLL block diagram



The PLLs are controlled via RCC_PLLxDIVR, RCC_PLLxFRACR, RCC_PLLxCFGR and RCC_CR registers.

The frequency of the reference clock provided to the PLLs (**refx_ck**) must range from 1 to 16 MHz. The DIVMx dividers of the *RCC PLL clock source selection register (RCC_PLL1CFGR)* must be properly programmed in order to match this condition. In addition, the PLLxRGE[1:0] field of the *RCC_PLLxCFGR* register must be set according to the reference input frequency to guarantee an optimal performance of the PLL.

The user application can then configure the proper VCO. The smaller range (150-420 MHz) must be chosen when the reference clock frequency is lower to 2 MHz.

To reduce the power consumption, it is recommended to configure the VCO output to the smaller range.

DIVNx loop divider must be programmed to achieve the expected frequency at VCO output. In addition, the VCO output range must be respected.

The PLLs operate in integer mode when the value of the SH_REG bit of the FRACNx shadow register is set to 0. The SH_REG bit is updated with the FRACNx value when PLLxFRACEN bit goes from 0 to 1. The sigma-delta modulator is designed in order to minimize the jitter impact while allowing very small frequency steps.

The PLLs can be enabled by setting PLLxON to 1. The PLLxRDY bits indicate that the PLL is ready (means locked).

*Note: Before enabling the PLLs, make sure that the reference frequency (**refx_ck**) provided to the PLL is stable, so the hardware does not allow changing DIVMx when the PLLx is ON and it is also not possible to change PLLSRC when one of the PLL is ON.*

The hardware prevents writing PLL1ON to 0 if the PLL1 is currently used to deliver the system clock. There are other hardware protections on the clock generators (refer to Section 10.4.1: HSE clock, Section 10.4.2: HSI clock and Section 10.4.3: CSI oscillator).

The following PLL parameters cannot be changed once the PLL is enabled: DIVNx, PLLxRGE, PLLxVCOSEL, DIVPx, DIVQx, DIVRx, DIVPxEN, DIVQxEN, and DIVRxEN.

To ensure an optimal behavior of the PLL when one of the post-divider (DIVP, DIVQ or DIVR) is not used, the application must set the enable bit (DIVyEN) as well as the corresponding post-divider bits (DIVP, DIVQ or DIVR) to 0.

If the above rules are not respected, the PLL output frequency is not guaranteed.

Output frequency computation

When the PLL is configured in integer mode (SH_REG = 0), the VCO frequency (F_{VCO}) is given by the following expression:

$$F_{VCO} = F_{REF\_CK} \times DIVN$$

$$F_{PLL\_y\_CK} = (F_{VCO} / (DIVy + 1)) \text{ with y = P, Q or R}$$

When the PLL is configured in fractional mode (SH_REG different from 0), the DIVN divider must be initialized before enabling the PLLs. However, it is possible to change the value of FRACNx on-the-fly without disturbing the PLL 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 each PLL, the VCO frequency is given by the following formula:

$$F_{VCO} = F_{ref\_ck} \times \left(DIVN + \frac{FRACN}{2^{(13)}} \right)$$

Note: For PLL1, DIVP can only take odd values.

The PLLs are disabled by hardware when:

- the system enters stop or standby mode
- an HSE failure occurs when HSE or PLL (clocked by HSE) are used as system clock

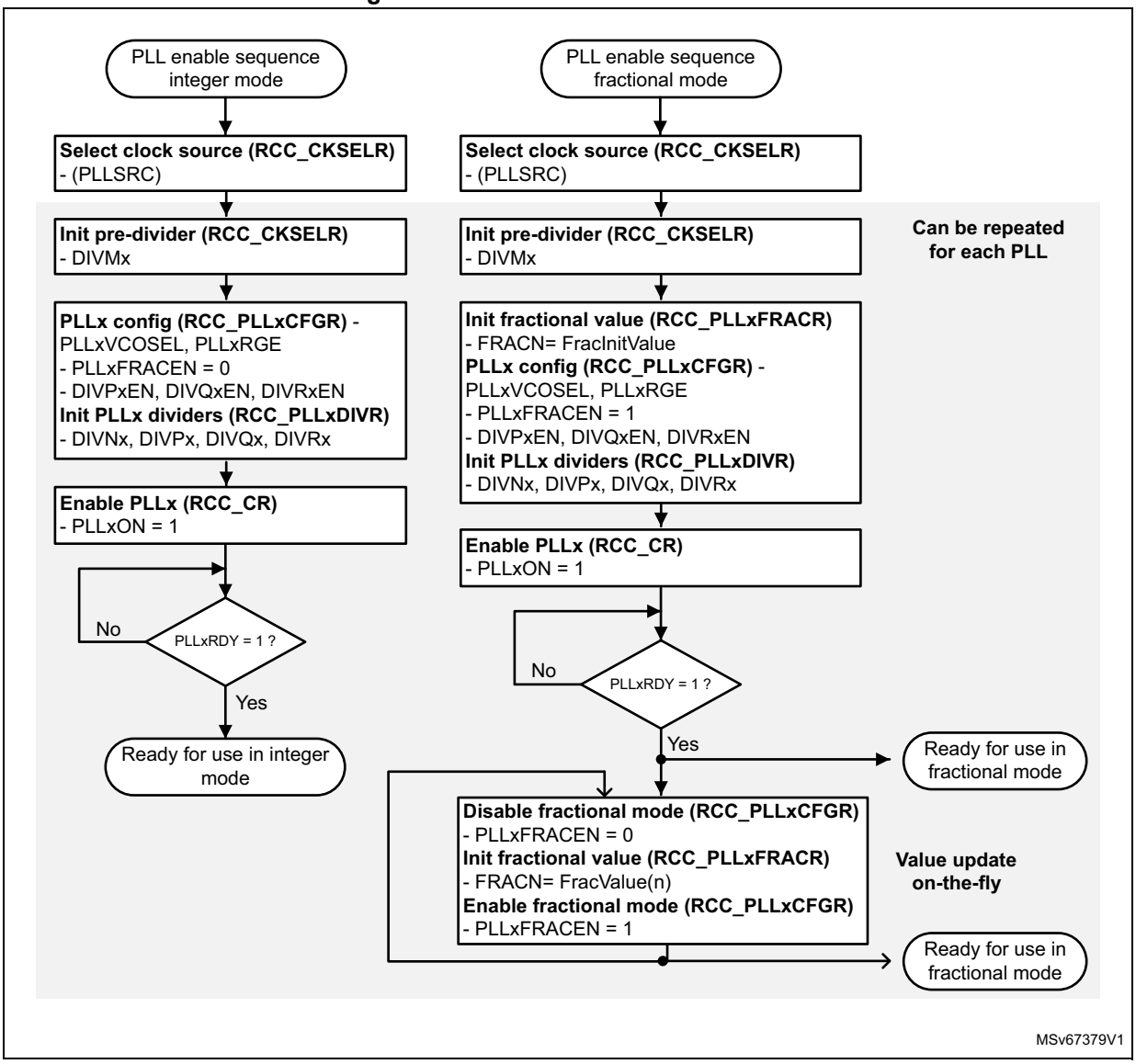
PLL initialization phase

Figure 37 shows the recommended PLL initialization sequence in integer and fractional mode. The PLLx are supposed to be disabled at the start of the initialization sequence:

1. Initialize the PLLs registers according to the required frequency.
   - Set PLLxFRACEN of *RCC_PLLxCFGR register* to 0 for integer mode.
   - For fractional mode, set FRACN to the required initial value (FracInitValue) and then set PLLxFRACEN to 1.
2. Once the PLLxON bit is set to 1, the user application must wait until PLLxRDY bit is set to 1. If the PLLx is in fractional mode, the PLLxFRACEN bit must not be set back to 0 as long as PLLxRDY = 0.
3. Once the PLLxRDY bit is set to 1, the PLLx is ready to be used.
4. If the application intends to tune the PLLx frequency on-the-fly (possible only in fractional mode), then:
   a) PLLxFRACEN must be set to 0. When PLLxFRACEN = 0, the sigma-eelta modulator is still operating with the value latched into SH_REG.
   b) A new value must be uploaded into PLLxFRACR (FracValue(n)).
   c) PLLxFRACEN must be set to 1, in order to latch the content of PLLxFRACR into its shadow register.

Note: When the PLLxRDY goes to 1, it means that the difference between the PLLx output frequency and the target value is lower than ±2%.

Figure 37. PLLs initialization flowchart



10.4.6 LSE clock

The LSE block can generate a clock from two possible sources:

- external crystal/ceramic resonator
- external user clock

External clock source (LSE bypass)

In this mode, an external clock source must be provided to OSC32_IN pin. The input clock can have a frequency up to 1 MHz and be low swing (analog) or digital. A duty cycle close to 50% is recommended.

This external clock is provided to the OSC32_IN pin while the OSC32_OUT pin can be used as a GPIO (see *Figure 32*).

In the case of an analog clock (low swing) the LSEBYP and LSEON bits must be set to 1 (*RCC backup domain control register (RCC_BDCR)*).

In case of a digital clock the LSEBYP and the LSEEXT bits must be set to 1 followed by setting the LSEON bit to 1 (*RCC backup domain control register (RCC_BDCR)*), if the RTC is used, the LSE bypass must not be configured in digital mode but in low swing analog mode (default value after reset).

External crystal/ceramic resonator (LSE crystal)

The LSE clock is generated from a 32.768 kHz crystal or ceramic resonator. It has the advantage to provide a low-power highly accurate clock source to the real-time clock (RTC) for clock/calendar or other timing functions.

The LSERDY flag of the *RCC backup domain control register (RCC_BDCR)*) 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 source interrupt enable register (RCC_CIER)*.

The LSE oscillator is switched ON and OFF using the LSEON bit. The LSE remains enabled when the system enters stop or standby mode.

In addition, the LSE clock can be driven to the MCO1 output and used as clock source for other application components.

The LSE also offers a programmable driving capability (LSEDRV[1:0]) that can be used to modulate the amplifier driving capability. This driving capability is chosen according to the external crystal/ceramic component requirement to ensure a stable oscillation.

The driving capability must be set before enabling the LSE oscillator.

10.4.7 LSI clock

The LSI acts as a low-power clock source that can be kept running when the system is in stop or standby mode for the independent watchdog (IWDG) and auto-wakeup unit (AWU).

The clock frequency is around 32 kHz. For more details, refer to the electrical characteristics section of the datasheet.

The LSI can be switched ON and OFF using the LSION bit. The LSIRDY flag indicates whether the LSI oscillator is stable or not. If an independent watchdog is started either by hardware or software, the LSI is forced ON and cannot be disabled.

The LSI remains enabled when the system enters stop or standby mode.

At LSI startup, the clock is not provided until the hardware sets the LSIRDY bit. An interrupt can be generated if enabled in the *RCC clock source interrupt enable register (RCC_CIER)*.

In addition, the LSI clock can be driven to the MCO2 output and used as a clock source for other application components.

Note: Bits LSION and LSIRDY bits are located into the RCC backup domain control register (RCC_BDCR).

10.4.8 System clock (SYSCLK) selection

Four different clock sources can be used to drive the system clock (SYSCLK):

- HSI oscillator
- CSI oscillator
- HSE oscillator
- PLL

The system clock maximum frequency is 250 MHz. After a system reset (or after leaving standby mode), the HSI oscillator, at 32 MHz, is selected as 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 (clock stable after startup delay or PLL locked). If a clock source that is not yet ready is selected, the switch occurs when the clock source becomes ready. Status bits in the *RCC clock control register (RCC_CR)* indicate which clocks are ready and which clock is currently used as a system clock.

10.4.9 Handling clock generators in stop and standby modes

When the whole system enters stop mode, all the clocks (system and kernel clocks) are stopped as well as the following clock sources:

- CSI, HSI (depending on HSIKERON, and CSIKERON bits)
- HSE
- PLL1 and PLL2
- HSI48

The content of the RCC registers is not altered except for PLL1ON, PLL2ON, HSEON, and HSI48ON that are set to 0.

Exiting stop mode

When the system exits stop mode via a wakeup event, the application can select which oscillator (HSI and/or CSI) is used to restart. The STOPWUCK bit selects the oscillator used as system clock. The STOPKERWUCK bit selects the oscillator used as kernel clock for peripherals. The STOPKERWUCK bit is useful if after a system Stop, a peripheral needs a kernel clock generated by an oscillator different from the one used for the system clock.

All these bits belong to the *Section 10.8.5: RCC clock configuration register 1 (RCC_CFGR1)*.

Table 65 gives a detailed description of their behavior.

Table 65. STOPWUCK and STOPKERWUCK description

STOPWUCK	STOPKERWUCK	-	Activated oscillator when the system exits stop mode	Distributed clocks when the system exits stop mode	
				System clock	Kernel clock
0	0	→	HSI	HSI	HSI
	1	→	HSI and CSI		HSI and/or CSI
1	0	→		CSI	
	1	→	CSI		CSI

During stop mode

There are two specific cases where the HSI or CSI can be enabled during system stop mode.

- When a dedicated peripheral requests the kernel clock:
  The peripheral receives the HSI or CSI according to the kernel clock source selected for this peripheral (via CKPERSEL[1:0]).
- When the HSIKERON or CSIKERON bits are set:
  The HSI and CSI are kept running during stop mode but the outputs are gated. The clock is then available immediately when the system exits stop mode or when a peripheral requests the kernel clock (see *Table 63* for details).

HSIKERON and CSIKERON bits belong to RCC source control register (RCC_CR).

Table 66 gives a detailed description of their behavior.

Table 66. HSIKERON and CSIKERON behavior

HSIKERON (CSIKERON)	-	HSI (CSI) state during stop mode	HSI (CSI) state setting time
0	→	OFF	$t_{su(HSI)}$ $t_{su(CSI)}$ (1)
1	→	Running and gated	Immediate

1. $t_{su(HSI)}$ and $t_{su(CSI)}$ are the startup times of the HSI and CSI oscillators (refer to the product datasheet for the values of these parameters).

When the microcontroller exists system standby mode, the HSI is selected as system and kernel clock. The RCC registers are reset to their initial values except for the RCC_RSR and

RCC_BDCR registers.

Note: The HSI and CSI outputs provide two clock paths:

- one path for the system clock (hsi_ck or csi_ck)
- one path for the peripheral kernel clock (hsi_ker_ck or csi_ker_ck).

When a peripheral requests the kernel clock in system stop mode, only the path providing the hsi_ker_ck or csi_ker_ck is activated.

10.4.10 Clock security system (CSS)

Clock security system on HSE

The clock security system can be enabled by software via the HSECSSON bit. The HSECSSON bit can be enabled even when the HSEON is set to 0.

The CSS on HSE is enabled by the hardware when the HSE is enabled and ready, and HSECSSON set to 1.

The CSS on HSE is disabled when the HSE is disabled. As a result, this function does not work when the system is in stop mode.

It is not possible to clear directly the HSECSSON bit by software.

The HSECSSON bit is cleared by hardware when a system reset occurs or when the system enters standby mode.

If a failure is detected on the HSE clock, the system automatically switches to the HSI, or CSI depending on the STOPWUCK bit configuration (in RCC clock configuration register 1 (RCC_CFGR1)) in order to provide a safe clock. The HSE is then automatically disabled, a clock failure event is sent to the break inputs of the advanced-control timer (TIM1), and an NMI is automatically generated to inform the application about the failure, thus allowing the MCU to perform rescue operations. If the HSE output was used as clock source for PLLs when the failure occurred, the PLLs are also disabled.

If an HSE clock failure occurs when the CSS is enabled, the CSS generates an interrupt that causes the automatic generation of an NMI. The HSECSSF flag in *RCC clock source interrupt flag register (RCC_CIFR)* is set to 1 to allow the application to identify the failure source. The NMI routine is executed indefinitely until the HSECSSF bit is cleared. As a consequence, the application must clear the HSECSSF flag in the NMI ISR by setting the HSECSSC bit in the *RCC clock source interrupt clear register (RCC_CICR)*.

Clock security system on LSE

A clock security system on LSE can be activated by software writing the LSECSSON bit in the *RCC backup domain control register (RCC_BDCR)*. 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 enabled) and ready (LSERDY set by hardware), and after the RTC clock has been selected by RTCSEL.

The CSS on LSE is working in all modes including VBAT. It works also under system reset (excluding power-on reset).

The clock security system on the LSE detects when the LSE disappears or in case of over frequency.

If a failure is detected on the external 32 kHz oscillator, the LSE clock is no longer supplied to the RTC, but no hardware action is made to the registers.

The CSS on LSE detection event is connected to the internal tamper 3 of the TAMP peripheral. The internal tamper 3 must be enabled (ITAMP3E = 1 in the TAMP_CR1 register) and the associated interrupt enabled (ITAMP3IE in TAMP_IER) in order to wake up from the low-power modes. This erases also the TAMP backup registers and backup SRAM unless the ITAMP3NOER = 1 in the TAMP_CR3 (see *Section 33: Tamper and backup registers (TAMP)* for more details).

In the case of the CSS on LSE detection event (LSECSSD = 1 in the RCC_BDCR), the software must then disable the LSECSSON bit, stop the defective 32 kHz oscillator

(disabling LSEON), and change the RTC clock source (no clock or LSI or HSE, with RTCSEL), or take any required action to secure the application.

Refer to datasheet for CSS on LSE electrical characteristics.

10.4.11 Clock output generation (MCO1/MCO2)

Two microcontroller clock output pins (MCO), MCO1 and MCO2, are available. A clock source can be selected for each output. The selected clock can be divided thanks to the configurable prescaler (refer to *Figure 31* for additional information on signal selection).

MCO1 and MCO2 outputs are controlled via MCO1PRE[3:0], MCO1[2:0], MCO2PRE[3:0] and MCO2[2:0] located in the *RCC clock configuration register 1 (RCC_CFGR1)*.

The GPIO port corresponding to each MCO pin must be programmed in alternate function mode.

The clock provided to the MCO outputs must not exceed the maximum pin speed (refer to the product datasheet for information on the supported pin speed).

- **LSCO**

  Another output (LSCO) allows one of the low-speed clocks below to be output onto the external LSCO pin:

  – LSI

  – LSE

  This output remains available in stop mode. This output is not available in standby and VBAT modes. The selection is controlled by the LSCOSEL bit and enabled with the LSCOEN in the *RCC backup domain control register (RCC_BDCR)*.

The MCO clock output requires the corresponding alternate function selected on the MCO pin. The LSCO pin must be left in default POR state.

10.4.12 Kernel clock selection

Some peripherals are designed to work with two different clock domains that operate asynchronously:

- a clock domain synchronous with the register and bus interface (**ckg_bus_perx** clock)
- a clock domain generally synchronous with the peripheral (kernel clock)

The benefit of having peripherals supporting these two clock domains is that the user application has more freedom to choose optimized clock frequency for the CPU, bus matrix and for the kernel part of the peripheral.

As a consequence, the user application can change the bus frequency without reprogramming the peripherals. As an example an on-going transfer with UART is not disturbed if its APB clock is changed on-the-fly.

The table below shows the kernel clock that the RCC can deliver to the peripherals. Each row of the table below represents a mux and the peripherals connected to its output.

Table 67. Kernel clock distribution overview

Peripherals	Clock mux control bits	pll1_q_ck	pll2_p_ck	pll2_q_ck	pll2_r_ck	sys_ck	bus clocks(1)	hse_ck	hsi_ker_ck	csi_ker_ck	hsi48_ck	lse_ck	lsi_ck	per_ck(2)	AUDIOCLK	Disabled
FDCAN1	FDCANSEL	1	-	2	-	-	-	0	-	-	-	-	-	-	-	-
I2C1	I2C1SEL	-	-	-	1	-	0	-	2	3	-	-	-	-	-	-
I2C2	I2C2SEL	-	-	-	1	-	0	-	2	3	-	-	-	-	-	-
I3C1	I3C1SEL	-	-	-	1	-	0	-	2	-	-	-	-	-	-	-
I3C2	I3C2SEL	-	-	-	1	-	0	-	2	-	-	-	-	-	-	-
LPTIM1	LPTIM1SEL	-	1	-	-	-	0	-	-	-	-	3	4	5	-	-
LPTIM2	LPTIM2SEL	-	1	-	-	-	0	-	-	-	-	3	4	5	-	-
TIM1	-	-	-	-	-	-	x	-	-	-	-	-	-	-	-	-
TIM2, TIM3, LPTIM2	TIMICSEL	-	-	-	-	-	-	-	1	1	-	-	-	-	-	0
RNG	RNGSEL	1	-	-	-	-	-	-	-	-	0	2	3	-	-	-
SPI(I2S)1	SPI1SEL	0	1	-	-	-	-	-	-	-	-	-	-	4	3	-
SPI(I2S)2	SPI2SEL	0	1	-	-	-	-	-	-	-	-	-	-	4	3	-
SPI(I2S)3	SPI3SEL	0	1	-	-	-	-	-	-	-	-	-	-	4	3	-
USARTx	USARTxSEL	-	-	1	-	-	0	-	3	4	-	5	-	-	-	-
LPUART 1	LPUART 1SEL	-	-	1	-	-	0	-	3	4	-	5	-	-	-	-
USB	USBSEL	1	-	2	-	-	-	-	-	-	3	-	-	-	-	0
ADCDAC(3)	ADCDACSEL	-	-	-	2	1	0	3	4	5	-	-	-	-	-	-
DAC1	DAC1SEL	-	-	-	-	-	-	-	-	-	-	0	1	-	-	-
RTC/AWU	RTCSEL	-	-	-	-	-	-	3 (4)	-	-	-	1	2	-	-	0
all	CKPERSEL	-	-	-	-	-	-	2	0	1	-	-	-	-	-	3

1. The bus clocks are the bus interface clocks to which the peripherals are connected, it can be APB or AHB clocks.
2. The per_ck clock could be hse_ck, hsi_ker_ck or csi_ker_ck according to CKPERSEL selection.
3. With a duty cycle close to 50%, meaning that DIV[P/Q/R]x values shall be even. For ADC, the duty cycle shall be 50% when supporting DDR.
4. Clock HSE divided by RTCPRE.

To reduce the number of switches, some peripherals share the same kernel clock source. Nevertheless, all peripherals have their dedicated enable signal.

Peripherals dedicated to audio applications

The audio peripherals generally need specific accurate frequencies, the kernel clock of the SPI(I2S)s can be generated by:

- PLL1 when the amount of active PLLs must be reduced (for SPI/I2S1 to 3)
- PLL2 for optimal flexibility in frequency generation
- HSE, HSI, or CSI for use-cases where the current consumption is critical
- AUDIOCLK when an external clock reference needs to be used

Peripherals dedicated to control and data transfer

Peripherals such as SPIs, I2Cs, I3Cs, UARTs do not need a specific kernel clock frequency but a clock fast enough to generate the correct baud rate, or the required bit clock on the serial interface. For that purpose the source can be selected among the following ones:

- PLL1 when the amount of active PLLs must be reduced
- PLL2 if better flexibility is required. As an example, this solution allows changing the frequency bus via PLL1 without affecting the speed of some serial interfaces.
- HSI or CSI for low-power use-cases or when the peripheral must quickly wake up from stop mode (such as UART, I2C or I3C)

Note: UARTs also need the LSE clock when high baud rates are not required.

RTC/AWU clock

The **rtc_ck** clock source can be one of the following:

- the **hse_1M_ck** (**hse_ck** divided by a programmable prescaler)
- the **lse_ck**
- the **lsi_ck** clock

The source clock is selected by programming the RTCSEL[1:0] bits in the *RCC backup domain control register (RCC_BDCR)* and the RTCPRE[5:0] bits in the *RCC clock configuration register 1 (RCC_CFGR1)*.

This selection cannot be modified without resetting the backup domain.

If the LSE is selected as RTC clock, the RTC works normally even if the backup or the V_{DD} supply disappears.

The LSE clock is in the backup domain, whereas the other oscillators are not. As a consequence:

- If LSE is selected as the RTC clock, the RTC continues working even if the V_{DD} supply is switched OFF, provided the V_{BAT} supply is maintained.
- If LSI is selected as the RTC clock, the AWU state is not guaranteed if the V_{DD} supply is powered off.
- If the HSE clock is used as RTC clock, the RTC state is not guaranteed if the V_{DD} supply is powered off, or if the V_{CORE} supply is powered off.

The **rtc_ck** clock is enabled through RTCEN bit located in the *RCC backup domain control register (RCC_BDCR)*.

The RTC bus interface clock (APB clock) is enabled through RTCAPBEN and RTCAPBLPEN bits located in RCC_APB3ENR/LPENR registers.

Note: To read the RTC calendar register when the APB clock frequency is less than seven times the RTC clock frequency (F_{APB} < 7 x F_{RTCLCK}), the software must read the calendar time and date registers twice. The data is correct if the second read access to RTC_TR gives the same result than the first one. Otherwise, a third read access must be performed.

Watchdog clocks

The RCC provides the clock for the two watchdog blocks available on the circuit. The independent watchdog (IWDG) is connected to the LSI. The window watchdog (WWDG) is connected to the APB clock.

If an independent watchdog is started by either hardware option or software access, the LSI is forced ON, and cannot be disabled. After the LSI oscillator setup delay, the clock is provided to the IWDG.

10.4.13 Clock measurement with TIMx and LPTIMx

The embedded timers (TIM2/3 and LPTIM1/2 in input capture mode) can be used to measure the frequency of clock sources used in the application.

10.4.14 Internal oscillator calibration

Most of the clock source generator frequencies can be measured by means of the input capture of TIMx or LPTIMx.

- **Calibrating the HSI or CSI with LSE:**

  The primary purpose of having the LSE connected to a TIMx or LPTIMx input capture is to be able to accurately measure the HSI or CSI. This requires to use the HSI or CSI as system clock source either directly or via PLL1. The number of system clock counts between consecutive edges of the LSE signal gives a measurement of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few tens of ppm) we can determine the internal clock frequency with the same resolution, and trim the source to compensate for manufacturing-process and/or temperature- and voltage-related frequency deviations.

  The basic concept consists in providing a relative measurement (such as HSI/LSE ratio). The precision is therefore tightly linked to the ratio between the two clock sources. The greater the ratio is, the more accurate the measurement is.

  The HSI and CSI oscillators have dedicated user-accessible calibration bits for this purpose (see *RCC CSI calibration register (RCC_CSICFGR)*). When HSI or CSI is used via the PLLx, the system clock can also be fine-tuned by using the fractional divider of the PLLs.

- **Calibrating the LSI with HSI:**

  The LSI frequency can also be measured: this is useful for applications that do not have a crystal. The ultra-low-power LSI oscillator has a large manufacturing process deviation. The LSI clock frequency can be measured using the more precise HSI clock source. Using this measurement, a more accurate RTC time base timeouts (when LSI is used as the RTC clock source) and/or an IWDG timeout with an acceptable accuracy can be obtained.

10.4.15 RTC and TAMP clock

The RTCCLK clock source is used by RTC and TAMP, and can be either the HSE / 32, LSE or LSI clock. It is selected by programming the RTCSEL[1:0] bits in the *RCC backup domain*

control register (RCC_BDCR). This selection cannot be modified without resetting the backup domain. The system must always be configured so as to get a PCLK frequency greater than or equal to the RTCCLK frequency for a proper operation of the RTC. The TAMP does not require any 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 33: Tamper and backup registers (TAMP)* for more details).

The LSE and the LSI clocks are in the backup domain, whereas the HSE clock is not. Consequently:

- If LSE or LSI is selected as RTC and TAMP clock, these peripherals continue to work even if the V_{DD} supply is switched off, provided the V_{BAT} supply is maintained.
- If the HSE clock divided by a prescaler is used as the RTC or TAMP clock, the RTC state is not guaranteed if the V_{DD} supply is powered off or if the internal voltage regulator is powered off (removing power from the core domain). Depending on the TAMP configuration, this one can remain functional if used in a mode that does not need any kernel clock.

When the RTC and TAMP clock is LSE or LSI, the RTC remains clocked and functional under system reset.

10.4.16 Timer clock

The timer clock frequencies are automatically defined by hardware.

There are two cases:

- If the APB prescaler equals 1, the timer clock frequencies are set to the APB domain frequency.
- Otherwise, they are set to twice (×2) the APB domain frequency.

10.4.17 Watchdog clock

If the independent watchdog (IWDG) is started by either hardware option or software access, the LSI oscillator is forced on and cannot be disabled. After the LSI oscillator temporization, the LSI 32 kHz clock is provided to the IWDG.

10.4.18 Peripherals clock gating

Peripherals clock gating in run mode

Each peripheral clock can be enabled by the corresponding EN bit in the RCC_AHBxENR and RCC_APBxENR registers.

When the peripheral clock is not active, read or write accesses 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 2 cycles of the peripheral bus clock.

Caution: Just after enabling the clock for a peripheral, the software must wait for these 2 clock cycles before accessing the peripheral registers.

Peripherals clock gating in sleep mode

When a peripheral is enabled, its clock can be automatically gated off when the device is in sleep mode, by clearing the peripheral LPEN bit in the RCC_AHBxLPENR and RCC_APBxLPENR registers. Both the EN and the LPEN bit of the peripheral must be set to keep the clock on in sleep mode.

10.4.19 Bus clock gating

In order to further reduce power consumption in Run or Sleep mode, APBx, or AHBx bus clock can be disabled through APBxDIS and AHBxDIS bits in RCC_CFGR2 register, when none of the peripherals mapped on such bus are used and when their clocks are disabled in RCC_APBxENR or RCC_AHBxENR.

10.5 RCC privilege protection modes

By default, after reset, all RCC registers can be read or written with both privileged and unprivileged access, except *RCC privilege configuration register (RCC_PRIVCFGR)* that can be written with privileged access only. RCC_PRIVCFGR can be read by privileged and unprivileged access.

The PRIV bit in RCC_PRIVCFGR can be written with privileged access only. This bit configures the privileged access of all RCC functions (as defined in *Table 68*, or by the GTZC for privileged peripherals, or by the peripheral itself in the case of privilege-aware peripherals).

When the PRIV bit is set in RCC_PRIVCFGR:

- Writing the RCC privileged bits is possible only with privileged access.
- The RCC privileged bits can be read only with privileged access except 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).

Table 68. RCC privilege configuration summary

RCC function	Privileged bits	Corresponding register
HSI	HSION, HSIKERON, HSIRDY	RCC_CR
	HSICAL[11:0], HSITRIM[6:0]	RCC_HSICFGR
	HSIRDYIE	RCC_CIER
	HSIRDYIF	RCC_CIFR
	HSIRDYC	RCC_CICR
HSE	HSEON, HSERDY, HSEBYP, HSECSSON, HSEEXT	RCC_CR
	HSERDYIE	RCC_CIER
	HSERDYIF, HSECSSF	RCC_CIFR
	HSERDYC, HSECSSC	RCC_CICR

Table 68. RCC privilege configuration summary (continued)

RCC function	Privileged bits	Corresponding register
CSI	CSISON, CSIKERON, CSISRDY	RCC_CR
	CSICAL[7:0], CSITRIM[5:0]	RCC_CSICFGR
	CSISRDYIE	RCC_CIER
	CSISRDYIF	RCC_CIFR
	CSISRDYIC	RCC_CICR
LSI	LSION, LSIRDY, LSIPREDIV, LSCOSEL, LSCOEN	RCC_BDCR
	LSIRDYIE	RCC_CIER
	LSIRDYIF	RCC_CIFR
	LSIRDYC	RCC_CICR
LSE	LSECSSON, LSECSSD, LSEDRV[1:0], LSEBYP, LSERDY, LSEON, LSESYSRDY, LSESYSEN, LSCOSEL, LSCOEN	RCC_BDCR
	LSERDYIE	RCC_CIER
	LSERDYF	RCC_CIFR
	LSERDYC	RCC_CICR
SYSCLK	SW[1:0], SWS[1:0], STOPWUCK, STOPKERWUCK, MCO1SEL[3:0], MCO1PRE[2:0], MCO2SEL[3:0], MCO2PRE[2:0]	RCC_CFGR
	SYSTICKSEL[1:0]	RCC_CCIPR4
	VOS[1:0]	PWR_VOSR
	TIMPRE	RCC_CFGR
PRESC	HPRE[3:0], PPRE1[2:0], PPRE2[2:0], PPRE3[2:0]	RCC_CFGR2
PLL1	PLL1SRC[1:0], PLL1RGE[1:0], PLL1FRACEN, PLL1M[3:0], PLL1VCOSEL, PLL1PEN, PLL1QEN, PLL1REN	RCC_PLL1CFGR
	PLL1N[8:0], PLL1P[6:0], PLL1Q[6:0], PLL1R[6:0]	RCC_PLL1DIVR
	PLL1FRACN[12:0]	RCC_PLL1FRACR
	PLL1RDY, PLL1ON	RCC_CR
	PLL1RDYIE	RCC_CIER
	PLL1RDYF	RCC_CIFR
	PLL1RDYC	RCC_CICR

Table 68. RCC privilege configuration summary (continued)

RCC function	Privileged bits	Corresponding register
PLL2	PLL2SRC[1:0], PLL2RGE[1:0], PLL2FRACEN, PLL2M[3:0], PLL2PEN, PLL2QEN, PLL2REN, PLL2VCOSEL	RCC_PLL2CFGR
	PLL2N[8:0], PLL2P[6:0], PLL2Q[6:0], PLL2R[6:0]	RCC_PLL2DIVR
	PLL2FRACN[12:0]	RCC_PLL2FRACR
	PLL2RDY, PLL2ON	RCC_CR
	PLL2RDYIE	RCC_CIER
	PLL2RDYF	RCC_CIFR
	PLL2RDYC	RCC_CICR
HSI48	HSI48ON, HSI48RDY	RCC_CR
	HSI48CAL[9:0]	RCC_CRRCR
	HSI48RDYE	RCC_CIER
	HSI48RDYF	RCC_CIFR
	HSI48RDYC	RCC_CICR
IPKERSECCFG	CKERPSEL[1:0]	RCC_CCIPR5
Remove reset flag	RMVF	RCC_RSR

10.6 RCC low-power modes

- AHB and APB peripheral clocks, including DMA clock, can be disabled by software.
- Sleep mode stops the CPU clock. The memory interface clocks (flash memory, caches, 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 the clocks of the peripherals connected to them are disabled.
- Stop mode stops all the clocks in the core domain and disables the PLLs, HSI, HSI48, CSI, and HSE oscillators. However, HSI (at up to 64 MHz) or CSI can be switched ON to generate a wakeup interrupt. LSI and LSE remain active in stop mode.
- Standby mode stops all the clocks in the core domain and disable the PLLs, HSI, HSI48, CSI, and HSE oscillators.

The CPU Deepsleep mode can be overridden for debugging by setting the DBG_STOP or DBG_STANDBY bit in the DBGMCU_CR register.

When exiting stop mode, the system clock is either HSI (at up to 64 MHz) or CSI, depending on the software configuration of STOPWUCK in the *RCC clock configuration register 1 (RCC_CFGR1)*. The frequency (range and user trim) of the HSI is the one configured before entering stop mode.

The other internal oscillator can be automatically woken up in addition to the one used by the system clock, in order to avoid waiting for the other oscillator wakeup time when the device is back in run mode. This is done thanks to STOPKERWUCK in RCC_CFGR1.

When leaving the standby mode, the system clock is HSI (32 MHz). The user trim is lost.

If a flash memory programming operation is ongoing, stop or standby mode entry is delayed until the flash memory interface access is finished. If an access to the APB domain is ongoing, stop or standby mode entry is delayed until the APB access is finished.

10.7 RCC interrupts

The table below summarizes the interrupt sources and the way to control them.

Table 69. 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	LSIRDYF	LSI ready	LSIRDYIE	Set LSIRDYC to 1	Yes	No
	LSERDYF	LSE ready	LSERDYIE	Set LSERDYC to 1	Yes	No
	HSIDRYF	HSI ready	HSIDRYIE	Set HSIRDYC to 1	Yes	No
	HSERDYF	HSE ready	HSERDYIE	Set HSERDYC to 1	Yes	No
	CSISRDYF	CSIS ready	CSISRDYIE	Set CSISRDYC to 1	Yes	No
	HSI48RDYF	HSI48 ready	HSI48RDYIE	Set HSI48RDYC to 1	Yes	No
	PLL1RDYF	PLL1 ready	PLL1RDYIE	Set PLL1RDYC to 1	Yes	No
	PLL2RDYF	PLL2 ready	PLL2RDYIE	Set PLL2RDYC to 1	Yes	No
TAMP	ITAMP3F(1)	LSE CSS failure	LSECSSON and ITAMP3E(1) and ITAMP3IE(1)	Set CITAMP3F(1) to 1	Yes	Yes
NMI	HSECSSF	HSE CSS failure	-(2)	Set HSECSSC to 1	Yes	No

1. The LSE CSS failure event (LSECSSD) is connected to TAMP internal tamper 3. In order to get the interrupt associated to this event, the internal tamper 3 must be enabled, and the internal tamper 3 interrupt must be enabled. The ITAMP3F, ITAMP3E, ITAMP3IE, and CITAMP3F bits are in the TAMP peripheral.
2. It is not possible to mask this interrupt when the security system feature is enabled (HSECSSON = 1).

10.8 RCC registers

10.8.1 RCC clock control register (RCC_CR)

Address offset: 0x000

Reset value: 0x0000 002B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	PLL2RDY	PLL2ON	PLL1RDY	PLL1ON	Res.	Res.	Res.	HSEEXT	HSECSSON	HSEBYP	HSERDY	HSEON
				r	rw	r	rw				rw	rs	rw	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	HSI48RDY	HSI48ON	Res.	CSIKERON	CSIRDY	CSION	Res.	Res.	HSIDIVF	HSIDIV[1:0]		HSIKERON	HSIRDY	HSION
		r	rw		rw	r	rw			r	rw	rw	rw	r	rw

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

Bit 27 **PLL2RDY**: PLL2 clock ready flag
Set by hardware to indicate that the PLL is locked.
0: PLL2 unlocked
1: PLL2 locked

Bit 26 **PLL2ON**: PLL2 enable
Set and cleared by software to enable PLL2.
Cleared by hardware when entering Stop or standby mode.
0: PLL2 OFF (default after reset)
1: PLL2 ON

Bit 25 **PLL1RDY**: PLL1 clock ready flag
Set by hardware to indicate that the PLL1 is locked.
0: PLL1 unlocked (default after reset)
1: PLL1 locked

Bit 24 **PLL1ON**: PLL1 enable
Set and cleared by software to enable PLL1.
Cleared by hardware when entering Stop or standby mode. Note that the hardware prevents writing this bit to 0, if the PLL1 output is used as the system clock.
0: PLL1 OFF (default after reset)
1: PLL1 ON

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **HSEEXT**: external high speed clock type in Bypass mode
Set and reset by software to select the external clock type (analog or digital).
The external clock must be enabled with the HSEON bit to be used by the device. The HSEEXT bit can be written only if the HSE oscillator is disabled.
0: HSE in analog mode (default after reset)
1: HSE in digital mode

Bit 19 **HSECSSON**: HSE clock security system enable
Set by software to enable clock security system on HSE.
This bit is “set only” (disabled by a system reset or when the system enters in standby mode). When HSECSSON is set, the clock detector is enabled by hardware when the HSE is ready and disabled by hardware if an oscillator failure is detected.
0: CSS on HSE OFF (clock detector OFF) (default after reset)
1: CSS on HSE ON (clock detector ON if the HSE oscillator is stable, OFF if not).

Bit 18 **HSEBYP**: HSE clock bypass
Set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the HSEON bit to be used by the device.
The HSEBYP bit can be written only if the HSE oscillator is disabled.
0: HSE oscillator not bypassed (default after reset)
1: HSE oscillator bypassed with an external clock

Bit 17 **HSERDY**: HSE clock ready flag
Set by hardware to indicate that the HSE oscillator is stable.
0: HSE clock is not ready (default after reset)
1: HSE clock is ready

Bit 16 **HSEON**: HSE clock enable
Set and cleared by software.
Cleared by hardware to stop the HSE when entering Stop or standby mode.
This bit cannot be cleared if the HSE is used directly as system clock, or if the HSE is selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set to 1).
0: HSE is OFF (default after reset)
1: HSE is ON

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

Bit 13 **HSI48RDY**: HSI48 clock ready flag
Set by hardware to indicate that the HSI48 oscillator is stable.
0: HSI48 clock is not ready (default after reset)
1: HSI48 clock is ready

Bit 12 **HSI48ON**: HSI48 clock enable
Set by software and cleared by software or by the hardware when the system enters to Stop or standby mode.
0: HSI48 is OFF (default after reset)
1: HSI48 is ON

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CSIKERON**: CSI clock enable in stop mode
Set and reset by software to force the CSI to ON, even in stop mode, in order to be quickly available as kernel clock for some peripherals. This bit has no effect on the value of CSION.
0: no effect on CSI (default after reset)
1: CSI is forced to ON even in stop mode

Bit 9 **CSIRDY**: CSI clock ready flag
Set by hardware to indicate that the CSI oscillator is stable. This bit is activated only if the RC is enabled by CSION (it is not activated if the CSI is enabled by CSIKERON or by a peripheral request).
0: CSI clock is not ready (default after reset)
1: CSI clock is ready

Bit 8 **CSION**: CSI clock enable

Set and reset by software to enable/disable CSI clock for system and/or peripheral.

Set by hardware to force the CSI to ON when the system leaves stop mode, if STOPWUCK = 1 or STOPKERWUCK = 1.

This bit cannot be cleared if the CSI is used directly as system clock, or if the CSI is selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set to 1).

0: CSI is OFF (default after reset)

1: CSI is ON

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **HSIDIVF**: HSI divider flag

Set and reset by hardware.

As a write operation to HSIDIV has not an immediate effect on the frequency, this flag indicates the current status of the HSI divider. HSIDIVF goes immediately to 0 when HSIDIV value is changed, and is set back to 1 when the output frequency matches the value programmed into HSIDIV.

0: new division ratio not yet propagated to hsi_ker_ck (default after reset)

1: hsi_ker_ck clock frequency reflects the new HSIDIV value (default register value when the clock setting is completed).

Bits 4:3 **HSIDIV[1:0]**: HSI clock divider

Set and reset by software.

These bits allow selecting a division ratio in order to configure the wanted HSI clock frequency.

The HSIDIV cannot be changed if the HSI is selected as reference clock for at least one enabled PLL (PLLxON bit set to 1). In that case, the new HSIDIV value is ignored.

00: division by 1, hsi_ker_ck = 64 MHz

01: division by 2, hsi_ker_ck = 32 MHz (default after system reset and exit from standby)

10: division by 4, hsi_ker_ck = 16 MHz

11: division by 8, hsi_ker_ck= 8 MHz

Bit 2 **HSIKERON**: HSI clock enable in stop mode

Set and reset by software to force the HSI to ON, even in stop mode, in order to be quickly available as kernel clock for peripherals. This bit has no effect on the value of HSION.

0: no effect on HSI (default after reset)

1: HSI is forced to ON even in stop mode

Bit 1 **HSIRDY**: HSI clock ready flag

Set by hardware to indicate that the HSI oscillator is stable.

0: HSI clock is not ready (default after reset)

1: HSI clock is ready

Bit 0 **HSION**: HSI clock enable

Set and cleared by software.

Set by hardware to force the HSI to ON when the product leaves stop mode, if STOPWUCK = 1 or STOPKERWUCK = 1.

Set by hardware to force the HSI to ON when the product leaves standby mode or in case of a failure of the HSE which is used as the system clock source.

This bit cannot be cleared if the HSI is used directly as system clock, or if the HSI is selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set to 1).

0: HSI is OFF

1: HSI is ON (default after reset)

10.8.2 RCC HSI calibration register (RCC_HSICFGR)

Address offset: 0x010

Reset value: 0x0040 0XXX

Reset value depends on the flash memory option bytes setting.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSITRIM[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.	HSICAL[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **HSITRIM[6:0]**: HSI clock trimming

Set by software to adjust calibration.

HSITRIM field is added to the engineering option bytes loaded during reset phase (FLASH_HSI_OPT) in order to form the calibration trimming value.

HSICAL = HSITRIM + FLASH_HSI_OPT.

After a change of HSITRIM it takes one system clock cycle before the new HSITRIM value is updated

Note: The reset value of the field is 0x40.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSICAL[11:0]**: HSI clock calibration

Set by hardware by option byte loading during system reset nreseort. Adjusted by software through trimming bits HSITRIM.

This field represents the sum of engineering option byte calibration value and HSITRIM bits value.

10.8.3 RCC clock recovery RC register (RCC_CRRCR)

Address offset: 0x014

Reset value: 0x0000 0XXX

Reset value depends on the flash memory option bytes setting.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	HSI48CAL[9:0]									
						r	r	r	r	r	r	r	r	r	r

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

Bits 9:0 **HSI48CAL[9:0]**: Internal RC 48 MHz clock calibration

Set by hardware by option-byte loading during system reset NRESET. Read-only.

10.8.4 RCC CSI calibration register (RCC_CSICFGR)

Address offset: 0x018

Reset value: 0x0020 00XX

Reset value depends on the flash memory option bytes setting.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSITRIM[5:0]					
										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.	CSICAL[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **CSITRIM[5:0]**: CSI clock trimming

Set by software to adjust calibration.

CSITRIM field is added to the engineering option bytes loaded during reset phase (FLASH_CSI_OPT) in order to form the calibration trimming value.

CSICAL = CSITRIM + FLASH_CSI_OPT.

Note: The reset value of the field is 0x20.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **CSICAL[7:0]**: CSI clock calibration

Set by hardware by option byte loading during system reset NRESET. Adjusted by software through trimming bits CSITRIM.

This field represents the sum of engineering option byte calibration value and CSITRIM bits value.

10.8.5 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 the access occurs during the clock source switch.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCO2SEL[2:0]			MCO2PRE[3:0]				MCO1SEL[2:0]			MCO1PRE[3:0]				Res.	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
TIMPRE	Res.	RTCPRE[5:0]						STOPKERWUCK	STOPWUCK	Res.	SWS[1:0]		Res.	SW[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw		r	r		rw	rw

Bits 31:29 **MCO2SEL[2:0]**: microcontroller clock output 2

Set and cleared by software. Clock source selection may generate glitches on MCO2.
It is highly recommended to configure these bits only after reset, before enabling the external oscillators and the PLLs.
000: system clock selected (**sys_ck**) (default after reset)
001: PLL2 oscillator clock selected (**pll2_p_ck**)
010: HSE clock selected (**hse_ck**)
011: PLL1 clock selected (**pll1_p_ck**)
100: CSI clock selected (**csi_ck**)
101: LSI clock selected (**lsi_ck**)
others: reserved

Bits 28:25 **MCO2PRE[3:0]**: MCO2 prescaler

Set and cleared by software to configure the prescaler of the MCO2. Modification of this prescaler may generate glitches on MCO2. It is highly recommended to change this prescaler only after reset, before enabling the external oscillators and the PLLs.
0000: prescaler disabled (default after reset)
0001: division by 1 (bypass)
0010: division by 2
0011: division by 3
0100: division by 4
...
1111: division by 15

Bits 24:22 **MCO1SEL[2:0]**: Microcontroller clock output 1

Set and cleared by software. Clock source selection may generate glitches on MCO1.
It is highly recommended to configure these bits only after reset, before enabling the external oscillators and the PLLs.
000: HSI clock selected (**hsi_ck**) (default after reset)
001: LSE oscillator clock selected (**lse_ck**)
010: HSE clock selected (**hse_ck**)
011: PLL1 clock selected (**pll1_q_ck**)
100: HSI48 clock selected (**hsi48_ck**)
others: reserved

Bits 21:18 **MCO1PRE[3:0]**: MCO1 prescaler

Set and cleared by software to configure the prescaler of the MCO1. Modification of this prescaler may generate glitches on MCO1. It is highly recommended to change this prescaler only after reset, before enabling the external oscillators and the PLLs.
0000: prescaler disabled (default after reset)
0001: division by 1 (bypass)
0010: division by 2
0011: division by 3
0100: division by 4
...
1111: division by 15

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **TIMPRE**: timers clocks prescaler selection

This bit is set and reset by software to control the clock frequency of all the timers connected to APB1 and APB2 domains.

0: The timers kernel clock is equal to **rcc_hclk1** if PPRE1 or PPRE2 corresponds to a division by 1 or 2, else it is equal to 2 x $\mathbf{F_{rcc\_pclk1}}$ or 2 x $\mathbf{F_{rcc\_pclk2}}$ (default after reset)

1: The timers kernel clock is equal to 2 x $\mathbf{F_{rcc\_pclk1}}$ or 2 x $\mathbf{F_{rcc\_pclk2}}$ if PPRE1 or PPRE2 corresponds to a division by 1, 2 or 4, else it is equal to 4 x $\mathbf{F_{rcc\_pclk1}}$ or 4 x $\mathbf{F_{rcc\_pclk2}}$

Bit 14 Reserved, must be kept at reset value.

Bits 13:8 **RTCPRE[5:0]**: HSE division factor for RTC clock

Set and cleared by software to divide the HSE to generate a clock for RTC.

Caution: The software must set these bits correctly to ensure that the clock supplied to the RTC is lower than 1 MHz. These bits must be configured if needed before selecting the RTC clock source.

000000: no clock (default after reset)

000001: no clock

000010: HSE/2

000011: HSE/3

000100: HSE/4

...

111110: HSE/62

111111: HSE/63

Bit 7 **STOPKERWUCK**: kernel clock selection after a wakeup from system Stop

Set and reset by software to select the kernel wakeup clock from system Stop.

0: HSI (at up to 64 MHz) selected as wakeup clock from system Stop (default after reset)

1: CSI selected as wakeup clock from system Stop

Bit 6 **STOPWUCK**: system clock selection after a wakeup from system Stop

Set and reset by software to select the system wakeup clock from system Stop.

The selected clock is also used as emergency clock for the clock security system (CSS) on HSE.

0: HSI (at up to 64 MHz) selected as wakeup clock from system Stop (default after reset)

1: CSI selected as wakeup clock from system Stop

Caution: STOPWUCK must not be modified when CSS is enabled (by HSECSSON bit) and the system clock is HSE (SWS = 10) or a switch on HSE is requested (SW =10).

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **SWS[1:0]**: system clock switch status
Set and reset by hardware to indicate which clock source is used as system clock.
00: HSI used as system clock (**hsi_ck**) (default after reset).
01: CSI used as system clock (**csi_ck**)
10: HSE used as system clock (**hse_ck**)
11: PLL1 used as system clock (**pll1_p_ck**)

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **SW[1:0]**: system clock and trace clock switch
Set and reset by software to select system clock and trace clock sources (**sys_ck**).
Set by hardware in order to:
-force the selection of the HSI or CSI (depending on STOPWUCK selection) when leaving a system stop mode
-force the selection of the HSI in case of failure of the HSE when used directly or indirectly as system clock
00: HSI selected as system clock (**hsi_ck**) (default after reset)
01: CSI selected as system clock (**csi_ck**)
10: HSE selected as system clock (**hse_ck**)
11: PLL1 selected as system clock (**pll1_p_ck** for **sys_ck**)

10.8.6 RCC CPU domain clock configuration register 2 (RCC_CFGR2)

Address offset: 0x020

Reset value: 0x0000 0000

1 or 2 wait states are inserted only if the access occurs during the clock source switch.

From 0 to 15 wait states are inserted if the access occurs when the APB or AHB prescalers values update is ongoing.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB3DIS	APB2DIS	APB1DIS	Res.	Res.	AHB2DIS	AHB1DIS
									rw	rw	rw			rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PPRE3[2:0]			Res.	PPRE2[2:0]			Res.	PPRE1[2:0]			HPRE[3:0]			
	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 **APB3DIS**: APB3 clock disable value.Set and cleared by software
This bit can be set in order to further reduce power consumption, when none of the APB3 peripherals are used and when their clocks are disabled in RCC_APB3ENR. When this bit is set, all the APB3 peripherals clocks are off.
0: APB3 clock enabled, distributed to peripherals according to their dedicated clock enable control bits
1: APB3 clock disabled

Bit 21 **APB2DIS**: APB2 clock disable value
This bit can be set in order to further reduce power consumption, when none of the APB2 peripherals are used and when their clocks are disabled in RCC_APB2ENR. When this bit is set, all the APB2 peripherals clocks are off.
0: APB2 clock enabled, distributed to peripherals according to their dedicated clock enable control bits
1: APB2 clock disabled

Bit 20 **APB1DIS**: APB1 clock disable value
This bit can be set in order to further reduce power consumption, when none of the APB1 peripherals (except IWDG) are used and when their clocks are disabled in RCC_APB1ENR. When this bit is set, all the APB1 peripherals clocks are off, except for IWDG.
0: APB1 clock enabled, distributed to peripherals according to their dedicated clock enable control bits
1: APB1 clock disabled

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **AHB2DIS**: AHB2 clock disable
This bit can be set in order to further reduce power consumption, when none of the AHB2 peripherals from RCC_AHB2ENR are used and when their clocks are disabled in RCC_AHB2ENR. When this bit is set, all the AHB2 peripherals clocks from RCC_AHB2ENR are off.
0: AHB2 clock enabled, distributed to peripherals according to their dedicated clock enable control bits
1: AHB2 clock disabled

Bit 16 **AHB1DIS**: AHB1 clock disable
This bit can be set in order to further reduce power consumption, when none of the AHB1 peripherals are used (except those listed below), and when their clocks are disabled in RCC_AHB1ENR. When this bit is set, all the AHB1 peripherals clocks from RCC_AHB1ENR are off, except for FLASH, BKPSRAM, ICACHE and SRAM1.
0: AHB1 clock enabled, distributed to peripherals according to their dedicated clock enable control bits
1: AHB1 clock disabled

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **PPRE3[2:0]**: APB low-speed prescaler (APB3)
Set and reset by software to control APB low-speed clocks division factor.
The clocks are divided with the new prescaler factor from 1 to 16 APB cycles after PPRE3 write.
0xx: rcc_pclk3 = rcc_hclk1
100: rcc_pclk3 = rcc_hclk1 / 2
101: rcc_pclk3 = rcc_hclk1 / 4
110: rcc_pclk3 = rcc_hclk1 / 8
111: rcc_pclk3 = rcc_hclk1 / 16

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **PPRE2[2:0]**: APB high-speed prescaler (APB2)
Set and reset by software to control APB high-speed clocks division factor.
The clocks are divided with the new prescaler factor from 1 to 16 APB cycles after PPRE2 write.
0xx: rcc_pclk2 = rcc_hclk1
100: rcc_pclk2 = rcc_hclk1 / 2
101: rcc_pclk2 = rcc_hclk1 / 4
110: rcc_pclk2 = rcc_hclk1 / 8
111: rcc_pclk2 = rcc_hclk1 / 16

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **PPRE1[2:0]**: APB low-speed prescaler (APB1)
Set and reset by software to control the division factor of rcc_pclk1.
The clock is divided by the new prescaler factor from 1 to 16 cycles of rcc_hclk after PPRE write.
0xx: rcc_pclk1 = rcc_hclk1 (default after reset)
100: rcc_pclk1 = rcc_hclk1 / 2
101: rcc_pclk1 = rcc_hclk1 / 4
110: rcc_pclk1 = rcc_hclk1 / 8
111: rcc_pclk1 = rcc_hclk1 / 16

Bits 3:0 **HPRE[3:0]**: AHB prescaler
Set and reset by software to control the division factor of rcc_hclk. Changing this division ratio has an impact on the frequency of all bus matrix clocks
0xxx: rcc_hclk = sys_ck (default after reset)
1000: rcc_hclk = sys_ck / 2
1001: rcc_hclk = sys_ck / 4
1010: rcc_hclk = sys_ck / 8
1011: rcc_hclk = sys_ck / 16
1100: rcc_hclk = sys_ck / 64
1101: rcc_hclk = sys_ck / 128
1110: rcc_hclk = sys_ck / 256
1111: rcc_hclk = sys_ck / 512

Caution: Care must be taken when using the voltage scaling. Due to the propagation delay of the new division factor, after a prescaler factor change and before lowering the VCORE voltage, this register must be read in order to check that the new prescaler value has been taken into account.
Depending on the clock source frequency and the voltage range, the software application must program a correct value in HPRE to make sure that the system frequency does not exceed the maximum frequency.

10.8.7 RCC PLL clock source selection register (RCC_PLL1CFGR)

Address offset: 0x028

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.	PLL1REN	PLL1QEN	PLL1PEN
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	PLL1M[5:0]						Res.	Res.	PLL1VCOSEL	PLL1FRACEN	PLL1RGE[1:0]		PLL1SRC[1:0]	
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **PLL1REN**: PLL1 DIVR divider output enable

Set and reset by software to enable the pll1_r_ck output of the PLL1.
To save power, DIVR1EN and DIVR1 bits must be set to 0 when the pll1_r_ck is not used.
This bit can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
0: pll1_r_ck output disabled (default after reset)
1: pll1_r_ck output enabled

Bit 17 **PLL1QEN**: PLL1 DIVQ divider output enable

Set and reset by software to enable the pll1_q_ck output of the PLL1.
In order to save power, when the pll1_q_ck output of the PLL1 is not used, the pll1_q_ck must be disabled.
This bit can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
0: pll1_q_ck output disabled (default after reset)
1: pll1_q_ck output enabled

Bit 16 **PLL1PEN**: PLL1 DIVP divider output enable

Set and reset by software to enable the pll1_p_ck output of the PLL1.
This bit can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
In order to save power, when the pll1_p_ck output of the PLL1 is not used, the pll1_p_ck must be disabled.
0: pll1_p_ck output disabled (default after reset)
1: pll1_p_ck output enabled

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

Bits 13:8 **PLL1M[5:0]**: prescaler for PLL1

Set and cleared by software to configure the prescaler of the PLL1.
The hardware does not allow any modification of this prescaler when PLL1 is enabled (PLL1ON = 1 or PLL1RDY = 1).
In order to save power when PLL1 is not used, the value of PLL1M must be set to 0.
000000: prescaler disabled (default after reset)
000001: division by 1 (bypass)
000010: division by 2
000011: division by 3
...
100000: division by 32
...
111111: division by 63

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **PLL1VCOSEL**: PLL1 VCO selection

Set and reset by software to select the proper VCO frequency range used for PLL1. This bit must be written before enabling the PLL1.
0: wide VCO range 128 to 560 MHz (default after reset)
1: medium VCO range 150 to 420 MHz

Bit 4 **PLL1FRACEN**: PLL1 fractional latch enable

Set and reset by software to latch the content of FRACN1 into the sigma-delta modulator.
In order to latch the FRACN1 value into the sigma-delta modulator, PLL1FRACEN must be set to 0, then set to 1. The transition 0 to 1 transfers the content of FRACN1 into the modulator.

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 bit must be written before enabling the PLL1.
00: PLL1 input (ref1_ck) clock range frequency between 1 and 2 MHz (default after reset)
01: PLL1 input (ref1_ck) clock range frequency between 2 and 4 MHz
10: PLL1 input (ref1_ck) clock range frequency between 4 and 8 MHz
11: PLL1 input (ref1_ck) clock range frequency between 8 and 16 MHz

Bits 1:0 **PLL1SRC[1:0]**: DIVMx and PLLs clock source selection

Set and reset by software to select the PLL clock source. These bits can be written only when all PLLs are disabled.
In order to save power, when no PLL is used, the value of PLL1SRC must be set to '00'. 00: no clock send to DIVMx divider and PLLs (default after reset).
01: HSI selected as PLL clock (hsi_ck)
10: CSI selected as PLL clock (csi_ck)
11: HSE selected as PLL clock (hse_ck)

10.8.8 RCC PLL clock source selection register (RCC_PLL2CFGR)

Address offset: 0x02C

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.	PLL2REN	PLL2QEN	PLL2PEN
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	PLL2M[5:0]						Res.	Res.	PLL2VCOSEL	PLL2FRACEN	PLL2RGE[1:0]		PLL2SRC[1:0]	
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **PLL2REN**: PLL2 DIVR divider output enable

Set and reset by software to enable the pll2_r_ck output of the PLL2.
To save power, DIVR2EN and DIVR2 bits must be set to 0 when the pll2_r_ck is not used.
0: pll2_r_ck output disabled (default after reset)
1: pll2_r_ck output enabled

Bit 17 **PLL2QEN**: PLL2 DIVQ divider output enable

Set and reset by software to enable the pll2_q_ck output of the PLL2.
To save power, when the pll2_q_ck output of the PLL2 is not used, the pll2_q_ck must be disabled.
0: pll2_q_ck output disabled (default after reset)
1: pll2_q_ck output enabled

Bit 16 **PLL2PEN**: PLL2 DIVP divider output enable

Set and reset by software to enable the pll2_p_ck output of the PLL2.
To save power, when the pll2_p_ck output of the PLL2 is not used, the pll2_p_ck must be disabled.
0: pll2_p_ck output disabled (default after reset)
1: pll2_p_ck output enabled

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

Bits 13:8 **PLL2M[5:0]**: prescaler for PLL2

Set and cleared by software to configure the prescaler of the PLL2.
The hardware does not allow any modification of this prescaler when PLL2 is enabled (PLL2ON = 1 or PLL2RDY = 1).
In order to save power when PLL2 is not used, the value of PLL2M must be set to 0.
000000: prescaler disabled (default after reset)
000001: division by 1 (bypass)
000010: division by 2
000011: division by 3
...
100000: division by 32
...
111111: division by 63

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **PLL2VCOSEL**: PLL2 VCO selection

Set and reset by software to select the proper VCO frequency range used for PLL2.
This bit must be written before enabling the PLL2.
0: wide VCO range 128 to 560 MHz (default after reset)
1: medium VCO range 150 to 420 MHz

Bit 4 **PLL2FRACEN**: PLL2 fractional latch enable

Set and reset by software to enable the pll2_p_ck output of the PLL2.
To save power, when the pll2_p_ck output of the PLL2 is not used, the pll2_p_ck must be disabled.
0: pll2_p_ck output disabled (default after reset)
1: pll2_p_ck output enabled

Bits 3:2 **PLL2RGE[1:0]**: PLL2 input frequency range

Set and reset by software to select the proper reference frequency range used for PLL2.
These bits must be written before enabling the PLL2.
00: PLL2 input (ref2_ck) clock range frequency between 1 and 2 MHz (default after reset)
01: PLL2 input (ref2_ck) clock range frequency between 2 and 4 MHz
10: PLL2 input (ref2_ck) clock range frequency between 4 and 8 MHz
11: PLL2 input (ref2_ck) clock range frequency between 8 and 16 MHz

Bits 1:0 **PLL2SRC[1:0]**: DIVMx and PLLs clock source selection

Set and reset by software to select the PLL clock source.
These bits can be written only when all PLLs are disabled.
In order to save power, when no PLL is used, the value of PLL2SRC must be set to '00'.
00: no clock send to DIVMx divider and PLLs (default after reset)
01: HSI selected as PLL clock (hsi_ck)
10: CSI selected as PLL clock (csi_ck)
11: HSE selected as PLL clock (hse_ck)

10.8.9 RCC PLL1 dividers register (RCC_PLL1DIVR)

Address offset: 0x034

Reset value: 0x0101 0280

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.	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 pll1_r_ck clock.
These bits can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
0000000: pll1_r_ck = vco1_ck
0000001: pll1_r_ck = vco1_ck / 2 (default after reset)
0000010: pll1_r_ck = vco1_ck / 3
0000011: pll1_r_ck = vco1_ck / 4
...
1111111: pll1_r_ck = vco1_ck / 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 pll1_q_ck clock.
These bits can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
0000000: pll1_q_ck = vco1_ck
0000001: pll1_q_ck = vco1_ck / 2 (default after reset)
0000010: pll1_q_ck = vco1_ck / 3
0000011: pll1_q_ck = vco1_ck / 4
...
1111111: pll1_q_ck = vco1_ck / 128

Bits 15:9 **PLL1P[6:0]**: PLL1 DIVP division factor

Set and reset by software to control the frequency of the pll1_p_ck clock.
These bits can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
Note that odd division factors are not allowed.
0000000: Not allowed
0000001: pll1_p_ck = vco1_ck / 2 (default after reset)
0000010: Not allowed
0000011: pll1_p_ck = vco1_ck / 4
...
1111111: pll1_p_ck = vco1_ck / 128

Bits 8:0 **PLL1N[8:0]**: Multiplication factor for PLL1VCO

Set and reset by software to control the multiplication factor of the VCO.

These bits can be written only when the PLL is disabled (PLL1ON = 0 and PLL1RDY = 0).

0x003: PLL1N = 4

0x004: PLL1N = 5

0x005: PLL1N = 6

...

0x080: PLL1N = 129 (default after reset)

...

0x1FF: PLL1N = 512

Others: reserved

10.8.10 RCC PLL1 fractional divider register (RCC_PLL1FRACR)

Address offset: 0x038

Reset value: 0x0000 0000

Access: no wait state; word and half-word 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
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.

Caution: The software must set correctly these bits to ensure that the VCO output frequency is between its valid frequency range, that is:

- 128 to 560 MHz if PLL1VCOSEL = 0
- 150 to 420 MHz if PLL1VCOSEL = 1

VCO output frequency = F_{ref1_ck} × (PLL1N + (PLL1FRACN / 2^{13})), with

- PLL1N between 8 and 420
- PLL1FRACN can be between 0 and 2^{13}- 1
- The input frequency F_{ref1_ck} must be between 1 and 16 MHz.

To change the PLL1FRACN value on-the-fly even if the PLL 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.

10.8.11 RCC PLL1 dividers register (RCC_PLL2DIVR)

Address offset: 0x03C

Reset value: 0x0101 0280

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.	PLL2R[6:0]							Res.	PLL2Q[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
PLL2P[6:0]								PLL2N[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 **PLL2R[6:0]**: PLL2 DIVR division factor

Set and reset by software to control the frequency of the pll2_r_ck clock.
These bits can be written only when the PLL1 is disabled (PLL2ON = 0 and PLL2RDY = 0).
0000000: pll2_r_ck = vco2_ck
0000001: pll2_r_ck = vco2_ck / 2 (default after reset)
0000010: pll2_r_ck = vco2_ck / 3
0000011: pll2_r_ck = vco2_ck / 4
...
1111111: pll2_r_ck = vco2_ck / 128

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **PLL2Q[6:0]**: PLL2 DIVQ division factor

Set and reset by software to control the frequency of the pll2_q_ck clock.
These bits can be written only when the PLL2 is disabled (PLL2ON = 0 and PLL2RDY = 0).
0000000: pll2_q_ck = vco2_ck
0000001: pll2_q_ck = vco2_ck / 2 (default after reset)
0000010: pll2_q_ck = vco2_ck / 3
0000011: pll2_q_ck = vco2_ck / 4
...
1111111: pll2_q_ck = vco2_ck / 128

Bits 15:9 **PLL2P[6:0]**: PLL2 DIVP division factor

Set and reset by software to control the frequency of the pll2_p_ck clock.
These bits can be written only when the PLL2 is disabled (PLL2ON = 0 and PLL2RDY = 0).
0000000: pll2_p_ck = vco2_ck
0000001: pll2_p_ck = vco2_ck / 2 (default after reset)
0000010: pll2_p_ck = vco2_ck / 3
0000011: pll2_p_ck = vco2_ck / 4
...
1111111: pll2_p_ck = vco2_ck / 128

Bits 8:0 **PLL2N[8:0]**: Multiplication factor for PLL2VCO

Set and reset by software to control the multiplication factor of the VCO.

These bits can be written only when the PLL is disabled (PLL2ON = 0 and PLL2RDY = 0).

0x003: PLL2N = 4

0x004: PLL2N = 5

0x005: PLL2N = 6

...

0x080: PLL2N = 129 (default after reset)

...

0x1FF: PLL2N = 512

Others: reserved

10.8.12 RCC PLL2 fractional divider register (RCC_PLL2FRACR)

Address offset: 0x040

Reset value: 0x0000 0000

Access: no wait state; word and half-word 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
PLL2FRACN[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 **PLL2FRACN[12:0]:** fractional part of the multiplication factor for PLL2 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 PLL2 VCO.

Caution: The software must set correctly these bits to ensure that the VCO output frequency is between its valid frequency range, that is:

- 128 to 560 MHz if PLL2VCOSEL = 0
- 150 to 420 MHz if PLL2VCOSEL = 1

VCO output frequency = F_{ref2_ck} × (PLL2N + (PLL2FRACN / 2^{13})), with

- PLL2N between 8 and 420
- PLL2FRACN can be between 0 and 2^{13}- 1
- The input frequency F_{ref2_ck} must be between 1 and 16 MHz.

To change the PLL2FRACN value on-the-fly even if the PLL is enabled, the application must proceed as follows:

- Set the bit PLL2FRACEN to 0
- Write the new fractional value into PLL2FRACN
- Set the bit PLL2FRACEN to 1

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

10.8.13 RCC clock source interrupt enable register (RCC_CIER)

Address offset: 0x050

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.	PLL2RDYIE	PLL1RDYIE	HSI48RDYIE	HSERDYIE	HSIRDYIE	CSIRDYIE	LSERDYIE	LSIRDYIE
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **PLL2RDYIE**: PLL2 ready interrupt enable
Set and reset by software to enable/disable interrupt caused by PLL2 lock.
0: PLL2 lock interrupt disabled (default after reset)
1: PLL2 lock interrupt enabled

Bit 6 **PLL1RDYIE**: PLL1 ready interrupt enable
Set and reset by software to enable/disable interrupt caused by PLL1 lock.
0: PLL1 lock interrupt disabled (default after reset)
1: PLL1 lock interrupt enabled

Bit 5 **HSI48RDYIE: HSI48 ready interrupt enable**
Set and reset by software to enable/disable interrupt caused by the HSI48 oscillator stabilization.
0: HSI48 ready interrupt disabled (default after reset)
1: HSI48 ready interrupt enabled

Bit 4 **HSERDYIE**: HSE ready interrupt enable
Set and reset by software to enable/disable interrupt caused by the HSE oscillator stabilization.
0: HSE ready interrupt disabled (default after reset)
1: HSE ready interrupt enabled

Bit 3 **HSIRDYIE**: HSI ready interrupt enable
Set and reset by software to enable/disable interrupt caused by the HSI oscillator stabilization.
0: HSI ready interrupt disabled (default after reset)
1: HSI ready interrupt enabled

Bit 2 **CSIRDYIE**: CSI ready interrupt enable
Set and reset by software to enable/disable interrupt caused by the CSI oscillator stabilization.
0: CSI ready interrupt disabled (default after reset)
1: CSI ready interrupt enabled

Bit 1 **LSERDYIE**: LSE ready interrupt enable
Set and reset by software to enable/disable interrupt caused by the LSE oscillator stabilization.
0: LSE ready interrupt disabled (default after reset)
1: LSE ready interrupt enabled

Bit 0 **LSIRDYIE**: LSI ready interrupt enable
Set and reset by software to enable/disable interrupt caused by the LSI oscillator stabilization.
0: LSI ready interrupt disabled (default after reset)
1: LSI ready interrupt enabled

10.8.14 RCC clock source interrupt flag register (RCC_CIFR)

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.	HSECSSF	Res.	Res.	PLL2RDYF	PLL1RDYF	HSI48RDYF	HSERDYF	HSIRDYF	CSIRDYF	LSERDYF	LSIRDYF
					r			r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **HSECSSF**: HSE clock security system interrupt flag
Reset by software by writing HSECSSC bit.
Set by hardware in case of HSE clock failure.
0: no clock security interrupt caused by HSE clock failure (default after reset)
1: clock security interrupt caused by HSE clock failure

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **PLL2RDYF**: PLL2 ready interrupt flag
Reset by software by writing PLL2RDYC bit.
Set by hardware when the PLL2 locks and PLL2RDYIE is set.
0: no clock-ready interrupt caused by PLL2 lock (default after reset)
1: clock ready interrupt caused by PLL2 lock

Bit 6 **PLL1RDYF**: PLL1 ready interrupt flag
Reset by software by writing PLL1RDYC bit.
Set by hardware when the PLL1 locks and PLL1RDYIE is set.
0: no clock-ready interrupt caused by PLL1 lock (default after reset)
1: clock ready interrupt caused by PLL1 lock

Bit 5 **HSI48RDYF**: HSI48 ready interrupt flag
Reset by software by writing HSI48RDYC bit.
Set by hardware when the HSI48 clock becomes stable and HSI48RDYIE is set.
0: no clock-ready interrupt caused by the HSI48 oscillator (default after reset)
1: clock ready interrupt caused by the HSI48 oscillator

Bit 4 **HSERDYF**: HSE ready interrupt flag
Reset by software by writing HSERDYC bit.
Set by hardware when the HSE clock becomes stable and HSERDYIE is set.
0: no clock-ready interrupt caused by the HSE (default after reset)
1: clock ready interrupt caused by the HSE

Bit 3 **HSIRDYF**: HSI ready interrupt flag
Reset by software by writing HSIRDYC bit.
Set by hardware when the HSI clock becomes stable and HSIRDYIE is set.
0: no clock-ready interrupt caused by the HSI (default after reset)
1: clock ready interrupt caused by the HSI

Bit 2 **CSIRDYF**: CSI ready interrupt flag
Reset by software by writing CSIRDYC bit.
Set by hardware when the CSI clock becomes stable and CSIRDYIE is set.
0: no clock-ready interrupt caused by the CSI (default after reset)
1: clock ready interrupt caused by the CSI

Bit 1 **LSERDYF**: LSE ready interrupt flag
Reset by software by writing LSERDYC bit.
Set by hardware when the LSE clock becomes stable and LSERDYIE is set.
0: no clock-ready interrupt caused by the LSE (default after reset)
1: clock ready interrupt caused by the LSE

Bit 0 **LSIRDYF**: LSI ready interrupt flag
Reset by software by writing LSIRDYC bit.
Set by hardware when the LSI clock becomes stable and LSIRDYIE is set.
0: no clock-ready interrupt caused by the LSI (default after reset)
1: clock ready interrupt caused by the LSI

10.8.15 RCC clock source interrupt clear register (RCC_CICR)

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	HSECSSC	Res.	Res.	PLL2RDYC	PLL1RDYC	HSI48RDYC	HSERDYC	HSIRDYC	CSIRDYC	LSERDYC	LSIRDYC
					rc_w1			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **HSECSSC**: HSE clock security system interrupt clear
Set by software to clear HSECSSF.
Reset by hardware when clear done.
0: HSECSSF no effect (default after reset)
1: HSECSSF cleared

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **PLL2RDYC**: PLL2 ready interrupt clear
Set by software to clear PLL2RDYF.
Reset by hardware when clear done.
0: PLL2RDYF no effect (default after reset)
1: PLL2RDYF cleared

Bit 6 **PLL1RDYC**: PLL1 ready interrupt clear
Set by software to clear PLL1RDYF.
Reset by hardware when clear done.
0: PLL1RDYF no effect (default after reset)
1: PLL1RDYF cleared

Bit 5 **HSI48RDYC**: HSI48 ready interrupt clear
Set by software to clear HSI48RDYF.
Reset by hardware when clear done.
0: HSI48RDYF no effect (default after reset)
1: HSI48RDYF cleared

Bit 4 **HSERDYC**: HSE ready interrupt clear
Set by software to clear HSERDYF.
Reset by hardware when clear done.
0: HSERDYF no effect (default after reset)
1: HSERDYF cleared

Bit 3 **HSIRDYC**: HSI ready interrupt clear
Set by software to clear HSIRDYF.
Reset by hardware when clear done.
0: HSIRDYF no effect (default after reset)
1: HSIRDYF cleared

Bit 2 **CSIRDYC**: HSI ready interrupt clear
Set by software to clear CSIRDYF.
Reset by hardware when clear done.
0: CSIRDYF no effect (default after reset)
1: CSIRDYF cleared

Bit 1 **LSERDYC**: LSE ready interrupt clear
Set by software to clear LSERDYF.
Reset by hardware when clear done.
0: LSERDYF no effect (default after reset)
1: LSERDYF cleared

Bit 0 **LSIRDYC**: LSI ready interrupt clear
Set by software to clear LSIRDYF.
Reset by hardware when clear done.
0: LSIRDYF no effect (default after reset)
1: LSIRDYF cleared

10.8.16 RCC AHB1 reset register (RCC_AHB1RSTR)

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.	RAMCFGRST	Res.
														rw	

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.	GPDMA2RST	GPDMA1RST
			rw											rw	rw

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

Bit 17 **RAMCFGRST**: RAMCFG block reset
Set and reset by software.
0: does not reset RAMCFG block (default after reset)
1: resets RAMCFG block

Bits 16:13 Reserved, must be kept at reset value.

Bit 12 **CRCRST**: CRC block reset Set and reset by software.
0: does not reset CRC block (default after reset)
1: resets CRC block

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **GPDMA2RST**: GPDMA2 block reset
Set and reset by software.
0: does not reset GPDMA2 block (default after reset)
1: resets GPDMA2 block

Bit 0 **GPDMA1RST**: GPDMA1 block reset
Set and reset by software.
0: does not reset GPDMA1 block (default after reset)
1: resets GPDMA1 block

10.8.17 RCC AHB2 peripheral reset register (RCC_AHB2RSTR)

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.	RNGRST	HASHRST	Res.
													rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DAC1RST	ADCRST	Res.	Res.	GPIOHRST	Res.	Res.	Res.	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
				rw	rw			rw				rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **RNGRST**: RNG block reset
Set and reset by software.
0: does not reset RNG block (default after reset)
1: resets RNG block

Bit 17 **HASHRST**: HASH block reset
Set and reset by software.
0: does not reset HASH block (default after reset)
1: resets HASH block

Bits 16:12 Reserved, must be kept at reset value.

Bit 11 **DAC1RST**: DAC block reset
Set and reset by software.
0: does not reset DAC block (default after reset)
1: resets DAC block

Bit 10 **ADCRST**: ADC1 block reset
Set and reset by software.
0: does not reset ADC1 block (default after reset)
1: resets ADC1 block

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHRST**: GPIOH block reset
Set and reset by software.
0: does not reset the GPIOH block (default after reset)
1: resets the GPIOH block

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

Bit 3 **GPIODRST**: GPIOD block reset
Set and reset by software.
0: does not reset the GPIOD block (default after reset)
1: resets the GPIOD block

Bit 2 **GPIOCRST**: GPIOC block reset
Set and reset by software.
0: does not reset the GPIOC block (default after reset)
1: resets the GPIOC block

Bit 1 **GPIOBRST**: GPIOB block reset
Set and reset by software.
0: does not reset the GPIOB block (default after reset)
1: resets the GPIOB block

Bit 0 **GPIOARST**: GPIOA block reset
Set and reset by software.
0: does not reset the GPIOA block (default after reset)
1: resets the GPIOA block

10.8.18 RCC APB1 peripheral low reset register (RCC_APB1LRSTR)

Address offset: 0x074

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.	CRSRST	I3C1RST	I2C2RST	I2C1RST	Res.	Res.	USART3RST	USART2RST	COMPRST
							rw	rw	rw	rw			rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3RST	SPI2RST	OPAMPRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM7RST	TIM6RST	Res.	Res.	TIM3RST	TIM2RST
rw	rw	rw								rw	rw			rw	rw

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

Bit 24 **CRSRST**: CRS block reset
Set and reset by software.
0: does not reset the CRS block (default after reset)
1: resets the CRS block

Bit 23 **I3C1RST**: I3C1 block reset
Set and reset by software.
0: does not reset the I3C1 block (default after reset)
1: resets the I3C1 block

Bit 22 **I2C2RST**: I2C2 block reset
Set and reset by software.
0: does not reset the I2C2 block (default after reset)
1: resets the I2C2 block

Bit 21 **I2C1RST**: I2C1 block reset
Set and reset by software.
0: does not reset the I2C1 block (default after reset)
1: resets the I2C1 block

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **USART3RST**: USART3 block reset
Set and reset by software.
0: does not reset the USART3 block (default after reset)
1: resets the USART3 block

Bit 17 **USART2RST**: USART2 block reset
Set and reset by software.
0: does not reset the USART2 block (default after reset)
1: resets the USART2 block

Bit 16 **COMPRST**: COMP block reset
Set and reset by software.
0: does not reset the COMP block (default after reset)
1: resets the COMP block

Bit 15 **SPI3RST**: SPI3 block reset
Set and reset by software.
0: does not reset the SPI3 block (default after reset)
1: resets the SPI3 block

Bit 14 **SPI2RST**: SPI2 block reset
Set and reset by software.
0: does not reset the SPI2 block (default after reset)
1: resets the SPI2 block

Bit 13 **OPAMPRST**: OPAMP block reset
Set and reset by software.
0: does not reset the OPAMP block (default after reset)
1: resets the OPAMP block

Bits 12:6 Reserved, must be kept at reset value.

Bit 5 **TIM7RST**: TIM7 block reset
Set and reset by software.
0: does not reset the TIM7 block (default after reset)
1: resets the TIM7 block

Bit 4 **TIM6RST**: TIM6 block reset
Set and reset by software.
0: does not reset the TIM6 block (default after reset)
1: resets the TIM6 block

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM3RST**: TIM3 block reset
Set and reset by software.
0: does not reset the TIM3 block (default after reset)
1: resets the TIM3 block

Bit 0 **TIM2RST**: TIM2 block reset
Set and reset by software.
0: does not reset the TIM2 block (default after reset)
1: resets the TIM2 block

10.8.19 RCC APB1 peripheral high reset register (RCC_APB1HRSTR)

Address offset: 0x078

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.	FDCANRST	Res.	Res.	Res.	LPTIM2RST	Res.	DTSRST	Res.	Res.	Res.
						rw				rw		rw			

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

Bit 9 **FDCANRST**: FDCAN1 block reset
Set and reset by software.
0: does not reset the FDCAN1 block (default after reset)
1: resets the FDCAN1 block

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2RST**: LPTIM2 block reset
Set and reset by software.
0: does not reset the LPTIM2 block (default after reset)
1: resets the LPTIM2 block

Bit 4 Reserved, must be kept at reset value.

Bit 3 **DTSRST**: DTS block reset
Set and reset by software.
0: does not reset the DTS block (default after reset)
1: resets the DTS block

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

10.8.20 RCC APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x07C

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.	USBRST	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.	USART1RST	Res.	SPI1RST	TIM1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw	rw											

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

Bit 24 **USBRST**: USB block reset
Set and reset by software.
0: does not reset the USB block (default after reset)
1: resets the USB block

Bits 23:15 Reserved, must be kept at reset value.

Bit 14 **USART1RST**: USART1 block reset
Set and reset by software.
0: does not reset the USART1 block (default after reset)
1: resets the USART1 block

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1RST**: SPI1 block reset
Set and reset by software.
0: does not reset the SPI1 block (default after reset)
1: resets the SPI1 block

Bit 11 **TIM1RST**: TIM1 block reset
Set and reset by software.
0: does not reset the TIM1 block (default after reset)
1: resets the TIM1 block

Bits 10:0 Reserved, must be kept at reset value.

10.8.21 RCC APB3 peripheral reset register (RCC_APB3RSTR)

Address offset: 0x080

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.	LPTIM1RST	Res.	I3C2RST	Res.	Res.	LPUART1RST	Res.	Res.	Res.	Res.	Res.	Res.
				rw		rw			rw						

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **LPTIM1RST**: LPTIM1 block reset
Set and reset by software.
0: does not reset the LPTIM1 block (default after reset)
1: resets the LPTIM1 block

Bit 10 Reserved, must be kept at reset value.

Bit 9 **I3C2RST**: I3C2RST block reset
Set and reset by software.
0: does not reset the I3C2RST block (default after reset)
1: resets the I3C2RST block

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **LPUART1RST**: LPUART1 block reset
Set and reset by software.
0: does not reset the LPUART1 block (default after reset)
1: resets the LPUART1 block

Bits 5:0 Reserved, must be kept at reset value.

10.8.22 RCC AHB1 peripherals clock register (RCC_AHB1ENR)

Address offset: 0x088

Reset value: 0x9000 0100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM1EN	Res.	Res.	BKPRAMEN	Res.	Res.	Res.	GTZC1EN	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGEN	Res.
rw			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.	FLITFEN	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA2EN	GPDMA1EN
			rw				rw							rw	rw

Bit 31 **SRAM1EN**:SRAM1 clock enable
Set and reset by software.
0: SRAM1 clock disabled
1: SRAM1 clock enabled (default after reset)

Bits 30:29 Reserved, must be kept at reset value.

Bit 28 **BKPRAMEN**: BKPRAM clock enable
Set and reset by software
0: BKPRAM peripheral clock disabled (default after reset)
1: BKPRAM peripheral clock enabled

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **GTZC1EN**: GTZC1 clock enable
Set and reset by software
0: GTZC1 peripheral clock disabled (default after reset)
1: GTZC1 peripheral clock enabled

Bits 23:18 Reserved, must be kept at reset value.

Bit 17 **RAMCFGEN**: RAMCFG clock enable
Set and reset by software.
0: RAMCFG peripheral clock disabled (default after reset)
1: RAMCFG peripheral clock enabled

Bits 16:13 Reserved, must be kept at reset value.

Bit 12 **CRCEN**: CRC clock enable
Set and reset by software.
0: CRC peripheral clock disabled (default after reset)
1: CRC peripheral clock enabled

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLITFEN**: Flash interface clock enable
Set and reset by software.
0: FLASH interface clock disabled
1: FLASH interface clock enabled (default after reset)

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

Bit 1 **GPDMA2EN**: GPDMA2 clock enable
Set and reset by software.
0: GPDMA2 peripheral clock disabled (default after reset)
1: GPDMA2 peripheral clock enabled

Bit 0 **GPDMA1EN**: GPDMA1 clock enable
Set and reset by software.
0: GPDMA1 peripheral clock disabled (default after reset)
1: GPDMA1 peripheral clock enabled

10.8.23 RCC AHB2 peripheral clock register (RCC_AHB2ENR)

Address offset: 0x08C

Reset value: 0x4000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SRAM2EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEN	HASHEN	Res.
	rw												rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DAC12EN	ADCEN	Res.	Res.	GPIOHEN	Res.	Res.	Res.	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
				rw	rw			rw				rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SRAM2EN**: SRAM2 clock enable
Set and reset by software.
0: SRAM2 clock disabled
1: SRAM2 clock enabled (default after reset)

Bits 29:19 Reserved, must be kept at reset value.

Bit 18 **RNGEN**: RNG clock enable
Set and reset by software.
0: RNG peripheral clock disabled (default after reset)
1: RNG peripheral clock enabled

Bit 17 **HASHEN**: HASH clock enable
Set and reset by software.
0: HASH peripheral clock disabled (default after reset)
1: HASH peripheral clock enabled

Bits 16:12 Reserved, must be kept at reset value.

Bit 11 **DAC12EN**: DAC clock enable
Set and reset by software.
0: DAC peripheral clock disabled (default after reset)
1: DAC peripheral clock enabled

Bit 10 **ADCEN**: ADC1 peripherals clock enabled
Set and reset by software.
0: ADC1 peripherals clock disabled (default after reset)
1: ADC1 peripherals clock enabled

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHEN**: GPIOH clock enable
Set and reset by software.
0: GPIOH peripheral clock disabled (default after reset)
1: GPIOH peripheral clock enabled

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

Bit 3 **GPIODEN**: GPIOD clock enable
Set and reset by software.
0: GPIOD peripheral clock disabled (default after reset)
1: GPIOD peripheral clock enabled

Bit 2 **GPIOCEN**: GPIOC clock enable
Set and reset by software.
0: GPIOC peripheral clock disabled (default after reset)
1: GPIOC peripheral clock enabled

Bit 1 **GPIOBEN**: GPIOB clock enable
Set and reset by software.
0: GPIOB peripheral clock disabled (default after reset)
1: GPIOB peripheral clock enabled

Bit 0 **GPIOAEN**: GPIOA clock enable
Set and reset by software.
0: GPIOA peripheral clock disabled (default after reset)
1: GPIOA peripheral clock enabled

10.8.24 RCC APB1 peripheral clock register (RCC_APB1LENR)

Address offset: 0x09C

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.	CRSEN	I3C1EN	I2C2EN	I2C1EN	Res.	Res.	USART3EN	USART2EN	COMPEN
							rw	rw	rw	rw			rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3EN	SPI2EN	OPAMPEN	Res.	WWDGEN	Res.	Res.	Res.	Res.	Res.	TIM7EN	TIM6EN	Res.	Res.	TIM3EN	TIM2EN
rw	rw	rw		rw						rw	rw			rw	rw

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

Bit 24 **CRSEN**: CRS clock enable
Set and reset by software.
0: CRS peripheral clock disabled (default after reset)
1: CRS peripheral clock enabled

Bit 23 **I3C1EN**: I3C1 clock enable
Set and reset by software.
0: I3C1 peripheral clock disabled (default after reset)
1: I3C1 peripheral clock enabled

Bit 22 **I2C2EN**: I2C2 clock enable
Set and reset by software.
0: I2C2 peripheral clock disabled (default after reset)
1: I2C2 peripheral clock enabled

Bit 21 **I2C1EN**: I2C1 clock enable
Set and reset by software.
0: I2C1 peripheral clock disabled (default after reset)
1: I2C1 peripheral clock enabled

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **USART3EN**: USART3 clock enable
Set and reset by software.
0: USART3 peripheral clock disabled (default after reset)
1: USART3 peripheral clock enabled

Bit 17 **USART2EN**: USART2 clock enable
Set and reset by software.
0: USART2 peripheral clock disabled (default after reset)
1: USART2 peripheral clock enabled

Bit 16 **COMPEN**: COMP clock enable
Set and reset by software.
0: COMP peripheral clock disabled (default after reset)
1: COMP peripheral clock enabled

Bit 15 **SPI3EN**: SPI3 clock enable
Set and reset by software.
0: SPI3 peripheral clock disabled (default after reset)
1: SPI3 peripheral clock enabled

Bit 14 **SPI2EN**: SPI2 clock enable
Set and reset by software.
0: SPI2 peripheral clock disabled (default after reset)
1: SPI2 peripheral clock enabled

Bit 13 **OPAMPEN**: OPAMP clock enable
Set and reset by software.
0: OPAMP peripheral clock disabled (default after reset)
1: OPAMP peripheral clock enabled

Bit 12 Reserved, must be kept at reset value.

Bit 11 **WWDGEN**: WWDG clock enable
Set and reset by software.
0: WWDG peripheral clock disabled (default after reset)
1: WWDG peripheral clock enabled

Bits 10:6 Reserved, must be kept at reset value.

Bit 5 **TIM7EN**: TIM7 clock enable
Set and reset by software.
0: TIM7 peripheral clock disabled (default after reset)
1: TIM7 peripheral clock enabled

Bit 4 **TIM6EN**: TIM6 clock enable
Set and reset by software.
0: TIM6 peripheral clock disabled (default after reset)
1: TIM6 peripheral clock enabled

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM3EN**: TIM3 clock enable
Set and reset by software.
0: TIM3 peripheral clock disabled (default after reset)
1: TIM3 peripheral clock enabled

Bit 0 **TIM2EN**: TIM2 clock enable
Set and reset by software.
0: TIM2 peripheral clock disabled (default after reset)
1: TIM2 peripheral clock enabled

10.8.25 RCC APB1 peripheral clock register (RCC_APB1HENR)

Address offset: 0x0A0

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.	FDCANEN	Res.	Res.	Res.	LPTIM2EN	Res.	DTSEN	Res.	Res.	Res.
						rw				rw		rw			

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

Bit 9 **FDCANEN**: FDCAN1 peripheral clock enable
Set and reset by software.
0: FDCAN1 peripheral clock disabled (default after reset)
1: FDCAN1 peripheral clock enabled

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2EN**: LPTIM2 clock enable
Set and reset by software.
0: LPTIM2 peripheral clock disabled (default after reset)
1: LPTIM2 peripheral clock enabled

Bit 4 Reserved, must be kept at reset value.

Bit 3 **DTSEN**: DTS clock enable
Set and reset by software.
0: DTS peripheral clock disabled (default after reset)
1: DTS peripheral clock enabled

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

10.8.26 RCC APB2 peripheral clock register (RCC_APB2ENR)

Address offset: 0x0A4

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.	USBEN	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.	USART1EN	Res.	SPI1EN	TIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw	rw											

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

Bit 24 **USBEN**: USB clock enable
Set and reset by software.
0: USB peripheral clock disabled (default after reset)
1: USB peripheral clock enabled

Bits 23:15 Reserved, must be kept at reset value.

Bit 14 **USART1EN**: USART1 clock enable
Set and reset by software.
0: USART1 peripheral clock disabled (default after reset)
1: USART1 peripheral clock enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1EN**: SPI1 clock enable
Set and reset by software.
0: SPI1 peripheral clock disabled (default after reset)
1: SPI1 peripheral clock enabled

Bit 11 **TIM1EN**: TIM1 clock enable
Set and reset by software.
0: TIM1 peripheral clock disabled (default after reset)
1: TIM1 peripheral clock enabled

Bits 10:0 Reserved, must be kept at reset value.

10.8.27 RCC APB3 peripheral clock register (RCC_APB3ENR)

Address offset: 0x0A8

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.	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.	I3C2EN	Res.	Res.	LPUART1EN	Res.	Res.	Res.	Res.	SBSEN	Res.
				rw		rw			rw					rw	

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **RTCAPBEN**: RTC APB interface clock enable
Set and reset by software.
0: RTC APB interface clock disabled (default after reset)
1: RTC APB interface clock enabled

Bits 20:12 Reserved, must be kept at reset value.

Bit 11 **LPTIM1EN**: LPTIM1 clock enable
Set and reset by software.
0: LPTIM1 peripheral clock disabled (default after reset)
1: LPTIM1 peripheral clock enabled

Bit 10 Reserved, must be kept at reset value.

Bit 9 **I3C2EN**: I3C2EN clock enable
Set and reset by software.
0: I3C2EN peripheral clock disabled (default after reset)
1: I3C2EN peripheral clock enabled

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **LPUART1EN**: LPUART1 clock enable
Set and reset by software.
0: LPUART1 peripheral clock disabled (default after reset)
1: LPUART1 peripheral clock enabled

Bits 5:2 Reserved, must be kept at reset value.

Bit 1 **SBSEN**: SBS clock enable
Set and reset by software.
0: SBS peripheral clock disabled (default after reset)
1: SBS peripheral clock enabled

Bit 0 Reserved, must be kept at reset value.

10.8.28 RCC AHB1 sleep clock register (RCC_AHB1LPENR)

Address offset: 0x0B0

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM1LPEN	Res.	ICACHELPEN	BKPRAMLPEN	Res.	Res.	Res.	GTZC1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGLPEN	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.	CRCLPEN	Res.	Res.	Res.	FLITFLPEN	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA2LPEN	GPDMA1LPEN
			rw				rw							rw	rw

Bit 31 **SRAM1LPEN**: SRAM1 clock enable during sleep mode
Set and reset by software
0: SRAM1 peripheral clock disabled during sleep mode
1: SRAM1 peripheral clock enabled during sleep mode (default after reset)

Bit 30 Reserved, must be kept at reset value.

Bit 29 **ICACHELPEN**: ICACHE clock enable during sleep mode
Set and reset by software
0: ICACHE peripheral clock disabled during sleep mode
1: ICACHE peripheral clock enabled during sleep mode (default after reset)

Bit 28 **BKPRAMLPEN**: BKPRAM clock enable during sleep mode
Set and reset by software
0: BKPRAM peripheral clock disabled during sleep mode
1: BKPRAM peripheral clock enabled during sleep mode (default after reset)

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **GTZC1LPEN**: GTZC1 clock enable during sleep mode
Set and reset by software
0: GTZC1 peripheral clock disabled during sleep mode
1: GTZC1 peripheral clock enabled during sleep mode (default after reset)

Bits 23:18 Reserved, must be kept at reset value.

Bit 17 **RAMCFGLPEN**: RAMCFG clock enable during sleep mode
Set and reset by software.
0: RAMCFG peripheral clock disabled during sleep mode
1: RAMCFG peripheral clock enabled during sleep mode (default after reset)

Bits 16:13 Reserved, must be kept at reset value.

Bit 12 **CRCLPEN**: CRC clock enable during sleep mode
Set and reset by software.
0: CRC peripheral clock disabled during sleep mode
1: CRC peripheral clock enabled during sleep mode (default after reset)

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLITFLPEN**: Flash interface (FLITF) clock enable during sleep mode
Set and reset by software.
0: FLITF peripheral clock disabled during sleep mode
1: FLITF peripheral clock enabled during sleep mode (default after reset)

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

Bit 1 **GPDMA2LPEN**: GPDMA2 clock enable during sleep mode
Set and reset by software.
0: GPDMA2 peripheral clock disabled during sleep mode
1: GPDMA2 peripheral clock enabled during sleep mode (default after reset)

Bit 0 **GPDMA1LPEN**: GPDMA1 clock enable during sleep mode
Set and reset by software.
0: GPDMA1 peripheral clock disabled during sleep mode
1: GPDMA1 peripheral clock enabled during sleep mode (default after reset)

10.8.29 RCC AHB2 sleep clock register (RCC_AHB2LPENR)

Address offset: 0x0B4

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SRAM2LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGLPEN	HASHLPEN	Res.
	rw												rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DAC12LPEN	ADCLPEN	Res.	Res.	GPIOHLPEN	Res.	Res.	Res.	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
				rw	rw			rw				rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SRAM2LPEN**: SRAM2 clock enable during sleep mode
Set and reset by software.
0: SRAM2 peripheral clock disabled during sleep mode
1: SRAM2 peripheral clock enabled during sleep mode (default after reset)

Bits 29:19 Reserved, must be kept at reset value.

Bit 18 **RNGLPEN**: RNG clock enable during sleep mode
Set and reset by software.
0: RNG peripheral clock disabled during sleep mode
1: RNG peripheral clock enabled during sleep mode (default after reset)

Bit 17 **HASHLPEN**: HASH clock enable during sleep mode
Set and reset by software.
0: HASH peripheral clock disabled during sleep mode
1: HASH peripheral clock enabled during sleep mode (default after reset)

Bits 16:12 Reserved, must be kept at reset value.

Bit 11 **DAC12LPEN**: DAC clock enable during sleep mode
Set and reset by software.
0: DAC peripheral clock disabled during sleep mode
1: DAC peripheral clock enabled during sleep mode (default after reset)

Bit 10 **ADCLPEN**: ADC1 peripherals clock enable during sleep mode
Set and reset by software.
0: ADC1 peripherals clock disabled during sleep mode
1: ADC1 peripherals clock enabled during sleep mode (default after reset)

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHLPEN**: GPIOH clock enable during sleep mode
Set and reset by software.
0: GPIOH peripheral clock disabled during sleep mode
1: GPIOH peripheral clock enabled during sleep mode (default after reset)

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

Bit 3 **GPIODLPEN**: GPIOD clock enable during sleep mode
Set and reset by software.
0: GPIOD peripheral clock disabled during sleep mode
1: GPIOD peripheral clock enabled during sleep mode (default after reset)

Bit 2 **GPIOCLPEN**: GPIOC clock enable during sleep mode
Set and reset by software.
0: GPIOC peripheral clock disabled during sleep mode
1: GPIOC peripheral clock enabled during sleep mode (default after reset)

Bit 1 **GPIOBLPEN**: GPIOB clock enable during sleep mode
Set and reset by software.
0: GPIOB peripheral clock disabled during sleep mode
1: GPIOB peripheral clock enabled during sleep mode (default after reset)

Bit 0 **GPIOALPEN**: GPIOA clock enable during sleep mode
Set and reset by software.
0: GPIOA peripheral clock disabled during sleep mode
1: GPIOA peripheral clock enabled during sleep mode (default after reset)

10.8.30 RCC APB1 sleep clock register (RCC_APB1LLPENR)

Address offset: 0x0C4

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRSLPEN	I3C1LPEN	I2C2LPEN	I2C1LPEN	Res.	Res.	USART3LPEN	USART2LPEN	COMPLPEN
							rw	rw	rw	rw			rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3LPEN	SPI2LPEN	OPAMPLPEN	Res.	WWDGLPEN	Res.	Res.	Res.	Res.	Res.	TIM7LPEN	TIM6LPEN	Res.	Res.	TIM3LPEN	TIM2LPEN
rw	rw	rw		rw						rw	rw			rw	rw

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

Bit 24 **CRSLPEN**: CRS clock enable during sleep mode
Set and reset by software.
0: CRS peripheral clock disabled during sleep mode
1: CRS peripheral clock enabled during sleep mode (default after reset)

Bit 23 **I3C1LPEN**: I3C1 clock enable during sleep mode
Set and reset by software.
0: I3C1 peripheral clock disabled during sleep mode
1: I3C1 peripheral clock enabled during sleep mode (default after reset)

Bit 22 **I2C2LPEN**: I2C2 clock enable during sleep mode
Set and reset by software.
0: I2C2 peripheral clock disabled during sleep mode
1: I2C2 peripheral clock enabled during sleep mode (default after reset)

Bit 21 **I2C1LPEN**: I2C1 clock enable during sleep mode
Set and reset by software.
0: I2C1 peripheral clock disabled during sleep mode
1: I2C1 peripheral clock enabled during sleep mode (default after reset)

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **USART3LPEN**: USART3 clock enable during sleep mode
Set and reset by software.
0: USART3 peripheral clock disabled during sleep mode
1: USART3 peripheral clock enabled during sleep mode (default after reset)

Bit 17 **USART2LPEN**: USART2 clock enable during sleep mode
Set and reset by software.
0: USART2 peripheral clock disabled during sleep mode
1: USART2 peripheral clock enabled during sleep mode (default after reset)

Bit 16 **COMPLPEN**: COMP clock enable during sleep mode
Set and reset by software.
0: COMP peripheral clock disabled during sleep mode
1: COMP peripheral clock enabled during sleep mode (default after reset)

Bit 15 **SPI3LPEN**: SPI3 clock enable during sleep mode
Set and reset by software.
0: SPI3 peripheral clock disabled during sleep mode
1: SPI3 peripheral clock enabled during sleep mode (default after reset)

Bit 14 **SPI2LPEN**: SPI2 clock enable during sleep mode
Set and reset by software.
0: SPI2 peripheral clock disabled during sleep mode
1: SPI2 peripheral clock enabled during sleep mode (default after reset)

Bit 13 **OPAMPLPEN**: OPAMP clock enable during sleep mode
Set and reset by software.
0: OPAMP peripheral clock disabled during sleep mode
1: OPAMP peripheral clock enabled during sleep mode (default after reset)

Bit 12 Reserved, must be kept at reset value.

Bit 11 **WWDGLPEN**: WWDG clock enable during sleep mode
Set and reset by software.
0: WWDG peripheral clock disabled during sleep mode
1: WWDG peripheral clock enabled during sleep mode (default after reset)

Bits 10:6 Reserved, must be kept at reset value.

Bit 5 **TIM7LPEN**: TIM7 clock enable during sleep mode
Set and reset by software.
0: TIM7 peripheral clock disabled during sleep mode
1: TIM7 peripheral clock enabled during sleep mode (default after reset)

Bit 4 **TIM6LPEN**: TIM6 clock enable during sleep mode
Set and reset by software.
0: TIM6 peripheral clock disabled during sleep mode
1: TIM6 peripheral clock enabled during sleep mode (default after reset)

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM3LPEN**: TIM3 clock enable during sleep mode
Set and reset by software.
0: TIM3 peripheral clock disabled during sleep mode
1: TIM3 peripheral clock enabled during sleep mode (default after reset)

Bit 0 **TIM2LPEN**: TIM2 clock enable during sleep mode
Set and reset by software.
0: TIM2 peripheral clock disabled during sleep mode
1: TIM2 peripheral clock enabled during sleep mode (default after reset)

10.8.31 RCC APB1 sleep clock register (RCC_APB1HLPENR)

Address offset: 0x0C8

Reset value: 0xFFFF 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.	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.	FDCANLPEN	Res.	Res.	Res.	LPTIM2LPEN	Res.	DTSLPEN	Res.	Res.	Res.
						rw				rw		rw			

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

Bit 9 **FDCANLPEN**: FDCAN1 peripheral clock enable during sleep mode
Set and reset by software.
0: FDCAN1 peripheral clock disabled during sleep mode
1: FDCAN1 peripheral clock enabled during sleep mode (default after reset)

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2LPEN**: LPTIM2 clock enable during sleep mode
Set and reset by software.
0: LPTIM2 peripheral clock disabled during sleep mode
1: LPTIM2 peripheral clock enabled during sleep mode (default after reset)

Bit 4 Reserved, must be kept at reset value.

Bit 3 **DTSLPEN**: DTS clock enable during sleep mode
Set and reset by software.
0: DTS peripheral clock disabled during sleep mode
1: DTS peripheral clock enabled during sleep mode (default after reset)

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

10.8.32 RCC APB2 sleep clock register (RCC_APB2LPENR)

Address offset: 0x0CC

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBLPEN	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.	USART1LPEN	Res.	SPI1LPEN	TIM1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw	rw											

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

Bit 24 **USBLPEN**: USB clock enable during sleep mode
Set and reset by software.
0: USB peripheral clock disabled during sleep mode
1: USB peripheral clock enabled during sleep mode (default after reset)

Bits 23:15 Reserved, must be kept at reset value.

Bit 14 **USART1LPEN**: USART1 clock enable during sleep mode
Set and reset by software.
0: USART1 peripheral clock disabled during sleep mode
1: USART1 peripheral clock enabled during sleep mode (default after reset)

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1LPEN**: SPI1 clock enable during sleep mode
Set and reset by software.
0: SPI1 peripheral clock disabled during sleep mode
1: SPI1 peripheral clock enabled during sleep mode (default after reset)

Bit 11 **TIM1LPEN**: TIM1 clock enable during sleep mode
Set and reset by software.
0: TIM1 peripheral clock disabled during sleep mode
1: TIM1 peripheral clock enabled during sleep mode (default after reset)

Bits 10:0 Reserved, must be kept at reset value.

10.8.33 RCC APB3 sleep clock register (RCC_APB3LPENR)

Address offset: 0x0D0

Reset value: 0xFFFF 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.	RTCAPBLPEN	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.	LPTIM1LPEN	Res.	I3C2LPEN	Res.	Res.	LPUART1LPEN	Res.	Res.	Res.	Res.	SBSLPEN	Res.
				rw		rw			rw					rw	

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **RTCAPBLPEN**: RTC APB interface clock enable during sleep mode
Set and reset by software.
0: RTC APB interface clock disabled during sleep mode
1: RTC APB interface clock enabled during sleep mode (default after reset)

Bits 20:12 Reserved, must be kept at reset value.

Bit 11 **LPTIM1LPEN**: LPTIM1 clock enable during sleep mode
Set and reset by software.
0: LPTIM1 peripheral clock disabled during sleep mode
1: LPTIM1 peripheral clock enabled during sleep mode (default after reset)

Bit 10 Reserved, must be kept at reset value.

Bit 9 **I3C2LPEN**: I3C2 clock enable during sleep mode
Set and reset by software.
0: I3C2 peripheral clock disabled during sleep mode
1: I3C2 peripheral clock enabled during sleep mode (default after reset)

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **LPUART1LPEN**: LPUART1 clock enable during sleep mode
Set and reset by software.
0: LPUART1 peripheral clock disabled during sleep mode
1: LPUART1 peripheral clock enabled during sleep mode (default after reset)

Bits 5:2 Reserved, must be kept at reset value.

Bit 1 **SBSLPEN**: SBS clock enable during sleep mode
Set and reset by software.
0: SBS peripheral clock disabled during sleep mode
1: SBS peripheral clock enabled during sleep mode (default after reset)

Bit 0 Reserved, must be kept at reset value.

10.8.34 RCC kernel clock configuration register (RCC_CCIPR1)

Address offset: 0x0D8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMICSEL	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.	USART3SEL[2:0]			USART2SEL[2:0]			USART1SEL[2:0]		
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMICSEL**: TIM2, TIM3 and LPTIM2 input capture source selection
Set and reset by software.
0: No internal clock available for timers input capture (default after reset)
1: hsi_ker_ck/1024, hsi_ker_ck/8 and csi_ker_ck/128 selected for timers input capture

Bits 30:9 Reserved, must be kept at reset value.

Bits 8:6 **USART3SEL[2:0]**: USART3 kernel clock source selection
Set and reset by software.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
others: reserved, the kernel clock is disabled

Bits 5:3 **USART2SEL[2:0]**: USART2 kernel clock source selection
Set and reset by software.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
others: reserved, the kernel clock is disabled

Bits 2:0 **USART1SEL[2:0]**: USART1 kernel clock source selection
Set and reset by software.
000: rcc_pclk2 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
others: reserved, the kernel clock is disabled

10.8.35 RCC kernel clock configuration register (RCC_CCIPR2)

Address offset: 0x0DC

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.	LPTIM2SEL[2:0]			Res.	LPTIM1SEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw		rw	rw	rw								

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:12 **LPTIM2SEL[2:0]**: LPTIM2 kernel clock source selection
000: rcc_pclk1 selected as kernel clock (default after reset)
001: pll2_p_ck selected as kernel clock
011: lse_ker_ck selected as kernel clock
100: lsi_ker_ck selected as kernel clock
101: per_ck selected as kernel clock
others: reserved, the kernel clock is disabled

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **LPTIM1SEL[2:0]**: LPTIM1 kernel clock source selection
000: rcc_pclk3 selected as kernel clock (default after reset)
001: pll2_p_ck selected as kernel clock
011: lse_ker_ck selected as kernel clock
100: lsi_ker_ck selected as kernel clock
101: per_ck selected as kernel clock
others: reserved, the kernel clock is disabled

Bits 7:0 Reserved, must be kept at reset value.

10.8.36 RCC kernel clock configuration register (RCC_CCIPR3)

Address offset: 0x0E0

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.	LPUART1SEL[2:0]			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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI3SEL[2:0]			SPI2SEL[2:0]			SPI1SEL[2:0]		
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **LPUART1SEL[2:0]**: LPUART1 kernel clock source selection
000: rcc_pclk3 s elected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
others: reserved, the kernel clock is disabled

Bits 23:9 Reserved, must be kept at reset value.

Bits 8:6 **SPI3SEL[2:0]**: SPI3 kernel clock source selection
Set and reset by software.
000: pll1_q_ck selected as kernel clock (default after reset)
001: pll2_p_ck selected as kernel clock
011: AUDIOCLK selected as kernel clock
100: per_ck selected as kernel clock
others: reserved, the kernel clock is disabled

Bits 5:3 **SPI2SEL[2:0]**: SPI2 kernel clock source selection
Set and reset by software.
000: pll1_q_ck selected as kernel clock (default after reset)
001: pll2_p_ck selected as kernel clock
011: AUDIOCLK selected as kernel clock
100: per_ck selected as kernel clock
others: reserved, the kernel clock is disabled

Bits 2:0 **SPI1SEL[2:0]**: SPI1 kernel clock source selection
Set and reset by software.
000: pll1_q_ck selected as kernel clock (default after reset)
001: pll2_p_ck selected as kernel clock
011: AUDIOCLK selected as kernel clock
100: per_ck selected as kernel clock
others: reserved, the kernel clock is disabled

10.8.37 RCC kernel clock configuration register (RCC_CCIPR4)

Address offset: 0x0E4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	I3C2SEL[1:0]		I3C1SEL[1:0]		Res.	Res.	Res.	Res.	I2C2SEL[1:0]		I2C1SEL[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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBSEL[1:0]		SYSTICKSEL[1:0]		Res.	Res.
										rw	rw	rw	rw		

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

Bits 27:26 **I3C2SEL[1:0]**: I3C2 kernel clock source selection
00: rcc_pclk3 selected as kernel clock (default after reset)
01: pll2_r_ck selected as kernel clock
10: hsi_ker_ck selected as kernel clock
11: no clock selected

Bits 25:24 **I3C1SEL[1:0]**: I3C1 kernel clock source selection
00: rcc_pclk1 selected as kernel clock (default after reset)
01: pll2_r_ck selected as kernel clock
10: hsi_ker_ck selected as kernel clock
11: no clock selected

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

Bits 19:18 **I2C2SEL[1:0]**: I2C2 kernel clock source selection
00: rcc_pclk1 selected as kernel clock (default after reset)
01: pll2_r_ck selected as kernel clock
10: hsi_ker_ck selected as kernel clock
11: csi_ker_ck selected as kernel clock

Bits 17:16 **I2C1SEL[1:0]**: I2C1 kernel clock source selection
00: rcc_pclk1 selected as kernel clock (default after reset)
01: pll2_r_ck selected as kernel clock
10: hsi_ker_ck selected as kernel clock
11: csi_ker_ck selected as kernel clock

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:4 **USBSEL[1:0]**: USB kernel clock source selection
00: no clock is selected as kernel clock (default after reset)
01: pll1_q_ck selected as kernel clock
10: pll2_q_ck selected as kernel clock
11: hsi48_ker_ck selected as kernel clock

Bits 3:2 **SYSTICKSEL[1:0]**: SYSTICK clock source selection
00: rcc_hclk/8 selected as clock source (default after reset)
01: lsi_ker_ck[1] selected as clock source
10: lse_ck[1] selected as clock source
11: reserved, the kernel clock is disabled

*Note: rcc_hclk frequency must be four times higher than lsi_ker_ck/lse_ck (period (LSI/LSE) ≥ 4 * period (HCLK).*

Bits 1:0 Reserved, must be kept at reset value.

10.8.38 RCC kernel clock configuration register (RCC_CCIPR5)

Address offset: 0x0E8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CKPERSEL[1:0]		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.	FDCANSEL [1:0]		Res.	Res.	RNGSEL[1:0]		DAC1SEL	ADCDACSEL[2:0]		
						rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 **CKPERSEL[1:0]**: per_ck clock source selection
- 00: hsi_ker_ck selected as kernel clock (default after reset)
- 01: csi_ker_ck selected as kernel clock
- 10: hse_ck selected as kernel clock
- 11: reserved, the per_ck clock is disabled

Bits 29:10 Reserved, must be kept at reset value.

Bits 9:8 **FDCANSEL[1:0]**: FDCAN1 kernel clock source selection
- 00: hse_ck selected as kernel clock (default after reset)
- 01: pll1_q_ck selected as kernel clock
- 10: pll2_q_ck selected as kernel clock
- 11: reserved, the kernel clock is disabled

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **RNGSEL[1:0]**: RNG kernel clock source selection
- 00: hsi48_ker_ck selected as kernel clock (default after reset)
- 01: pll1_q_ck selected as kernel clock
- 10: lse_ck selected as kernel clock
- 11: lsi_ker_ck selected as kernel clock

Bit 3 **DAC1SEL**: DAC1 sample and hold clock source selection

This bit is used to select the DAC1 sample and hold clock source (dac_hold_ck).
- 0: LSE selected
- 1: LSI selected

Bits 2:0 **ADCDACSEL[2:0]**: ADC and DAC kernel clock source selection
- 000: rcc_hclk selected as kernel clock (default after reset)
- 001: sys_ck selected as kernel clock
- 010: pll2_r_ck selected as kernel clock
- 011: hse_ck selected as kernel clock
- 100: hsi_ker_ck selected as kernel clock
- 101: csi_ker_ck selected as kernel clock
- others: reserved, the kernel clock is disabled

10.8.39 RCC backup domain control register (RCC_BDCR)

Address offset: 0x0F0

Reset value: 0x0000 0000

Reset by backup domain reset.

Access: 0 ≤ wait state ≤ 3, word, half-word, and byte access.

Wait states are inserted in case of successive accesses to this register.

Note: After a system reset, the RCC_BDCR register is write-protected (except bits 27:26 bits). To modify the backup domain bits, the DBP bit in the PWR_DBPCR register must be set to 1. RCC_BDCR bits (except bits 27:24 and bit 16) are only reset after a backup domain reset (see Section 10.3.3: Backup domain reset). Any other reset does not have any effect on these bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	LSIRDY	LSION	LSCOSEL	LSCOEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VSWRST
				r	rw	rw	rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Res.	Res.	Res.	Res.	Res.	RTCSEL[1:0]		LSEEXT	LSECSSD	LSECSSON	LSEDRV[1:0]		LSEBYP	LSERDY	LSEON
rw						rw	rw	rw	r	rw	rw	rw	rw	r	rw

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

Bit 27 **LSIRDY**: LSI oscillator ready
Set and cleared by hardware to indicate when the LSI oscillator is stable.
After the LSION bit is cleared, LSIRDY goes low after three internal low-speed oscillator clock cycles.
This bit is set when the LSI is used by IWDG or RTC, even if LSION = 0.
0: LSI oscillator not ready
1: LSI oscillator ready

Bit 26 **LSION**: LSI oscillator enable
Set and cleared by software.
0: LSI oscillator off
1: LSI oscillator on

Bit 25 **LSCOSEL**: Low-speed clock output selection
Set and cleared by software.
0: LSI clock selected
1: LSE clock selected

Bit 24 **LSCOEN**: Low-speed clock output (LSCO) enable
Set and cleared by software.
0: LSCO output disabled
1: LSCO output enabled

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **VSWRST**: VSwitch domain software reset
Set and reset by software.
0: reset not activated (default after Backup domain reset)
1: resets the entire VSW domain

Bit 15 **RTCEN**: RTC clock enable
Set and reset by software.
0: rtc_ck disabled (default after Backup domain reset)
1: rtc_ck enabled

Bits 14:10 Reserved, must be kept at reset value.

Bits 9:8 **RTCSEL[1:0]**:RTC clock source selection
Set by software to select the clock source for the RTC.
These bits can be written only one time (except in case of failure detection on LSE).
These bits must be written before LSECSSON is enabled.
The VSWRST bit can be used to reset them, then it can be written one time again.
If HSE is selected as RTC clock, this clock is lost when the system is in stop mode or in case of a pin reset (NRST).
00: no clock (default after Backup domain reset)
01: LSE selected as RTC clock
10: LSI selected as RTC clock
11: HSE divided by RTCPRE value selected as RTC clock

Bit 7 **LSEEXT**: low-speed external clock type in bypass mode
Set and reset by software to select the external clock type (analog or digital).
The external clock must be enabled with the LSEON bit, to be used by the device.
The LSEEXT bit can be written only if the LSE oscillator is disabled.
0: LSE in analog mode (default after Backup domain reset)
1: LSE in digital mode (do not use if RTC is active).

Bit 6 **LSECSSD**: LSE clock security system failure detection
Set by hardware to indicate when a failure has been detected by the clock security system on the external 32 kHz oscillator.
0: no failure detected on 32 kHz oscillator (default after Backup domain reset)
1: failure detected on 32 kHz oscillator

Bit 5 **LSECSSON**: LSE clock security system enable
Set by software to enable the clock security system on 32 kHz oscillator.
LSECSSON must be enabled after LSE is enabled (LSEON enabled) and ready (LSERDY set by hardware) and after RTCSEL is selected.
Once enabled, this bit cannot be disabled, except after a LSE failure detection (LSECSSD = 1). In that case the software must disable LSECSSON.
0: CSS on 32 kHz oscillator OFF (default after Backup domain reset)
1: CSS on 32 kHz oscillator ON

Bits 4:3 **LSEDRV[1:0]**: LSE oscillator driving capability
Set by software to select the driving capability of the LSE oscillator.
These bit can be written only if LSE oscillator is disabled (LSEON = 0 and LSERDY = 0).
00: lowest drive (default after Backup domain reset)
01: medium-low drive
10: medium-high drive
11: highest drive

Bit 2 **LSEBYP**: LSE oscillator bypass
Set and reset by software to bypass oscillator in debug mode. This bit must not be written when the LSE is enabled (by LSEON) or ready (LSERDY = 1)
0: LSE oscillator not bypassed (default after Backup domain reset)
1: LSE oscillator bypassed

Bit 1 **LSERDY**: LSE oscillator ready
Set and reset by hardware to indicate when the LSE is stable.
This bit needs 6 cycles of lse_ck clock to fall down after LSEON has been set to 0.
0: LSE oscillator not ready (default after Backup domain reset)
1: LSE oscillator ready

Bit 0 **LSEON**: LSE oscillator enabled
Set and reset by software.
0: LSE oscillator OFF (default after Backup domain reset)
1: LSE oscillator ON

10.8.40 RCC reset status register (RCC_RSR)

Address offset: 0x0F4

Reset value: 0x0C00 0000

Reset by power-on reset only.

Access: 0 ≤ wait state ≤ 3, word, half-word, and byte access.

Wait states are inserted in case of successive accesses to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	BORRSTF	PINRSTF	Res.	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 Stop or standby mode entry, whereas the corresponding nRST_STOP, nRST_STBY option bit is cleared.
Cleared by writing to the RMVF bit.
0: No illegal low-power mode reset occurred
1: Illegal low-power mode reset occurred

Bit 30 **WWDGRSTF**: window watchdog reset flag
Reset by software by writing the RMVF bit.
Set by hardware when a window watchdog reset occurs.
0: no window watchdog reset occurred from WWDG (default after power-on reset)
1: window watchdog reset occurred from WWDG

Bit 29 **IWDGRSTF**: independent watchdog reset flag
Reset by software by writing the RMVF bit.
Set by hardware when an independent watchdog reset occurs.
0: no independent watchdog reset occurred (default after power-on reset)
1: independent watchdog reset occurred

Bit 28 **SFTRSTF**: system reset from CPU reset flag
Reset by software by writing the RMVF bit.
Set by hardware when the system reset is due to CPU.The CPU can generate a system reset by writing SYSRESETREQ bit of AIRCR register of the core M33.
0: no CPU software reset occurred (default after power-on reset)
1: a system reset has been generated by the CPU

Bit 27 **BORRSTF**: BOR reset flag
Reset by software by writing the RMVF bit.
Set by hardware when a BOR reset occurs (pwr_bor_rst).
0: no BOR reset occurred
1: BOR reset occurred (default after power-on reset)

Bit 26 **PINRSTF**: pin reset flag (NRST)
Reset by software by writing the RMVF bit.
Set by hardware when a reset from pin occurs.
0: no reset from pin occurred
1: reset from pin occurred (default after power-on reset)

Bits 25:24 Reserved, must be kept at reset value.

Bit 23 **RMVF**: remove reset flag
Set and reset by software to reset the value of the reset flags.
0: reset of the reset flags not activated (default after power-on reset)
1: resets the value of the reset flags

Bits 22:0 Reserved, must be kept at reset value.

10.8.41 RCC privilege configuration register (RCC_PRIVCFGR)

Address offset: 0x114

Reset value: 0x0000 0000

Access: no wait state; 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.	PRIV	Res.
														rw	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PRIV**: RCC functions privilege configuration
Set and reset by software. This bit can be written only by privileged access.
0: Read and write to RCC functions can be done by privileged or unprivileged access.
1: Read and write to RCC functions can be done by privileged access only.

Bit 0 Reserved, must be kept at reset value.

10.8.42 RCC register map

Table 70. RCC 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	RCC_CR	Res.	Res.	Res.	Res.	PLL2RDY	PLL2ON	PLL1RDY	PLL1ON	Res.	Res.	Res.	HSEEXT	HSECSSON	HSEBYP	HSERDY	HSEON	Res.	Res.	HSI48RDY	HSI48ON	Res.	CSIKERON	CSIRDY	CSION	Res.	Res.	HSIDIVF	HSIDIV[1:0]		HSIKERON	HSIRDY	HSION
	Reset value					0	0	0	0				0	0	0	0	0			0	0		0	0	0			1	0	1	0	1	1
0x010	RCC_HSICFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSITRIM[6:0]							Res.	Res.	Res.	Res.	HSICAL[11:0]											
	Reset value										1	0	0	0	0	0	0					X	X	X	X	X	X	X	X	X	X	X	X
0x014	RCC_CRRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSI48CAL[9:0]									
	Reset value																							X	X	X	X	X	X	X	X	X	X
0x018	RCC_CSICFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSITRIM[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSICAL[7:0]							
	Reset value											1	0	0	0	0	0									X	X	X	X	X	X	X	X
0x01C	RCC_CFGR1	MCO2SEL[2:0]			MCO2PRE[3:0]				MCO1SEL[2:0]			MCO1PRE[3:0]			Res.	Res.	TIMPRE	Res.	RTCPRE[5:0]							STOPKERWUCK	STOPWUCK	Res.	SWS[2:0]		Res.	SW[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
0x020	RCC_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB3DIS	APB2DIS	APB1DIS	Res.	Res.	AHB2DIS	AHB1DIS	Res.	PPRE3[2:0]			Res.	PPRE2[2:0]			Res.	PPRE1[2:0]			HPRE[3:0]			
	Reset value										0	0	0			0	0		0	0	0		0	0	0		0	0	0	0	0	0	0	0
0x028	RCC_PLL1CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL1REN	PLL1QEN	PLL1PEN	Res.	Res.	PLL1M[5:0]						Res.	Res.	PLL1VCOSEL	PLL1FRACEN	PLL1RGE[1:0]		PLL1SRC[1:0]	
	Reset value														0	0	0			0	0	0	0	0	0			0	0	0	0	0	0

Table 70. RCC 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	RCC_PLL2CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL2REN	PLL2QEN	PLL2PEN	Res.	Res.	PLL2M[5:0]						Res.	Res.	PLL2VCOSEL	PLL2FRACEN	PLL2RGE[1:0]		PLL2SRC[1:0]	
	Reset value														0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x034	RCC_PLL1DIVR	Res.	PLL1R[6:0]							Res.	PLL1Q[6:0]							PLL1P[6:0]							PLL1N[8:0]								
	Reset value		0	0	0	0	0	0	1		0	0	0	0	0	0	1	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	RCC_PLL2DIVR	Res.	PLL2R[6:0]							Res.	PLL2Q[6:0]							PLL2P[6:0]							PLL2N[8:0]								
	Reset value		0	0	0	0	0	0	1		0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0x040	RCC_PLL2FRACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL2FRACN[12:0]												Res.	Res.	Res.	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0				
0x050	RCC_CIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL2RDYIE	PLL1RDYIE	HSI48RDYIE	HSERDYIE	HSIRDYIE	CSIRDYIE	LSERDYIE	LSIRDYIE
	Reset value																									0	0	0	0	0	0	0	0
0x054	RCC_CIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSECSSF	Res.	Res.	PLL2RDYF	PLL1RDYF	HSI48RDYF	HSERDYF	HSIRDYF	CSIRDYF	LSERDYF	LSIRDYF
	Reset value																						0			0	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.	Res.	Res.	Res.	Res.	Res.	Res.	HSECSSC	Res.	Res.	PLL2RDYC	PLL1RDYC	HSI48RDYC	HSERDYC	HSIRDYC	CSIRDYC	LSERDYC	LSIRDYC
	Reset value																						0			0	0	0	0	0	0	0	0
0x060	RCC_AHB1RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGRST	Res.	Res.	Res.	Res.	CRCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA2RST	GPDMA1RST
	Reset value															0					0											0	0

Table 70. RCC 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	RCC_AHB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGRST	HASHRST	Res.	Res.	Res.	Res.	Res.	DAC1RST	ADCRST	Res.	Res.	GPIOHRST	Res.	Res.	Res.	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
	Reset value														0	0						0	0			0				0	0	0	0
0x074	RCC_ APB1LRSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRSRST	I3C1RST	I2C2RST	I2C1RST	Res.	Res.	USART3RST	USART2RST	COMPRST	SPI3RST	SPI2RST	OPAMPRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM7RST	TIM6RST	Res.	Res.	TIM3RST	TIM2RST
	Reset value								0	0	0	0			0	0	0	0	0	0								0	0				0	0
0x078	RCC_ APB1HRSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCAN1RST	Res.	Res.	Res.	LPTIM2RST	Res.	DTSRST	Res.	Res.	Res.
	Reset value																							0				0		0			
0x07C	RCC_APB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART1RST	Res.	SPI1RST	TIM1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0										0		0	0												
0x080	RCC_APB3RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM1RST	Res.	I3C2RST	Res.	Res.	LPUART1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																					0		0			0							
0x088	RCC_AHB1ENR	SRAM1EN	Res.	Res.	BKPRAMEN	Res.	Res.	Res.	GTZC1EN	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGEN	Res.	Res.	Res.	Res.	CRCEN	Res.	Res.	Res.	FLITFEN	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA2EN	GPDMA1EN
	Reset value	1			1				0							0					0				1							0	0
0x08C	RCC_AHB2ENR	Res.	SRAM2EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEN	HASHEN	Res.	Res.	Res.	Res.	Res.	DAC12EN	ADCEN	Res.	Res.	GPIOHEN	Res.	Res.	Res.	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	Reset value		1												0	0						0	0			0				0	0	0	0
0x09C	RCC_APB1LENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRSEN	I3C1EN	I2C2EN	I2C1EN	Res.	Res.	USART3EN	USART2EN	COMPEN	SPI3EN	SPI2EN	OPAMPEN	Res.	WWDGEN	Res.	Res.	Res.	Res.	Res.	Res.	TIM7EN	TIM6EN	Res.	Res.	TIM3EN	TIM2EN
	Reset value								0	0	0	0			0	0	0	0	0	0		0							0	0			0	0
0x0A0	RCC_APB1HENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANEN	Res.	Res.	Res.	LPTIM2EN	Res.	DTSEN	Res.	Res.	Res.
	Reset value																							0				0		0			
0x0A4	RCC_APB2ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART1EN	Res.	SPI1EN	TIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0										0		0	0												

Table 70. RCC 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
0x0A8	RCC_APB3ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM1EN	Res.	I3C2EN	Res.	Res.	LPUART1EN	Res.	Res.	Res.	Res.	SBSEN	Res.
	Reset value											0										0		0			0					0	
0x0B0	RCC_ AHB1LPENR	SRAM1LPEN	Res.	ICACHELPEN	BKPRAMLPEN	Res.	Res.	Res.	GTZC1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGLPEN	Res.	Res.	Res.	Res.	CRCLPEN	Res.	Res.	Res.	FLITFLPEN	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA2LPEN	GPDMA1LPEN
	Reset value	1		1	1				1							1					1				1							1	1
0x0B4	RCC_ AHB2LPENR	Res.	SRAM2LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGLPEN	HASHLPEN	Res.	Res.	Res.	Res.	Res.	DAC12LPEN	ADCLPEN	Res.	Res.	GPIOHLPEN	Res.	Res.	Res.	Res.	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
	Reset value		1												1	1						1	1			1					1	1	1	1
0x0C4	RCC_ APB1LLPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRSLPEN	I3C1LPEN	I2C2LPEN	I2C1LPEN	Res.	Res.	USART3LPEN	USART2LPEN	COMPLPEN	SPI3LPEN	SPI2LPEN	OPAMPLPEN	Res.	WWDGLPEN	Res.	Res.	Res.	Res.	Res.	TIM7LPEN	TIM6LPEN	Res.	Res.	TIM3LPEN	TIM2LPEN
	Reset value								1	1	1	1			1	1	1	1	1	1		1						1	1			1	1
0x0C8	RCC_ APB1HLPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANLPEN	Res.	Res.	Res.	LPTIM2LPEN	Res.	DTSLPEN	Res.	Res.	Res.
	Reset value																							1				1		1			
0x0CC	RCC_ APB2LPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART1LPEN	Res.	SPI1LPEN	TIM1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								1										1		1	1											
0x0D0	RCC_ APB3LPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM1LPEN	Res.	I3C2LPEN	Res.	Res.	LPUART1LPEN	Res.	Res.	Res.	Res.	SBSLPEN	Res.
	Reset value											1										1		1			1					1	
0x0D8	RCC_CCIPR1	TIMICSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART3SEL[2:0]			USART2SEL[2:0]			USART1SEL[2:0]		
	Reset value	0																							0	0	0	0	0	0	0	0	0
0x0DC	RCC_CCIPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2SEL[2:0]			Res.	Res.	LPTIM1SEL[2:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0	0	0		0	0	0								

Table 70. RCC 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
0x0E0	RCC_CCIPR3	Res.	Res.	Res.	Res.	Res.	LPUART1SEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI3SEL[2:0]			SPI2SEL[2:0]			SPI1SEL[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
0x0E4	RCC_CCIPR4	Res.	Res.	Res.	Res.	I3C2SEL[1:0]		I3C1SEL[1:0]		Res.	Res.	Res.	Res.	I2C2SEL[1:0]		I2C1SEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBSEL[1:0]		SYSTICKSEL[1:0]		Res.	Res.
	Reset value					0	0	0	0					0	0	0	0											0	0	0	0		
0x0E8	RCC_CCIPR5	CKPERSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANSEL[1:0]		Res.	Res.	RNGSEL[1:0]		DAC1SEL	ADCDACSEL[2:0]		
	Reset value	0	0																					0	0			0	0	0	0	0	0
0x0F0	RCC_BDCR	Res.	Res.	Res.	Res.	LSIRDY	LSION	LSCOSEL	LSCOEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VSWRST	RTCEN	Res.	Res.	Res.	Res.	Res.	RTCSEL[1:0]		LSEEXT	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	0
0x0F4	RCC_RSR	LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	BORRSTF	PINRSTF	Res.	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	0	1	1			0																							
0x0F8 to 0x10C	Reserved	Reserved																															
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.	PRIV	Res.
	Reset value																															0	

Refer to *Section 2.2* for the register boundary addresses.

11 Clock recovery system (CRS)

11.1 CRS introduction

The clock recovery system (CRS) is an advanced digital controller acting on the internal fine-granularity trimmable RC oscillator HSI48. The CRS provides powerful means to evaluate the oscillator output frequency, based on comparison with a selectable synchronization signal. The CRS can perform automatic trimming adjustments based on the measured frequency error value, while keeping the possibility of a manual trimming.

The CRS is ideally suited to provide a precise clock to the USB peripheral. In this case, the synchronization signal can be derived from the start-of-frame (SOF) packet signalization on the USB bus, sent by a USB host at 1 ms intervals.

The synchronization signal can also be derived from alternative synchronization or generated by the user software.

11.2 CRS main features

- Selectable synchronization source with programmable prescaler and polarity (see *Section 11.4.2: CRS internal signals*)
- Possibility to generate synchronization pulses by software
- Automatic oscillator trimming capability with no need of CPU action
- Manual control option for faster startup convergence
- 16-bit frequency error counter with automatic error value capture and reload
- Programmable limit for automatic frequency error value evaluation and status reporting
- Maskable interrupts/events:
  - Expected synchronization (ESYNC)
  - Synchronization OK (SYNCOK)
  - Synchronization warning (SYNCWARN)
  - Synchronization or trimming error (ERR)

11.3 CRS implementation

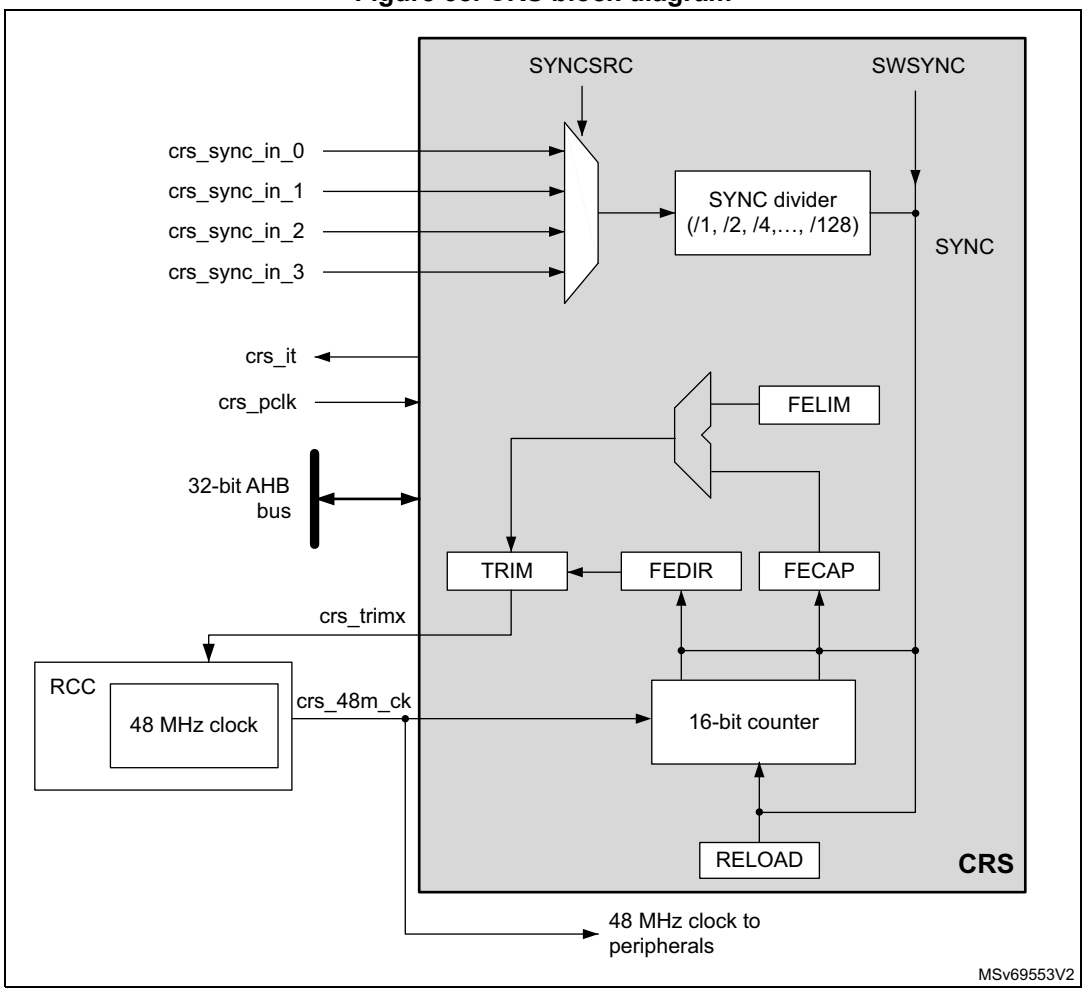
Table 71. CRS features

Feature	CRS
TRIM width	6 bits

11.4 CRS functional description

11.4.1 CRS block diagram

Figure 38. CRS block diagram



11.4.2 CRS internal signals

Below the list of CRS internal signals.

Table 72. CRS internal input/output signals

Internal signal name	Signal type	Description
crs_it	Digital output	CRS interrupt
crs_pclk	Digital input	AHB bus clock
crs_48m_ck	Digital input	HSI48 48 MHz clock
crs_trim[0:5]	Digital output	HSI48 oscillator smooth trimming value

Table 72. CRS internal input/output signals (continued)

Internal signal name	Signal type	Description
crs_sync_in_0	Digital input	SYNC signal source
crs_sync_in_1		
crs_sync_in_2		
crs_sync_in_3		

Table 73. CRS interconnection

Internal signal name	Description
crs_sync_in_0	GPIO selected as SYNC signal source
crs_sync_in_1	LSE selected as SYNC signal source
crs_sync_in_2	USB SOF selected as SYNC signal source (default)
crs_sync_in_3	Reserved

11.4.3 Synchronization input

The CRS synchronization source (crs_sync_in_x) can be selected through SYNCSRC[1:0] bitfield of CRS_CFGR register (refer to *Section 11.4.2: CRS internal signals* for the possible sources). For a better robustness of the crs_sync_in_x input, a simple digital filter (2 out of 3 majority votes, sampled by the 48 MHz clock) is implemented to filter out glitches. In addition, the source signal has a configurable polarity (selected through SYNCPOL bit of CRS_CFGR). The signal can then be divided by a programmable binary prescaler to obtain a synchronization signal in a suitable frequency range (usually around 1 kHz).

For more information on the CRS synchronization source configuration, refer to CRS_CFGR register.

It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

For more information on the CRS synchronization source configuration, refer to *CRS configuration register (CRS_CFGR)*.

It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

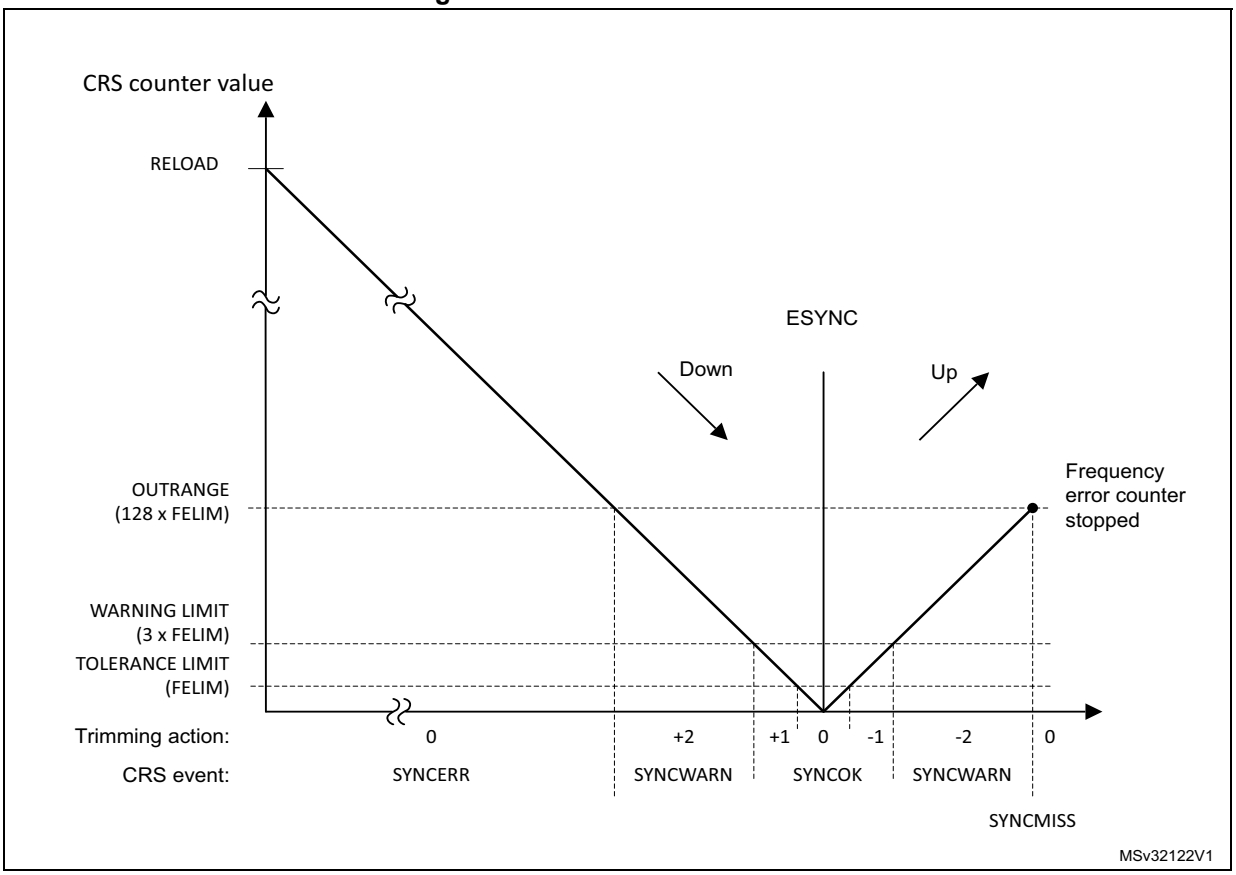
11.4.4 Frequency error measurement

The frequency error counter is a 16-bit down/up counter, reloaded with the RELOAD value on each SYNC event. It starts counting down until it reaches the zero value, where the ESYNC (expected synchronization) event is generated. Then it starts counting up to the OUTRANGE limit, where it eventually stops (if no SYNC event is received), and generates a SYNCMISS event. The OUTRANGE limit is defined as the frequency error limit (FELIM field of the CRS_CFGR register) multiplied by 128.

When the SYNC event is detected, the actual value of the frequency error counter and its counting direction are stored in the FECAP (frequency error capture) field and in the FEDIR (frequency error direction) bit of the CRS_ISR register. When the SYNC event is detected during the down-counting phase (before reaching the zero value), it means that the actual

frequency is lower than the target (the TRIM value must be incremented). When it is detected during the up-counting phase, it means that the actual frequency is higher (the TRIM value must be decremented).

Figure 39. CRS counter behavior



11.4.5 Frequency error evaluation and automatic trimming

The measured frequency error is evaluated by comparing its value with a set of limits:

- TOLERANCE LIMIT, given directly in the FELIM field of the CRS_CFGR register
- WARNING LIMIT, defined as 3 × FELIM value
- OUTRANGE (error limit), defined as 128 × FELIM value

The result of this comparison is used to generate the status indication and also to control the automatic trimming, which is enabled by setting the AUTOTRIMEN bit in the CRS_CR register:

- When the frequency error is below the tolerance limit, it means that the actual trimming value in the TRIM field is the optimal one, hence no trimming action is needed.
  - SYNCOK status indicated
  - TRIM value not changed in AUTOTRIM mode
- When the frequency error is below the warning limit but above or equal to the tolerance limit, it means that some trimming action is necessary but that adjustment by one trimming step is enough to reach the optimal TRIM value.
  - SYNCOK status indicated

  - TRIM value adjusted by one trimming step in AUTOTRIM mode
- When the frequency error is above or equal to the warning limit but below the error limit, a stronger trimming action is necessary, and there is a risk that the optimal TRIM value is not reached for the next period.
  - SYNCWARN status indicated
  - TRIM value adjusted by two trimming steps in AUTOTRIM mode
- When the frequency error is above or equal to the error limit, the frequency is out of the trimming range. This can also happen when the SYNC input is not clean, or when some SYNC pulse is missing (for example when one USB SOF is corrupted).
  - SYNCERR or SYNCMISS status indicated
  - TRIM value not changed in AUTOTRIM mode

Note: If the actual value of the TRIM field is close to its limits and the automatic trimming can force it to overflow or underflow, the TRIM value is set to the limit, and the TRIMOVF status is indicated.

In AUTOTRIM mode (AUTOTRIMEN bit set in the CRS_CR register), the TRIM field of CRS_CR is adjusted by hardware and is read-only.

11.4.6 CRS initialization and configuration

RELOAD value

The RELOAD value must be selected according to the ratio between the target frequency and the frequency of the synchronization source after prescaling. This value is decreased by 1, to reach the expected synchronization on the zero value. The formula is the following:

$$RELOAD = (f_{TARGET} / f_{SYNC}) - 1$$

The reset value of the RELOAD field corresponds to a target frequency of 48 MHz and a synchronization signal frequency of 1 kHz (SOF signal from USB).

FELIM value

The selection of the FELIM value is closely coupled with the HSI48 oscillator characteristics and its typical trimming step size. The optimal value corresponds to half of the trimming step size, expressed as a number of oscillator clock ticks. The following formula can be used:

$$FELIM = (f_{TARGET} / f_{SYNC}) * STEP[\%] / 100\% / 2$$

The result must be always rounded up to the nearest integer value to obtain the best trimming response. If frequent trimming actions are not needed in the application, the hysteresis can be increased by slightly increasing the FELIM value.

The reset value of the FELIM field corresponds to $(f_{TARGET} / f_{SYNC}) = 48000$, and to a typical trimming step size of 0.14%.

Note: The trimming step size depends upon the product, check the datasheet for accurate setting.

Caution: There is no hardware protection from a wrong configuration of the RELOAD and FELIM fields, this can lead to an erratic trimming response. The expected operational mode requires proper setup of the RELOAD value (according to the synchronization source frequency), which is also greater than 128 * FELIM value (OUTRANGE limit).

11.5 CRS in low-power modes

Table 74. Effect of low-power modes on CRS

Mode	Description
Sleep	No effect. CRS interrupts cause the device to exit the Sleep mode.
Stop	CRS registers are frozen. The CRS stops operating until the Stop mode is exited and the HSI48 oscillator is restarted.
Standby	The peripheral is powered down and must be reinitialized after exiting Standby mode.
Shutdown	The peripheral is powered down and must be reinitialized after exiting Shutdown mode.

11.6 CRS interrupts

Table 75. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Clear flag bit
Expected synchronization	ESYNCF	ESYNCIE	ESYNCC
Synchronization OK	SYNCOKF	SYNCOKIE	SYNCOKC
Synchronization warning	SYNCWARNF	SYNCWARNIE	SYNCWARNC
Synchronization or trimming error (TRIMOVF, SYNCMISS, SYNCERR)	ERRF	ERRIE	ERRC

11.7 CRS registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed only by words (32-bit).

11.7.1 CRS control register (CRS_CR)

Address offset: 0x00

Reset value: 0x0000 2000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	TRIM[5:0]						SW SYNC	AUTO TRIMEN	CEN	Res.	ESYNC IE	ERR IE	SYNC WARNIE	SYNC OKIE
		rw	rw	rw	rw	rw	rw	rt_w1	rw	rw		rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **TRIM[5:0]**: HSI48 oscillator smooth trimming

These bits provide a user-programmable trimming value to the HSI48 oscillator. They can be programmed to adjust to variations in voltage and temperature that influence the oscillator frequency.

The default value is 32, corresponding to the middle of the trimming interval. The trimming step is specified in the product datasheet. A higher TRIM value corresponds to a higher output frequency.

When the AUTOTRIMEN bit is set, this field is controlled by hardware and is read-only.

Bit 7 **SWSYNC**: Generate software SYNC event

This bit is set by software in order to generate a software SYNC event. It is automatically cleared by hardware.

0: No action

1: A software SYNC event is generated.

Bit 6 **AUTOTRIMEN**: Automatic trimming enable

This bit enables the automatic hardware adjustment of TRIM bits according to the measured frequency error between two SYNC events. If this bit is set, the TRIM bits are read-only. The TRIM value can be adjusted by hardware by one or two steps at a time, depending on the measured frequency error value. Refer to *Section 11.4.5* for more details.

0: Automatic trimming disabled, TRIM bits can be adjusted by the user.

1: Automatic trimming enabled, TRIM bits are read-only and under hardware control.

Bit 5 **CEN**: Frequency error counter enable

This bit enables the oscillator clock for the frequency error counter.

0: Frequency error counter disabled

1: Frequency error counter enabled

When this bit is set, the CRS_CFGR register is write-protected and cannot be modified.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **ESYNCIE**: Expected SYNC interrupt enable

0: Expected SYNC (ESYNCF) interrupt disabled

1: Expected SYNC (ESYNCF) interrupt enabled

Bit 2 **ERRIE**: Synchronization or trimming error interrupt enable

0: Synchronization or trimming error (ERRF) interrupt disabled

1: Synchronization or trimming error (ERRF) interrupt enabled

Bit 1 **SYNCWARNIE**: SYNC warning interrupt enable

0: SYNC warning (SYNCWARNF) interrupt disabled

1: SYNC warning (SYNCWARNF) interrupt enabled

Bit 0 **SYNCOKIE**: SYNC event OK interrupt enable

0: SYNC event OK (SYNCOKF) interrupt disabled

1: SYNC event OK (SYNCOKF) interrupt enabled

11.7.2 CRS configuration register (CRS_CFGR)

This register can be written only when the frequency error counter is disabled (the CEN bit is cleared in CRS_CR). When the counter is enabled, this register is write-protected.

Address offset: 0x04

Reset value: 0x2022 BB7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SYNCPOL	Res.	SYNCSRC[1:0]		Res.	SYNCDIV[2:0]			FELIM[7:0]							
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
RELOAD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SYNCPOL**: SYNC polarity selection

This bit is set and cleared by software to select the input polarity for the SYNC signal source.

0: SYNC active on rising edge (default)

1: SYNC active on falling edge

Bit 30 Reserved, must be kept at reset value.

Bits 29:28 **SYNCSRC[1:0]**: SYNC signal source selection

These bits are set and cleared by software to select the SYNC signal source (see *Section 11.4.2: CRS internal signals*).

00: crs_sync_in_0 selected as SYNC signal source

01: crs_sync_in_1 selected as SYNC signal source

10: crs_sync_in_2 selected as SYNC signal source

11: crs_sync_in_3 selected as SYNC signal source

Note: When using USB LPM (link power management) and the device is in Sleep mode, the periodic USB SOF is not generated by the host. No SYNC signal is therefore provided to the CRS to calibrate the 48 MHz clock on the run. To guarantee the required clock precision after waking up from Sleep mode, the LSE clock or the SYNC pin must be used as SYNC signal.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **SYNCDIV[2:0]**: SYNC divider

These bits are set and cleared by software to control the division factor of the SYNC signal.

000: SYNC not divided (default)

001: SYNC divided by 2

010: SYNC divided by 4

011: SYNC divided by 8

100: SYNC divided by 16

101: SYNC divided by 32

110: SYNC divided by 64

111: SYNC divided by 128

Bits 23:16 **FELIM[7:0]**: Frequency error limit

FELIM contains the value to be used to evaluate the captured frequency error value latched in the FECAP[15:0] bits of the CRS_ISR register. Refer to *Section 11.4.5* for more details about FECAP evaluation.

Bits 15:0 **RELOAD[15:0]**: Counter reload value

RELOAD is the value to be loaded in the frequency error counter with each SYNC event. Refer to *Section 11.4.4* for more details about counter behavior.

11.7.3 CRS interrupt and status register (CRS_ISR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FECAP[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
FEDIR	Res.	Res.	Res.	Res.	TRIM OVF	SYNC MISS	SYNC ERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNC WARNF	SYNC OKF
r					r	r	r					r	r	r	r

Bits 31:16 **FECAP[15:0]**: Frequency error capture

FECAP is the frequency error counter value latched in the time of the last SYNC event. Refer to *Section 11.4.5* for more details about FECAP usage.

Bit 15 **FEDIR**: Frequency error direction

FEDIR is the counting direction of the frequency error counter latched in the time of the last SYNC event. It shows whether the actual frequency is below or above the target.

0: Up-counting direction, the actual frequency is above the target

1: Down-counting direction, the actual frequency is below the target

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **TRIMOVF**: Trimming overflow or underflow

This flag is set by hardware when the automatic trimming tries to over- or under-flow the TRIM value. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No trimming error signaled

1: Trimming error signaled

Bit 9 **SYNCMISS**: SYNC missed

This flag is set by hardware when the frequency error counter reaches value FELIM * 128 and no SYNC is detected, meaning either that a SYNC pulse was missed, or the frequency error is too big (internal frequency too high) to be compensated by adjusting the TRIM value, hence some other action must be taken. At this point, the frequency error counter is stopped (waiting for a next SYNC), and an interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No SYNC missed error signaled

1: SYNC missed error signaled

Bit 8 **SYNCERR**: SYNC error

This flag is set by hardware when the SYNC pulse arrives before the ESYNC event and the measured frequency error is greater than or equal to FELIM * 128. This means that the frequency error is too big (internal frequency too low) to be compensated by adjusting the TRIM value, and that some other action has to be taken. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No SYNC error signaled

1: SYNC error signaled

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **ESYNCF**: Expected SYNC flag

This flag is set by hardware when the frequency error counter reached a zero value. An interrupt is generated if the ESYNCIE bit is set in the CRS_CR register. It is cleared by software by setting the ESYNCC bit in the CRS_ICR register.

0: No expected SYNC signaled

1: Expected SYNC signaled

Bit 2 **ERRF**: Error flag

This flag is set by hardware in case of any synchronization or trimming error. It is the logical OR of the TRIMOVF, SYNCMISS and SYNCERR bits. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software in reaction to setting the ERRC bit in the CRS_ICR register, which clears the TRIMOVF, SYNCMISS and SYNCERR bits.

0: No synchronization or trimming error signaled

1: Synchronization or trimming error signaled

Bit 1 **SYNCWARNF**: SYNC warning flag

This flag is set by hardware when the measured frequency error is greater than or equal to FELIM * 3, but smaller than FELIM * 128. This means that to compensate the frequency error, the TRIM value must be adjusted by two steps or more. An interrupt is generated if the SYNCWARNIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCWARNC bit in the CRS_ICR register.

0: No SYNC warning signaled

1: SYNC warning signaled

Bit 0 **SYNCOKF**: SYNC event OK flag

This flag is set by hardware when the measured frequency error is smaller than FELIM * 3. This means that either no adjustment of the TRIM value is needed or that an adjustment by one trimming step is enough to compensate the frequency error. An interrupt is generated if the SYNCOKIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCOKC bit in the CRS_ICR register.

0: No SYNC event OK signaled

1: SYNC event OK signaled

11.7.4 CRS interrupt flag clear register (CRS_ICR)

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESYNCC	ERRC	SYNC WARNC	SYNC OKC
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ESYNCC**: Expected SYNC clear flag
Writing 1 to this bit clears the ESYNCF flag in the CRS_ISR register.

Bit 2 **ERRC**: Error clear flag
Writing 1 to this bit clears TRIMOVF, SYNCMISS, and SYNCERR bits and consequently also the ERRF flag in the CRS_ISR register.

Bit 1 **SYNCWARNC**: SYNC warning clear flag
Writing 1 to this bit clears the SYNCWARNF flag in the CRS_ISR register.

Bit 0 **SYNCOKC**: SYNC event OK clear flag
Writing 1 to this bit clears the SYNCOKF flag in the CRS_ISR register.

11.7.5 CRS register map

Table 76. CRS register map and reset values

Offset	Register name Reset value	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	CRS_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]						SWSYNC	AUTOTRIMEN	CEN	Res.	ESYNCIE	ERRIE	SYNCWARNIE	SYNCOKIE
	Reset value	0x0000 2000																															
0x04	CRS_CFGR	SYNCPOL	Res.	SYNC SRC [1:0]		Res.	SYNC DIV [2:0]			FELIM[7:0]								RELOAD[15:0]															
	Reset value	0		1	0		0	0	0	0	0	1	0	0	0	1	0	1	0	1	1	1	0	1	1	0	1	1	1	1	1	1	1
0x08	CRS_ISR	FECAP[15:0]																FEDIR	Res.	Res.	Res.	Res.	TRIMOVF	SYNCMISS	SYNCERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNCWARNF	SYNCOKF
	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
0x0C	CRS_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.	ESYNCC	ERRC	SYNCWARNC	SYNCOKC
	Reset value																													0	0	0	0

Refer to *Section 2.2* for the register boundary addresses.

12 General-purpose I/Os (GPIO)

12.1 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 bits 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 high-speed low-voltage register (GPIOx_HSLVR).

12.2 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
- I/Os state retention during Standby mode

12.3 GPIO functional description

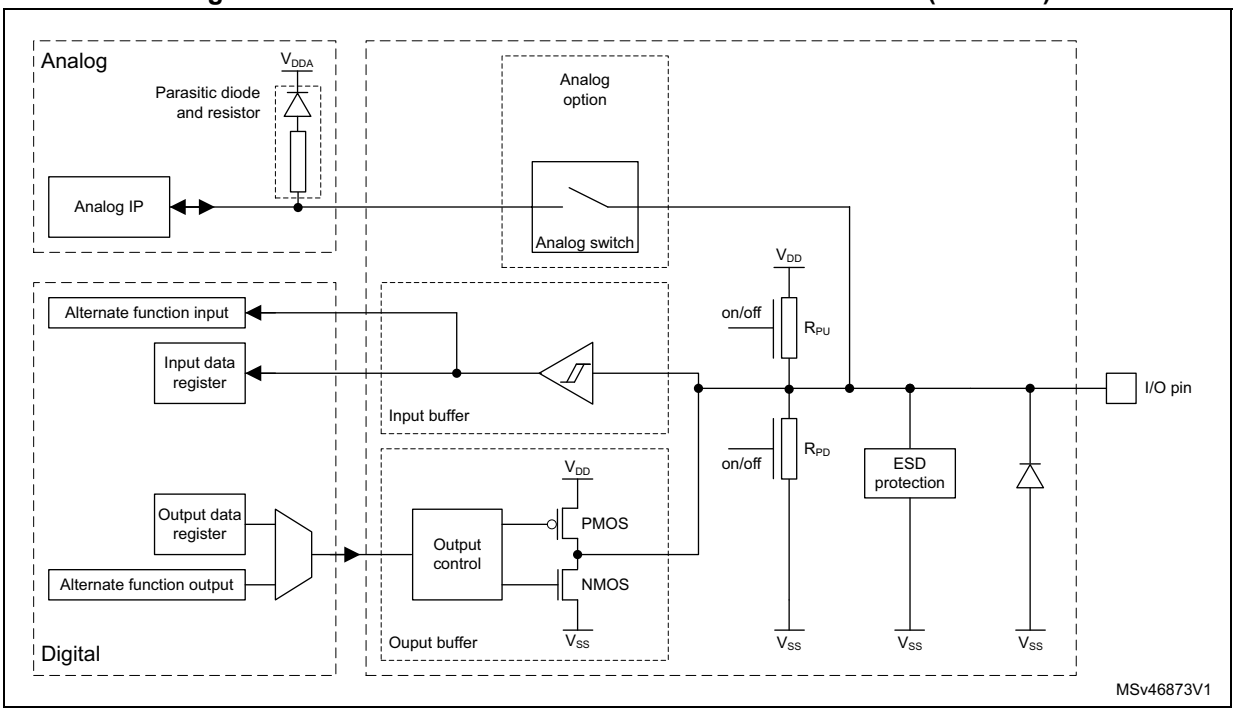
Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- 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 the modify access.

The figure below shows the basic structure of a three-volt or five-volt tolerant GPIO (TT or FT). The *Table 77* gives the possible port bit configurations.

Figure 40. Structure of three-volt or five-volt tolerant GPIO (TT or FT)



Note: On a TT GPIO, the analog switch is not present and replaced by a direct connection. The analog bloc parasitic circuitry does not allow five-volt tolerance.

Table 77. 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)	

Table 77. Port bit configuration[1] (continued)

MODE(i) [1:0]	OTYPE(i)	OSPEED(i) [1:0]		PUPD(i) [1:0]		I/O configuration	
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		
	x	x	x	1	1		

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

12.3.1 General-purpose I/O (GPIO)

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

When the pin is configured as output, the value written to 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 HI-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 not depending on the value in the GPIOx_PUPDR register.

12.3.2 I/O 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. In 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 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.

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 AFx 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_OSPEEDR registers respectively.
  - Configure the desired I/O as an alternate function in the GPIOx_MODER register.
- Cortex-M33 alternate function (EVENTOUT)
  - The Cortex-M33 output EVENTOUT signal can be output as alternate function on I/O pin. An event can be signaled through the configured pin after executing SEV instruction
  - EVENTOUT signal can be used internally as a trigger for some peripherals (see *Section 14: Peripherals interconnect matrix*)
- **Additional functions:**
  - For the ADC, DAC, OPAMP and COMP, configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADC, DAC, OPAMP and COMP registers.
  - For the additional functions like RTC, WKUPx and oscillators, configure the required function in the related RTC, PWR and RCC registers. These functions have priority over the configuration in the standard GPIO registers.

Refer to the “Alternate function mapping” table in the device datasheet for the detailed mapping of the alternate function I/O pins.

12.3.3 I/O 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.

12.3.4 I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (*GPIO x port input data register (GPIOx_IDR) (x = A to D, H)* and *GPIO x port output data register (GPIOx_ODR) (x = A to D, H)*).

GPIOx_ODR stores the data to be output, it is read/write accessible. The data input through the I/O are stored into the input data register (GPIOx_IDR), a read-only register.

12.3.5 I/O 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). The bit set reset register has twice the size of GPIOx_ODR.

To each bit in GPIOx_ODR, correspond two control bits in GPIOx_BSRR: BS(i) and BR(i). When written to 1, BS(i) sets the corresponding ODR(i) bit. When written to 1, BR(i) resets the ODR(i) corresponding bit.

Writing any bit to 0 in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR. If there is an attempt to both 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.

12.3.6 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, GPIOx_AFRH and GPIOx_HSLVR.

To write the GPIOx_LCKR register, a specific write/read sequence must be applied. When the right LOCK sequence is applied to the 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 a word (32-bit long) access to the GPIOx_LCKR register due to the fact that GPIOx_LCKR bit 16 must be set at the same time as the [15:0] bits.

12.3.7 I/O 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.

12.3.8 External interrupt/wakeup 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 17: Extended interrupts and event controller (EXTI)*.

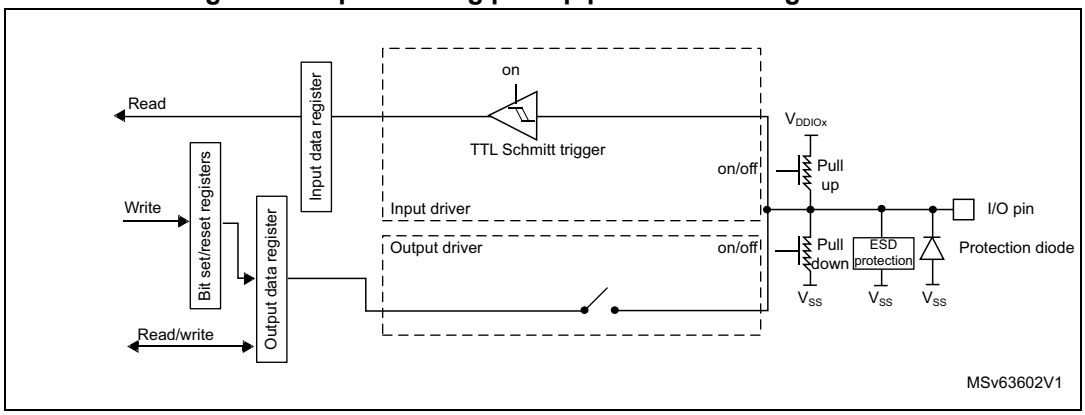
12.3.9 Input configuration

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.
- A read access to the input data register provides the I/O state.

The figure below shows the input configuration of the I/O port bit.

Figure 41. Input floating/pull-up/pull-down configurations



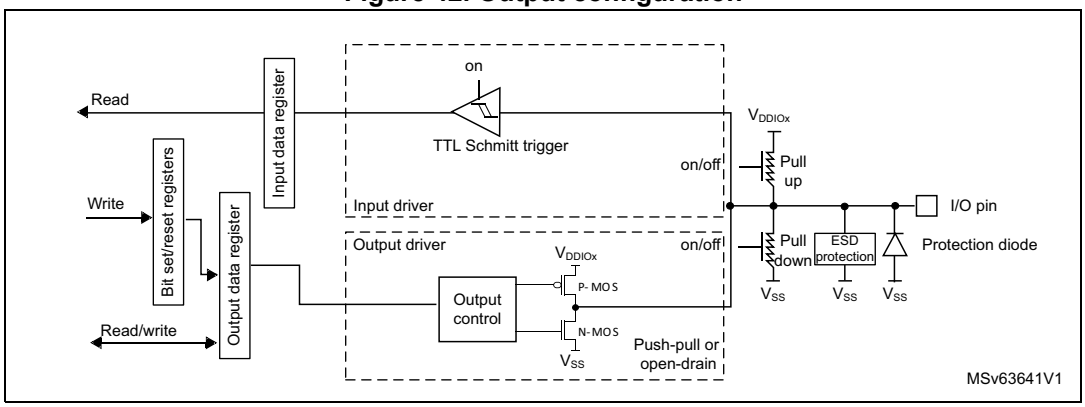
12.3.10 Output configuration

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 Hi-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.
- A read access to the input data register gets the I/O state.
- A read access to the output data register gets the last written value.

The figure below shows the output configuration of the I/O port bit.

Figure 42. Output configuration



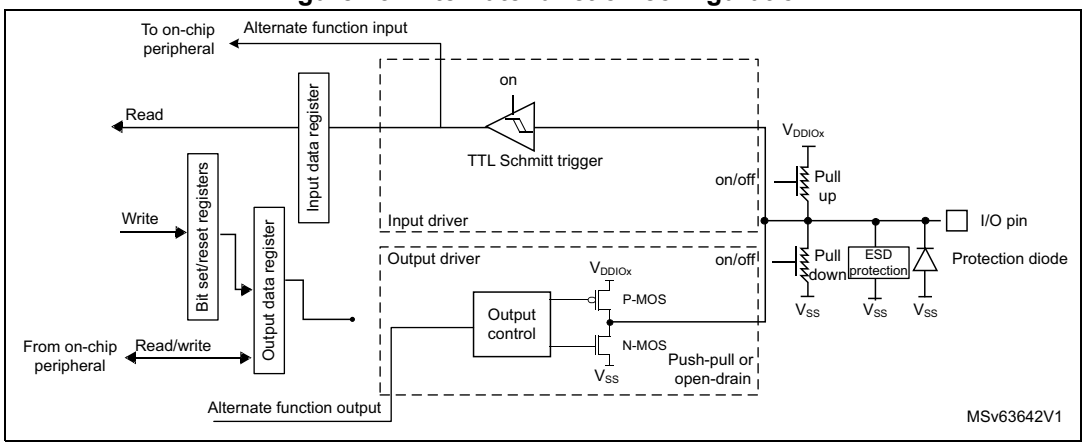
12.3.11 Alternate function configuration

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.
- A read access to the input data register gets the I/O state.

The figure below shows the alternate function configuration of the I/O port bit.

Figure 43. Alternate function configuration



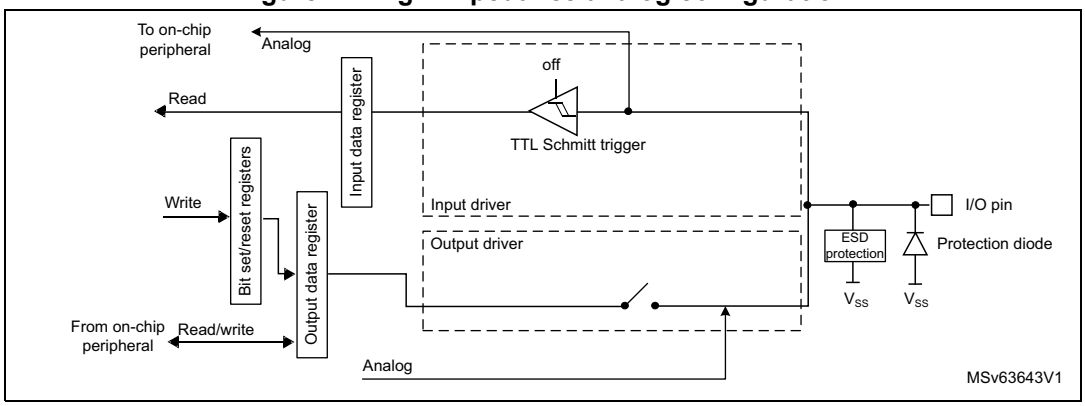
12.3.12 Analog configuration

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-up and pull-down resistors are disabled by hardware.
- Read access to the input data register gets the value 0.

The figure below shows the high-impedance, analog-input configuration of the I/O port bits.

Figure 44. High-impedance analog configuration



12.3.13 Using the HSE or LSE oscillator pins as GPIOs

When the HSE or LSE oscillator is switched off (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the HSE or LSE oscillator is switched on (by setting the HSEON or LSEON bit in the RCC_CSR 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 OSC_OUT or OSC32_OUT pin can still be used as normal GPIO.

12.3.14 Using the GPIO pins in the RTC supply domain

The PC13/PC14/PC15 GPIO functionality is lost when the core supply 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 set in an analog input mode.

For details about I/O control by the RTC, refer to *Section 32.3: RTC functional description*

12.3.15 I/Os state retention during standby mode

In the Standby mode, the I/Os are by default in floating state.

If the IORETEN bit in the PWR_IORETR register is set, the I/Os state is sampled during standby entry. The state of I/Os is applied to the pin via pull-up and pull-down resistors. The pull-up and pull-down resistors remains applied after Standby wakeup until the IORETEN bit in the PWR_IORETR register is cleared by software.

12.3.16 Privileged and unprivileged modes

All GPIO registers can be read and written by privileged and unprivileged accesses.

12.3.17 High-speed low-voltage mode (HSLV)

Some I/Os have the capability to increase their maximum speed at low voltage by configuring them in HSLV mode. The I/O HSLV bit controls whether the I/O output speed is optimized to operate at 3.3 V (default setting) or at 1.8 V (HSLV = 1).

Caution: The I/O HSLV configuration bit must not be set if the I/O supply (V_{DD} or V_{DDIO2}) is above 2.7 V. Setting it while the voltage is higher than 2.7 V can damage the device. The I/O HSLV bit can be set only when the corresponding option bit is activated (IO_VDD_HSLV or IO_VDDIO2_HSLV depending on the I/O supply, refer to *Section 7.4.1: About option bytes*). There is no hardware protection associated to this feature so it is recommended to use it only as a static configuration for fixed I/O supply.

12.3.18 I/O compensation cell

The I/O commutation slew rate (tfall / trise) can be adapted by software depending on Process, Voltage and Temperatures conditions, in order to reduce the I/O noise on power supply. Refer to *Section 13: System configuration, boot, and security (SBS)* for more details.

12.4 GPIO registers

This section gives a detailed description of the GPIO registers.

The peripheral registers can be written in word, half word or byte mode.

12.4.1 GPIO x port mode register (GPIOx_MODER) (x = A to D, H)

Address offset:0x00

Reset value: 0xABFF FFFF (for port A)

Reset value: 0xFF33 FEBF (for port B)

Reset value: 0xFFFF FFFF (for ports C)

Reset value: 0x0000 0030 (for ports D)

Reset value: 0x0000 000F (for ports H)

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]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MODEy[1:0]:** Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O mode.

00: Input mode

01: General purpose output mode

10: Alternate function mode

11: Analog mode (reset state)

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

12.4.2 GPIO x port output type register (GPIOx_OTYPER) (x = A to D, H)

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
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
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 **OTy:** Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O output type.

0: Output push-pull (reset state)

1: Output open-drain

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

12.4.3 GPIO x port output speed register (GPIOx_OSPEEDR) (x = A to D, H)

Address offset: 0x08

Reset value: 0x0C00 0000 (for port A)

Reset value: 0x0000 00C0 (for port B)

Reset value: 0x0000 0000 (for the other ports)

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]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSPEEDy[1:0]**: Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O output speed.

00: Low speed

01: Medium speed

10: High speed

11: Very-high speed

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.

The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

12.4.4 GPIO x port pull-up/pull-down register (GPIOx_PUPDR) (x = A to D, H)

Address offset: 0x0C

Reset value: 0x6400 0000 (for port A)

Reset value: 0x0000 0100 (for port B)

Reset value: 0x0000 0000 (for the other ports)

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]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PUPDy[1:0]:** Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O pull-up or pull-down

00: No pull-up, pull-down

01: Pull-up

10: Pull-down

11: Reserved

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

12.4.5 GPIO x port input data register (GPIOx_IDR) (x = A to D, H)

Address offset: 0x10

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	ID4	ID3	ID2	ID1	ID0
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 **IDy:** Port x input data I/O pin y (y = 15 to 0)

These bits are read-only. They contain the input value of the corresponding I/O port.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

12.4.6 GPIO x port output data register (GPIOx_ODR) (x = A to D, H)

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
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
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 **ODy:** Port output data I/O pin y (y = 15 to 0)

These bits can be read and written by software.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOx_BSRR or GPIOx_BRR registers (x = A to D and H).

The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

12.4.7 GPIO x port bit set/reset register (GPIOx_BSRR) (x = A to D, H)

Address offset: 0x18

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	BR4	BR3	BR2	BR1	BR0
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
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port x reset I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

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.

The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Bits 15:0 **BSy:** Port x set I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODy bit

1: Sets the corresponding ODy bit

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

12.4.8 GPIO x port configuration lock register (GPIOx_LCKR) (x = A to D, H)

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 GPIOx_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).

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.	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	LCK4	LCK3	LCK2	LCK1	LCK0
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.

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

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 LCK[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 15:0 **LCKy:** Port x lock I/O pin y (y = 15 to 0)

These bits are read/write but can only be written when the LCKK bit is 0

0: Port configuration not locked

1: Port configuration locked

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

12.4.9 GPIO x alternate function low register (GPIOx_AFRL) (x = A to D, H)

Address offset: 0x20

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]				AFSEL4[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
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFSELy[3:0]:** Alternate function selection for port x I/O pin y (y = 7 to 0)
These bits are written by software to configure alternate function I/Os.
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

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

12.4.10 GPIO x alternate function high register (GPIOx_AFRH) (x = A to D, H)

Address offset: 0x24

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 31:0 **AFSELy[3:0]:** Alternate function selection for port x I/O pin y (y = 15 to 8)
These bits are written by software to configure alternate function I/Os.
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

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

12.4.11 GPIO x port bit reset register (GPIOx_BRR) (x = A to D, H)

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
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
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 **BRy:** Port x reset IO pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODy bit
1: Reset the corresponding ODy bit

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

12.4.12 GPIO x high-speed low-voltage register (GPIOx_HSLVR) (x = A to D, H)

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
HSLV 15	HSLV 14	HSLV 13	HSLV 12	HSLV 11	HSLV 10	HSLV9	HSLV8	HSLV7	HSLV6	HSLV5	HSLV4	HSLV3	HSLV2	HSLV1	HSLV0
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 **HSLVy:** Port x high-speed low-voltage configuration (y = 15 to 0)

These bits are written by software to optimize the I/O speed when the I/O supply is low.

Each bit is active only if the corresponding IO_VDD_HSLV/IO_VDDIO2_HSLV user option bit is set. It must be used only if the I/O supply voltage is below 2.7 V.

Setting these bits when the I/O supply (VDD or VDDIO2) is higher than 2.7 V may be destructive.

0: I/O speed optimization disabled

1: I/O speed optimization enabled

Note: Not all I/Os support the HSLV mode. Refer to the I/O structure in the corresponding datasheet for the list of I/Os supporting this feature. Other I/Os HSLV configuration must be kept at reset value.

The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

12.4.13 GPIO register map

Table 78. GPIO 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
0x00	**GPIOx_MODER (x = A to D, H)**	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]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value for port A	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	1	1	1	1
	Reset value for port B	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1
	Reset value for port C	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value for port D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
	Reset value for port H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
0x04	**GPIOx_OTYPER (x = A to D, H)**	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	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 78. GPIO 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
0x08	**GPIOx_ OSPEEDR (x = A to D, H)**	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]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value for port A	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value for port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
	Reset value for ports C, D and H	0	0	0	0	0	0	0	0	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	**GPIOx_PUPDR (x = A to D, H)**	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]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value for port A	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	0	0	0	0
	Reset value for port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	Reset value for ports C, D and H	0	0	0	0	0	0	0	0	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	**GPIOx_IDR (x = A to D, H)**	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	ID4	ID3	ID2	ID1	ID0
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x14	**GPIOx_ODR (x = A to D, H)**	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	OD4	OD3	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**GPIOx_BSRR (x = A to D, H)**	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	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	0	0	0	0
0x1C	**GPIOx_LCKR (x = A to D, H)**	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	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**GPIOx_AFRL (x = A to D, H)**	AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]				AFSEL3[3:0]				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	0	0	0	0	0	0	0	0
0x24	**GPIOx_AFRH (x = A to D, H)**	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
0x28	**GPIOx_BRR (x = A to D, H)**	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	BR4	BR3	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**GPIOx_HSLVR (x = A to D, H)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSLV15	HSLV14	HSLV13	HSLV12	HSLV11	HSLV10	HSLV9	HSLV8	HSLV7	HSLV6	HSLV5	HSLV4	HSLV3	HSLV2	HSLV1	HSLV0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2* for the register boundary addresses.

13 System configuration, boot, and security (SBS)

13.1 SBS introduction

The STM32H503xx devices feature a set of configuration registers located in the SBS. On top of various device configurations this SBS peripheral controls key boot and security features, including debug control.

13.2 SBS main features

- System configuration
  - Manage robustness feature
  - Enable/disable the FMP high-drive mode of some I/Os
  - Manage the I/O compensation cell
  - Configure register security access
- Boot control
  - Upon system reset, configure the Cortex-M33 boot address and the temporal isolation level depending on the current configuration (such as PRODUCT_STATE (sbs_product_state).
  - Manage the temporal isolation feature implemented thanks to the hide protect level (HDPL) monotonic counter
- Debug control
  - Control the opening of the device debug interface, ensuring the sequencing of events that guarantee the device security

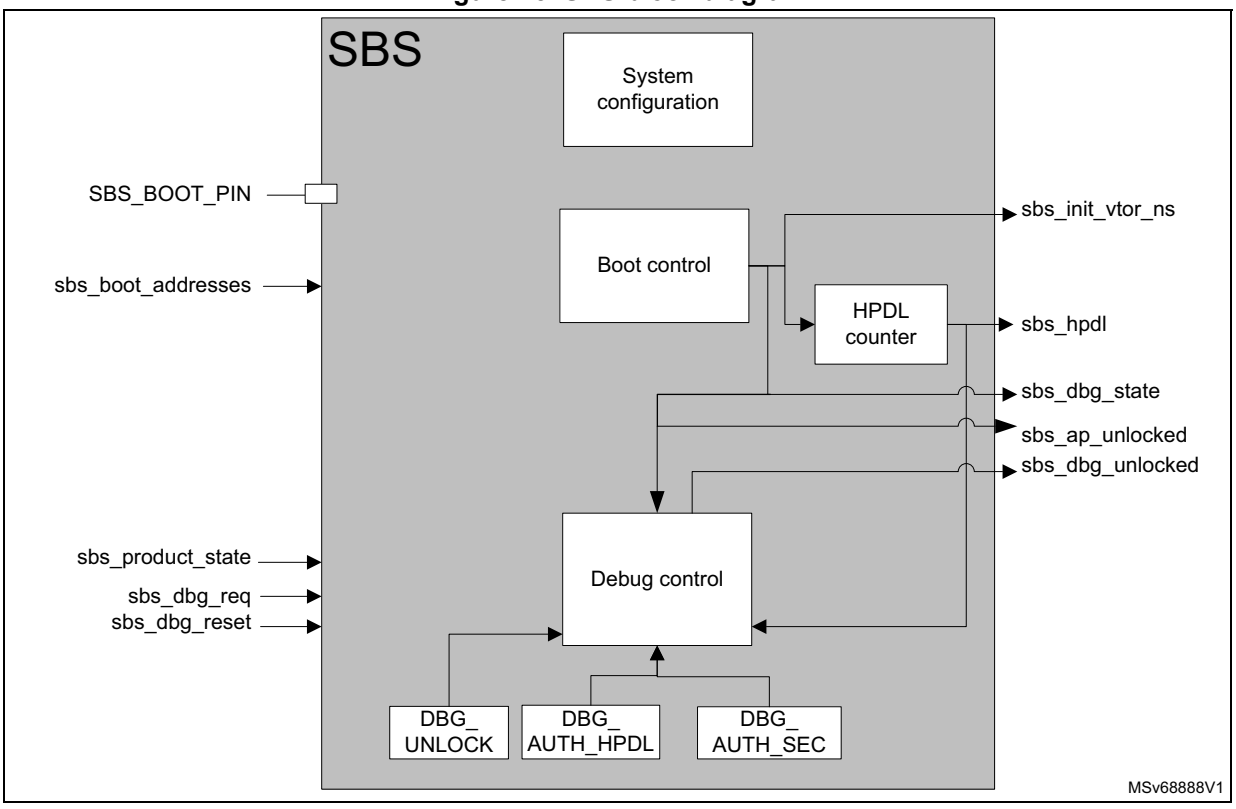
13.3 SBS functional description

13.3.1 SBS block diagram

The figure below shows the SBS block diagram, including the four main functions:

- System configuration
- Boot control
- Debug control

Figure 45. SBS block diagram



13.3.2 SBS signals

The table below details the user-relevant internal signals that interface the SBS.

Table 79. SBS internal input/output signals

Signal name	Type	Description
SBS_BOOT_PIN	Input	Select booting on user flash memory or bootloader
sbs_boot_addresses		List of addresses defined by the flash memory: – BOOT_ADDR_NS: boot address
sbs_product_state		Signal based on the PRODUCT_STATE option byte to activate the different security mechanisms depending on the product use. Expected values are described in *Section 7: Embedded flash memory (FLASH)*.
sbs_dbg_req		Launch the debug authentication protocol when booting.

Table 79. SBS internal input/output signals (continued)

Signal name	Type	Description
sbs_init_vtor_ns	Output	Vector address for Cortex-M33
sbs_hdpl		HDPL (temporal isolation level, ID of the boot level, used to isolate boot levels) This signal reflects a monotonic counter that can be incremented from 0 to 3, and that is reset to zero only on software reset or POR.
sbs_ap_unlocked		Control the Cortex-M33 access port.
sbs_dbg_unlocked		Unlock the debug (when set to 1) for the Cortex-M33
sbs_dbg_state		Populate the debug state of the flash memory with OPEN (0xB4), CLOSE_SEC (0x8A), or CLOSE (0x51).
sbs_dbg_reset	Input	Reset SBS_DBGCR (coming from the RCC). Can be reset by a system reset or POR, when configured through DBGMCU.

13.3.3 SBS reset and clocks

The SBS configuration port is clocked by the AHB bus clock. There is a general reset and a debug configuration reset controlled in DBGMCU.

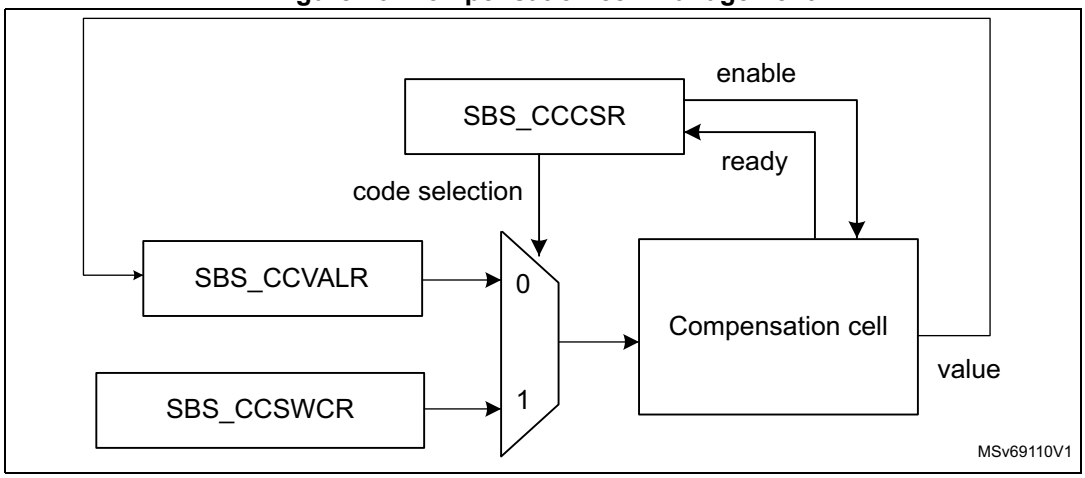
13.3.4 SBS system configuration

SBS 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. This value depends on the process, voltage, temperature (PVT) operating conditions. These bits are used to control the output impedance in the I/O buffer, and the slew rate of the I/O commutation (the t_{fall} and t_{rise} time), in order to reduce the I/O noise on the power supply.

As shown in the figure below, the compensation cell is split in two blocks: one block to provide an optimal code for the current PVT, and one block to drive the block controlled by the software.

Figure 46. Compensation cell management



The compensation cell value can be read when the READY flag is set in SBS_CCCSR. With CODESEL in SBS_CCCSR, the application can select the value to apply between two options: the code from the cell or the code from SBS_CCSWCR.

Two compensation cells are embedded in STM32H503xx devices:

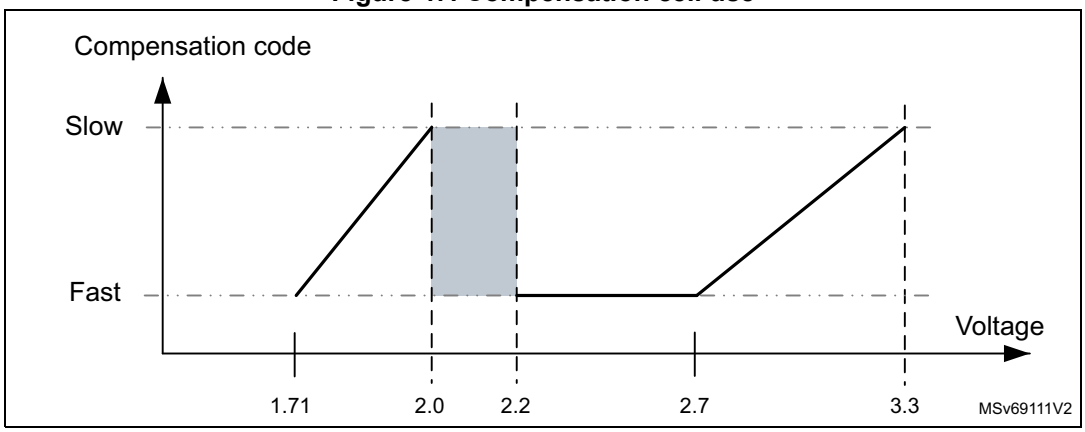
- one for the I/Os supplied by V_{DDIO} power rail
- one for the I/Os supplied by V_{DDIO2} power rail

By default, the compensation cells are disabled, and a fixed code is applied to all the I/Os.

Note: The compensation cell can be used only when 2.7 V ≤ V_{DDIOx} ≤ 3.6 V or 1.71 V ≤ V_{DDIOx} ≤ 2 V (see the figure below).

Note: The compensation cell can be used only when the CSI oscillator is enabled, see Section 10: Reset and clock control (RCC) for more details on the CSI oscillator.

Figure 47. Compensation cell use

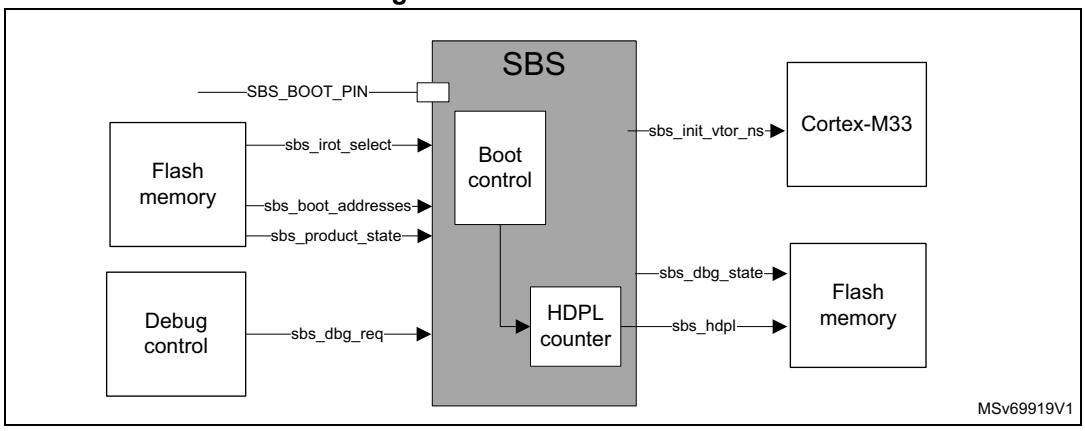


13.3.5 SBS boot control

The SBS can be used to control the boot entry points considering the product settings. The main boot control actions are listed below:

- Boot when launching a debug authentication sequence.
- Select boot between the bootloader or the user flash memory boot.
- Initialize the HDPL boot value.

Figure 48. SBS boot control



The boot configurations are selected considering the product settings:

- SBS_BOOT_PIN: to select booting on user flash memory
  - sbs_boot_addresses: list of addresses defined by the flash memory: BOOT_ADDR_NS: boot address
- PRODUCT_STATE: option byte to activate the different security mechanisms depending on the product use. Expected values are described in *Section 7: Embedded flash memory (FLASH)*.
- sbs_dbg_req: used to launch the debug authentication protocol when booting

The boot control logic sets the following data:

- sbs_init_vtor_ns: vector address for Cortex-M33
- HDPL (see the description below)

SBS HDPL (temporal isolation level) management

The HDPL is a monotonic counter incremented during the boot stages. The HDPL is reset to its default value only after a power-on or a system reset. This default value (0 or 1) depends on the device life cycle, as defined in the boot logic.

The STM32H503xx devices use HDPL information to automatically isolate code and its associated secrets (like keys) during the boot process. Incrementing HPDL ensures that private code and data for one boot stage cannot be directly accessible from later boot stages.

The HDPL is used by the user flash memory. See *Section 7: Embedded flash memory (FLASH)* for more details.

The HDPL can take the value from 0 to 3. When reaching the value 3, HDPL keeps this value until reset. The current HDPL value is readable in the HDPL bitfield in SBS_HDPLSR.

To increment the HDPL by one, the application must write 0x6A to INCR_HDPL in SBS_HDPLCR. After such increment, and before doing any subsequent action, the user must check that the HDPL has effectively been incremented reading SBS_HDPLSR.

Table 80. HDPL encoded values

HDPL	Code
0	0xB4
1	0x51
2	0x8A
3	0x6F
	All other values

Table 81. SBS boot logic

Inputs				Outputs		
sbs_product_state	**sbs_dbg_req**	**SBS_BOOT_PIN**	**sbs_boot_addresses**	**sbs_init_vtor_ns**	**sbs_hdpl**	**sbs_dbg_state (FLASH_STATE)**
Any except Locked	1	x	BOOT_DBG_AUTH_ADD	x	1	Depends on PRODUCT_STATE
Open	0	0	BOOT_ADDR_NS	BOOT_ADDR_NS	1	OPEN
	0	1	START_ADD_BOOT_LOADER	x	≥1	OPEN
Provisioning	0	x	START_ADD_BOOT_LOADER	x	≥1	CLOSE
iROT-Provisioned	0	x	BOOT_ADDR_NS	BOOT_ADDR_NS	1	CLOSE
Closed	0	x	BOOT_ADDR_NS	BOOT_ADDR_NS	1	CLOSE
Locked	0	x	BOOT_ADDR_NS	BOOT_ADDR_NS	1	CLOSE

13.3.6 SBS debug control

The SBS debug control is used to manage debug opening, taking care on the product context (PRODUCT_STATE, HDPL) on register settings, or through a debug authentication control.

When the debug is forbidden, debug access port AP0 (via mailbox), Cortex-M33 access port, and CPU debug interface are locked. In this situation, the debugger cannot access the CPU and no effective debug can be done. Refer to the *Section 41: Debug support (DBG)* for more details.

Authenticated debug sequence

1. The external host requests to launch the debug authentication protocol, via the DBGMCU access port mailbox. The rest of the device is kept under reset.
2. SBS selects the ST-DA (debug authentication library) boot address and requests the CPU to be released from reset.
3. The CPU running the ST-DA library executes the debug authentication protocol in the system flash memory. If the device is closed, the access port mailbox is closed until ST-DA acknowledges the authentication sequence start request.
4. As soon as the right password is entered from the host, the full regression is launched.

Note: Only full regression can be controlled (no debug reopening), thanks to a password authentication method. The product has to be previously provisioned with the HASH of the password.

Debug reset

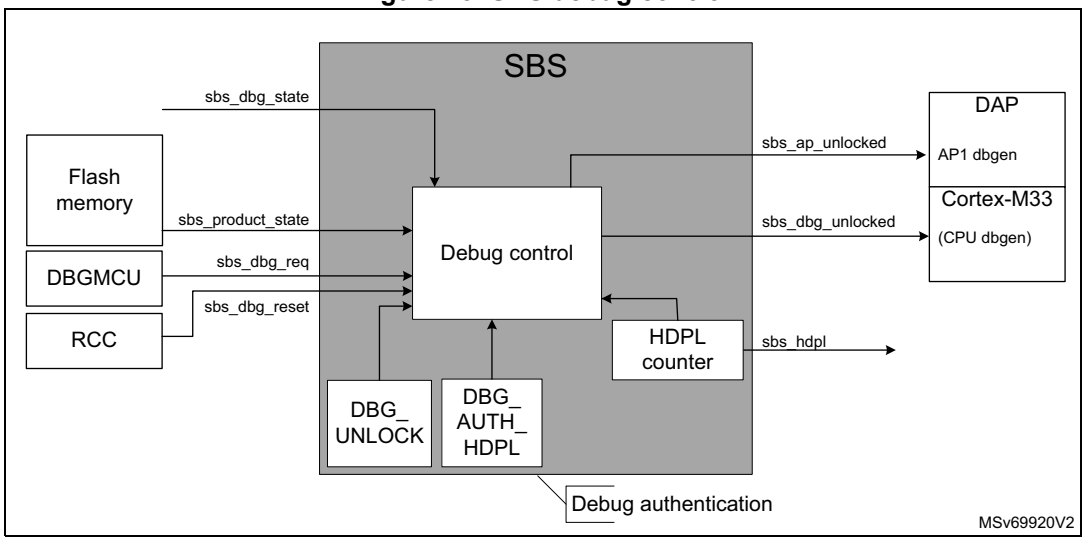
The debug opening may have the configuration to be reset by system or power-on reset, depending on a DBGMCU register field. A specific reset pin, DBGRESETn coming from the RCC, is used to reset the SBS_DBGCR register.

Debug locking

The debug configuration can be locked thanks to DBGCFG_LOCK in SBS_DBGLOCKR. SBS_DBGCR is then not anymore writable.

When DBGCFG_LOCK is set to 1, it can only be reset by the hardware DBGRESETn reset.

Figure 49. SBS debug control



Inputs used to control the debug opening

- sbs_dbg_req: input signal that a host requests to launch the debug authentication protocol. When this signal is set, the boot address is set to launch the debug authentication library.
- sbs_product_state: provide the current PRODUCT_STATE (from flash memory). See *Section 7: Embedded flash memory (FLASH)* for more details.
- sbs_dbg_reset: reset information coming from the RCC, and reset SBS_DBGCR if this register is configured to be reset.

Configuration

- DBG_UNLOCK in SBS_DBGCR: debug unlock when DBG_AUTH_HDPL is reached
- AP_UNLOCK in SBS_DBGCR: access port unlock
- DBG_AUTH_HDPL in SBS_DBGCR: authenticated debug temporal isolation level. Define the HDPL value from which the debug can be opened. Value can only be from 1 to 3.
- DBGC_LOCK in SBS_DBGLOCKR: lock the current debug configuration (released only by a reset).

Outputs

- sbs_ap_unlocked: signal controlling the Cortex-M33 access port
  - 1: 0xB4
  - 0: all other codes
- sbs_dbg_unlocked: unlock the debug (when set to 1) for the Cortex-M33.
  - 1: 0xB4
  - 0: all other codes
- sbs_dbg_state: translation of the sbs_product_state into debug status values (OPEN (0xB4), CLOSE_SEC (0x8A), or CLOSE (0x51))

Table 82. Debug authentication logic

	Input signal	Debug authentication control		Output signal
sbs_dbg_state	**sbs_ hdpl**	**sbs_dbg_ unlock**	**sbs_dbg_ auth_hdpl**	**sbs_dbg_ unlocked**
OPEN	1	x	x	1
	2	x	x	1
	3	x	x	1
CLOSED	x	0	x	0
	1	1	0	0
	1	1	-	1
	1	1	2	0
	2	1	1	1
	2	1	2	1
	2	1	3	0
	3	1	1	1
	3	1	2	1
	3	1	3	1

13.4 SBS interrupts

SBS does not support interrupts.

13.5 SBS registers

13.5.1 SBS temporal isolation control register (SBS_HDPLCR)

Address offset: 0x010

Reset value: 0x0000 00B4

Reset: system reset

Register security: no restriction

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	INCR_HDPL[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **INCR_HDPL[7:0]**: increment HDPL value
0xB4: no increment
0x6A: recommended value to increment HDPL level by one
Other: all other values allow an HDPL level increment.

13.5.2 SBS temporal isolation status register (SBS_HDPLSR)

Address offset: 0x014

Reset value: 0xXXXX XXXX

The reset value depends on boot case, HDPL0 or HDPL1. The boot logic defines the value out of reset. See *Section 13.3.5: SBS boot control*.

Reset: system reset

Register security: no restriction

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	HDPL[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **HDPL[7:0]**: temporal isolation level

This bitfield returns the current temporal isolation level.

0xB4: HDPL0, hide protection level reserved for ST code and data

0x51: HDPL1, hide protection level to be used to execute and protect an immutable root of trust (IROT) stage. First code to be executed at each reset of the platform. To ensure secure boot, this code must be immutable)

0x8A: HDPL2, hide protection level to be used to execute and protect an updatable Root of Trust (UROT) stage. UROT is executed from IROT to complete the immutable root of trust execution during boot stages

0x6F: HDPL3, hide protection level to be used to execute the application. It is a temporal isolation level that can be used to run the application when it has to be isolated from IROT and UROT)

13.5.3 SBS debug control register (SBS_DBGCR)

Address offset: 0x020

Reset value: 0x0000 0000

Reset: debug reset (system reset or power-on reset)

Register security: HDPL0/1 (RAZ/WI otherwise)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_AUTH_HDPL[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
DBG_UNLOCK[7:0]								AP_UNLOCK[7:0]							
rw	rw	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 23:16 **DBG_AUTH_HDPL[7:0]**: authenticated debug temporal isolation level

Writing to this bitfield defines at which HDPL the authenticated debug opens.

0x51: HDPL1

0x8A: HDPL2

0x6F: HDPL3

Note: Writing any other values is ignored. Reading any other value means that the debug never opens.

Bits 15:8 **DBG_UNLOCK[7:0]**: debug unlock when DBG_AUTH_HDPL is reached

Write 0xB4 to this bitfield to open the debug when HDPL in SBS_HDPLSR equals to DBG_AUTH_HDPL in this register.

Bits 7:0 **AP_UNLOCK[7:0]**: access port unlock

Write 0xB4 to this bitfield to open the device access port.

13.5.4 SBS debug lock register (SBS_DBGLOCKR)

Address offset: 0x024

Reset value: 0x0000 00B4

Reset: debug reset (system reset or power-on reset)

Register security: HDPL0/1 (RAZ/WI otherwise)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	DBGCFG_LOCK[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DBGCFG_LOCK[7:0]**: debug configuration lock

Reading this bitfield returns 0x6A if the bitfield value is different from 0xB4.
0xC3 is the recommended value to lock the debug configuration using this bitfield.
0xB4: Writes to SBS_DBGCR allowed (default)
Other: Writes to SBS_DBGCR ignored

13.5.5 SBS product mode and configuration register (SBS_PMCR)

Address offset: 0x100

Reset value: 0x0000 0000

Reset: system reset

Register security: there is no access restriction.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PB8_FMPLUS	PB7_FMPLUS	PB6_FMPLUS
													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.

Bit 18 **PB8_FMPLUS**: Fast-mode Plus command on PB(8)

0: Fast-mode Plus mode on PB(8) is disabled
1: Fast-mode Plus mode on PB(8) is enabled

Bit 17 **PB7_FMPLUS**: Fast-mode Plus command on PB(7)

0: Fast-mode Plus mode on PB(7) is disabled
1: Fast mode plus mode on PB(7) is enabled

Bit 16 **PB6_FMPLUS**: Fast-mode Plus command on PB(6)
0: Fast-mode Plus mode on PB(6) is disabled
1: Fast-mode Plus mode on PB(6) is enabled

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 Reserved, must be kept at reset value.

13.5.6 SBS FPU interrupt mask register (SBS_FPUIMR)

Address offset: 0x104

Reset value: 0x0000 001F

Reset: system reset

This register is only accessible through a privilege transaction.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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]**: FPU interrupt enable
Set and cleared by software to enable the Cortex-M33 FPU interrupts
FPU_IE[5]: inexact interrupt enable (interrupt disabled 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

13.5.7 SBS memory erase status register (SBS_MESR)

Address offset: 0x108

Power-on reset value: 0b0000 0000 0000 0000 0000 0000 0000 0000

System reset value: 0b0000 0000 0000 0000 0000 0000 0000 000U

Register security: always accessible

31	30	29	28	27	26	25	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 erase status
This bit is set by hardware when ICACHE erase is completed after potential tamper detection or a Product State regression (refer to *Section 33: Tamper and backup registers (TAMP)* for more details).
This bit is cleared by software by writing 1 to it.
0: ICACHE erase on going
1: ICACHE erase ended

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, BKPSRAM, and ICACHE erase is completed after power-on reset or tamper detection or Product State regression (refer to *Section 33: Tamper and backup registers (TAMP)* for more details).
This bit is not reset by system reset and is cleared by software by writing 1 to it.
0: memory erase on going if not yet cleared by software
1: memory erase done

13.5.8 SBS compensation cell for I/Os control and status register (SBS_CCCSR)

Address offset: 0x110

Reset value: 0x0000 0000

Reset: system reset

Register security: always accessible

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	RDY2	RDY1	Res.	Res.	Res.	Res.	CS2	EN2	CS1	EN1
						r	r					rw	rw	rw	rw

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

Bit 9 **RDY2**: VDDIO2 compensation cell ready flag
This bit provides the status of the VDDIO2 compensation cell.
0: VDDIO2 compensation cell not ready
1: VDDIO2 compensation cell ready (code value provided by the cell can be used)

Bit 8 **RDY1**: VDDIO compensation cell ready flag
This bit provides the status of the compensation cell.
0: VDDIO compensation cell not ready
1: VDDIO compensation cell ready (code value provided by the cell can be used)

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **CS2**: code selection for VDDIO2 power rail (reset value set to 1)
This bit selects the code to be applied for the I/O compensation cell.
0: Code from the cell (available in SBS_CCVALR)
1: Code from SBS_CCSWCR

Bit 2 **EN2**: enable compensation cell for VDDIO2 power rail
This bit enables the I/O compensation cell.
0: I/O compensation cell disabled
1: I/O compensation cell enabled

Bit 1 **CS1**: code selection for VDDIO power rail (reset value set to 1)
This bit selects the code to be applied for the I/O compensation cell.
0: Code from the cell (available in the SBS_CCVALR)
1: Code from SBS_CCSWCR

Bit 0 **EN**1: enable compensation cell for VDDIO power rail
This bit enables the I/O compensation cell.
0: I/O compensation cell disabled
1: I/O compensation cell enabled

13.5.9 SBS compensation cell for I/Os value register (SBS_CCVALR)

Address offset: 0x114

Reset value: 0x0000 0088

Reset: system reset

Register security: always accessible

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
APSRC2[3:0]				ANSRC2[3:0]				APSRC1[3:0]				ANSRC1[3:0]			
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 15:12 **APSRC2[3:0]**: compensation value for the PMOS transistor
This value is provided by the cell and the processor must interpret it to compensate the slew rate in the functional range.

Bits 11:8 **ANSRC2[3:0]**: Compensation value for the NMOS transistor
This value is provided by the cell and the processor must interpret it to compensate the slew rate in the functional range.

Bits 7:4 **APSRC1[3:0]**: compensation value for the PMOS transistor
This value is provided by the cell and the processor must interpret it to compensate the slew rate in the functional range.

Bits 3:0 **ANSRC1[3:0]**: compensation value for the NMOS transistor
This value is provided by the cell and the processor must interpret it to compensate the slew rate in the functional range.

13.5.10 SBS compensation cell for I/Os software code register (SBS_CCSWCR)

Address offset: 0x118

Reset value: 0x0000 7878

Reset: system reset

Register security: always accessible

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SW_APSRC2[3:0]				SW_ANSRC2[3:0]				SW_APSRC1[3:0]				SW_ANSRC1[3:0]			
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 15:12 **SW_APSRC2[3:0]**: PMOS compensation code for the V_{DDIO} power rails
This bitfield is written by software to define an I/O compensation cell code for the PMOS transistors of the VDDIO power rail. This code is applied to the I/O when CS2 is set in SBS_CCSR.

Bits 11:8 **SW_ANSRC2[3:0]**: NMOS compensation code for VDDIO power rails
This bitfield is written by software to define an I/O compensation cell code for the NMOS transistors of the VDD power rail. This code is applied to the I/O when CS2 is set in SBS_CCSR.

Bits 7:4 **SW_APSRC1[3:0]**: PMOS compensation code for the VDD power rails
This bitfield is written by software to define an I/O compensation cell code for PMOS transistors of the VDDIO power rail. This code is applied to the I/O when CS1 is set in SBS_CCSR.

Bits 3:0 **SW_ANSRC1[3:0]**: NMOS compensation code for VDD power rails
This bitfield is written by software to define an I/O compensation cell code for NMOS transistors of the VDD power rail. This code is applied to the I/O when CS1 is set in SBS_CCSR.

13.5.11 SBS Class B register (SBS_CFGR2)

Address offset: 0x120

Reset value: 0x0000 0000

Reset: system reset

Register security: always accessible

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

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ECCL**: ECC lock

This bit is set and cleared by software. It can be used to enable and lock the flash memory double ECC error with break input of TIM1.

0: double ECC error flag disconnected to timer break inputs

1: double ECC error flag connected to timer break inputs

Bit 2 **PVDL**: PVD lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the PVD connection with TIM1 break inputs.

0: PVD interrupt disconnected from timer break inputs. PVD_EN and PVD_SEL[2:0] in the PWR registers are read/write.

1: PVD interrupt is connected to timer break inputs. PVD_EN and PVD_SEL[2:0] in the PWR registers are read only

Bit 1 **SEL**: SRAM ECC error lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the SRAM double ECC error signal with break input of TIM1.

0: SRAM double ECC error flag disconnected from timer break inputs

1: SRAM double ECC error flag connected to timer break inputs

Bit 0 **CLL**: core lockup lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the lockup (HardFault) output of the Cortex-M33 with TIM1 break inputs.

0: lockup output disconnected from timer break inputs

1: lockup output connected to timer break inputs

13.5.12 SBS CPU lock register (SBS_CNSLCKR)

Address offset: 0x144

Reset value: 0x0000 0000

Reset: system reset

Register security: This register can be read and written by privileged access only. Unprivileged access is RAZ/WI.

This register is used to lock the configuration of the MPU and VTOR_NS registers of the Cortex-M33.

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **LOCKNSMPU**: MPU register lock

This bit is set by software and cleared only by a system reset. When set, this bit disables write access to MPU_CTRL_NS, MPU_RNR_NS and MPU_RBAR_NS registers.

0: MPU registers write enabled

1: MPU registers write disabled

Bit 0 **LOCKNSVTOR**: VTOR_NS register lock

This bit is set by software and cleared only by a system reset.

0: VTOR_NS register write enabled

1: VTOR_NS register write disabled

13.5.13 SBS flift ECC NMI mask register (SBS_ECCNMIR)

Address offset: 0x14C

Reset value: 0x0000 0000

Reset: system reset

Register security: This register is only accessible through a privilege transaction.

This register sets up the expected behavior on NMI regarding double ECC errors from the flash memory.

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

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

Bit 0 **ECCNMI_MASK_EN**: NMI behavior setup when a double ECC error occurs on FLITF data part

0: NMI generated if a double ECC error in the FLITF data part

1: NMI not generated if a double ECC error in the FLITF data part

13.5.14 SBS register map

Table 83. SBS 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
0x010	**SBS_HDPLCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INCR_HDPL[7:0]							
	Reset value																									1	0	1	1	0	1	0	0
0x014	**SBS_HDPLSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPL[7:0]							
	Reset value																									x	x	x	x	x	x	x	x
0x020	**SBS_DBGCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_AUTH_HDPL[7:0]								DBG_UNLOCK[7:0]								AP_UNLOCK[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

Table 83. SBS 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
0x024	**SBS_DBGLOCKR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGCFG_LOCK[7:0]							
	Reset value																									1	0	1	1	0	1	0	0
0x028 to 0x030	**Reserved**	Reserved																															
0x038 to 0x09C	**Reserved**	Reserved																															
0x0A4 to 0x0BC	**Reserved**	Reserved																															
0x0C4 to 0x0FC	**Reserved**	Reserved																															
0x100	**SBS_PMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PB8_FMPLUS	PB7_FMPLUS	PB6_FMPLUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0																
0x104	**SBS_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
0x108	**SBS_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
0x110	**SBS_CCCSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDY2	RDY1	Res.	Res.	Res.	Res.	CS2	EN2	CS1	EN1
	Reset value																							0	0					0	0	0	0
0x114	**SBS_CCVALR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APSRC2 [3:0]				ANSRC2 [3:0]				APSRC1 [3:0]				ANSRC1 [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0
0x118	**SBS_CCSWCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW_APSRC2 [3:0]				SW_ANSRC2 [3:0]				SW_APSRC1 [3:0]				SW_ANSRC1 [3:0]			
	Reset value																	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0
0x120	**SBS_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	SEL	CLL
	Reset value																													0	0	0	0
0x124 to 0x140	**Reserved**	Reserved																															
0x144	**SBS_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
0x14C	**SBS_ECCNMIR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCNMI_MASK_EN
	Reset value																																0

Refer to *Section 2.2* for the register boundary addresses.

14 Peripherals interconnect matrix

14.1 Introduction

Several peripherals have direct connections between them. This allows autonomous communication and or synchronization between peripherals, saving CPU resources, thus power supply consumption.

In addition, these hardware connections avoid software latency and allow design of predictable system.

Depending on peripherals, these interconnections can operate in various power modes: Run, Sleep, and Stop modes.

14.2 Connection summary

Table 84. Peripherals interconnect matrix(1) (2)

Source	Destination														
	TIM1	TIM2	TIM3	TIM6	TIM7	LPTIM1	LPTIM2	ADC1	OPAMP1	DAC1/2	COMP1	GPDMA	TAMP	RTC	DTS
TIM1	-	*1*	*1*	-	-	-	-	*2*	-	*4*	*10*	-	-	-	-
TIM2	*1*	-	*1*	-	-	-	-	*2*	-	*4*	*10*	*13*	-	-	-
TIM3	*1*	*1*	-	-	-	-	-	*2*	-	*4*	*10*	-	-	-	-
TIM6	-	-	-	-	-	-	-	*2*	-	*4*	-	-	-	-	-
TIM7	-	-	-	-	-	-	-	*2*	-	*4*	-	-	-	-	-
LPTIM1	-	-	-	-	-	-	*6*	*2*	-	*4*	*10*	*13*	-	-	*5*
LPTIM2	-	-	-	-	-	*6*	-	*2*	-	*4*	*10*	*13*	-	-	*5*
ADC1	*3*	-	-	-	-	-	-	-	-	-	-	-	*16*	-	-
DAC1/2	-	-	-	-	-	-	-	-	*14*	-	*14*	-	-	-	-
OPAMP	-	-	-	-	-	-	-	*14*	-	-	-	-	-	-	-
COMP1	*11*	*11*	*11*	-	-	*8*	*8*	-	-	-	-	*13*	-	-	-
GPDMA1	-	-	-	-	-	*8*	*8*	-	-	-	-	*13*	-	-	-
GPDMA2	-	-	-	-	-	*8*	*8*	-	-	-	-	*13*	-	-	-
EXTI	-	-	-	-	-	-	-	*2*	-	*4*	-	*13*	-	-	*5*
RTC wakeup	-	*9*	-	-	-	-	-	-	-	-	-	*13*	-	-	-
RTC alarm	-	-	-	-	-	*8*	*8*	-	-	-	-	*13*	-	-	-
RTC calendar overflow	-	-	-	-	-	-	-	-	-	-	-	-	*16*	-	-
TAMP	-	-	-	-	-	*8*	*8*	-	-	-	-	*13*	-	*17*	-
HSE	-	-	*7*	-	-	*7*	-	-	-	-	-	-	-	-	-
LSE	-	*7*	-	-	-	*7*	-	-	-	*15*	-	-	-	-	-
CSS on LSE	-	-	-	-	-	-	-	-	-	-	-	-	*16*	-	-
CSS on HSE	*12*	-	-	-	-	-	-	-	-	-	-	-	*16*	-	-
CSI	*7*	*7*	*7*	-	-	-	*7*	-	-	-	-	-	-	-	-
HSI	*7*	*7*	*7*	-	-	-	*7*	-	-	-	-	-	-	-	-
LSI	-	*7*	-	-	-	*7*	-	-	-	*15*	-	-	-	-	-
MCO1	-	*7*	*7*	-	-	*7*	-	-	-	-	-	-	-	-	-

Table 84. Peripherals interconnect matrix[1] [2] (continued)

Source	Destination														
	TIM1	TIM2	TIM3	TIM6	TIM7	LPTIM1	LPTIM2	ADC1	OPAMP1	DAC1/2	COMP1	GPDMA	TAMP	RTC	DTS
MCO2	-	*7*	*7*	-	-	-	*7*	-	-	-	-	-	-	-	-
VCORE	-	-	-	-	-	-	-	*14*	-	-	-	-	-	-	-
VREFINT	-	-	-	-	-	-	-	*14*	-	-	*14*	-	-	-	-
T sensor (V_{SENSE})	-	-	-	-	-	-	-	*14*	-	-	*14*	-	-	-	-
VBAT	-	-	-	-	-	-	-	*14*	-	-	*14*	-	-	-	-
VBAT and temperature monitor	-	-	-	-	-	-	-	-	-	-	-	-	*16*	-	-
System errors	*12*	-	-	-	-	-	-	-	-	-	-	-	-	-	-
EVENTOUT	-	-	-	-	-	*8*	*8*	-	-	-	-	*13*	-	-	-

1. Numbers in this table are links to corresponding subsections of *Section 14.3*.
2. "-" means no interconnect.

14.3 Interconnection details

14.3.1 Master to slave interconnection for timers

From timer (TIM1/TIM2/TIM3) to timer (TIM1/TIM2/TIM3)

Purpose

Some of 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.

A description of the feature is provided in *Section 27.4.23: Timer synchronization*.

The synchronization modes are detailed in:

- *Section 26.3.30* for advanced-control timers TIM1
- *Section 27.4.22* for general-purpose timers TIM2/TIM3

Triggering signals

The output (from master) is on signal TIMx_TRGO (and TIMx_TRGO2 for TIM1) following a configurable timer event. The input (to slave) is on signals TIMx_ITR0/ITR1/ITR2.

The possible master/slave connections are given in:

- *Table 171: Internal trigger connection* for advanced-control timers TIM1
- *Table 194: TIMx internal trigger connection* for general-purpose timers TIM2/TIM3

Active power mode

Timers are optionally active in Run and Sleep modes. The effects of low-power modes on TIMx are given in:

- *Table 183: Effect of low-power modes on TIM1*
- *Table 201: Effect of low-power modes on TIM2/TIM3*

14.3.2 Triggers to ADCs

From EXTI, timers (TIM1/TIM2/TIM3/TIM6/TIM7) and LP timers (LPTIM1/LPTIM2) to ADC1

Purpose

A conversion, or a sequence of conversions, can be triggered either by software or by an external event (such as timer output channel, timer trigger output or input pins with EXTI). For ADC1, if the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, then external events can trigger a conversion with the selected polarity. More details in:

- *Section 19.4.18: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN,JEXTSEL, JEXTEN)*
- EXTEN[1:0] defined in *ADC configuration register (ADC_CFGR)*
- JEXTEN[1:0] defined in *ADC injected sequence register (ADC_JSQR)*

General-purpose timers (TIM2/TIM3), basic timer (TIM6), advanced-control timers (TIM1) can be used to generate the ADC triggering event through the timer outputs tim_oc and tim_trgo (TIMx synchronization described in *Section 26.3.31: ADC triggers* for TIM1)

The general-purpose timer TIM2, basic timers (TIM6), and advanced-control timers (TIM1) can be used to generate the ADC triggering event through the timer outputs tim_oc and tim_trgo. Low-power timers (LPTIM1/LPTIM2) can be used to generate the ADC triggering event through the LPTIM channels in addition to EXTI on channel 11 and 15.

Triggering signals

For ADC1, the input triggering signals and the description of the interconnection between ADC1, and timers, are given in:

- *Table 110: ADC interconnection*
- *Section 19.4.18: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN,JEXTSEL, JEXTEN)*
- *Section 19.4.25: Timing diagrams example (single/continuous modes, hardware/software triggers)*

Active power mode

This interconnection is active in Run and Sleep modes, assuming that its trigger event line is active as well. The timers are active in Run and Sleep mode only. The effects of low-power modes are given in:

- *Table 183: Effect of low-power modes on TIM1*
- *Table 201: Effect of low-power modes on TIM2/TIM3*
- *Table 222: Effect of low-power modes on the LPTIM*

14.3.3 ADC analog watchdogs as triggers to timers

From ADC1 to TIM1

Purpose

The internal analog watchdog output signals coming from ADC1 is connected to on-chip timer. ADC1 can provide trigger event through analog watchdog signals to advanced-control timer (TIM1) in order to reset, start, stop or clock the counter.

Settings description of the ADC analog watchdog and timer trigger, are provided in:

- *Section 26.3.6: External trigger input* for TIM1
- *Table 172: Interconnect to the tim_etr input multiplexer* for the internal ADC1 sources connected to TIM1 (tim_etr) input multiplexer
- *Section 19.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)* for the ADC1/ADC_AWDx signal output generation

Triggering signals

The output (from ADC) is on signals ADC_AWDx, with x = 1, 2, 3 (three watchdogs per ADC). The input (to timer) is on signal TIMx_ETR (external trigger).

Active power mode

ADC1 is active in Run and Sleep modes.

14.3.4 Triggers to DAC

From timer (TIM1/TIM2/TIM3/TIM6/TIM7), Low-power timers (LPTIM1/LPTIM2) and EXTI to DAC (DAC1/DAC2)

Purpose

General-purpose timers (TIM2/TIM3), basic timers (TIM6/TIM7), advanced-control timer (TIM1), LP timers (LPTIM1/LPTIM2) outputs channels (lptim1_ch1 and lptim2_ch1) and EXTI can be used as triggering event to start a DAC conversion.

Triggering signals

The output (from timer) on the TIMx_TRGO signal and from LP timers are directly connected to corresponding DAC inputs.

The selection of input triggers on DAC is provided in:

- *Table 132: DAC interconnection*
- *Section 21.4.8: DAC trigger selection*

Active power mode

This interconnect is active in Run, Sleep, and Stop modes.

14.3.5 Triggers to DTS

From EXTI, LP timers (LPTIM1/LPTIM2) to DTS.

Purpose

A temperature measurement, can be triggered either by software, by LP timers, or by an external event.

Triggering signals

The input triggering signals and the description of the interconnection between DTS, and LP timers, are given in *Table 125: Trigger configuration*.

14.3.6 LP timer to LP timer connection

From LPTIM1 to LPTIM2 and from LPTIM2 to LPTIM1.

Purpose

LPTIM1 output channel can be used as clock source to increment the LPTIM2 counter through lptim2_in1_mux3 input.

LPTIM2 output channel can be used as clock source to increment the LPTIM1 counter through lptim1_in1_mux3 input.

Triggering signals

LPTIM1 output lptim1_ch2 is the input to lptim2_in1_mux3.

LPTIM2 output lptim2_ch2 is the input to lptim1_in1_mux3.

The LPTIM1/2 input1 connection is given in *Table 215: LPTIM1/2 input 1 connections*.

Active power mode

This feature is available under Run, Sleep and Stop modes.

14.3.7 Internal clock sources to timers

From HSE, LSE, LSI, CSI, HSI and MCO1/MCO2 to timers (TIM2/TIM3) and LP timers (LPTIM1/LPTIM2)

Purpose

A timer input or timer counter can receive different clock sources and can be used to calibrate internal oscillator on a reference clock for example.

External clocks (HSE, LSE), internal clocks (LSI, CSI, HSI) and microcontroller output clock (MCO1 and MCO2) can be used as input to timers:

- HSE, LSE, CSI, HSI and LSI are assigned to general purpose timers TIM2/TIM3 as external inputs signals. CSI/HSI/LSI can be selected as counter clock provided by an external clock source in mode1 (tim_ti1 or tim_ti2 signals) and mode2 (external trigger input tim_etr_in). Inputs assignment and clock selection description are detailed in:
  - *Section 27.4.5: Clock selection* for TIM2/TIM3
  - External clock mode1: *Table 190: Interconnect to the tim_ti1 input multiplexer* for tim_ti1_in1 (LSI-TIM2), tim_ti1_in2 (LSE-TIM2), tim_ti1_in4 (RCC_HSE_1MHz-TIM3). *Table 191: Interconnect to the tim_ti2 input multiplexer* for tim_ti1_in1 (HSI/1024 -TIM2), tim_ti1_in2 (CSI/128 -TIM2), tim_ti1_in3 (HSI/1024 -TIM3), tim_ti1_in1 (CSI/128 -TIM3)
  - External clock mode2: *Table 195: Interconnect to the tim_etr input multiplexer*, tim_etr3 (LSE), for TIM2
- Microcontroller output clock (MCO1/MCO2) is connected as external input to general-purpose timers TIM2/TIM3. This allows the calibration of the HSI/CSI system clocks (with TIM2 and LSE) or LSI (with TIM3 and HSE). This is also used to precisely

measure LSI or CSI (with TIM3 and HSI) oscillator frequency. When the low-speed external (LSE) oscillator is used, no additional hardware connections are required.

- RCC_HSE_1MHz, LSI and LSE can be selected as input capture 2 to LPTIM1 as described in *Table 218: LPTIM1/2 input capture 2 connections*.
- MCO1 and MCO2 can be selected as input capture 1 to LPTIM1 and LPTIM2 respectively as described in *Table 215: LPTIM1/2 input 1 connections*.
- HSI/1024, CSI/128 and HSI/8 can be selected as input capture 2 to LPTIM2 as described in *Table 218: LPTIM1/2 input capture 2 connections*.

Triggering signals

lptim_ic2_mux1 LPTIM input capture selection can be set in the LPTIM configuration register 2 (LPTIM_CFGR2). For timers, the internal clock signal can be selected as counter clock provided by an external clock source in mode1 (tim_ti1_in) and mode2 (external trigger input tim_etr_in).

Active power mode

This feature is available under Run and Sleep modes.

14.3.8 Triggers to low-power timers

From comparator (COMP1), CPU (EVENTOUT), GPDMA1/2, TAMP and RTC alarm to LP timers (LPTIM1/LPTIM2)

Purpose

LPTIM1/LPTIM2 counters may be started either by software or after the detection of an active edge on one of the eight trigger inputs (see *Section 29.4.7: Trigger multiplexer*).

RTC alarm A/B, TAMP input detection, COMP1_OUT, GPDMA transfert complete, and CPU EVENTOUT can be used as trigger to start LPTIM counters (LPTIM1/2)..

Triggering signals

This trigger feature is described in and the following sections. The input selection is described in *Table 214: LPTIM1/2 external trigger connections*.

Active power mode

This interconnection remains active down to Stop mode.

14.3.9 RTC wakeup as inputs to timers

From RTC to timers (TIM2)

Purpose

RTC wakeup interrupt can be used as input to general-purpose timer (TIM2) channel 1.

Triggering signals

RTC wakeup signal is connected to tim_ti1_in3 signal as described in *Table 190: Interconnect to the tim_ti1 input multiplexer* for TIM2.

Active power mode

This interconnection is active down to Stop. Timers are not active but the count is performed at wakeup.

14.3.10 Blanking sources to comparator

From timers (TIM1/TIM2/TIM3) to comparator COMP1

Purpose

Advanced-control timer TIM1 and general-purpose timers (TIM2/TIM3) can be used as blanking window input to COMP1. The blanking function is described in *Section 22.3.6: Comparator output blanking function*.

The blanking sources are given in *Comparator configuration register 1 (COMP_CFGR1)*, BLANKING[3:0].

Triggering signals

Timer output signal TIMx_Ocx are the inputs to blanking source of COMP1.

Active power mode

This feature is available under Run and Sleep modes.

14.3.11 Comparators as inputs, trigger or break signals to timers

From comparator COMP1 to timers (TIM1/TIM2/TIM3)

Purpose

The comparators (COMP1) output can be connected to timers (TIM1/TIM2/TIM3) input captures, TIMx_ETR or timer break signals. The connection to ETR is described in *Section 26.3.6: External trigger input*.

Comparators (COMP1) output values can also generate break input signals for timers (such TIM1). The sources for break (tim_brk) channel are one of the following:

- external: connected to one of the TIMx_BKIN pin (as per selection done in the AFIO controller) with polarity selection and optional digital filtering
- internal: coming from comparators, tim_brk_cmpx input (refer to *Section 26.3.2: TIM1 pins and internal signals* for product specific implementation).

Triggering signals

The tim_etr and tim_brk signals connected to TIM1 (coming from COMP1) are given in:

- tim_etr (*Table 172: Interconnect to the tim_etr input multiplexer*): external trigger internal input bus

  These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulse width control.
- tim_brk (*Table 173: Timer break interconnect* and *Table 174: Timer break2 interconnect*)
- *Section 26.3.6: External trigger input*
- *Section 26.3.18: Using the break function*

For TIM2/TIM3, the sources connected to the tim_ti[1..4] input multiplexers coming from comparators and some other peripherals, are given in:

- *Table 190: Interconnect to the tim_ti1 input multiplexer*
- *Table 191: Interconnect to the tim_ti2 input multiplexer*
- *Table 192: Interconnect to the tim_ti3 input multiplexer*
- *Table 193: Interconnect to the tim_ti4 input multiplexer*
- *Table 195: Interconnect to the tim_etr input multiplexer*
- *Section 27.4.22: Timers and external trigger synchronization*
- *Section 27.5.26: TIMx timer input selection register (TIMx_TISEL)(x = 2, 3)*

Active power mode

Run, Sleep and wakeup capability in Stop mode for trigger sources. Input and break remain active in same low-power modes as timers activity, on Run and Sleep modes.

14.3.12 System errors as break signals to timers

From system errors to timers.

Purpose

CSS, CPU lockup, SRAM1/SRAM2 ECC double errors, FLASH ECC double-error detection and PVD can generate system errors in the form of timer break toward timer TIM1.

The purpose of the break function is to protect power switches driven by PWM signals generated by the timer.

See *Section 13.5.11: SBS Class B register (SBS_CFGR2)* for more details.

Triggering signals

The possible sources of break are described in:

- *Section 26.3.18: Using the break function* for TIM1
- *Table 175: System break interconnect* for TIM1

Active power mode

Timers are optionally active in Run and Sleep modes. The effects of low-power modes on TIMx are given in:

- *Table 183: Effect of low-power modes on TIM1*
- *Table 201: Effect of low-power modes on TIM2/TIM3*

14.3.13 Triggers to GPDMA1/2

From EXTI, RTC (alarm/wakeup), TAMP, timers (TIM2), comparators (COMP1), GPDMA1/2 transfer complete (gpdma1_chx_tcf/gpdma2_chx_tcf), to GPDMA1/2.

Purpose

A GPDMA trigger can be assigned to a GPDMA channel x. A programmed GPDMA 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 LLI link transfer. More details are given in the sections below:

- *Section 15.3.7: GPDMA triggers*
- *Section 15.4.12: GPDMA triggered transfer*
- *GPDMA channel x transfer register 2 (GPDMA_CxTR2)* for more details on:
  - Trigger selection TRIGSEL[5:0] field
  - Trigger mode (LLI) defined by TRIGM[1:0].
  - Trigger polarity as defined by TRIGPOL[1:0]

Triggering signals

GPDMA trigger mapping is specified in *Table 87: Programmed GPDMA1/2 request*, according to GPDMA_CxTR2.TRIGSEL[5:0].

Active power mode

Assuming sources are active down to Sleep modes, this interconnection remains functional in Sleep mode.

Refer to:

- *Section 15.6: GPDMA in low-power modes*

14.3.14 Internal analog signals to analog peripherals

From internal analog source to ADC1, comparator COMP1 and OPAMP1

Purpose

The internal reference voltage (VREFINT), the internal temperature sensor (VSENSE) and VBAT monitoring channel are connected to ADC1 input channels. In addition, the internal digital core voltage (VCORE) is connected to ADC1 input channel.

DAC channels (DAC1_OUT/DAC2_OUT), VREFINT, VSENSE and VBAT monitoring channel are connected to comparator (COMP1).

OPAMP1 output can be connected to ADC1 input channel through the GPIO. DAC1_OUT1 can be connected to OPAMP1_VINP.

Refer to *Table 131: DAC internal input/output signals* for:

- dac_out1 analog output DAC channel1, output, for on-chip peripherals
- dac_out2 analog output DAC channel2, output, for on-chip peripherals

This is according:

- *Section 19.2: ADC main features*
- *Section 19.4.11: Channel selection (SQRx, JSQRx)*
- *Table 149: Possible connections for OPAMP*
- *Section 22.3.2: COMP pins and internal signals*

Active power mode

These interconnections remain in Stop modes if the selected peripheral is kept active. Refer to:

- *Section 22.4: COMP low-power modes*
- *Section 23.4: OPAMP low-power modes*
- *Section 21.5: DAC in low-power modes*

14.3.15 Clock source for the DAC sample and hold mode

LSI/LSE to DAC1/DAC2

Purpose

DAC1/DAC2 can run in Stop mode. The sample and hold block and its associated registers use the LSI or LSE clock source (dac_hold_ck) in Stop mode.

Refer to *Table 131: DAC internal input/output signals*:

dac_hold_ck, Input, DAC low-power clock used in sample and hold mode

Active power mode

This feature remains available down to Stop mode.

14.3.16 Internal tamper sources

From internal peripherals, clocks or monitoring to tamper.

Purpose

In order to detect any abnormal activity or tentative to corrupt the device, tampers are introduced and alert the system of such undesired event. Different actions can be taken in consequences.

- The backup domain voltage monitor can generate a tamper event when the battery voltage is above the functional range. Refer to *Section 9.6.6: Backup domain voltage monitoring* for more details.
- The temperature monitor can generate a tamper event when the temperature is above or below the functional range. Refer to *Section 9.6.7: Temperature monitoring* for more details.
- The clock security system on LSE can generate a tamper event in case of LSE failure detection (when the LSE disappears or in case of over frequency). Refer to *Section 10.4.10: Clock security system (CSS)* for more details
- The clock security system on HSE can generate a tamper event in case of HSE failure detection (when the HSE disappears or in case of over frequency). Refer to *Section 10.4.10: Clock security system (CSS)* for more details

List of tamper sources can be found in *Table 246: TAMP interconnection*.

Active power mode

This interconnection is active in all power modes if the tamper source is activated.

14.3.17 Output from tamper to RTC

From TAMP to RTC

Purpose

The RTC can timestamp a tamper event in order to retrieve history in time of such detection. The RTC can also control GPIOs and set a signal based on tamp or alarm status outside the MCU.

Refer to *Section 32.3.3: GPIOs controlled by the RTC and TAMP* for more details.

Active power mode

This interconnection remains active in all power modes.

15 General purpose direct memory access controller (GPDMA)

15.1 GPDMA introduction

The general purpose direct memory access (GPDMA) controller is a bus master and system peripheral.

The GPDMA 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.

15.2 GPDMA main features

- Dual 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

  Transfer 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 GPDMA channels:
  - Per channel FIFO for queuing source and destination transfers (see *Section 15.3.2*)
  - Intra-channel GPDMA transfers chaining via programmable linked-list into memory, supporting two execution modes: run-to-completion and link step mode
  - Intra-channel and inter-channel GPDMA transfers chaining via programmable GPDMA input triggers connection to GPDMA 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 reordering, packing or unpacking, 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 burst transfers

  - 2D source and destination addressing: programmable signed address offsets between successive burst transfers (non-contiguous addressing within a block, combined with programmable signed address offsets between successive blocks, at a second 2D/repeated block level, for a reduced set of channels (see *Section 15.3.2*)
  - Support for scatter-gather (multi-buffer transfers), data interleaving and deinterleaving via 2D addressing
  - Programmable GPDMA request and trigger selection
  - Programmable GPDMA 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 GPDMA linked-list control registers
  - Channel abort and restart
- Debug:
  - Channel suspend and resume support
  - Channel status reporting, including FIFO level, and event flags
- Privileged/unprivileged support:
  - Support for privileged and unprivileged GPDMA transfers, independently at a channel level
  - Privileged-aware AHB slave port

15.3 GPDMA implementation

15.3.1 GPDMA instances

There are two instances of the GPDMA in the devices, named as GPDMA1 and GPDMA2.

Each GPDMA instance has the same channel-based implementation and is connected to the same requests and triggers, as detailed in the following sub-sections.

15.3.2 GPDMA channels

A given GPDMA channel x is implemented with the following features and intended usage. To make the best use of the GPDMA performances, the table below lists some general recommendations, allowing the user to select and allocate a channel, given its implemented FIFO size and the requested GPDMA transfer.

Table 85. GPDMA1/2 channel implementation

Channel x	Hardware parameters		Features
	dma_fifo_size[x]	dma_addressing[x]	
x = 0 to 3	2	0	Channel x (x = 0 to 3) is implemented with: – a FIFO of 8 bytes, 2 words – fixed/contiguously incremented addressing These channels must be typically allocated for GPDMA transfers between an APB or AHB peripheral and SRAM.
x = 4 to 5	4	0	Channel x (x = 4 to 5) is implemented with: – a FIFO of 32 bytes, 8 words – fixed/contiguously incremented addressing These channels may be used for GPDMA transfers between a demanding AHB peripheral and SRAM, or for transfers from/to external memories.
x = 6 to 7	4	1	Channel x (x = 6 to 7) is implemented with: – a FIFO of 32 bytes, 8 words – 2D addressing These channels may be used for GPDMA transfers between a demanding AHB peripheral and SRAM, or for transfers from/to external memories.

15.3.3 GPDMA in low-power modes

The GPDMA wake-up feature is implemented in the device low-power modes as per the table below.

Table 86. GPDMA1/2 wake-up in low-power modes

Feature	Low-power modes
Wake-up	GPDMA1/2 in Sleep mode

15.3.4 GPDMA requests

A GPDMA request from a peripheral can be assigned to a GPDMA channel x, via REQSEL[7:0] in GPDMA_CxTR2, provided that SWREQ = 0.

The GPDMA requests mapping is specified in the table below.

Table 87. Programmed GPDMA1/2 request

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
0	adc1_dma
1	Reserved
2	dac1_ch1_dma
3	dac1_ch2_dma
4	tim6_upd_dma

Table 87. Programmed GPDMA1/2 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
5	tim7_upd_dma
6	spi1_rx_dma
7	spi1_tx_dma
8	spi2_rx_dma
9	spi2_tx_dma
10	spi3_rx_dma
11	spi3_tx_dma
12	i2c1_rx_dma
13	i2c1_tx_dma
14	Reserved
15	i2c2_rx_dma
16	i2c2_tx_dma
17	Reserved
18	Reserved
19	Reserved
20	Reserved
21	usart1_rx_dma
22	usart1_tx_dma
23	usart2_rx_dma
24	usart2_tx_dma
25	usart3_rx_dma
26	usart3_tx_dma
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved
32	Reserved
33	Reserved
34	Reserved
35	Reserved
36	Reserved
37	Reserved
38	Reserved
39	Reserved

Table 87. Programmed GPDMA1/2 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
40	Reserved
41	Reserved
42	Reserved
43	Reserved
44	Reserved
45	lpuart1_rx_dma
46	lpuart1_tx_dma
47	Reserved
48	Reserved
49	Reserved
50	Reserved
51	Reserved
52	Reserved
53	Reserved
54	Reserved
55	Reserved
56	Reserved
57	Reserved
58	tim1_cc1_dma
59	tim1_cc2_dma
60	tim1_cc3_dma
61	tim1_cc4_dma
62	tim1_upd_dma
63	tim1_trg_dma
64	tim1_com_dma
65	Reserved
66	Reserved
67	Reserved
68	Reserved
69	Reserved
70	Reserved
71	Reserved
72	tim2_cc1_dma
73	tim2_cc2_dma
74	tim2_cc3_dma

Table 87. Programmed GPDMA1/2 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
75	tim2_cc4_dma
76	tim2_upd_dma
77	tim3_cc1_dma
78	tim3_cc2_dma
79	tim3_cc3_dma
80	tim3_cc4_dma
81	tim3_upd_dma
82	tim3_trg_dma
83	Reserved
84	Reserved
85	Reserved
86	Reserved
87	Reserved
88	Reserved
89	Reserved
90	Reserved
91	Reserved
92	Reserved
93	Reserved
94	Reserved
95	Reserved
96	Reserved
97	Reserved
98	Reserved
99	Reserved
100	Reserved
101	Reserved
102	lptim1_ic1_dma
103	lptim1_ic2_dma
104	lptim1_ue_dma
105	lptim2_ic1_dma
106	lptim2_ic2_dma
107	lptim2_ue_dma
108	Reserved
109	Reserved

Table 87. Programmed GPDMA1/2 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
110	Reserved
111	hash_in_dma
112	Reserved
113	Reserved
114	Reserved
115	Reserved
116	Reserved
117	Reserved
118	Reserved
119	Reserved
120	i3c1_rx_dma
121	i3c1_tx_dma
122	i3c1_tc_dma
123	i3c1_rs_dma
124	Reserved
125	Reserved
126	Reserved
127	Reserved
128	Reserved
129	Reserved
130	Reserved
131	Reserved
132	Reserved
133	Reserved
134	Reserved
135	Reserved
136	i3c2_rx
137	i3c2_tx
138	i3c2_tc
139	i3c2_rs

15.3.5 GPDMA block requests

Some GPDMA requests must be programmed as a block request, and not as a burst request. Then BREQ in GPDMA_CxTR2 must be set for a correct GPDMA execution of the requested peripheral transfer at the hardware level.

Table 88. Programmed GPDMA1/2 request as a block request

GPDMA block requests
lptim1_ue_dma
lptim2_ue_dma

15.3.6 GPDMA channels with peripheral early termination

A GPDMA channel, if implemented with this feature, can support the early termination of the data transfer from the peripheral which does also support this feature.

Table 89. GPDMA1/2 channel with peripheral early termination

GPDMA channel x with peripheral early termination
x = 0 and x = 7

This GPDMA support is activated when the channel x is programmed with GPDMA_CxTR2.PFREQ = 1. Then, the peripheral itself can initiate and request a data transfer completion, before that the GPDMA has transferred the whole block (see *Section 15.4.14* for more details).

Table 90. Programmed GPDMA1/2 request with peripheral early termination

Programmed GPDMA channel x request with peripheral early termination
i3c1_rx_dma
i3c2_rx_dma

15.3.7 GPDMA triggers

A GPDMA trigger can be assigned to a GPDMA channel x, via TRIGSEL[5:0] in GPDMA_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).

Table 91.Programmed GPDMA1/2 trigger

GPDMA_CxTR2.TRIGSEL[5:0]	Selected GPDMA trigger
0	exti0
1	exti1
2	exti2
3	exti3
4	exti4
5	exti5
6	exti6
7	exti7
8	tamp_trg1

Table 91.Programmed GPDMA1/2 trigger (continued)

GPDMA_CxTR2.TRIGSEL[5:0]	Selected GPDMA trigger
9	tamp_trg2
10	Reserved
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	gpdma1_ch0_tc
19	gpdma1_ch1_tc
20	gpdma1_ch2_tc
21	gpdma1_ch3_tc
22	gpdma1_ch4_tc
23	gpdma1_ch5_tc
24	gpdma1_ch6_tc
25	gpdma1_ch7_tc
26	gpdma2_ch0_tc
27	gpdma2_ch1_tc
28	gpdma2_ch2_tc
29	gpdma2_ch3_tc
30	gpdma2_ch4_tc
31	gpdma2_ch5_tc
32	gpdma2_ch6_tc
33	gpdma2_ch7_tc
34	tim2_trgo
35	Reserved
36	Reserved
37	Reserved
38	Reserved
39	Reserved
40	Reserved
41	Reserved
42	Reserved
43	Reserved

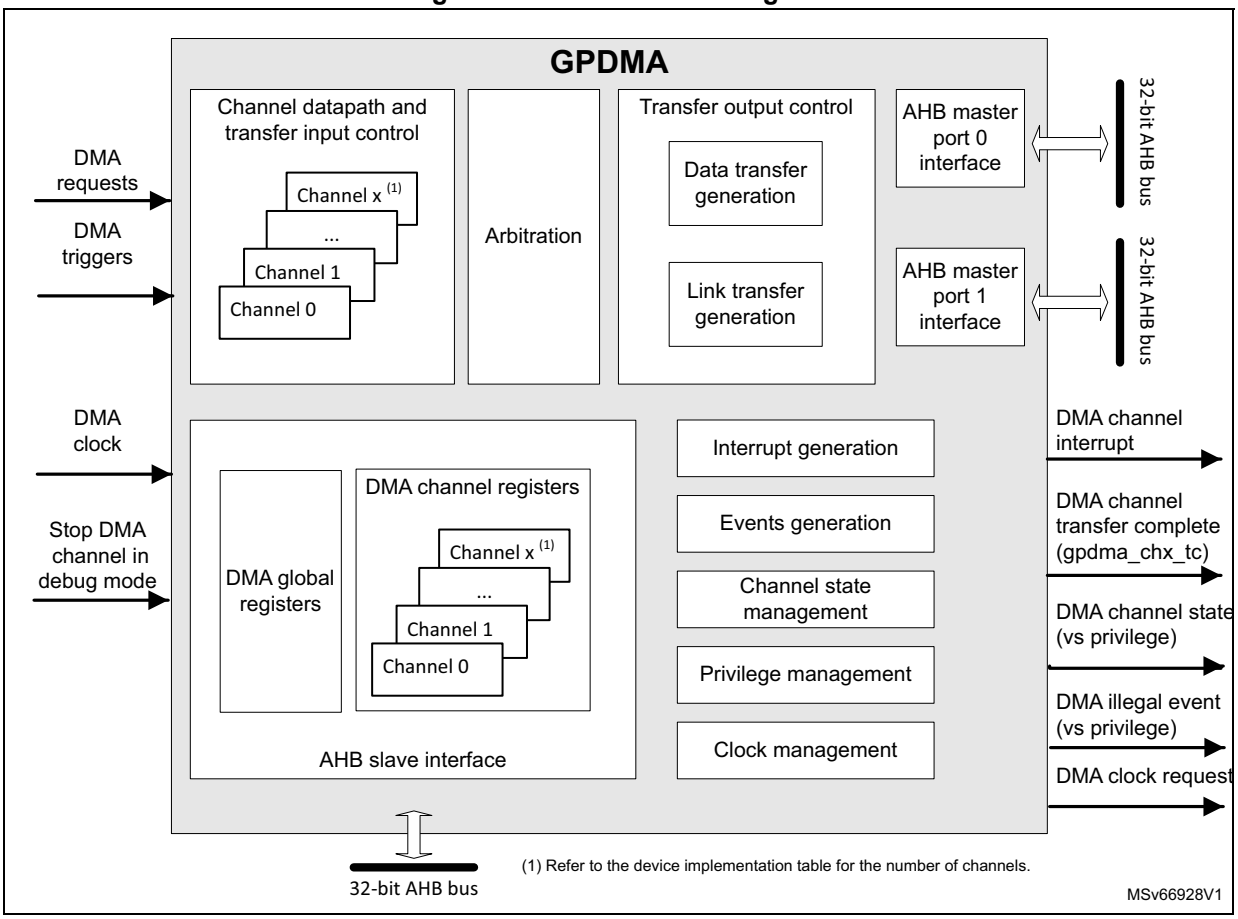
Table 91.Programmed GPDMA1/2 trigger (continued)

GPDMA_CxTR2.TRIGSEL[5:0]	Selected GPDMA trigger
44	comp1_out
45	eventout

15.4 GPDMA functional description

15.4.1 GPDMA block diagram

Figure 50. GPDMA block diagram



Note: The DMA illegal event is not generated on the STM32H503xx devices.

15.4.2 GPDMA channel state and direct programming without any linked-list

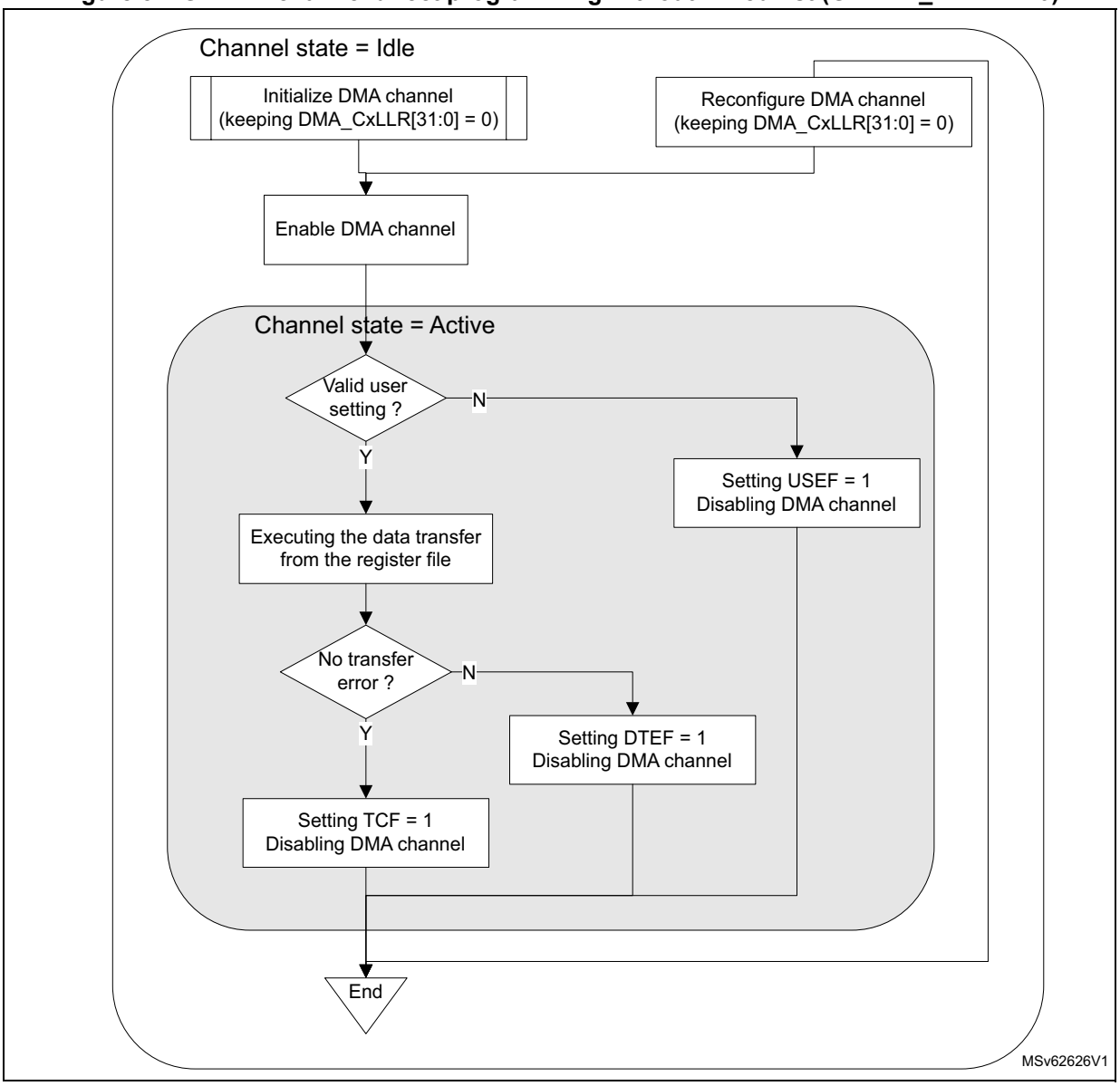
After a GPDMA reset, a GPDMA channel x is in idle state. When the software writes 1 in GPDMA_CxCR.EN, the channel takes into account the value of the different channel configuration registers (GPDMA_CxXXX), switches to the active/non-idle state and starts to execute the corresponding requested data transfers.

After enabling/starting a GPDMA channel transfer by writing 1 in GPDMA_CxCR.EN, a GPDMA channel interrupt on a complete transfer notifies the software that the GPDMA

channel is back in idle state (EN is then deasserted by hardware) and that the channel is ready to be reconfigured then enabled again.

Figure 51 illustrates this GPDMA direct programming without any linked-list (GPDMA_CxLLR = 0).

Figure 51. GPDMA channel direct programming without linked-list (GPDMA_CxLLR = 0)



15.4.3 GPDMA channel suspend and resume

The software can suspend on its own a channel still active, with the following sequence:

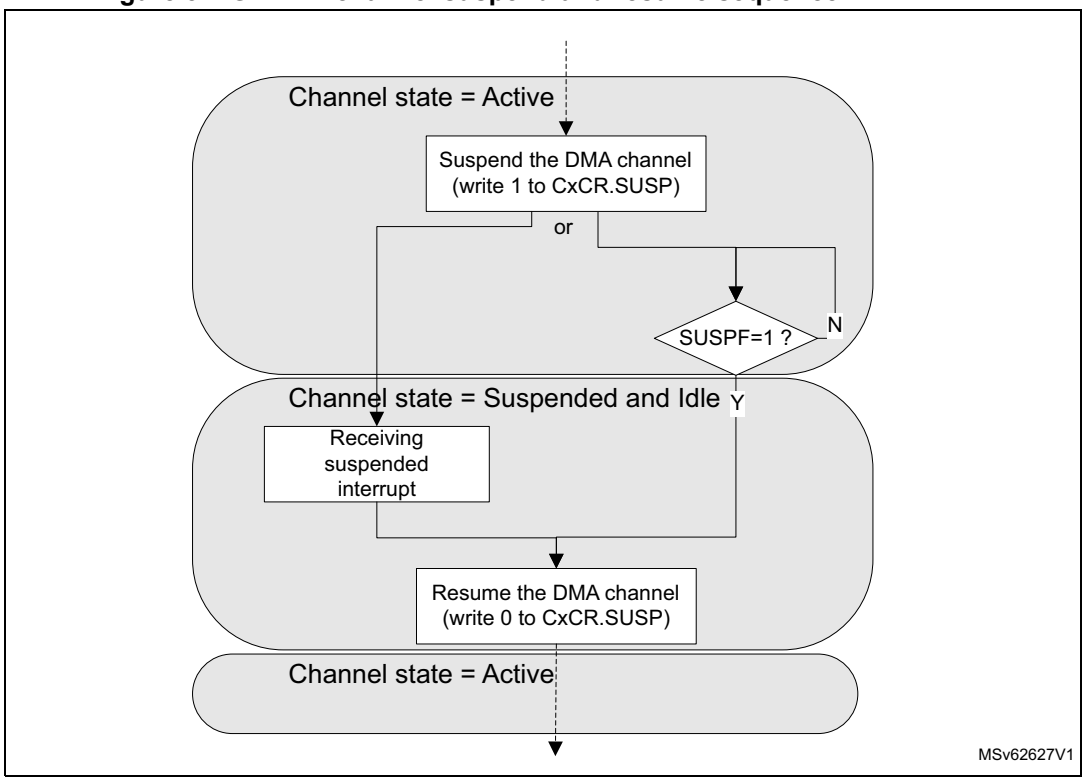
1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls the suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of

any ongoing GPDMA transfer over its master ports. Then the software can observe, in a steady state, any read register or register field that is hardware modifiable.

Note: An ongoing GPDMA transfer can be a data transfer (a source/destination burst 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 GPDMA_CxCR.SUSP.

Figure 52. GPDMA channel suspend and resume sequence



Note: A suspend and resume sequence does not impact the GPDMA_CxCR.EN bit. Suspending a channel (transfer) does not suspend a started trigger detection.

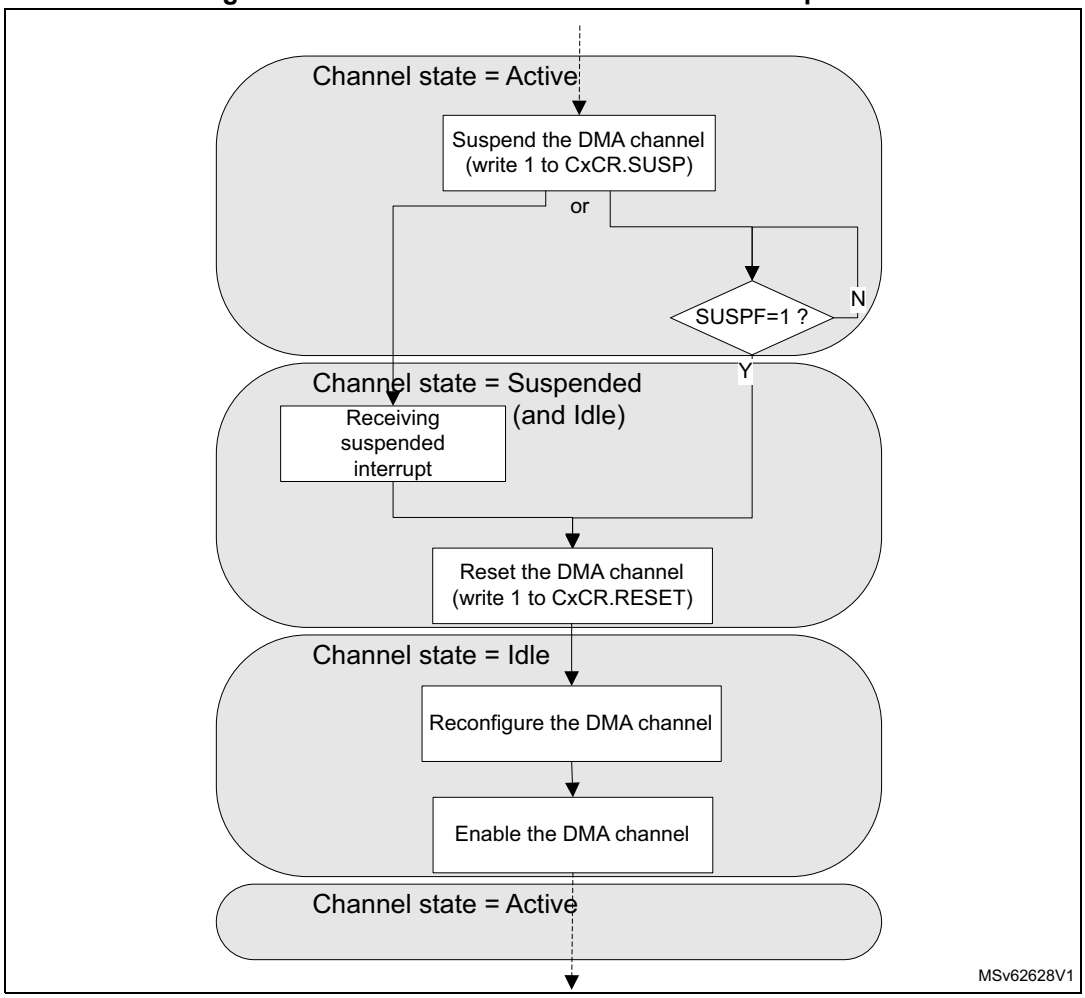
15.4.4 GPDMA channel abort and restart

Alternatively, like for aborting a continuous GPDMA 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 GPDMA_CxCR.SUSP bit.
2. The software polls suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master port.
3. The software resets the channel by writing 1 to GPDMA_CxCR.RESET. This causes the reset of the FIFO, the reset of the channel internal state, the reset of the GPDMA_CxCR.EN bit, and the reset of the GPDMA_CxCR.SUSP bit.

4. The software safely reconfigures the channel. The software must reprogram the hardware-modified GPDMA_CxBR1, GPDMA_CxSAR, and GPDMA_CxDAR registers.
5. In order to restart the aborted then reprogrammed channel, the software enables it again by writing 1 to the GPDMA_CxCR.EN bit.

Figure 53. GPDMA channel abort and restart sequence



15.4.5 GPDMA 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 GPDMA_CxLBAR)
- the link address offset (LA[15:2] field in GPDMA_CxLLR). The linked-list register GPDMA_CxLLR is the updated result from the data structure of the previous LLIn of the channel x.

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, plus UB2 and UT3, in GPDMA_CxLLR.

In linked-list mode, each GPDMA linked-list register (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR or GPDMA_CxLLR, plus GPDMA_CxTR3 or GPDMA_CxBR2) is conditionally and automatically updated from the next linked-list data structure in the memory, following the current value of the GPDMA_CxLLR register that was conditionally updated from the linked-list data structure of the previous LLI.

Static linked-list data structure

For example, when the update bits (UT1, UT2, UB1, USA, UDA and ULL, plus UB2 and UT3) in GPDMA_CxLLR are all asserted, the linked-list data structure in the memory is maximal with:

- channel x (x = 0 to 5) contiguous 32-bit locations, including GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR (see *Figure 54*) and including the first linked-list register file (LLI_0) and the next LLIs (such as LLI_1, LLI_2) in the memory
- channel x (x = 6 to 7), contiguous 32-bit locations, including GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2 (see *Figure 55*), and including the first linked-list register file (LLI_0) and the next LLIs (such as LLI_1, LLI_2) in the memory

Figure 54. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x

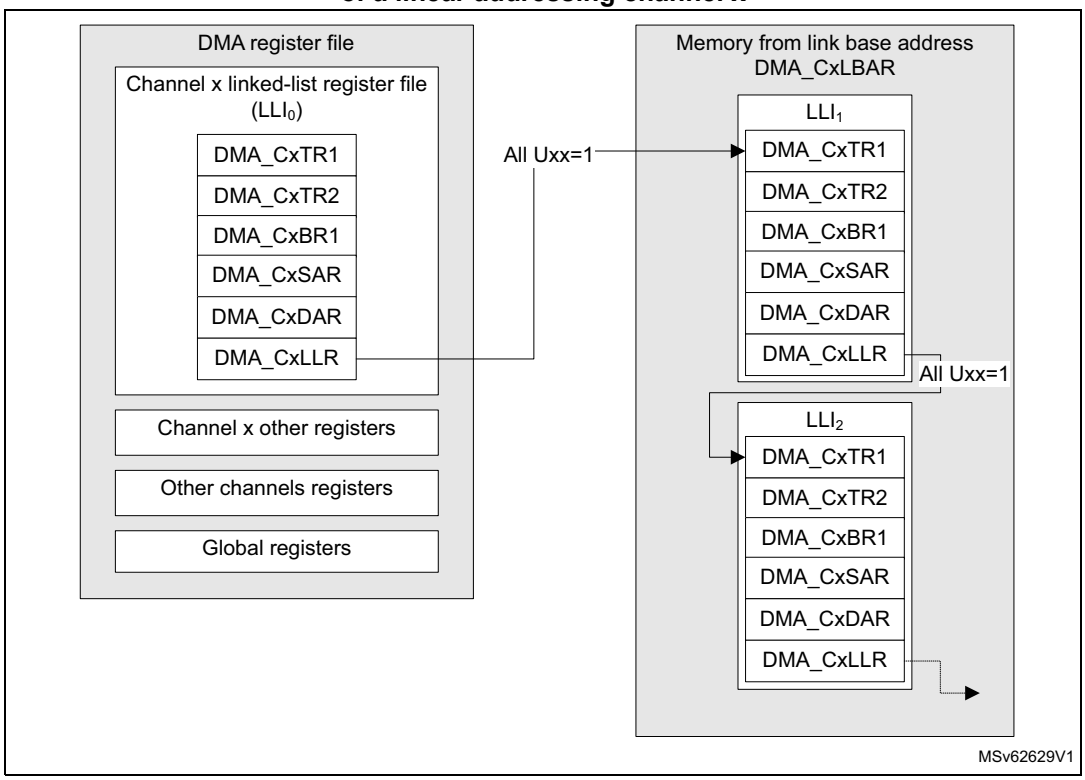
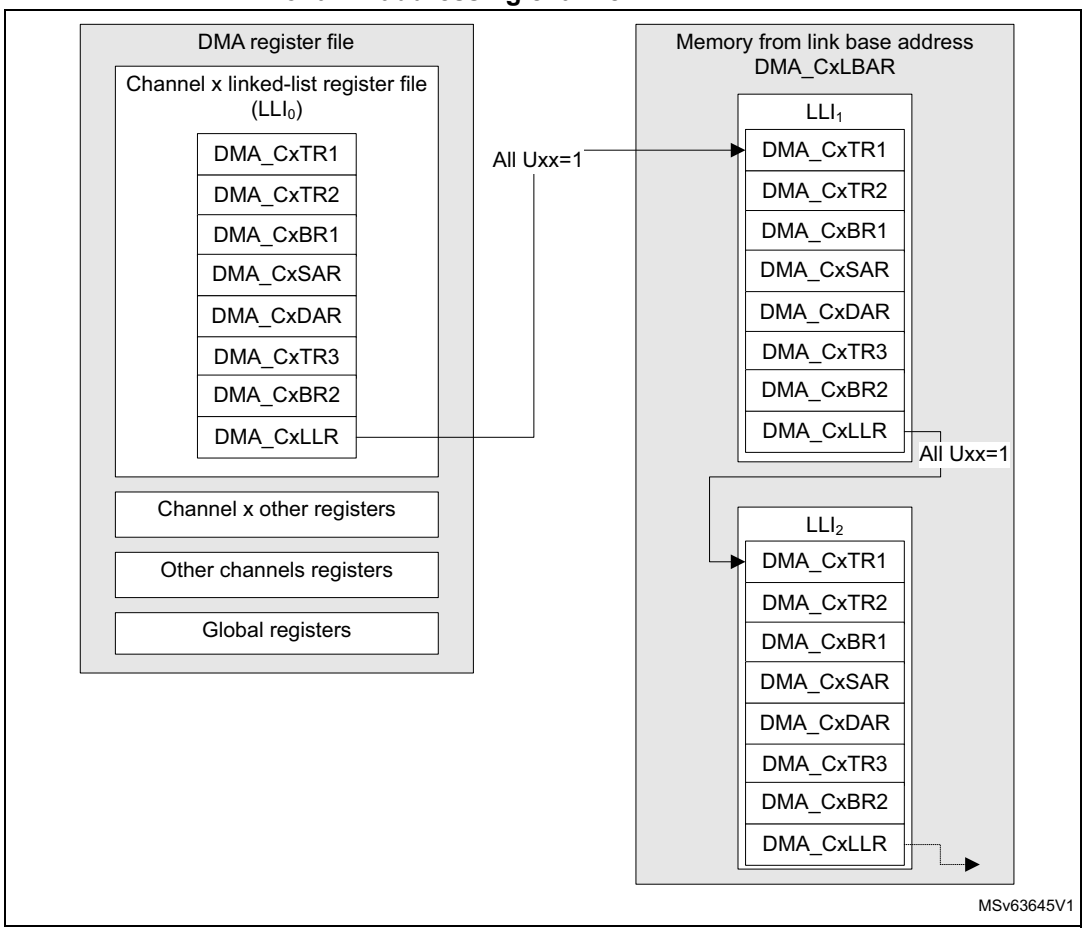


Figure 55. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x



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 GPDMA_CxLLR is pointing to the updated value of the GPDMA_CxTR2 instead of the GPDMA_CxTR1 which is not to be modified (see *Figure 56*).

Example: if UT1 = UB1 = USA = 0 and if UT3 = UB2 = 0, when channel x is with 2D addressing, and if UT2 = UDA = ULL = 1, the next LLI does not contain an (updated) value for GPDMA_CxTR1, nor GPDMA_CxBR1, nor GPDMA_CxSAR, nor GPDMA_CxTR3, nor GPDMA_CxBR2 when channel x is with 2D addressing. The next LLI contains an updated value for GPDMA_CxTR2, GPDMA_CxDAR, and GPDMA_CxLLR, as shown in *Figure 57*.

Figure 56. GPDMA dynamic linked-list data structure of a linear addressing channel x

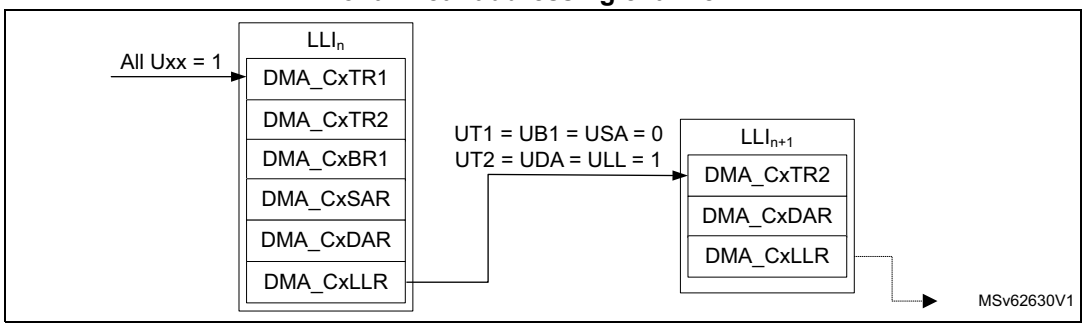
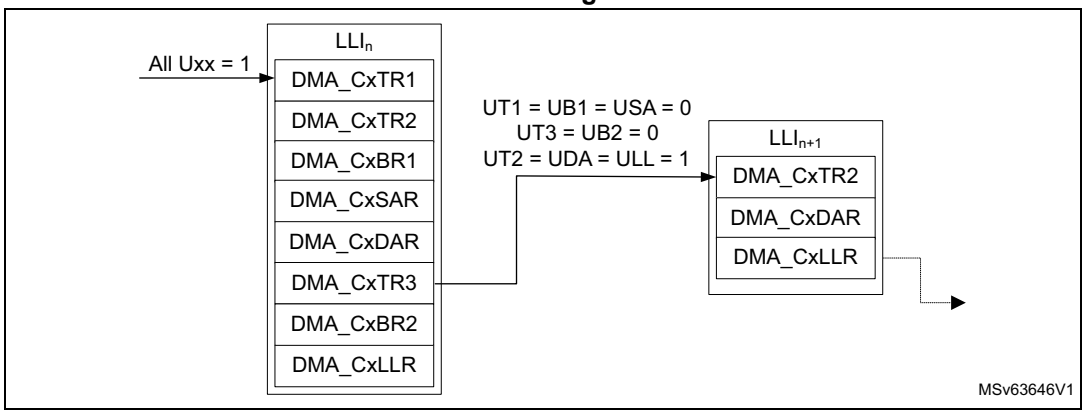


Figure 57. GPDMA dynamic linked-list data structure of a 2D addressing channel x



The user must program GPDMA_CxLLR for each LLI_n to be 32-bit aligned and not to exceed the 64-Kbyte addressable space pointed by GPDMA_CxLBAR.

15.4.6 Linked-list item transfer execution

A LLI_n transfer is the sequence of:

1. a data transfer: GPDMA executes the data transfer as described by the GPDMA internal register file (this data transfer can be void/null for LLI_0)
2. a conditional link transfer: GPDMA automatically and conditionally updates its internal register file by the data structure of the next LLI_{n+1}, as defined by the GPDMA_CxLLR value of the LLI_n.

Note: The initial data transfer as defined by the internal register file (LLI_0) can be null (GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxTR2.PFREQ = 0) provided that the conditional update bit UB1 in GPDMA_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 GPDMA usage, a GPDMA channel x can be executed as described by the full linked-list (run-to-completion mode, GPDMA_CxCR.LSM = 0) or a GPDMA channel x can be programmed for a single execution of a LLI (link step mode, GPDMA_CxCR.LSM = 1), as described in the next sections.

15.4.7 GPDMA channel state and linked-list programming in run-to-completion mode

When GPDMA_CxCR.LSM = 0 (in full list execution mode, execution of the full sequence of LLIs, named run-to-completion mode), a GPDMA channel x is initially programmed, started by writing 1 to GPDMA_CxCR.EN, and after completed at channel level. The channel transfer is:

- configured with at least the following:
  - the first LLI_0, internal linked-list register file: GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2
  - the last LLI_N, described by the linked-list data structure in memory, as defined by the GPDMA_CxLLR reflecting the before last LLI_{N-1}
- completed when GPDMA_CxLLR[31:0] = 0, GPDMA_CxBR1.BRC[10:0] = 0, and GPDMA_CxBR1.BNDT[15:0] = 0, at the end of the last LLI_{N-1} transfer

GPDMA_CxLLR[31:0] = 0 is the condition of a linked-list based channel completion and means the following:

- The 16 low significant bits GPDMA_CxLLR.LA[15:0] of the next link address are null.
- All the update bits GPDMA_CxLLR.Uxx are null (UT1, UT2, UB1, USA, UDA and ULL, plus UB2 and UT3).

The channel may never be completed when GPDMA_CxLLR.LSM = 0:

- If the last LLI_N is recursive, pointing to itself as a next LLI:
  - either GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated by the same value
  - or GPDMA_CxLLR.ULL = 0
- If LLI_N is pointing to a previous LLI

In the regular data transfer completion at a block level, GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0 (if present). Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 15.4.14* for more details).

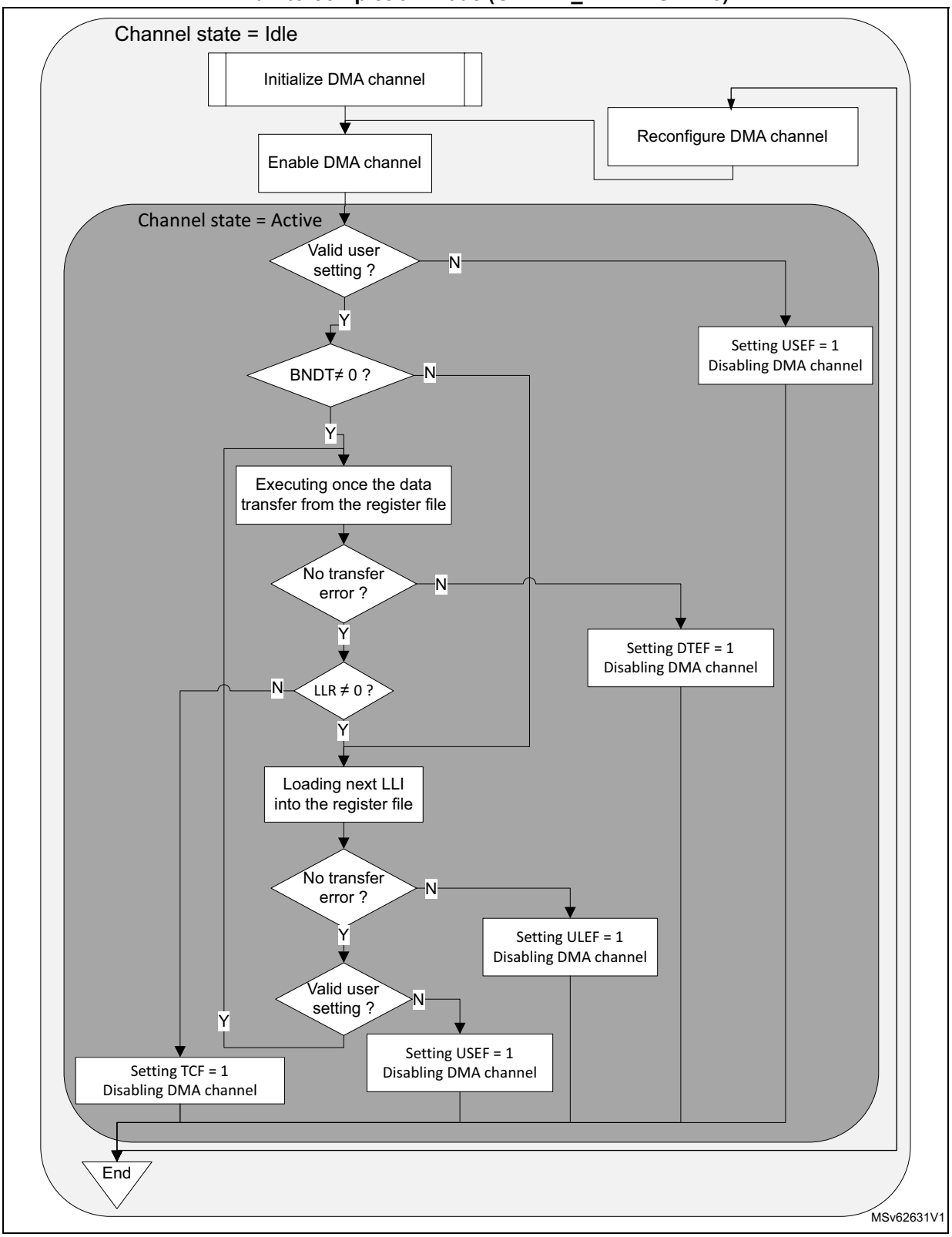
In the typical run-to-completion mode, the allocation of a GPDMA channel, including its fine programming, is done once during the GPDMA initialization. In order to have a reserved data communication link and GPDMA service during run-time, for continuously repeated transfers (from/to a peripheral respectively 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 58 depicts the GPDMA channel execution and its registers programming in run-to-completion mode.

Note: Figure 58 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_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.

In Figure 58, BNDT ≠ 0 is the typical condition for starting the first data transfer in this figure. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 15.3.6).

Figure 58. GPDMA channel execution and linked-list programming in run-to-completion mode (GPDMA_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 GPDMA_CxTR2 in the data structure of the LLI_{n-1}:

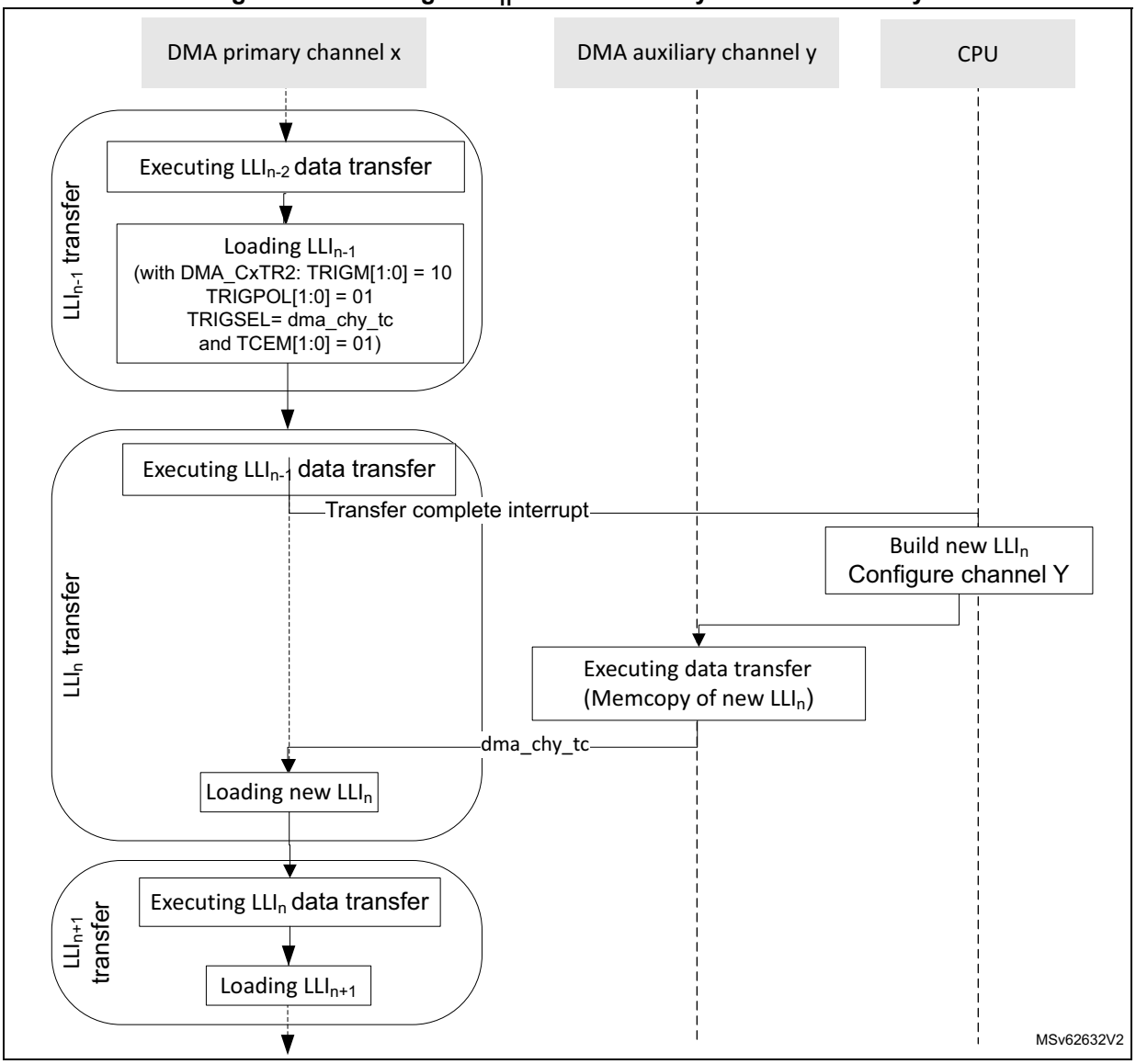
- TRIGM[1:0] = 10 (link transfer triggering mode)
- TRIGPOL[1:0] = 01 or 10 (rising or falling edge)
- TRIGSEL[5:0] (see *Section 15.3.7* for the trigger selection details)

Another auxiliary channel y can be used to store the channel x LLI_n in the memory and to generate a transfer complete event gpdma_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 59 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 59. Inserting a LLI_n with an auxiliary GPDMA channel y



15.4.8 GPDMA channel state and linked-list programming in link step mode

When GPDMA_CxCR.LSM = 1 (in link step execution mode, single execution of one LLI), 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 GPDMA channel transfer can be programmed at LLI level, started by writing 1 into GPDMA_CxCR.EN, and after completed at LLI level:

- The current LLI_n transfer is described with:
  - GPDMA_CxTR1 defines the source/destination elementary single/burst transfers.
  - GPDMA_CxBR1 defines the number of bytes at a block level (BNDT[15:0]) and, for channel x (x = 6 to 7), the number of blocks at a 2D/repeated block level (BRC[10:0]+1) and the incrementing/decrementing mode for address offsets.

  - GPDMA_CxTR2 defines the input control (request, trigger) and the output control (transfer complete event) of the transfer.
  - GPDMA_CxSAR/GPDMA_CxDAR define the source/destination transfer start address.
  - GPDMA_CxTR3 for channel x (x = 6 to 7) defines the source/destination additional address offset between burst transfers.
  - GPDMA_CxBR2 for channel x (x = 6 to 7) defines the source/destination additional address offset between blocks at a 2D/repeated block level.
  - GPDMA_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 GPDMA_CxBR1.BRC[10:0] = 0, and GPDMA_CxBR1.BNDT[15:0] = 0
  - after the (conditional) update of the GPDMA 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 GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated by the same value, or GPDMA_CxLLR.ULL = 0), a channel in link step mode is completed after each repeated single execution of this LLI.

In the regular data transfer completion at a block level, GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0. Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 15.4.14* for more details).

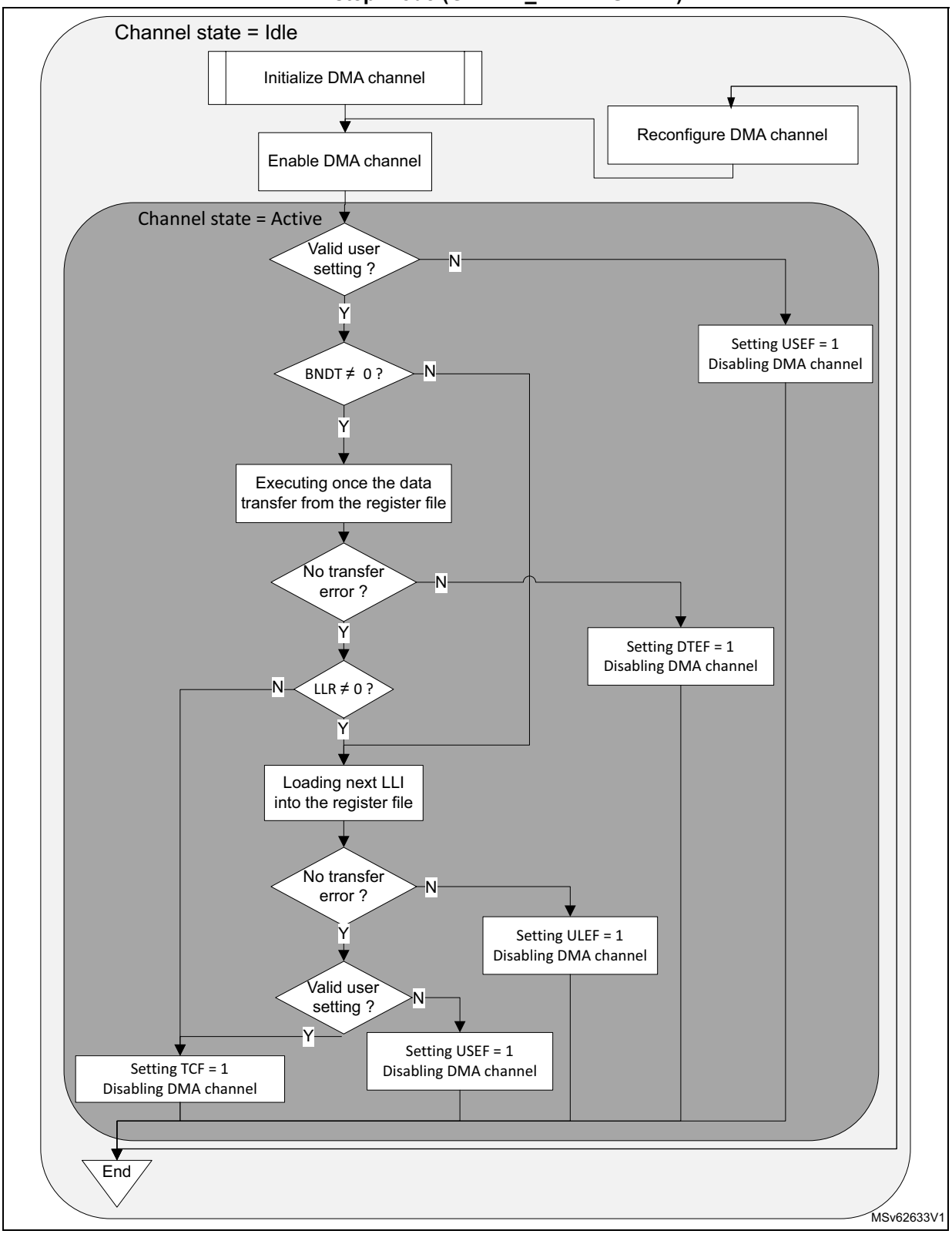
The link step mode can be used to elaborate dynamically LLIs in memory during run-time. The software can be facilitated by using a static data structure for any LLI_n (all update bits of GPDMA_CxLLR have a static value, LLI_n.LLR.LA = LLI_{n-1}.LLR.LA + constant).

Figure 60 depicts the GPDMA channel execution mode, and its programming in link step mode.

Note: Figure 60 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_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.

In Figure 60, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 15.3.6).

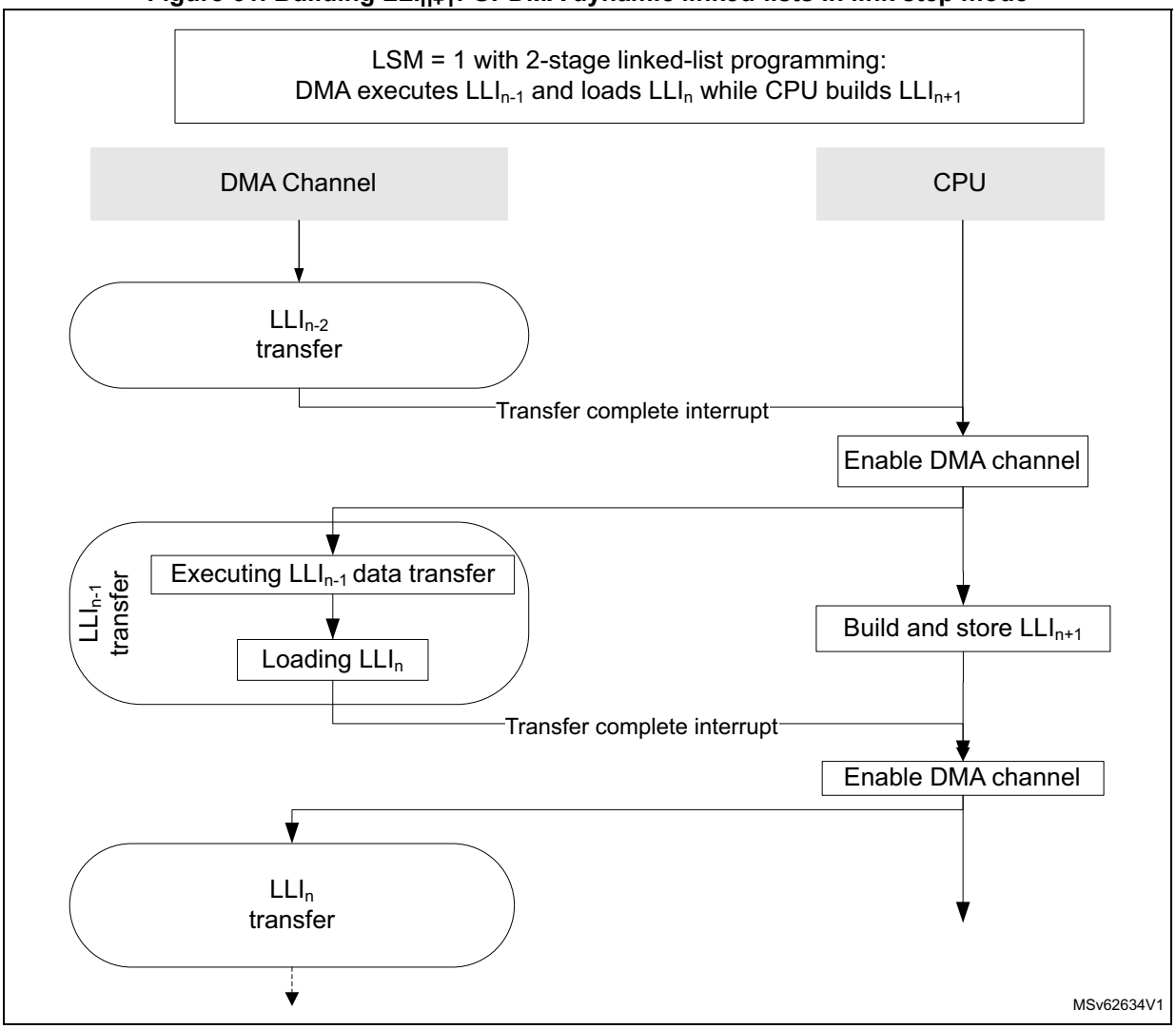
Figure 60. GPDMA channel execution and linked-list programming in link step mode (GPDMA_CxCR.LSM = 1)



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 GPDMA executed the transfer from the LLI_{n-1} and loaded the LLI_n from the memory, as shown in *Figure 61*.

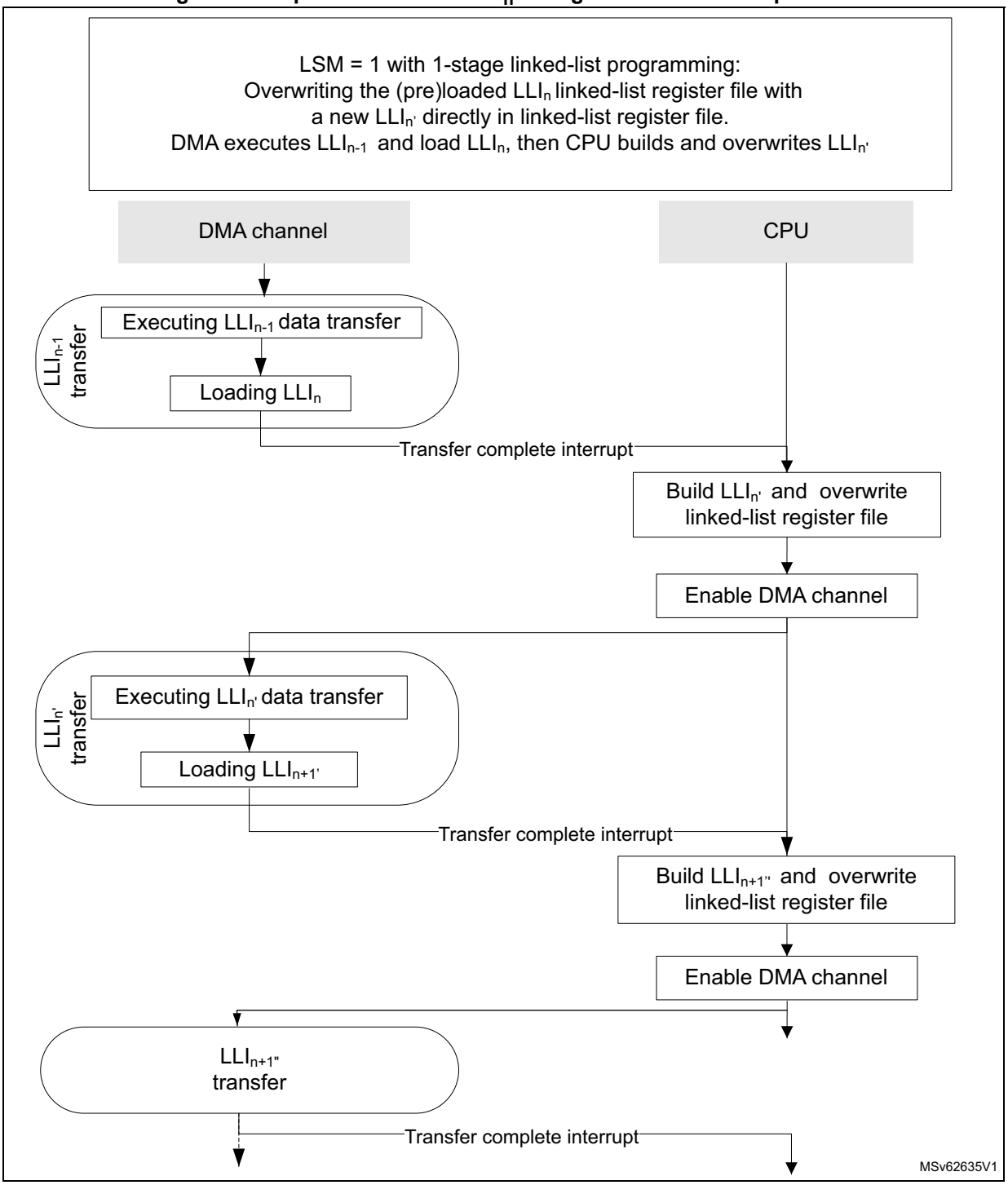
Figure 61. Building LLI_{n+1}: GPDMA dynamic linked-lists in link step mode



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 GPDMA 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 62*.

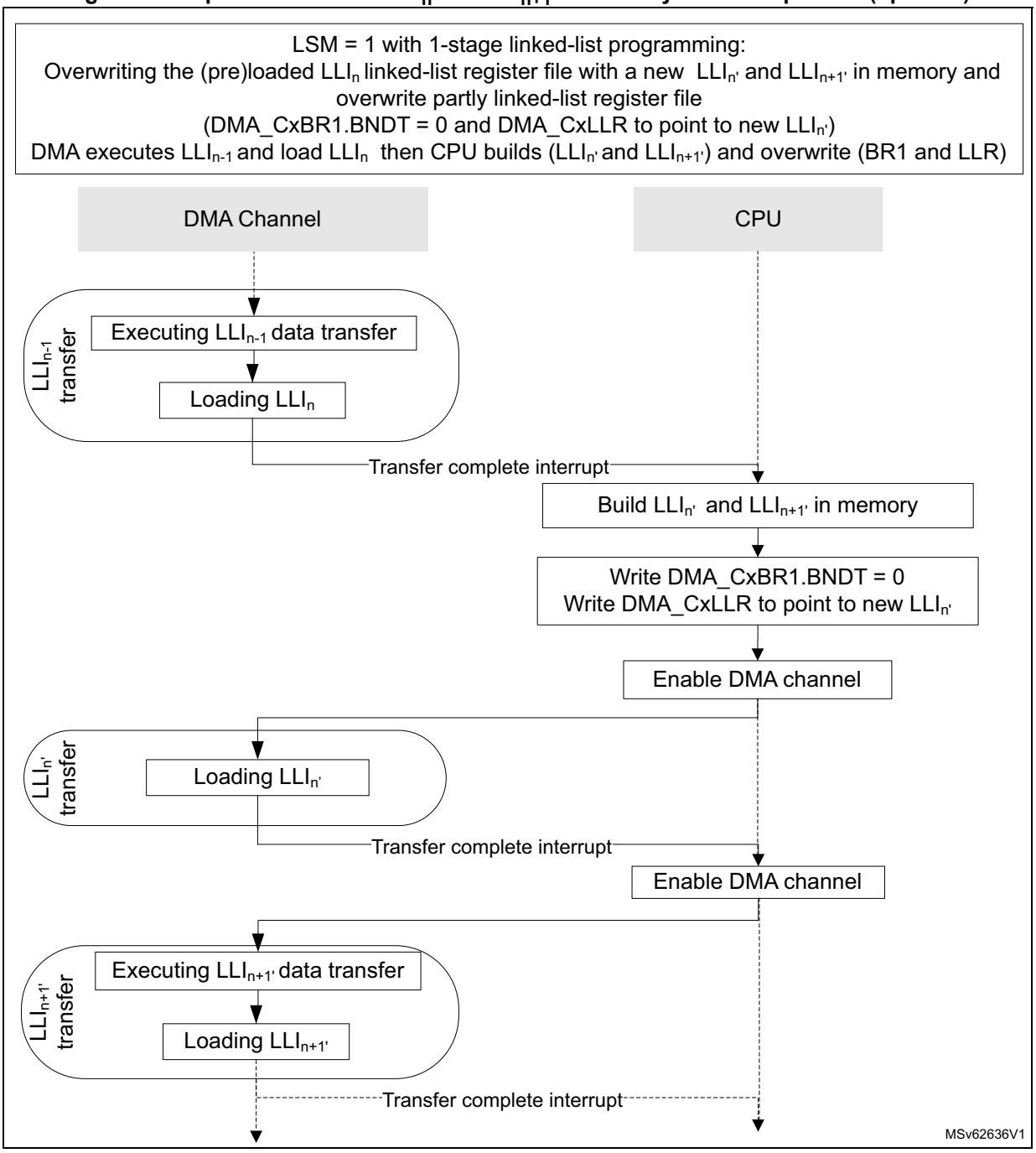
Figure 62. Replace with a new $LLI_{n'}$ in register file in link step mode



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 GPDMA 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 (GPDMA_CxBR1.BNDT[15:0] to be null and GPDMA_CxLLR to point to new $LLI_{n')}$ as shown in *Figure 63*.

Figure 63. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 1)



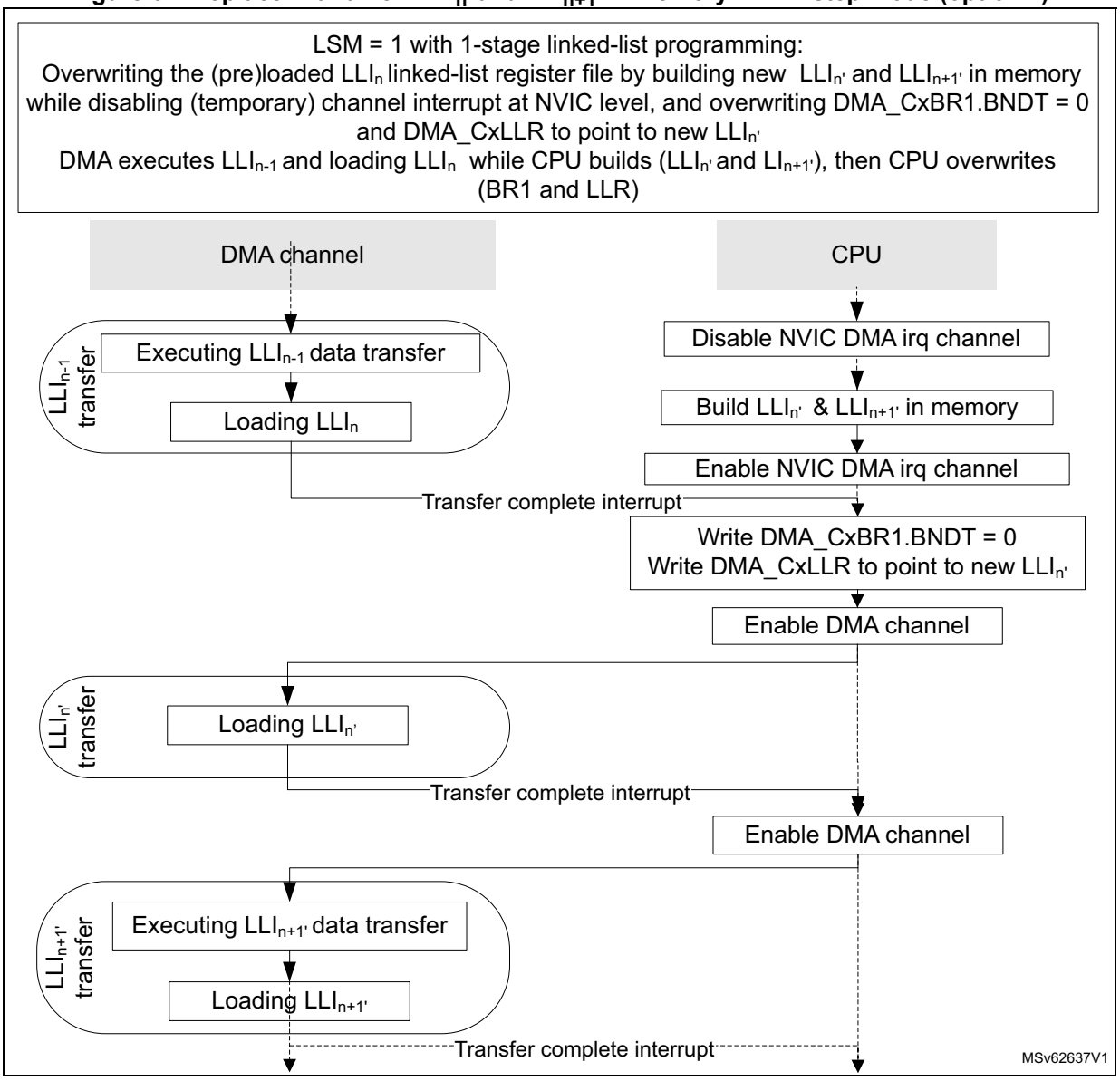
Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode

Other software implementations exist. Meanwhile GPDMA 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 GPDMA_CxBR1.BNDT[15:0] to be null and GPDMA_CxLLR to point to new $LLI_{n'}$, as shown in *Figure 64*.

Figure 64. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 2)



15.4.9 GPDMA channel state and linked-list programming

The software can reconfigure a channel when the channel is disabled (GPDMA_CxCR.EN = 0) and update the execution mode (GPDMA_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 GPDMA_CxLLR of this LLI_{n+1}), before that this LLI_{n+1} is loaded/used by the GPDMA channel
- abort and reconfigure the channel with a LSM update (see *Section 15.4.4*.)

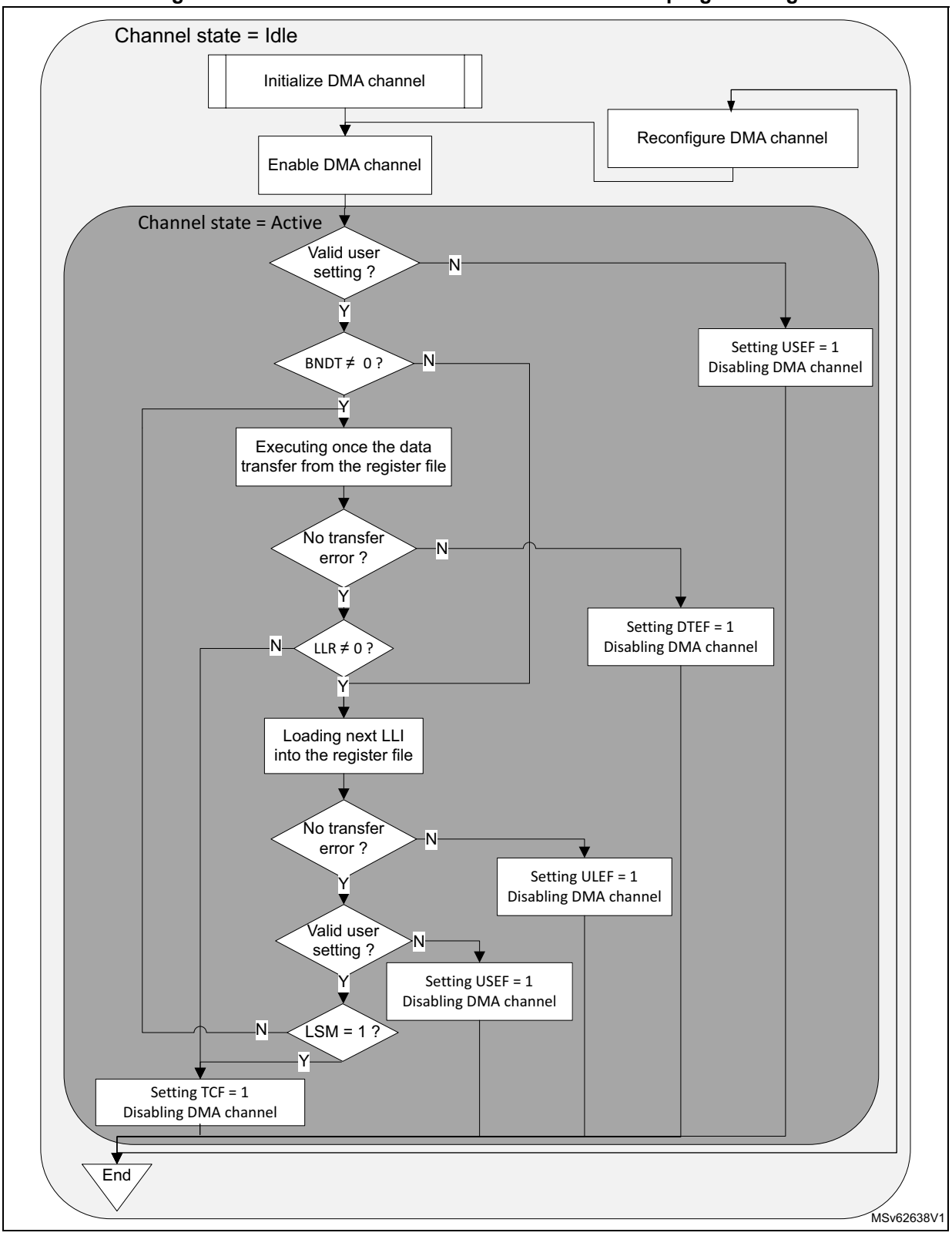
In link step mode, the software can clear LSM after each a single execution of any LLI, during LLI_{n-1}.

Figure 65 shows the overall and unified GPDMA linked-list programming, whatever is the execution mode.

Note: Figure 65 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_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.

In Figure 65, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 15.3.6).

Figure 65. GPDMA channel execution and linked-list programming



15.4.10 GPDMA FIFO-based transfers

There is a single transfer operation mode: the FIFO mode. There are FIFO-based transfers. Any channel x is implemented with a dedicated FIFO whose size is defined by dma_fifo_size[x] (see *Section 15.3.2* for more details).

GPDMA burst

A programmed transfer at the lowest level is a GPDMA burst.

A GPDMA burst is a burst of data received from the source, or a burst of data sent to the destination. A source (and destination) burst is programmed with a burst length by the field SBL_1[5:0] (respectively DBL_1[5:0]), and with a data width defined by the field SDW_LOG2[1:0] (respectively DDW_LOG2[1:0]) in the GPDMA_CxTR1 register.

The addressing mode after each data (named beat) of a GPDMA burst is defined by SINC and DINC in GPDMA_CxTR1, for source and destination respectively: either a fixed addressing or an incremented addressing with contiguous data.

The start and next addresses of a GPDMA source/destination burst (defined by GPDMA_CxSAR and GPDMA_CxDAR) must be aligned with the respective data width.

The table below lists the main characteristics of a GPDMA burst.

Table 92. Programmed GPDMA source/destination burst

SDW_LOG2[1:0] DDW_LOG2[1:0]	Data width (bytes)	SINC/DINC	SBL_1[5:0] DBL_1[5:0]	Burst length (data/beats)	Next data/ beat address	Next burst address	Burst address alignment
00	1	0 (fixed)	n = 0 to 63(1)	n+1	+ 0	+ 0	1
01	2						2
10	4						4
00	1	1 (contiguously incremented)			+ 1	+ (n + 1)	1
01	2				+ 2	+ 2 * (n + 1)	2
10	4				+ 4	+ 4 * (n + 1)	4
11	forbidden user setting, causing USEF generation and none burst to be issued.						

1. When S/DBL_1[5:0] = 0, burst is of length 1. Then burst can be also named as single.

The next burst address in the above table is the next source/destination default address pointed by GPDMA_CxSAR or GPDMA_CxDAR, once the programmed source/destination burst is completed. This default value refers to the fixed/contiguously incremented address.

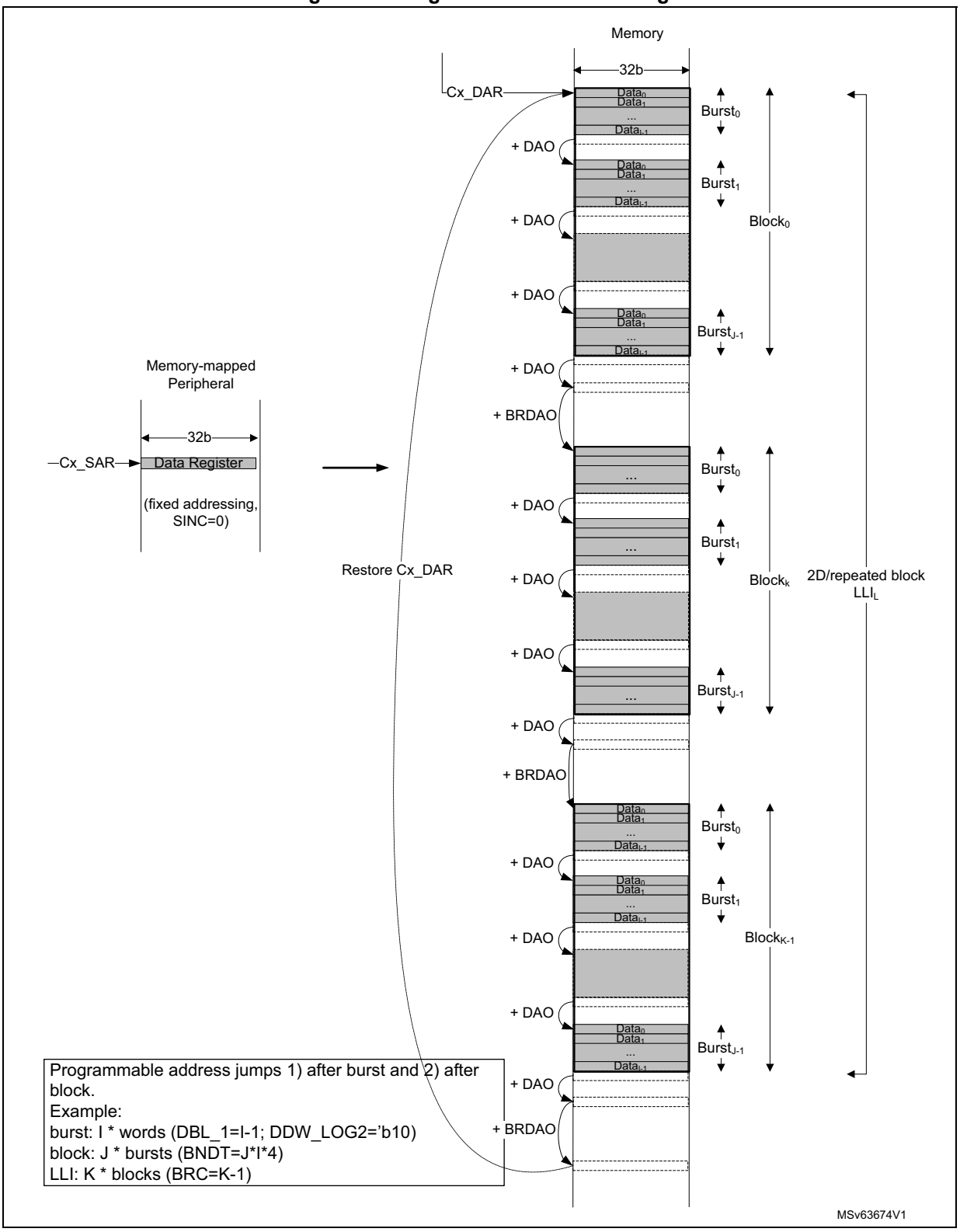
GPDMA burst with 2D addressing (channel x = 6 to 7)

When the channel has additional 2D addressing feature, this default value refers to the value without taking into account the two programmed incremented or decremented offsets. These two additional offsets (with a null default value) are applied:

- after each completed source/destination burst, as defined respectively by GPDMA_CxTR2.SAO[12:0]/DAO[12:0] and GPDMA_CxBR1.SDEC/DDEC
- after each completed block, as defined respectively by GPDMA_CxBR2.BRSAO[15:0]/BRDAO[15:0] and GPDMA_CxBR1.BRSDEC/BRDDEC)

Then, a 2D/repeated block can be addressed with a first programmed address jump after each completed burst, and with a second programmed address jump after each block, as depicted by *Figure 66* with a 2D destination buffer.

Figure 66. Programmed 2D addressing



GPDMA FIFO-based burst

In FIFO-mode, a transfer generally consists of two pipelined and separated burst transfers:

- one burst from the source to the FIFO over the allocated source master port, as defined by GPDMA_CxTR1.SAP
- one burst from the FIFO to the destination over the allocated destination master port, as defined by GPDMA_CxTR1.DAP

GPDMA source burst

The requested source burst transfer to the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to get one new burst from the source):

when FIFO level ≤ $2^{dma\_fifo\_size[x]}$ - (SBL_1[5:0]+1) * $2^{SDW\_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- $2^{dma\_fifo\_size[x]}$ is the half of the FIFO size of the channel x, in bytes (see *Section 15.3.2* for the implementation details and dma_fifo_size[x] value).
- (SBL_1[5:0]+1) * $2^{SDW\_LOG2[1:0]}$ is the size of the programmed source burst transfer, in bytes.

Based on the channel priority (GPDMA_CxCR.PRIO[1:0]), this ready FIFO-based source transfer is internally arbitrated versus the other requested and active channels.

GPDMA destination burst

The requested destination burst transfer from the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to push one new burst to the destination):

when FIFO level ≥ (DBL_1[5:0]+1) * $2^{DDW\_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- (DBL_1[5:0]+1) * $2^{DDW\_LOG2[1:0]}$ is the size of the programmed destination burst transfer, in bytes.

Based on the channel priority, this ready FIFO-based destination transfer is internally arbitrated versus the other requested and active channels.

GPDMA burst vs source block size, 1-Kbyte address boundary and FIFO size

The programmed source/destination GPDMA burst is implemented with an AHB burst as is, unless one of the following conditions is met:

- When half of the FIFO size of the channel x is lower than the programmed source/destination burst size, the programmed source/destination GPDMA burst is implemented with a series of singles or bursts of a lower size, each transfer being of a size that is lower or equal than half of the FIFO size, without any user constraint.
- if the source block size (GPDMA_CxBR1.BNDT[15:0]) is not a multiple of the source burst size but is a multiple of the data width of the source burst (GPDMA_CxTR1.SDW_LOG2[1:0]), the GPDMA modifies and shortens bursts into singles or bursts of lower length, in order to transfer exactly the source block size, without any user constraint.

- if the source/destination burst transfer have crossed the 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol, without any user constraint.
- If the source/destination burst length exceeds 16 on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol, without any user constraint.

In any case, the GPDMA keeps ensuring source/destination data (and address) integrity without any user constraint. The current FIFO level (software readable in GPDMA_CxSR) is compared to and updated with the effective transfer size, and the GPDMA re-arbitrates between each AHB single or burst transfer, possibly modified.

Based on the channel priority, each single or burst of a lower burst size versus the programmed burst, is internally arbitrated versus the other requested and active channels.

Note: In linked-list mode, the GPDMA read transfers related to the update of the linked-list parameters from the memory to the internal GPDMA registers, are scheduled over the link allocated port, as programmed by GPDMA_CxCR.LAP.

GPDMA data handling: byte-based reordering, packing/unpacking, padding/truncation, sign extension and left/right alignment

The data handling is controlled by GPDMA_CxTR1. The source/destination data width of the programmed burst is byte, half-word or word, as per the SDW_LOG2[1:0] and DDW_LOG2[1:0] fields (see *Table 93*).

The user can configure the data handling between transferred data from the source and transfer to the destination. More specifically, programmed data handling is orderly performed with:

1. Byte-based source reordering
   - If SBX = 1 and if source data width is a word, the two bytes of the unaligned half-word at the middle of each source data word are exchanged.
2. Data width conversion by packing, unpacking, padding or truncation, if destination data width is different than the source data width, depending on PAM[1:0]:
   - If destination data width > source data width, the post SBX source data is either right-aligned and padded with 0 s, or sign extended up to the destination data width, or is FIFO queued and packed up to the destination data width.
   - If destination data width < source data width, the post SBX data is either right-aligned and left-truncated down to the destination data width, or is FIFO queued and unpacked and streamed down to the destination data width.
3. Byte-based destination re-ordering:
   - If DBX = 1 and if the destination data width is not a byte, the two bytes are exchanged within the aligned post PAM[1:0] half-words.
   - If DHX = 1 and if the destination data width is neither a byte nor a half-word, the two aligned half-words are exchanged within the aligned post PAM[1:0] words.

Note: Left-alignment with 0s-padding can be achieved by programming both a right-alignment with a 0s-padding and a destination byte-based re-ordering.

The table below lists the possible data handling from the source to the destination.

Table 93. Programmed data handling

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
00	Byte	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$	x	00	Byte	xx	x	x	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, 0P)	0		$0B_3,0B_2,0B_1,0B_0$
							1		B_30,B_20,B_10,B_00
						01 (RA, SE)	0		SB_3,SB_2,SB_1,SB_0
							1		B_3S,B_2S,B_1S,B_0S
						1x (PACK)	0		$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)	0	0	$000B_1,000B_0$
							1		$00B_10,00B_00$
							0	1	$0B_100,0B_000$
							1		B_1000,B_0000
						01 (RA, SE)	0	0	$SSSB_1,SSSB_0$
							1		SSB_1S,SSB_0S
							0	1	SB_1SS,SB_0SS
							1		B_1SSS,B_0SSS
						1x (PACK)	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
01	Half-word	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$		00	Byte	00 (RA, LT)	x	x	B_6,B_4,B_2,B_0
						01 (LA, RT)			B_7,B_5,B_3,B_1
						1x (UNPACK)			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$

Table 93. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
01	Half-word	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$	x	01	Half-word	xx	0	x	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)	0	0	$00B_3B_2,00B_1B_0$
							1		$00B_2B_3,00B_0B_1$
							0	1	B_3B_200,B_1B_000
							1		B_2B_300,B_0B_100
						01 (RA, SE)	0	0	SSB_3B_2,SSB_1B_0
							1		SSB_2B_3,SSB_0B_1
							0	1	B_3B_2SS,B_1B_0SS
							1		B_2B_3SS,B_0B_1SS
						1x (PACK)	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
10	Word	$B_7B_6B_5B_4,B_3B_2B_1B_0$	0	00	Byte	00 (RA, LT)	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)			B_{15},B_{11},B_7,B_3
						10 (UNPACK)			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, LT)	0		B_5B_4,B_1B_0
							1		B_4B_5,B_0B_1
						01 (LA, RT)	0		B_7B_6,B_3B_2
							1		B_6B_7,B_2B_3
						1x (UNPACK)	0		$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$

Table 93. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
10	Word	$B_7B_6B_5B_4$, $B_3B_2B_1B_0$	0	10	Word	xx	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
			1	00	Byte	00 (RA, LT)	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)			B_{15},B_{11},B_7,B_3
						1x (UNPACK)			$B_7,B_5,B_6,B_4,B_3,B_1,B_2,B_0$
				01	Half-word	00 (RA, LT)	0		B_6B_4,B_2B_0
							1		B_4B_6,B_0B_2
						01 (LA, RT)	0		B_7B_5,B_3B_1
							1		B_5B_7,B_1B_3
						1x (UNPACK)	0		$B_7B_5,B_6B_4,B_3B_1,B_2B_0$
							1		$B_5B_7,B_4B_6,B_1B_3,B_0B_2$
				10	Word	xx	0	0	$B_7B_5B_6B_4,B_3B_1B_2B_0$
							1		$B_5B_7B_4B_6,B_1B_3B_0B_2$
							0	1	$B_6B_4B_7B_5,B_2B_0B_3B_1$
							1		$B_4B_6B_5B_7,B_0B_2B_1B_3$

1. Data stream is timely ordered starting from the byte with the lowest index (B_0).
2. RA= right aligned, LA = left aligned, RT = right truncated, LT = left truncated, 0P = zero bit padding up to the destination data width, SE = sign bit extended up to the destination data width.

15.4.11 GPDMA transfer request and arbitration

GPDMA transfer request

As defined by GPDMA_CxTR2, a programmed GPDMA 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 GPDMA hardware peripheral request is driven by the REQSEL[7:0] field (see *Section 15.3.4*). The selected hardware request can be one of the following:
  - an hardware request from a peripheral configured in GPDMA mode (for a transfer from/to the peripheral data register respectively to/from the memory)
  - an hardware request from a peripheral for its control registers update from the memory
  - an 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 GPDMA request via GPDMA_CxTR.REQSEL[7:0] to two different channels, 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.

GPDMA transfer request for arbitration

A ready FIFO-based GPDMA source single/burst transfer (from the source address to the FIFO) to be scheduled over the allocated master port (GPDMA_CxTR1.SAP) is arbitrated based on the channel priority (GPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested GPDMA transfers to the same master port.

A ready FIFO-based GPDMA destination single/burst transfer (from the FIFO to the destination address) to be scheduled over the allocated master port (GPDMA_CxTR1.DAP) is arbitrated based on the channel priority (GPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested GPDMA transfers to the same master port.

An arbitrated GPDMA requested link transfer consists of one 32-bit read from the linked-list data structure in memory to one of the linked-list registers (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR or GPDMA_CxLLR, plus GPDMA_CxTR3, GPDMA_CxBR2). Each 32-bit read from memory is arbitrated with the same channel priority as for data transfers, in order to be scheduled over the allocated master port (GPDMA_CxCR.LAP).

Whatever the requested data transfer is programmed with a software request for a memory-to- memory transfer (GPDMA_CxTR2.SWREQ = 1), or with a hardware request (GPDMA_CxTR2.SWREQ = 0) for a memory-to-peripheral transfer or a peripheral-to-memory transfer and whatever is the hardware request type, re-arbitration occurs after each granted single/burst transfer.

When an hardware request is programmed from a destination peripheral (GPDMA_CxTR2.SWREQ = 0 and GPDMA_CxTR2.DREQ = 1), the first memory read of a (possibly 2D/repeated) block (the first ready FIFO-based source burst request), is gated by the occurrence of the corresponding and selected hardware request. This first read request to memory is not taken into account earlier by the arbiter (not as soon as the block transfer is enabled and executable).

GPDMA arbitration

The GPDMA arbitration is directed from the 4-grade assigned channel priority (GPDMA_CxCR.PRIO[1:0]). The arbitration policy, as illustrated in *Figure 67*, is defined by:

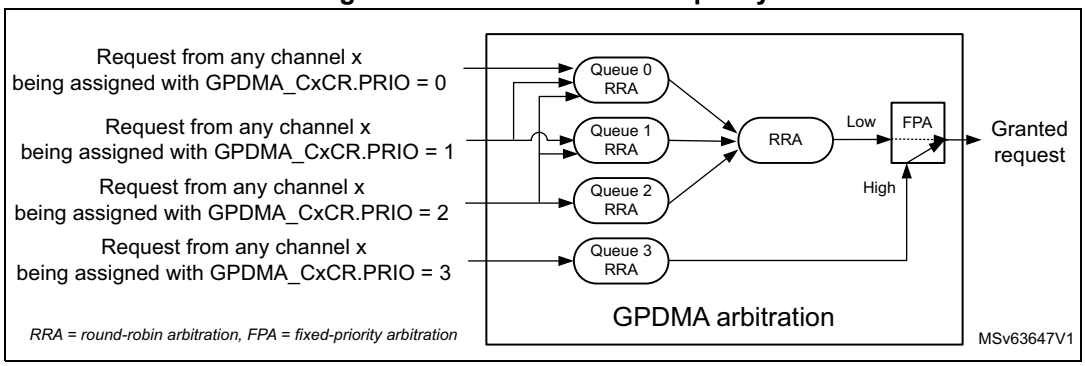
- one high-priority traffic class (queue 3), dedicated to the assigned channels with priority 3, for time-sensitive channels

  This traffic class is granted via a fixed-priority arbitration against any other low-priority traffic class. Within this class, requested single/burst 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/burst 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 67. GPDMA arbitration policy



GPDMA 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 GPDMA 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 GPDMA 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 GPDMA 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 GPDMA 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 GPDMA 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.

15.4.12 GPDMA triggered transfer

A programmed GPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event, as defined by GPDMA_CxTR2.TRIGPOL[1:0] and GPDMA_CxTR2.TRIGSEL[5:0] (see *Section 15.3.7* for the trigger selection).

The triggered transfer, as defined by the trigger mode in GPDMA_CxTR2.TRIGM[1:0], can be at LLI data transfer level, to condition the first burst read of a block, the first burst read of a 2D/repeated block for channel x (x = 6 to 7), or each programmed single read. The trigger mode can also be programmed to condition the LLI link transfer (see TRIGM[1:0] in GPDMA_CxTR2 for more details).

Trigger hit memorization and trigger overrun flag generation

The GPDMA 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).

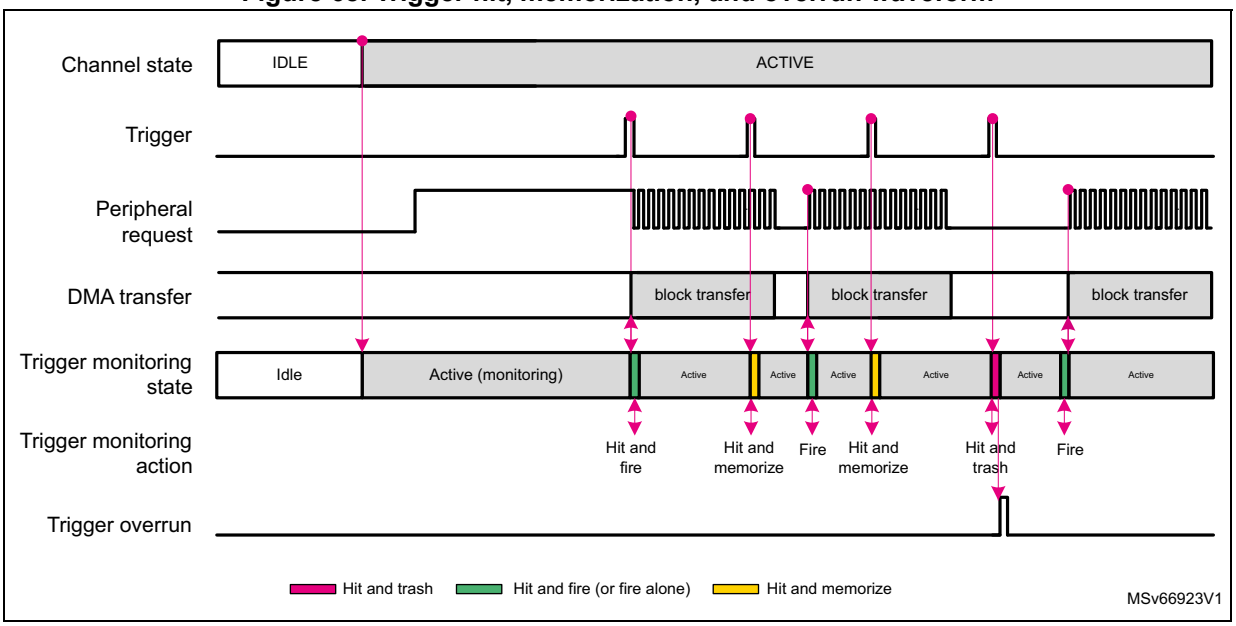
The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer. If a new trigger is detected, this hit is internally memorized to grant the next transfer, as long as the defined rising/falling edge and TRIGSEL[5:0] are not modified, and the channel is enabled.

Transferring a next $LLI_{n+1,}$ that updates the GPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] 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 (GPDMA_CxSR.TOF = 1) and an interrupt is generated if enabled (if GPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

Figure 68 illustrates the trigger hit, memorization and overrun in the configuration example with a block-level trigger mode and a rising edge trigger polarity.

Figure 68. 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.

15.4.13 GPDMA circular buffering with linked-list programming

GPDMA circular buffering for memory-to-peripheral and peripheral-to-memory transfers, with a linear addressing channel

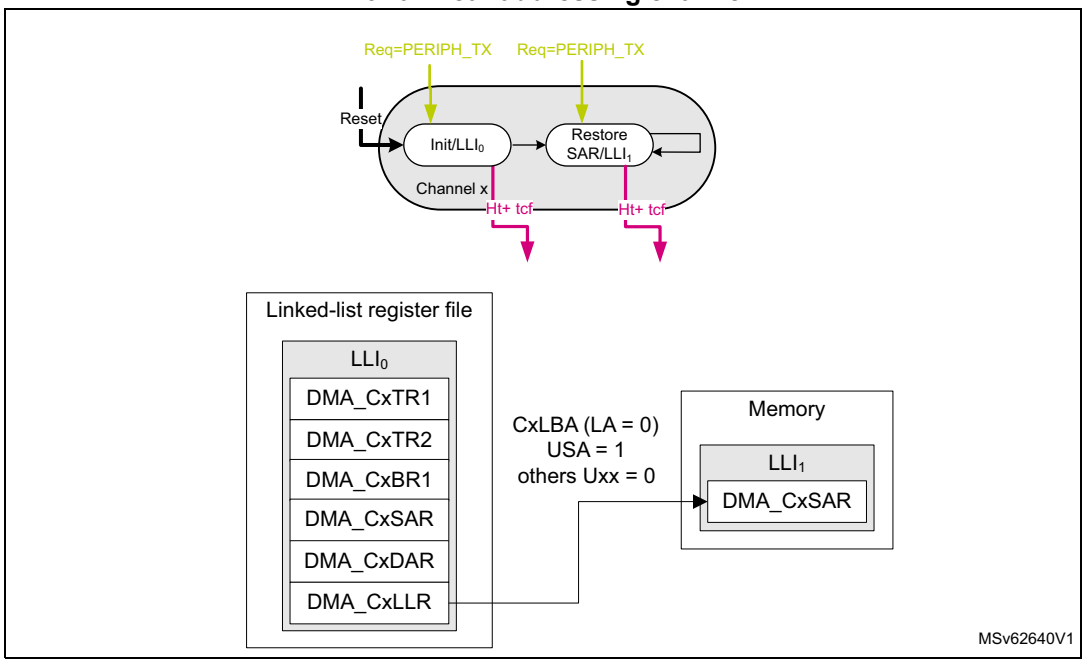
For a circular buffering, with a continuous memory-to-peripheral (or peripheral-to-memory) transfer, the software must set up a channel with half transfer and complete transfer

events/interrupts generation (GPDMA_CxCR.HTIE = 1 and GPDMA_CxCR.TCIE = 1), in order to enable a concurrent buffer software processing.

LLI_0 is configured for the first block transfer with the linear addressing channel. 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). GPDMA automatically reloads the initially programmed GPDMA_CxBR1.BNDT[15:0] when a block transfer is completed, and there is no need to restore GPDMA_CxBR1.

Figure 69 illustrates this programming with a linear addressing GPDMA channel and a source circular buffer.

Figure 69. GPDMA circular buffer programming: update of the memory start address with a linear addressing channel

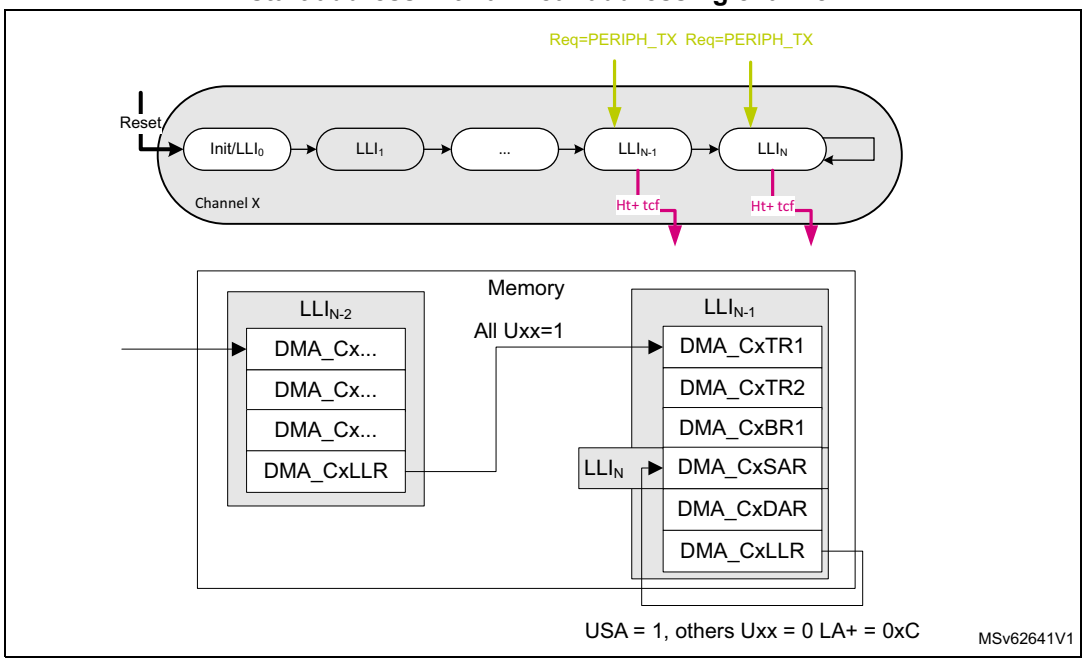


Note: With a 2D addressing channel, the user may use a single LLI with GPDMA_CxBR1.BRC[10:0] = 1, and program a negative memory block address offset with GDMA_CxBR2 and GDMA_CxBR1, in order to jump back to the memory source or the destination start address.

If circular buffering must be executed after some other transfers over the shared GPDMA 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).

Figure 70 illustrates this programming with a linear addressing shared GPDMA channel, and a source circular buffer.

Figure 70. Shared GPDMA channel with circular buffering: update of the memory start address with a linear addressing channel



15.4.14 GPDMA transfer in peripheral flow-control mode

A peripheral with the peripheral flow-control mode feature can decide to early terminate a GPDMA block transfer, provided that the allocated channel is implemented with this feature (see *Section 15.3.6*).

If the related GPDMA channel x is also programmed in peripheral flow-control mode (GPDMA_CxTR2.PFREQ = 1):

- The GPDMA block transfer starts as follows:
  - If GPDMA_CxBR1.BNDT[15:0] ≠ 0, the programmed value is internally taken into account by the GPDMA hardware.
  - If GPDMA_CxBR1.BNDT[15:0] = 0, the GPDMA hardware internally considers a 64-Kbyte value for the maximum source block size to be transferred.
- The GPDMA block transfer is completed as soon as the first occurrence of one of the following condition occurs:
  - when GPDMA_CxBR1.BNDT[15:0] = 0
  - when the peripheral early terminates the block. The complete transfer event is generated if programmed, depending on GPDMA_CxTR2 (see *GPDMA channel x transfer register 2 (GPDMA_CxTR2)*). Then the software can read the current number of transferred bytes from the source (GPDMA_CxBR1.BNDT[15:0]), and/or read the current source or destination address of the buffer in memory (GPDMA_CxSAR[31:0] or GPDMA_CxDAR[31:0]).

In peripheral flow-control mode:

- a destination peripheral with a hardware requested transfer is not supported: memory-to-peripheral transfer is not supported.
- Data packing from a source peripheral is not supported.
- 2D/repeated block is not supported.
- GPDMA_CxBR1.BNDT[15:0] must be programmed as a multiple of the source (peripheral) burst size.

15.4.15 GPDMA privileged/unprivileged channel

Any channel x is a privileged or unprivileged hardware resource, as configured by a privileged agent via GPDMA_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 GPDMA_PRIVCFGR that is 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 GPDMA memory into its register file, is automatically performed with privileged (respectively unprivileged) transfers, via the GPDMA_CxCR.LAP allocated master port.

The GPDMA generates a privileged bus that reflects GPDMA_PRIVCFGR, to keep the other peripherals informed of the privileged/unprivileged state of each GPDMA channel x.

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 GPDMA_CxCR.EN = 0 before the unprivileged software can reprogram the GPDMA_CxCR for a next transfer. The privileged software may reset not only the channel x (GPDMA_CxCR.RESET = 1) but also the full channel x register file to its reset value.

15.4.16 GPDMA error management

The GPDMA 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 or a burst) to the source or the destination:

- The source or destination target reports an AHB error.
- The programmed channel transfer is stopped (GPDMA_CxCR.EN cleared by the GPDMA hardware). The channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and the data error (GPDMA_CxSR.DTEF = 1).

- After a GPDMA 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 GPDMA data transfer error, the user must issue a channel reset (set GPDMA_CxCR.RESET) to reset the hardware GPDMA channel data path and the content of the FIFO, before the user enables again the same channel for a next transfer.

Link transfer error

On a tentative update of a GPDMA channel register from the programmed LLI in the memory:

- The linked-list memory reports an AHB error.
- The programmed channel transfer is stopped (GPDMA_CxCR.EN cleared by the GPDMA hardware), the channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and the link error (GPDMA_CxSR.ULEF = 1).
- After a GPDMA link error, the user must perform a debug session, taking care of the product-defined memory mapping of the linked-list data structure (GPDMA_CxLBAR and GPDMA_CxLLR), including the protection attributes.
- After a GPDMA link error, the user must explicitly write the linked-list register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2), before the user enables again the same channel for a next transfer.

User setting error

On a tentative execution of a GPDMA transfer with an unauthorized user setting:

- The programmed channel transfer is disabled (GPDMA_CxCR.EN forced and cleared by the GPDMA hardware) preventing the next unauthorized programmed data transfer from being executed. The channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and a user setting error (GPDMA_CxSR.USEF = 1).
- After a GPDMA user setting error, the user must perform a debug session, taking care of the GPDMA 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 (GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxLLR.UB1 = 0)
  - a programmed non-null source block size being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxBR1.BNDT[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - when in packing/unpacking mode (if PAM[1] = 1), a programmed non-null source block size being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxBR1.BNDT[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
  - a programmed unaligned source start address, being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxSAR[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - for channel x (x = 6 to 7): a programmed unaligned source address offset being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxTR3.SAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])

- for channel x (x = 6 to 7): a programmed unaligned block repeated source address offset being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxBR2.BRSAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
- a programmed unaligned destination start address, being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxDAR[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
- for channel x (x = 6 to 7): a programmed unaligned destination address offset being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxTR3.DAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
- for channel x (x = 6 to 7): a programmed unaligned block repeated destination address offset being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxBR2.BRDAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
- a programmed double-word source data width (GPDMA_CxTR1.SDW_LOG2[1:0] = 11)
- a programmed double-word destination data width (GPDMA_CxTR1.DDW_LOG2[1:0] = 11)
- a programmed linked-list item LLI_{n+1} with a null data transfer (GPDMA_CxLLR.UB1 = 1 and GPDMA_CxBR1. BNDT = 0)

15.5 GPDMA 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, GPDMA_CxSR.SUSPF is not altered by a suspension from the programmable control bit in the DBGMCU module. In this case, GPDMA_CxSR.IDLEF can be checked to know the completion status of the channel suspension.

15.6 GPDMA in low-power modes

Table 94. Effect of low-power modes on GPDMA

Mode	Description
Sleep	No effect. GPDMA interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the GPDMA registers is kept when entering Stop mode.
Standby	The GPDMA is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 15.3.3* to know if any Stop mode is supported.

15.7 GPDMA interrupts

There is one GPDMA interrupt line for each channel, and separately for each CPU (if several ones in the devices).

Table 95. GPDMA interrupt requests

Interrupt acronym	Interrupt event	Interrupt enable	Event flag	Event clear method
GPDMA_CHx	Transfer complete	GPDMA_CxCR.TCIE	GPDMA_CxSR.TCF	Write 1 to GPDMA_CxFCR.TCF
	Half transfer	GPDMA_CxCR.HTIE	GPDMA_CxSR.HTF	Write 1 to GPDMA_CxFCR.HTF
	Data transfer error	GPDMA_CxCR.DTEIE	GPDMA_CxSR.DTEF	Write 1 to GPDMA_CxFCR.DTEF
	Update link error	GPDMA_CxCR.ULEIE	GPDMA_CxSR.ULEF	Write 1 to GPDMA_CxFCR.ULEF
	User setting error	GPDMA_CxCR.USEIE	GPDMA_CxSR.USEF	Write 1 to GPDMA_CxFCR.USEF
	Suspended	GPDMA_CxCR.SUSPIE	GPDMA_CxSR.SUSPF	Write 1 to GPDMA_CxFCR.SUSPF
	Trigger overrun	GPDMA_CxCR.TOFIE	GPDMA_CxSR.TOF	Write 1 to GPDMA_CxFCR.TOF

A GPDMA 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 gpdma_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 (GPDMA_CxCR.xxIE = 1)
- the corresponding event flag is cleared (GPDMA_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 GPDMA_CxFCR.xxF).

TCF (transfer complete) and HTF (half transfer) events generation is controlled by GPDMA_CxTR2.TCEM[1:0] as follows:

- A transfer complete event is a block transfer complete, a 2D/repeated 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 GPDMA_CxTR2.TCEM[1:0].

- A half transfer event is an half block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode GPDMA_CxTR2.TCEM[1:0].

  A half-block transfer occurs when half of the source block size bytes (rounded-up integer of GPDMA_CxBR1.BNDT[15:0] / 2) is transferred to the destination.

  A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded-up integer of (GPDMA_CxBR1.BRC[10:0] + 1) / 2) is transferred to the destination.

See *GPDMA channel x transfer register 2 (GPDMA_CxTR2)* for more details.

A transfer error rises in one of the following situations:

- during a single/burst data transfer from the source or to the destination (DTEF)
- during an update of a GPDMA channel register from the programmed LLI in memory (ULEF)
- during a tentative execution of a GPDMA channel with an unauthorized setting (USEF)

  The user must perform a debug session to correct the GPDMA channel programming versus the USEF root causes list (see *Section 15.4.16*).

A trigger overrun is described in *Trigger hit memorization and trigger overrun flag generation*.

15.8 GPDMA registers

The GPDMA registers must be accessed with an aligned 32-bit word data access.

15.8.1 GPDMA privileged configuration register (GPDMA_PRIVCFGR)

Address offset: 0x04

Reset value: 0x0000 0000

A write access to this register must be privileged. A read access can be privileged or unprivileged.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN =1.

This register must be programmed at a bit level, at the initialization/closure of a GPDMA channel (GPDMA_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

15.8.2 GPDMA masked interrupt status register (GPDMA_MISR)

Address offset: 0x0C

Reset value: 0x0000 0000

This is a read register.

This register contains the masked interrupt status bit MISx for each channel x. It is a logical OR of all the GPDMA_CxSR flags, each source flag being enabled by the corresponding GPDMA_CxCR interrupt enable bit.

Every bit is deasserted by hardware when writing 1 to the corresponding GPDMA_CxFCR flag clear bit.

This register can mix privileged and unprivileged information, depending on the privileged state of each channel GPDMA_PRIVCFGR.PRIVx. A privileged software can read the full interrupt status. An unprivileged software is restricted to read the status of unprivileged 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 channel x (x = 7 to 0)
0: no interrupt occurred on channel x
1: an interrupt occurred on channel x

15.8.3 GPDMA channel x linked-list base address register (GPDMA_CxLBAR)

Address offset: 0x50 + 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 GPDMA_PRIVCFGR.PRIVx.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_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 GPDMA transfers, are conditionally and automatically updated.

This 64-Kbyte aligned channel x linked-list base address is offset by the 16-bit GPDMA_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 GPDMA channel x

Bits 15:0 Reserved, must be kept at reset value.

15.8.4 GPDMA channel x flag clear register (GPDMA_CxFCR)

Address offset: 0x5C+ 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This is a write register privileged or unprivileged, depending on the privileged state of the channel x (GPDMA_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: corresponding TOF flag cleared

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.

15.8.5 GPDMA channel x status register (GPDMA_CxSR)

Address offset: 0x60 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0001

This is a read register, reporting the channel status.

This register is privileged or non-privileged, depending on the privileged state of the channel (GPDMA_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.	FIFOL[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.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLEF
	r	r	r	r	r	r	r								r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **FIFOL[7:0]**: monitored FIFO level
Number of available write beats in the FIFO, in units of the programmed destination data width (see GPDMA_CxTR1.DDW_LOG2[1:0], in units of bytes, half-words, or words).

Note: After having suspended an active transfer, the user may need to read FIFOL[7:0], additionally to GPDMA_CxBR1.BDNT[15:0] and GPDMA_CxBR1.BRC[10:0], to know how many data have been transferred to the destination. Before reading, the user may wait for the transfer to be suspended (GPDMA_CxSR.SUSPF = 1).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOF**: trigger overrun flag
0: no trigger overrun event
1: a trigger overrun event occurred

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 either a half block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode (GPDMA_CxTR2.TCEM[1:0]).

A half block transfer occurs when half of the bytes of the source block size (rounded up integer of GPDMA_CxBR1.BNDT[15:0]/2) has been transferred to the destination.

A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded up integer of (GPDMA_CxBR1.BRC[10:0] + 1) / 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 either a block transfer complete, a 2D/repeated 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 (GPDMA_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 deasserted by hardware when the channel is enabled (GPDMA_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).

15.8.6 GPDMA channel x control register (GPDMA_CxCR)

Address offset: 0x64 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of the channel x (GPDMA_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.	LAP	LSM
								rw	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 GPDMA 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 21:18 Reserved, must be kept at reset value.

Bit 17 **LAP**: linked-list allocated port
This bit is used to allocate the master port for the update of the GPDMA linked-list registers from the memory.
0: port 0 (AHB) allocated
1: port 1 (AHB) allocated
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 16 **LSM**: Link step mode
0: channel executed for the full linked-list and completed at the end of the last LLI (GPDMA_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 and UT3 = UB2 = 0). Then GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0.
1: channel executed **once** for the current LLI
First the (possible 1D/repeated) block transfer is executed as defined by the current internal register file until GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0. Secondly the next linked-list data structure is conditionally uploaded from memory as defined by GPDMA_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 (channel with an ongoing GPDMA transfer over its master ports).
The software must write 0 in order to resume a suspended channel, following the programming sequence detailed in *Figure 52*.
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 FIFO, 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 (GPDMA_CxSR.SUSPF = 1 and GPDMA_CxSR.IDLEF = GPDMA_CxCR.EN = 1)
- channel in disabled state (GPDMA_CxSR.IDLEF = 1 and GPDMA_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 (GPDMA_CxBR1, GPDMA_CxSAR, and GPDMA_CxDAR) before enabling again the channel (see the programming sequence in *Figure 53*).
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 deasserted 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

15.8.7 GPDMA channel x transfer register 1 (GPDMA_CxTR1)

Address offset: 0x90 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is privileged or non-privileged, depending on the privileged state of the channel x in GPDMA_PRIVCFGR.PRIVx.

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed. Then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory if GPDMA_CxLLR.UT1 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DAP	Res.	Res.	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2[1:0]	
	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.	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2[1:0]	
	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **DAP**: destination allocated port

This bit is used to allocate the master port for the destination transfer

0: port 0 (AHB) allocated

1: port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **DHX**: destination half-word exchange

If the destination data size is shorter than a word, this bit is ignored.

If the destination data size is a word:

0: no halfword-based exchanged within word

1: the two consecutive (post PAM) half-words are exchanged in each destination word.

Bit 26 **DBX**: destination byte exchange

If the destination data size is a byte, this bit is ignored.

If the destination data size is not a byte:

0:no byte-based exchange within half-word

1: the two consecutive (post PAM) bytes are exchanged in each destination half-word.

Bits 25:20 **DBL_1[5:0]**: destination burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If DBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width DDW_LOG2[1:0].

Note: If a burst transfer crossed a 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with GPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 19 **DINC**: destination incrementing burst

0: fixed burst

1: contiguously incremented burst

The destination address, pointed by GPDMA_CxDAR, is kept constant after a burst beat/single transfer, or is incremented by the offset value corresponding to a contiguous data after a burst beat/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 burst, in bytes
00: byte
01: half-word (2 bytes)
10: word (4 bytes)
11: user setting error reported and no transfer issued

Note: A destination burst transfer must have an aligned address with its data width (start address GPDMA_CxDAR[2:0] and if present address offset GPDMA_CxTR3.DAO[2:0], versus DDW_LOG2[1:0]). Otherwise a user setting error is reported and no transfer is issued.

When configured in packing mode (PAM[1] = 1 and destination data width different from source data width), a source block size must be a multiple of the destination data width (see GPDMA_CxBR1.BNDT[2:0] versus DDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued.

Setting a 8-byte data width causes a user setting error to be reported and none transfer is issued.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **SAP**: source allocated port
This bit is used to allocate the master port for the source transfer
0: port 0 (AHB) allocated
1: port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 13 **SBX**: source byte exchange within the unaligned half-word of each source word
If the source data width is shorter than a word, this bit is ignored.
If the source data width is a word:
0:no byte-based exchange within the unaligned half-word of each source word
1: the two consecutive bytes within the unaligned half-word of each source word are exchanged.

Bits 12:11 **PAM[1:0]**: padding/alignment mode
If DDW_LOG2[1:0] = SDW_LOG2[1:0]: if the data width of a burst destination transfer is equal to the data width of a burst source transfer, these bits are ignored.
Else, in the following enumerated values, the condition PAM_1 is when destination data width is higher than source data width, and the condition PAM_2 is when source data width is higher than destination data width.
Condition: PAM_1
00: source data is transferred as right aligned, padded with 0s up to the destination data width
01: source data is transferred as right aligned, sign extended up to the destination data width
10-11: successive source data are FIFO queued and packed at the destination data width, in a left (LSB) to right (MSB) order (named little endian), before a destination transfer
Condition: PAM_2
00: source data is transferred as right aligned, left-truncated down to the destination data width
01: source data is transferred as left-aligned, right-truncated down to the destination data width
10-11: source data is FIFO queued and unpacked at the destination data width, to be transferred in a left (LSB) to right (MSB) order (named little endian) to the destination

Note: If the transfer from the source peripheral is configured with peripheral flow-control mode (SWREQ = 0 and PFREQ = 1 and DREQ = 0), and if the destination data width > the source data width, packing is not supported.

Bit 10 Reserved, must be kept at reset value.

Bits 9:4 **SBL_1[5:0]**: source burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If SBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width SDW_LOG2[1:0].

Note: If a burst transfer crossed a 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with GPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 3 **SINC**: source incrementing burst

0: fixed burst

1: contiguously incremented burst

The source address, pointed by GPDMA_CxSAR, is kept constant after a burst beat/single transfer or is incremented by the offset value corresponding to a contiguous data after a burst beat/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 burst in bytes

00: byte

01: half-word (2 bytes)

10: word (4 bytes)

11: user setting error reported and no transfer issued

Note: A source block size must be a multiple of the source data width (GPDMA_CxBR1.BNDT[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and no transfer is issued.

A source burst transfer must have an aligned address with its data width (start address GPDMA_CxSAR[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and none transfer is issued.

Setting a 8-byte data width causes a user setting error to be reported and no transfer is issued.

15.8.8 GPDMA channel x transfer register 2 (GPDMA_CxTR2)

Address offset: 0x94 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (the hardware deasserted GPDMA_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 GPDMA from the memory, if GPDMA_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.	TRIGSEL[5:0]					
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
TRIGM[1:0]		Res.	PFREQ	BREQ	DREQ	SWREQ	Res.	REQSEL[7:0]							
rw	rw		rw	rw	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 GPDMA_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 GPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half transfer event is generated.

01: channel x (x = 0 to 5), same as 00, channel x (x = 6 to 7), at 2D/repeated block level (when GPDMA_CxBR1.BRC[10:0] = 0 and GPDMA_CxBR1.BNDT[15:0] = 0). The complete (and the half) transfer event is generated at the end (respectively half of the end) of the 2D/repeated block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half transfer event is generated.

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 is a block transfer or a 2D/repeated block transfer for channel x (x = 6 to 7), if any data transfer.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_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 GPDMA_CxLLR.LA[15:2] to zero and clears all the GPDMA_CxLLR update bits (UT1, UT2, UB1, USA, UDA and ULL, plus UT3 and UB2). 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[5:0].

00: no trigger (masked trigger event)

01: trigger on the rising edge

10: trigger on the falling edge

11: same as 00

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **TRIGSEL[5:0]**: trigger event input selection

These bits select the trigger event input of the GPDMA transfer (as per *Section 15.3.7*), 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 (GPDMA_CxCR.EN asserted) with TRIGPOL[1:0] = 00 or 11, these TRIGM[1:0] bits are ignored.

Else, a GPDMA transfer is conditioned by at least one trigger hit:

00: at block level: the first burst read of each block transfer is conditioned by one hit trigger (channel x (x = 6 to 7), for each block if a 2D/repeated block is configured with GPDMA_CxBR1.BRC[10:0] ≠ 0).

01: channel x (x = 0 to 5), same as 00; channel x (x = 6 to 7), at 2D/repeated block level. The first burst read of a 2D/repeated block transfer is conditioned by one hit trigger.

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 burst level: If SWREQ = 1, each programmed burst read is conditioned by one hit trigger. If SWREQ = 0, each programmed burst that is requested by the selected peripheral, is conditioned by one hit trigger.

– If the peripheral is programmed as a source (DREQ = 0) of the LLI data transfer, each programmed burst read is conditioned.

– If the peripheral is programmed as a destination (DREQ = 1) of the LLI data transfer, each programmed burst write is conditioned. The first memory burst read of a (possibly 2D/repeated) block, also named as the first ready FIFO-based source burst, is gated by the occurrence of both the hardware request and the first trigger hit.

The GPDMA 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] = 01 or respectively TRIGPOL[1:0] = 10).

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[5:0] is not modified, and the channel is enabled.

Transferring a next LLI_{n+1} that updates the GPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] 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. A trigger overrun flag is reported (GPDMA_CxSR.TOF = 1), and an interrupt is generated if enabled (GPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

Note: When the source block size is not a multiple of the source burst size and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, if TRIGM[1:0] = 11 and (SWREQ = 1 or (SWREQ = 0 and DREQ = 0)), the shortened burst transfer (by singles or/and by bursts of lower length) is conditioned once by the trigger.

When the programmed destination burst is internally shortened by singles or/and by bursts of lower length (versus FIFO size, versus block size, 1-Kbyte boundary address crossing): if the trigger is conditioning the programmed destination burst (if TRIGM[1:0] = 11 and SWREQ = 0 and DREQ = 1), this shortened destination burst transfer is conditioned once by the trigger.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PFREQ**: Hardware request in peripheral flow control mode

Important: If a given channel x is not implemented with this feature, this bit is reserved and PFREQ is not present (see Section 15.3.2 for the list of the implemented channels with this feature).

If the channel x is activated (GPDMA_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 in GPDMA control mode. The GPDMA is programmed with GPDMA_CxTR1.BNDT[15:0] and this is internally used by the hardware for the block transfer completion.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in peripheral control mode. The GPDMA block transfer can be early completed by the peripheral itself (see *Section 15.3.6* for more details).

Note: In peripheral flow control mode, there are the following restrictions:

- no 2D/repeated block support (GPDMA_CxBR1.BRC[10:0] must be set to 0)

- the peripheral must be set as the source of the transfer (DREQ = 0).

- data packing to a wider destination width is not supported (if destination width > source data width, GPDMA_CxTR1.PAM[1] must be set to 0).

- GPDMA_CxBR1.BNDT[15:0] must be programmed as a multiple of the source (peripheral) burst size.

Bit 11 **BREQ**: Block hardware request

If the channel x is activated (GPDMA_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 burst level.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a block level (see *Section 15.3.4*).

Bit 10 **DREQ**: destination hardware request

This bit is ignored if channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else:

0: selected hardware request driven by a source peripheral (request signal taken into account by the GPDMA transfer scheduler over the source/read port)

1: selected hardware request driven by a destination peripheral (request signal taken into account by the GPDMA transfer scheduler over the destination/write port)

Note: If the channel x is activated (GPDMA_CxCR.EN is asserted) with SWREQ = 0 and PFREQ = 1 (peripheral hardware request with peripheral flow-control mode), any software assertion to this DREQ bit is ignored: in peripheral flow-control mode, only a peripheral-to-memory transfer is supported.

Bit 9 **SWREQ**: software request

This bit is internally taken into account when GPDMA_CxCR.EN is asserted.

0: no software request. The selected hardware request REQSEL[7:0] is taken into account.

1: software request for a memory-to-memory transfer. The default selected hardware request as per REQSEL[7:0] is ignored.

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **REQSEL[7:0]**: GPDMA hardware request selection

These bits are ignored if channel x is activated (GPDMA_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 15.3.4*.

Caution: The user must not assign a same input hardware request (same REQSEL[7:0] value) to different active GPDMA channels (GPDMA_CxCR.EN = 1 and GPDMA_CxTR2.SWREQ = 0 for these channels). GPDMA 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.

15.8.9 GPDMA channel x block register 1 (GPDMA_CxBR1)

Address offset: 0x98 + 0x80 * x (x = 0 to 5)

Reset value: 0x0000 0000

This register is privileged or non-privileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has deasserted GPDMA_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 GPDMA_CxLLR.UB1 = 1, this register is automatically updated by the GPDMA from the next LLI in memory.
- If GPDMA_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 value for the field BNDT[15:0].
- If all the update bits GPDMA_CxLLR.Uxx are null and if GPDMA_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 GPDMA_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 GPDMA_CxLLR.UB1 = 1, this field is updated by the LLI in the memory.
- if GPDMA_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 GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] = 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
- if GPDMA_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 GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

When configured in packing mode (GPDMA_CxTR1.PAM[1] = 1 and destination data width different from source data width), a non-null source block size must be a multiple of the destination data width (BNDT[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

15.8.10 GPDMA channel x alternate block register 1 (GPDMA_CxBR1)

Address offset: 0x98 + 0x80 * x (x = 6 to 7)

Reset value: 0x0000 0000

This register is privileged or non-privileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has deasserted GPDMA_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 GPDMA_CxLLR.UB1 = 1, this register is automatically updated by the GPDMA from the next LLI in memory.
- If GPDMA_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 value for the fields BNDT[15:0] and BRC[10:0].
- If all the update bits GPDMA_CxLLR.Uxx are null and if GPDMA_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 the fields BNDT[15:0] and BRC[10:0] after each execution of this final LLI

- If GPDMA_CxLLR = 0, BNDT[15:0] and BRC[10:0] are kept as null, channel x is completed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BRDDEC	BRSDEC	DDEC	SDEC	Res.	BRC[10:0]										
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
BNDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BRDDEC**: Block repeat destination address decrement

0: at the end of a block transfer, the GPDMA_CxDAR register is updated by adding the programmed offset GPDMA_CxBR2.BRDAO to the current GPDMA_CxDAR value (current destination address)

1: at the end of a block transfer, the GPDMA_CxDAR register is updated by subtracting the programmed offset GPDMA_CxBR2.BRDAO from the current GPDMA_CxDAR value (current destination address)

Note: On top of this increment/decrement (depending on BRDDEC), GPDMA_CxDAR is in the same time also updated by the increment/decrement (depending on DDEC) of the GPDMA_CxTR3.DAO value, as it is usually done at the end of each programmed burst transfer.

Bit 30 **BRSDEC**: Block repeat source address decrement

0: at the end of a block transfer, the GPDMA_CxSAR register is updated by adding the programmed offset GPDMA_CxBR2.BRSAO to the current GPDMA_CxSAR value (current source address)

1: at the end of a block transfer, the GPDMA_CxSAR register is updated by subtracting the programmed offset GPDMA_CxBR2.BRSAO from the current GPDMA_CxSAR value (current source address)

Note: On top of this increment/decrement (depending on BRSDEC), GPDMA_CxSAR is in the same time also updated by the increment/decrement (depending on SDEC) of the GPDMA_CxTR3.SAO value, as it is done after any programmed burst transfer.

Bit 29 **DDEC**: destination address decrement

0: At the end of a programmed burst transfer to the destination, the GPDMA_CxDAR register is updated by adding the programmed offset GPDMA_CxTR3.DAO to the current GPDMA_CxDAR value (current destination address)

1: At the end of a programmed burst transfer to the destination, the GPDMA_CxDAR register is updated by subtracting the programmed offset GPDMA_CxTR3.DAO to the current GPDMA_CxDAR value (current destination address)

Bit 28 **SDEC**: source address decrement

0: At the end of a programmed burst transfer from the source, the GPDMA_CxSAR register is updated by adding the programmed offset GPDMA_CxTR3.SAO to the current GPDMA_CxSAR value (current source address)

1: At the end of a programmed burst transfer from the source, the GPDMA_CxSAR register is updated by subtracting the programmed offset GPDMA_CxTR3.SAO to the current GPDMA_CxSAR value (current source address)

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 **BRC[10:0]**: Block repeat counter

This field contains the number of repetitions of the current block (0 to 2047).

When the channel is enabled, this field becomes read-only. After decrements, this field indicates the remaining number of blocks, excluding the current one. This counter is hardware decremented for each completed block transfer.

Once the last block transfer is completed (BRC[10:0] = BNDT[15:0] = 0):

– If GPDMA_CxLLR.UB1 = 1, all GPDMA_CxBR1 fields are updated by the next LLI in the memory.

– If GPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this field is internally restored to the programmed value.

– if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] ≠ 0, this field is internally restored to the programmed value (infinite/continuous last LLI).

– if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI and data transfer.

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 GPDMA_CxLLR.UB1 = 1, this field is updated by the LLI in the memory.

- if GPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this field is internally restored to the programmed value.

- if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] ≠ 0, this field is internally restored to the programmed value (infinite/continuous last LLI).

- if GPDMA_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 GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

When configured in packing mode (GPDMA_CxTR1.PAM[1] = 1 and destination data width different from source data width), a non-null source block size must be a multiple of the destination data width (BNDT[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

15.8.11 GPDMA channel x source address register (GPDMA_CxSAR)

Address offset: 0x9C + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the source start address of a transfer.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer from the source.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_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 (GPDMA_CxTR1.SINC), this field is kept fixed or incremented by the data width (GPDMA_CxTR1.SDW_LOG2[1:0]) after each burst source data, reflecting the next address from which data is read.

During the channel activity, this address is updated after each completed source burst, consequently to:

– the programmed source burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (GPDMA_CxTR1.SINC = 1), then the additional address offset value is the programmed burst size, as defined by GPDMA_CxTR1.SBL_1[5:0] and GPDMA_CxTR1.SDW_LOG2[1:0]

– the additional source incremented/decremented offset value as programmed by GPDMA_CxBR1.SDEC and GPDMA_CxTR3.SAO[12:0].

– once/if completed source block transfer, for a channel x with 2D addressing capability (x = 6 to 7). additional block repeat source incremented/decremented offset value as programmed by GPDMA_CxBR1.BRSDEC and GPDMA_CxBR2.BRSAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by GPDMA from the memory, provided the LLI is set with GPDMA_CxLLR.USA = 1.

Note: A source address must be aligned with the programmed data width of a source burst (SA[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

When the source block size is not a multiple of the source burst size and is a multiple of the source data width, the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional GPDMA_CxTR3.SAO[12:0] is not applied.

15.8.12 GPDMA channel x destination address register (GPDMA_CxDAR)

Address offset: 0xA0 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the destination start address of a transfer.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer to the destination.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory if GPDMA_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 (GPDMA_CxTR1.DINC), this field is kept fixed or incremented by the data width (GPDMA_CxTR1.DDW_LOG2[1:0]) after each burst destination data, reflecting the next address from which data is written.

During the channel activity, this address is updated after each completed destination burst, consequently to:

– the programmed destination burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (GPDMA_CxTR1.DINC = 1), then the additional address offset value is the programmed burst size, as defined by GPDMA_CxTR1.DBL_1[5:0] and GPDMA_CxTR1.DDW_LOG2[1:0]

– the additional destination incremented/decremented offset value as programmed by GPDMA_CxBR1.DDEC and GPDMA_CxTR3.DAO[12:0].

– once/if completed destination block transfer, for a channel x with 2D addressing capability (x = 6 to 7), the additional block repeat destination incremented/decremented offset value as programmed by GPDMA_CxBR1.BRDDEC and GPDMA_CxBR2.BRDAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by the GPDMA from the memory, provided the LLI is set with GPDMA_CxLLR.UDA = 1.

Note: A destination address must be aligned with the programmed data width of a destination burst (DA[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

15.8.13 GPDMA channel x transfer register 3 (GPDMA_CxTR3)

Address offset: 0xA4 + 0x80 * x (x = 6 to 7)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_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 the GPDMA from the memory if GPDMA_CxLLR.UT3 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DAO[12:0]												
			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.	Res.	SAO[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **DAO[12:0]**: destination address offset increment

The destination address, pointed by GPDMA_CxDAR, is incremented or decremented (depending on GPDMA_CxBR1.DDEC) by this offset DAO[12:0] for each programmed destination burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (GPDMA_CxTR1.DINC = 1).

Note: A destination address offset must be aligned with the programmed data width of a destination burst (DAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **SAO[12:0]**: source address offset increment

The source address, pointed by GPDMA_CxSAR, is incremented or decremented (depending on GPDMA_CxBR1.SDEC) by this offset SAO[12:0] for each programmed source burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (GPDMA_CxTR1.SINC = 1).

Note: A source address offset must be aligned with the programmed data width of a source burst (SAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued.

When the source block size is not a multiple of the destination burst size, and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional GPDMA_CxTR3.SAO[12:0] is not applied.

15.8.14 GPDMA channel x block register 2 (GPDMA_CxBR2)

Address offset: 0xA8 + 0x80 * x (x = 6 to 7)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a 2D/repeated block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.UB2 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BRDAO[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
BRSAO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **BRDAO[15:0]**: Block repeated destination address offset

For a channel with 2D addressing capability, this field is used to update (by addition or subtraction depending on GPDMA_CxBR1.BRDDEC) the current destination address (GPDMA_CxDAR) at the end of a block transfer.

Note: A block repeated destination address offset must be aligned with the programmed data width of a destination burst (BRDAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued. BRDAO[15:0] must be set to 0 in peripheral flow-control mode (if GPDMA_CxTR2.PFREQ = 1).

Bits 15:0 **BRSAO[15:0]**: Block repeated source address offset

For a channel with 2D addressing capability, this field is used to update (by addition or subtraction depending on GPDMA_CxBR1.BRSDEC) the current source address (GPDMA_CxSAR) at the end of a block transfer.

Note: A block repeated source address offset must be aligned with the programmed data width of a source burst (BRSAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued. BRSAO[15:0] must be set to 0 in peripheral flow-control mode (if GPDMA_CxTR2.PFREQ = 1).

15.8.15 GPDMA channel x linked-list address register (GPDMA_CxLLR)

Address offset: 0xCC + 0x80 * x (x = 0 to 5)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_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 GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_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 the GPDMA from the memory if GPDMA_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 GPDMA_CxTR1 from memory

This bit controls the update of GPDMA_CxTR1 from the memory during the link transfer.
0: no GPDMA_CxTR1 update
1: GPDMA_CxTR1 update

Bit 30 **UT2**: Update GPDMA_CxTR2 from memory

This bit controls the update of GPDMA_CxTR2 from the memory during the link transfer.
0: no GPDMA_CxTR2 update
1: GPDMA_CxTR2 update

Bit 29 **UB1**: Update GPDMA_CxBR1 from memory

This bit controls the update of GPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if GPDMA_CxLLR ≠ 0, the linked-list is not completed.
GPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.
0: no GPDMA_CxBR1 update from memory (GPDMA_CxBR1.BNDT[15:0] restored if any link transfer)
1: GPDMA_CxBR1 update

Bit 28 **USA**: update GPDMA_CxSAR from memory

This bit controls the update of GPDMA_CxSAR from the memory during the link transfer.
0: no GPDMA_CxSAR update
1: GPDMA_CxSAR update

Bit 27 **UDA**: Update GPDMA_CxDAR register from memory
This bit is used to control the update of GPDMA_CxDAR from the memory during the link transfer.
0: no GPDMA_CxDAR update
1: GPDMA_CxDAR update

Bits 26:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update GPDMA_CxLLR register from memory
This bit is used to control the update of GPDMA_CxLLR from the memory during the link transfer.
0: no GPDMA_CxLLR update
1: GPDMA_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 GPDMA 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 GPDMA internal register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR).

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.

15.8.16 GPDMA channel x alternate linked-list address register (GPDMA_CxLLR)

Address offset: 0xCC + 0x80 * x (x = 6 to 7)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_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 GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_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 the GPDMA from the memory if GPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	UT3	UB2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	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 GPDMA_CxTR1 from memory

This bit controls the update of GPDMA_CxTR1 from the memory during the link transfer.

0: no GPDMA_CxTR1 update

1: GPDMA_CxTR1 update

Bit 30 **UT2**: Update GPDMA_CxTR2 from memory

This bit controls the update of GPDMA_CxTR2 from the memory during the link transfer.

0: no GPDMA_CxTR2 update

1: GPDMA_CxTR2 update

Bit 29 **UB1**: Update GPDMA_CxBR1 from memory

This bit controls the update of GPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if GPDMA_CxLLR ≠ 0, the linked-list is not completed.
GPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.

0: no GPDMA_CxBR1 update from memory (GPDMA_CxBR1.BNDT[15:0] restored if any link transfer)

1: GPDMA_CxBR1 update

Bit 28 **USA**: update GPDMA_CxSAR from memory

This bit controls the update of GPDMA_CxSAR from the memory during the link transfer.

0: no GPDMA_CxSAR update

1: GPDMA_CxSAR update

Bit 27 **UDA**: Update GPDMA_CxDAR register from memory
This bit is used to control the update of GPDMA_CxDAR from the memory during the link transfer.
0:no GPDMA_CxDAR update
1: GPDMA_CxDAR update

Bit 26 **UT3**: Update GPDMA_CxTR3 from memory
This bit controls the update of GPDMA_CxTR3 from the memory during the link transfer.
0: no GPDMA_CxTR3 update
1: GPDMA_CxTR3 update

Bit 25 **UB2**: Update GPDMA_CxBR2 from memory
This bit controls the update of GPDMA_CxBR2 from the memory during the link transfer.
0: no GPDMA_CxBR2 update
1: GPDMA_CxBR2 update

Bits 24:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update GPDMA_CxLLR register from memory
This bit is used to control the update of GPDMA_CxLLR from the memory during the link transfer.
0: no GPDMA_CxLLR update
1: GPDMA_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 GPDMA 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 GPDMA internal register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR).
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.

15.8.17 GPDMA register map

Table 96. GPDMA 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
0x04	GPDMA_ 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
0x08	Reserved	Reserved																															
0x0C	GPDMA_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
0x14 - 0x4C	Reserved	Reserved																															
0x50+0x80 * x (x=0 to 7)	GPDMA_ 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																

Table 96. GPDMA 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
0x54 to 0x58+0x80*x (x=0 to 7)	Reserved	Reserved																															
0x5C+0x80 * x (x=0 to 7)	GPDMA_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								
0x60+ 0x80 * x (x=0 to 7)	GPDMA_CxSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOL[7:0]								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	0		0	0	0	0	0	0	0								1
0x64+ 0x80 * x (x=0 to 7)	GPDMA_CxCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	LAP	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	0
0x68 to 0x8C+0x80 * x (x=0 to 7)	Reserved	Reserved																															
0x90+0x80 * x (x=0 to 7)	GPDMA_CxTR1	Res.	DAP	Res.	Res.	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2 [1:0]		Res.	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2 [1:0]	
	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
0x94+0x80 * x (x=0 to 7)	GPDMA_CxTR2	TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL[1:0]		Res.	Res.	TRIGSEL[5:0]						TRIGM[1:0]		Res.	PFREQ	BREQ	DREQ	SWREQ	Res.	REQSEL[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
0x98+0x80 * x (x=0 to 5)	GPDMA_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
0x98+0x80 * x (x=6 to 7)	GPDMA_CxBR1	BRDDEC	BRSDEC	DDEC	SDEC	Res.	BRC[10:0]											BNDT[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
0x9C+0x80 * x (x=0 to 7)	GPDMA_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
0xA0+0x80 * x (x=0 to 7)	GPDMA_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
0xA4+0x80 * x (x=6 to 7)	GPDMA_CxTR3	Res.	Res.	Res.	DAO[12:0]													Res.	Res.	Res.	SAO[12:0]												
	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
0xA8+0x80 * x (x=6 to 7)	GPDMA_CxBR2	BRDAO[15:0]																BRSAO[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
0xCC+0x80 * x (x=0 to 5)	GPDMA_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		
0xCC+0x80 * x (x=6 to 7)	GPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	UT3	UB2	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	0	0		

Refer to *Section 2.2* for the register boundary addresses.

16 Nested vectored interrupt controller (NVIC)

16.1 NVIC main features

- 134 maskable interrupt channels (not including the 16 Cortex-M33 with FPU interrupt lines)
- 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.

All interrupts including the core exceptions are managed by the NVIC.

16.2 SysTick calibration value register

The Cortex-M33 embeds one SysTick timer. The 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. In order 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 SYST_RVR register:

- When SysTick clock source is CPU clock HCLK
  reload value = (HCLK x STCALIB) - 1
- When SysTick clock source is external clock (HCLK/8)
  reload value = ((HCLK/8) x STCALIB) - 1

The HCLK refers to the AHB frequency value in MHz.

Example: SysTick clock source is CPU clock HCLK of 100 MHz, to match a time base of 1 ms:

SysTick reload value = (100 x STCALIB) - 1 = 0x1869F

16.3 Interrupt and exception vectors

The grey rows in the table below describe the vectors without specific position.

Table 97. STM32H503xx vector table

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000 0000
-	-4	fixed	Reset	Reset	0x0000 0004
-	-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000 0008
-	-1	fixed	HardFault	Hard fault. All classes of fault	0x0000 000C
-	0	settable	MemManage	Memory management	0x0000 0010
-	1	settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
-	2	settable	UsageFault	Undefined instruction or invalid state	0x0000 0018
-	-	-	-	Reserved	0x0000 001C - 0x0000 0028
-	4	-	SVC	System service call via SWI instruction	0x0000 002C
-	5	-	Debug Monitor	Monitor	0x0000 0030
-	-	-	-	Reserved	0x0000 0034
-	6	settable	PendSV	Pendable request for system service	0x0000 0038
-	7	settable	SysTick	System tick timer	0x0000 003C
0	8	settable	WWDG	Window watchdog interrupt	0x0000 0040
1	9	settable	PVD_AVD	Power voltage monitor/ Analog voltage monitor	0x0000 0044
2	10	settable	RTC	RTC global interrupts	0x0000 0048
-	-	-	-	Reserved	0x0000 004C
4	12	settable	TAMP	Tamper global interrupts	0x0000 0050
5	13	settable	RAMCFG	RAM configuration global interrupt	0x0000 0054
6	14	settable	FLASH	Flash global interrupt	0x0000 0058
-	-	-	-	Reserved	0x0000 005C
-	-	-	-	Reserved	0x0000 0060
9	17	settable	RCC	RCC global interrupt	0x0000 0064
-	-	-	-	Reserved	0x0000 0068
11	19	settable	EXTI0	EXTI Line0 interrupt	0x0000 006C
12	20	settable	EXTI1	EXTI Line1 interrupt	0x0000 0070
13	21	settable	EXTI2	EXTI Line2 interrupt	0x0000 0074

Table 97. STM32H503xx vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
14	22	settable	EXTI3	EXTI Line3 interrupt	0x0000 0078
15	23	settable	EXTI4	EXTI Line4 interrupt	0x0000 007C
16	24	settable	EXTI5	EXTI Line5 interrupt	0x0000 0080
17	25	settable	EXTI6	EXTI Line6 interrupt	0x0000 0084
18	26	settable	EXTI7	EXTI Line7 interrupt	0x0000 0088
19	27	settable	EXTI8	EXTI Line8 interrupt	0x0000 008C
20	28	settable	EXTI9	EXTI Line9 interrupt	0x0000 0090
21	29	settable	EXTI10	EXTI Line10 interrupt	0x0000 0094
22	30	settable	EXTI11	EXTI Line11 interrupt	0x0000 0098
23	31	settable	EXTI12	EXTI Line12 interrupt	0x0000 009C
24	32	settable	EXTI13	EXTI Line13 interrupt	0x0000 00A0
25	33	settable	EXTI14	EXTI Line14 interrupt	0x0000 00A4
26	34	settable	EXTI15	EXTI Line15 interrupt	0x0000 00A8
27	35	settable	GPDMA1_CH0	GPDMA1 channel 0 global interrupt	0x0000 00AC
28	36	settable	GPDMA1_CH1	GPDMA1 channel 1 global interrupt	0x0000 00B0
29	37	settable	GPDMA1_CH2	GPDMA1 channel 2 global interrupt	0x0000 00B4
30	38	settable	GPDMA1_CH3	GPDMA1 channel 3 global interrupt	0x0000 00B8
31	39	settable	GPDMA1_CH4	GPDMA1 channel 4 global interrupt	0x0000 00BC
32	40	settable	GPDMA1_CH5	GPDMA1 channel 5 global interrupt	0x0000 00C0
33	41	settable	GPDMA1_CH6	GPDMA1 channel 6 global interrupt	0x0000 00C4
34	42	settable	GPDMA1_CH7	GPDMA1 channel 7 global interrupt	0x0000 00C8
35	43	settable	IWDG	Independent watchdog interrupt	0x0000 00CC
-	-	-	-	Reserved	0x0000 00D0
37	45	settable	ADC1	ADC1 global interrupt	0x0000 00D4
38	46	settable	DAC1	DAC1 global interrupt	0x0000 00D8
39	47	settable	FDCAN1_IT0	FDCAN1 Interrupt 0	0x0000 00DC
40	48	settable	FDCAN1_IT1	FDCAN1 Interrupt 1	0x0000 00E0
41	49	settable	TIM1_BRK/TIM1_TERR/ TIM1_IERR	TIM1 break/TIM1 transition error/TIM1 index error	0x0000 00E4
42	50	settable	TIM1_UP	TIM1 update	0x0000 00E8

Table 97. STM32H503xx vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
43	51	settable	TIM1_TRG_COM/TIM1_DIR/TIM1_IDX	TIM1 trigger and commutation/TIM1 direction change interrupt/TIM1 index	0x0000 00EC
44	52	settable	TIM1_CC	TIM1 capture compare interrupt	0x0000 00F0
45	53	settable	TIM2	TIM2 global interrupt	0x0000 00F4
46	54	settable	TIM3	TIM3 global interrupt	0x0000 00F8
-	-	-	-	Reserved	0x0000 00FC - 0x0000 0100
49	57	settable	TIM6	TIM6 global interrupt	0x0000 0104
50	58	settable	TIM7	TIM7 global interrupt	0x0000 0108
51	59	settable	I2C1_EV	I2C1 event interrupt	0x0000 010C
52	60	settable	I2C1_ER	I2C1 error interrupt	0x0000 0110
53	61	settable	I2C2_EV	I2C2 event interrupt	0x0000 0114
54	62	settable	I2C2_ER	I2C2 error interrupt	0x0000 0118
55	63	settable	SPI1	SPI1 global interrupt	0x0000 011C
56	64	settable	SPI2	SPI2 global interrupt	0x0000 0120
57	65	settable	SPI3	SPI3 global interrupt	0x0000 0124
58	66	settable	USART1	USART1 global interrupt	0x0000 0128
59	67	settable	USART2	USART2 global interrupt	0x0000 012C
60	68	settable	USART3	USART3 global interrupt	0x0000 0130
-	-	-	-	Reserved	0x0000 0134 - 0x0000 0138
63	71	settable	LPUART1	LPUART1 global interrupt or LPUART1 R wakeup or LPUART1 T wakeup through EXTI line	0x0000 013C
64	72	settable	LPTIM1 OR LPTIM1_AIT	LPTIM1 global interrupt or LPTimer1 AIT through EXTI line	0x0000 0140
-	-	-	-	Reserved	0x0000 0144 - 0x0000 0154
70	78	settable	LPTIM2 OR LPTIM2_AIT	LPTIM2 global interrupt or LPTimer2 AIT through EXTI line	0x0000 0158
-	-	-	-	Reserved	0x0000 015C - 0x0000 0164
74	82	settable	USB	USB global interrupt	0x0000 0168

Table 97. STM32H503xx vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
75	83	settable	CRS	Clock recovery system global interrupt	0x0000 016C
-	-	-	-	Reserved	0x0000 0170 - 0x0000 018C
90	98	settable	GPDMA2_CH0	GPDMA2 channel0 global interrupt	0x0000 0190
91	99	settable	GPDMA2_CH1	GPDMA2 channel1 global interrupt	0x0000 0194
92	100	settable	GPDMA2_CH2	GPDMA2 channel2 global interrupt	0x0000 0198
93	101	settable	GPDMA2_CH3	GPDMA2 channel3 global interrupt	0x0000 019C
94	102	settable	GPDMA2_CH4	GPDMA2 channel4 global interrupt	0x0000 01A0
95	103	settable	GPDMA2_CH5	GPDMA2 channel5 global interrupt	0x0000 01A4
96	104	settable	GPDMA2_CH6	GPDMA2 channel6 global interrupt	0x0000 01A8
97	105	settable	GPDMA2_CH7	GPDMA2 channel7 global interrupt	0x0000 01AC
-	-	-	-	Reserved	0x0000 01B0 - 0x0000 01C0
103	111	settable	FPU	Floating point interrupt	0x0000 01C4
104	112	settable	ICACHE	Instruction cache global interrupt	0x0000 01C8
-	-	-	-	Reserved	0x0000 01CC - 0x0000 01E8
113	121	settable	DTS OR DTS_WKUP	DTS interrupt or DTS AIT through EXTI line	0x0000 01EC
114	122	settable	RNG	RNG global interrupt	0x0000 01F0
-	-	-	-	Reserved	0x0000 01F4 - 0x0000 01F8
117	125	settable	HASH	HASH interrupt	0x0000 01FC
-	-	-	-	Reserved	0x0000 0200 - 0x0000 0210
123	131	settable	I3C1_EV	I3C1 event interrupt	0x0000 0214
124	132	settable	I3C1_ER	I3C1 error interrupt	0x0000 0218
-	-	-	-	Reserved	0x0000 021C - 0x0000 0230
131	139	settable	I3C2_EV	I3C2 event interrupt	0x0000 0234
132	140	settable	I3C2_ER	I3C2 error interrupt	0x0000 0238
133	141	settable	COMP	COMP global interrupt	0x0000 023C

17 Extended interrupts and event controller (EXTI)

The extended interrupts and event controller (EXTI) manages the individual CPU and system wakeup through configurable event inputs. It provides wakeup 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 wakeup 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.

17.1 EXTI main features

The EXTI main features are the following:

- 54 input events supported
- All event inputs allow the possibility to wake up the system.
- Events that do not have an associated wakeup 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 wakeup 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 wakeup, interrupt and event generation
- Software trigger possibility
- EXTI IO port selection

17.2 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 71*.

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 wakeup, interrupt and event outputs, and their masking.

The EXTI mux provides the IO port selection on to the EXTI event signal.

Figure 71. EXTI block diagram

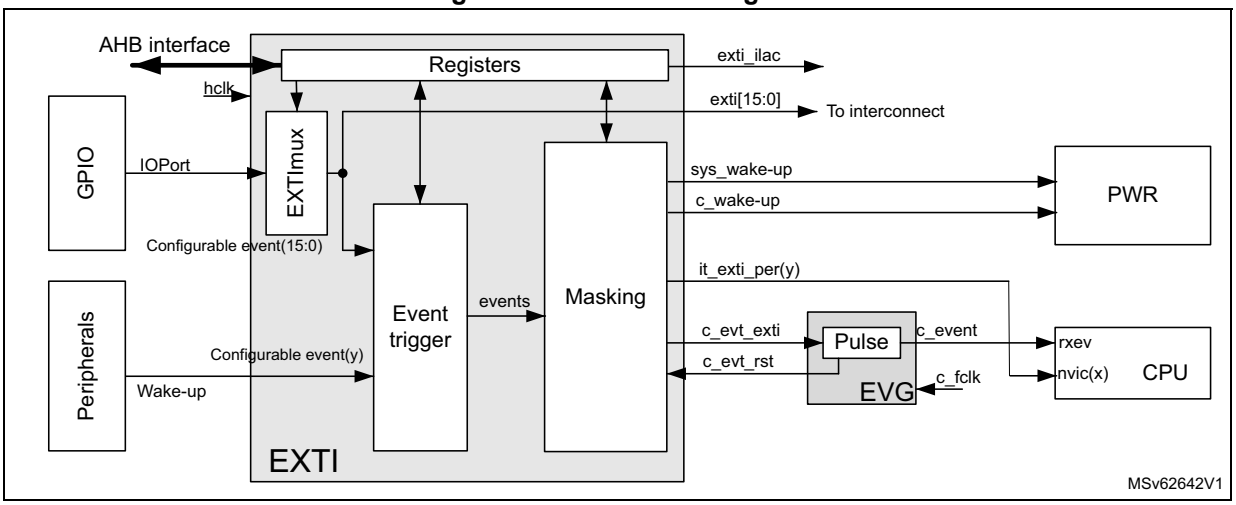


Table 98. EXTI pin overview

Pin name	I/O	Description
AHB interface	I/O	EXTI register bus interface.
hclk	I	AHB bus clock and EXTI system clock
Configurable event(y)	I	Asynchronous wakeup events from peripherals that do not have an associated interrupt and flag in the peripheral
exti_ilac	O	Illegal access event
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 wakeup request to PWR for ck_sys and hclk
c_wakeup	O	Wakeup request to PWR for CPU, synchronous to hclk

Table 99. 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

17.2.1 EXTI connections between peripherals and CPU

Some peripherals are able to generate wakeup or interrupt events when the system is in Stop mode, are connected to the EXTI.

- Peripheral wakeup 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 allow 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 wakeup signals are connected to the PWR and are used to wake up the system and the CPU sub-system bus clocks.

17.2.2 EXTI interrupt/event mapping

The EXTI lines are connected as shown in the table below.

Table 100. EXTI line connections

EXTI Line	Line source	Line type
0-15	GPIO	Configurable
16	PVD/AVD output	Configurable
17	RTC	-
18	Reserved	-
19	TAMP	-
20	Reserved	-
21	I2C1 wakeup	-
22	I2C2 wakeup	-
23	Reserved	-
24	I3C wakeup	-
25	USART1 wakeup	-
26	USART2 wakeup	-
27	USART3 wakeup	-
28	I3C2 wakeup	-
29	COMP1 output wakeup	-
30 - 36	Reserved	-
37	LPUART1 wakeup	-
38	LPTIM1	-
39	LPTIM2	-
40	SPI1 wakeup	-
41	SPI2 wakeup	-
42	SPI3 wakeup	-
43 - 46	Reserved	-
47	USB wakeup	-

Table 100. EXTI line connections (continued)

EXTI Line	Line source	Line type
48	Reserved	-
49	LPTIM2 CH1	-
50	DTS wakeup	Configurable

17.3 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 wakeup and interrupt enable in the

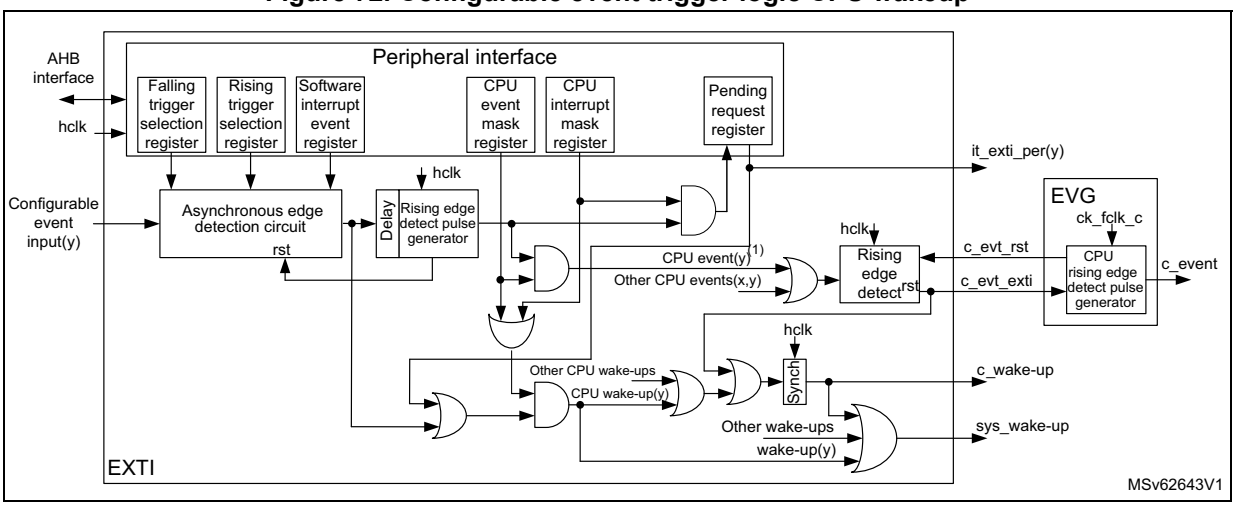
  EXTI CPU wakeup with interrupt mask register (EXTI_IMR1)
- CPU wakeup and event enable

  EXTI CPU wakeup with interrupt mask register (EXTI_IMR1)

17.3.1 EXTI configurable event input wakeup

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 wakeup event.

Figure 72. Configurable event trigger logic CPU wakeup



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 wakeup (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 ORed 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.

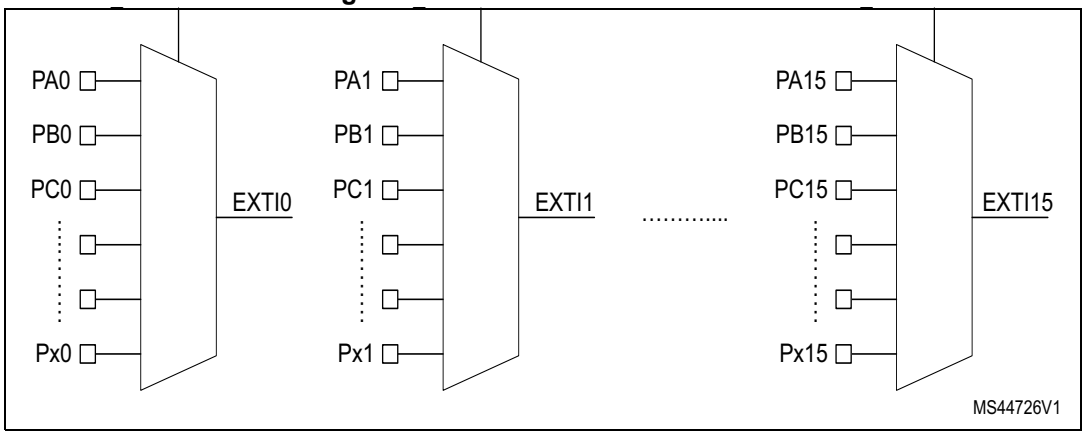
When a CPU wakeup (interrupt) or CPU event is enabled, the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees that the EXTI hclk clock is woken up before the asynchronous edge detection circuit is reset.

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.

17.3.2 EXTI mux selection

The EXTI mux allows the selection of GPIOs as interrupts and wakeup. 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 external interrupt selection register (EXTI_EXTICR1)*.

Figure 73. 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.

17.4 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 wakeup generation is conditioned by the CPU interrupt mask and CPU event mask.

Table 101. 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(1)	CPU event	CPU wakeup
0	0	No	Masked	Masked	Masked
	1	No	Masked	Yes	Yes
1	0	Status latched	Yes	Masked	Yes(2)
	1	Status latched	Yes	Yes	Yes

1. The single exti(n) interrupt goes to the CPU. If no interrupt is required for CPU(m), the exti(n) interrupt must be masked in the CPU NVIC.
2. Only if CPU interrupt is enabled in EXTI_IMR.IMn.

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.

17.5 EXTI event protection

The EXTI is able to protect event register bits from being modified by unprivileged accesses. The protection is individually activated per input event via the register bits in EXTI_PRIVCFGR. At EXTI level, the protection consists in preventing the following unauthorized write access:

- Change the settings of the privileged configurable events.
- Change the masking of the privileged input events.
- Clear pending status of the privileged input events.

Table 102. Register protection overview

Register name	Access type	Protection(1)
EXTI_RTSR	RW	Privilege can be bit-wise enabled in EXTI_PRIVCFGR.
EXTI_FTSR	RW	
EXTI_SWIER	RW	
EXTI_RPR	RW	
EXTI_FPR	RW	
EXTI_PRIVCFGR	RW	Always privilege.
EXTI_EXTICRn	RW	Privilege can be bit-wise enabled in EXTI_PRIVCFGR.

Table 102. Register protection overview

Register name	Access type	Protection(1)
EXTI_IMR	RW	Privilege can be bit-wise enabled in EXTI_PRIVCFGR.
EXTI_EMR	RW	

1. Privilege is enabled with the individual Input event (EXTI_PRIVCFGR register).

17.5.1 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.

17.6 EXTI registers

The EXTI register map is divided in the following sections:

Table 103. EXTI register map sections

Address offset	Description
0x000 - 0x01C	General configurable event [31:0] configuration
0x020 - 0x03C	General configurable event [57:32] configuration
0x060 - 0x06C	EXTI IO port mux selection
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.

17.6.1 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 **RTx:** Rising trigger event configuration bit of configurable event input x[(1)] (x = 16 to 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.

17.6.2 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 **FTx:** Falling trigger event configuration bit of configurable event input x [(1)] (x = 16 to 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.

17.6.3 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 **SWIx:** Software interrupt on event x (x = 16 to 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.

17.6.4 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
															rw
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
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 **RPIFx:** configurable event inputs x rising edge pending bit (x = 16 to 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.

17.6.5 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
															rw
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
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 **FPIFx:** configurable event inputs x falling edge pending bit (x = 16 to 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.

17.6.6 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 privilege capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	Res.	PRIV22	PRIV21	Res.	PRIV19	Res.	PRIV17	PRIV16
		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 31:30 Reserved, must be kept at reset value.

Bits 29:24 **PRIVx:** Privilege enable on event input x (x = 29 to 24)

0: Event privilege disabled (unprivileged)

1: Event privilege enabled (privileged)

Bit 23 Reserved, must be kept at reset value.

Bits 22:21 **PRIVx:** Privilege enable on event input x (x = 22 to 21)
0: Event privilege disabled (unprivileged)
1: Event privilege enabled (privileged)

Bit 20 Reserved, must be kept at reset value.

Bit 19 **PRIV19:** Privilege enable on event input 19
0: Event privilege disabled (unprivileged)
1: Event privilege enabled (privileged)

Bit 18 Reserved, must be kept at reset value.

Bits 17:0 **PRIVx:** Privilege enable on event input x (x = 17 to 0)
0: Event privilege disabled (unprivileged)
1: Event privilege enabled (privileged)

17.6.7 EXTI rising trigger selection register 2 (EXTI_RTSR2)

Address offset: 0x020

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.	RT53	Res.	Res.	RT50	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 31:22, 20:19, 17:0 Reserved, must be kept at reset value.

Bit 21 **RT53:** Rising trigger event configuration bit of configurable event input x[(1)]
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

Bit 18 **RT50:** Rising trigger event configuration bit of configurable event input x[(1)]
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.

17.6.8 EXTI falling trigger selection register 2 (EXTI_FTSR2)

Address offset: 0x024

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.	FT53	Res.	Res.	FT50	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 31:22, 20:19, 17:0 Reserved, must be kept at reset value.

Bit 21 **FT53:** Falling trigger event configuration bit of configurable event input x (1)

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.

Bit 18 **FT50:** Falling trigger event configuration bit of configurable event input x (1)

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.

17.6.9 EXTI software interrupt event register 2 (EXTI_SWIER2)

Address offset: 0x028

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.	SWI53	Res.	Res.	SWI50	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 31:22, 20:19, 17:0 Reserved, must be kept at reset value.

Bit 21 **SWI53:** Software interrupt on event x
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.

Bit 18 **SWI50:** Software interrupt on event x
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.

17.6.10 EXTI rising edge pending register 2 (EXTI_RPR2)

Address offset: 0x02C

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.	RPIF53	Res.	Res.	RPIF50	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 31:22, 20:19, 17:0 Reserved, must be kept at reset value.

Bit 21 **RPIF53:** configurable event inputs x rising edge pending bit
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.

Bit 18 **RPIF50:** configurable event inputs x rising edge pending bit
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.

17.6.11 EXTI falling edge pending register 2 (EXTI_FPR2)

Address offset: 0x030

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.	FPIF53	Res.	Res.	FPIF50	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 31:22, 20:19, 17:0 Reserved, must be kept at reset value.

Bit 21 **FPIF53:** configurable event inputs x falling edge pending bit
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.

Bit 18 **FPIF50:** configurable event inputs x falling edge pending bit
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.

17.6.12 EXTI privilege configuration register 2 (EXTI_PRIVCFGR2)

Address offset: 0x038

Reset value: 0x0000 0000

This register provides privileged write access protection. An unprivileged read returns the register data.

Contains only register bits for privilege 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.	PRIV53	Res.	Res.	PRIV50	PRIV49	Res.
										rw			rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV47	Res.	Res.	Res.	Res.	PRIV42	PRIV41	PRIV40	PRIV39	PRIV38	PRIV37	Res.	Res.	Res.	Res.	Res.
rw					rw	rw	rw	rw	rw	rw					

Bits 31:22, 20:19, 16, 14:11, 4:0 Reserved, must be kept at reset value.

Bit 21 **PRIV53:** Privilege enable on event input x
0: Event privilege disabled (unprivileged)
1: Event privilege enabled (privileged)

Bits 18:17 **PRIVx:** Privilege enable on event input x (x = 50 to 49)
0: Event privilege disabled (unprivileged)
1: Event privilege enabled (privileged)

Bit 15 **PRIV47:** Privilege enable on event input x
0: Event privilege disabled (unprivileged)
1: Event privilege enabled (privileged)

Bits 10:5 **PRIVx:** Privilege enable on event input x (x = 42 to 37)
0: Event privilege disabled (unprivileged)
1: Event privilege enabled (privileged)

17.6.13 EXTI external interrupt selection register (EXTI_EXTICR1)

Address offset: 0x060

(EXTI mux 0, 1, 2, 3)

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 selectio
These bits are written by software to select the source input for EXTI3 external interrupt.
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.
0x00: PA[3] pin
0x01: PB[3] pin
0x02: PC[3] 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_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: PA[2] pin
0x01: PB[2] pin
0x02: PC[2] pin
0x03: PD[2] 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_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: PA[1] pin
0x01: PB[1] pin
0x02: PC[1] pin
0x07: PH[1] 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_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: PA[0] pin
0x01: PB[0] pin
0x02: PC[0] pin
0x07: PH[0] pin
Others: reserved

17.6.14 EXTI external interrupt selection register (EXTI_EXTICR2)

Address offset: 0x064 EXTI mux 4, 5, 6, 7

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_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: PA[7] pin
0x01: PB[7] pin
0x02: PC[7] 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_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: PA[6] pin
0x01: PB[6] pin
0x02: PC[6] 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_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: PA[5] pin
0x01: PB[5] pin
0x02: PC[5] 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_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.
0x00: PA[4] pin
0x01: PB[4] pin
0x02: PC[4] pin
Others: reserved

17.6.15 EXTI external interrupt selection register (EXTI_EXTICR3)

Address offset: 0x068 EXTI mux 8, 9, 10, 11

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_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: PA[11] pin
0x02: PC[11] 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_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: PA[10] pin
0x01: PB[10] pin
0x02: PC[10] 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_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: PA[9] pin
0x02: PC[9] 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_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: PA[8] pin
0x01: PB[8] pin
0x02: PC[8] pin
Others: reserved

17.6.16 EXTI external interrupt selection register (EXTI_EXTICR4)

Address offset: 0x06C EXTI mux 12, 13, 14, 15

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_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: PA[15] pin
0x01: PB[15] pin
0x02: PC[15] 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_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: PA[14] pin
0x01: PB[14] pin
0x02: PC[14] 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_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: PA[13] pin
0x01: PB[13] pin
0x02: PC[13] pin
Others: reserved

Bits 7:0 **EXTI12[7:0]:** EXTI12 GPIO port selection

These bits are written by software to select the source input for EXTI12 external interrupt.
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: PA[12] pin
0x01: PB[12] pin
0x02: PC[12] pin
Others: reserved

17.6.17 EXTI CPU wakeup with interrupt mask register (EXTI_IMR1)

Address offset: 0x080

Reset value: 0xFFFE 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.	IM29	IM28	IM27	IM26	IM25	IM24	Res.	IM22	IM21	Res.	IM19	Res.	IM17	IM16
		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
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 31:30, 23, 20, 18 Reserved, must be kept at reset value.

Bits 29:24 **IMx:** CPU wakeup with interrupt mask on event input x [1] (x = 29 to 24)
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: Wakeup with interrupt request from input event x is masked.
1: Wakeup with interrupt request from input event x is unmasked.

Bits 22:21 **IMx:** CPU wakeup with interrupt mask on event input x [1] (x = 22 to 21)
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: Wakeup with interrupt request from input event x is masked.
1: Wakeup with interrupt request from input event x is unmasked.

Bit 19 **IM19:** CPU wakeup with interrupt mask on event input x [1]
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: Wakeup with interrupt request from input event x is masked.
1: Wakeup with interrupt request from input event x is unmasked.

Bits 17:0 **IMx:** CPU wakeup with interrupt mask on event input x [1] (x = 17 to 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: Wakeup with interrupt request from input event x is masked.
1: Wakeup 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.

17.6.18 EXTI CPU wakeup 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.	EM29	EM28	EM27	EM26	EM25	EM24	Res.	EM22	EM21	Res.	EM19	Res.	EM17	EM16
		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
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 31:30, 23, 20, 18 Reserved, must be kept at reset value.

Bits 29:24 **EMx:** CPU wakeup with event generation mask on event input x (x = 29 to 24)

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: Wakeup with event generation from Line x is masked.

1: Wakeup with event generation from Line x is unmasked.

Bits 22:21 **EMx:** CPU wakeup with event generation mask on event input x (x = 22 to 21)

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: Wakeup with event generation from Line x is masked.

1: Wakeup with event generation from Line x is unmasked.

Bit 19 **EM19:** CPU wakeup with event generation mask on event input x

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: Wakeup with event generation from Line x is masked.

1: Wakeup with event generation from Line x is unmasked.

Bits 17:0 **EMx:** CPU wakeup with event generation mask on event input x (x = 17 to 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: Wakeup with event generation from Line x is masked.

1: Wakeup with event generation from Line x is unmasked.

17.6.19 EXTI CPU wakeup with interrupt mask register 2 (EXTI_IMR2)

Address offset: 0x090

Reset value: 0x001B FFFF

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.	IM53	Res.	Res.	IM50	IM49	Res.
										rw			rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM47	Res.	Res.	Res.	Res.	IM42	IM41	IM40	IM39	IM38	IM37	Res.	Res.	Res.	Res.	Res.
rw					rw	rw	rw	rw	rw	rw					

Bits 31:22, 20:19, 16, 14:11, 4:0 Reserved, must be kept at reset value.

Bit 21 **IM53:** CPU wakeup with interrupt mask on event input x [1]

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: Wakeup with interrupt request from input event x is masked.
1: Wakeup with interrupt request from input event x is unmasked.

Bits 18:17 **IMx:** CPU wakeup with interrupt mask on event input x [1] (x = 50 to 49)

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: Wakeup with interrupt request from input event x is masked.
1: Wakeup with interrupt request from input event x is unmasked.

Bit 15 **IM47:** CPU wakeup with interrupt mask on event input x [1]

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: Wakeup with interrupt request from input event x is masked.
1: Wakeup with interrupt request from input event x is unmasked.

Bits 10:5 **IMx:** CPU wakeup with interrupt mask on event input x [1] (x = 42 to 37)

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: Wakeup with interrupt request from input event x is masked.
1: Wakeup 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.

17.6.20 EXTI CPU wakeup with event mask register 2 (EXTI_EMR2)

Address offset: 0x094

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.	EM53	Res.	Res.	EM50	EM49	Res.
										rw			rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM47	Res.	Res.	Res.	Res.	EM42	EM41	EM40	EM39	EM38	EM37	Res.	Res.	Res.	Res.	Res.
rw					rw	rw	rw	rw	rw	rw					

Bits 31:22, 20:19, 16, 14:11, 4:0 Reserved, must be kept at reset value.

Bit 21 **EM53:** CPU wakeup with event generation mask on event input x

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: Wakeup with event generation from Line x is masked.

1: Wakeup with event generation from Line x is unmasked.

Bits 18:17 **EMx:** CPU wakeup with event generation mask on event input x (x = 50 to 49)

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: Wakeup with event generation from Line x is masked.

1: Wakeup with event generation from Line x is unmasked.

Bit 15 **EM47:** CPU wakeup with event generation mask on event input x

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: Wakeup with event generation from Line x is masked.

1: Wakeup with event generation from Line x is unmasked.

Bits 10:5 **EMx:** CPU wakeup with event generation mask on event input x (x = 42 to 37)

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: Wakeup with event generation from Line x is masked.

1: Wakeup with event generation from Line x is unmasked.

17.6.21 EXTI register map

Table 104. 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.	RPIF1	RPIF1	RPIF1	RPIF1	RPIF1	RPIF1	RPIF1	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
0x018	**EXTI_ PRIVCFGR1**	Res.	Res.	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	Res.	PRIV22	PRIV21	Res.	PRIV19	Res.	PRIV17	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	0	0	0	0	0	0	0	0	0	0
0x020	**EXTI_RTSR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT53	Res.	Res.	RT50	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0			0																		
0x024	**EXTI_FTSR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT53	Res.	Res.	FT50	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0			0																		
0x028	**EXTI_SWIER2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI53	Res.	Res.	SWI50	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0			0																		
0x02C	**EXTI_RPR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF53	Res.	Res.	RPIF50	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0			0																		
0x030	**EXTI_FPR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF53	Res.	Res.	FPIF50	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0			0																		
0x038	**EXTI_ PRIVCFGR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV53	Res.	Res.	PRIV50	PRIV49	Res.	PRIV47	Res.	Res.	Res.	Res.	PRIV42	PRIV41	PRIV40	PRIV39	PRIV38	PRIV37	Res.	Res.	Res.	Res.	Res.
	Reset value											0			0	0		0					0	0	0	0	0	0					
0x03C 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

Table 104. 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
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]								EXTI12[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
0x074 to 0x07C	**Reserved**	Reserved																															
0x080	**EXTI_IMR1**	Res.	Res.	IM29	IM28	IM27	IM26	IM25	IM24	Res.	IM22	IM21	Res.	IM19	Res.	IM17	IM16	IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
	Reset value			1	1	1	1	1	1		1	1		1		1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	**EXTI_EMR1**	Res.	Res.	EM29	EM28	EM27	EM26	EM25	EM24	Res.	EM22	EM21	Res.	EM19	Res.	EM17	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	0	0	0	0	0	0	0	0	0	0
0x090	**EXTI_IMR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM53	Res.	Res.	IM50	IM49	Res.	IM47	Res.	Res.	Res.	Res.	IM42	IM41	IM40	IM39	IM38	IM37	Res.	Res.	Res.	Res.	Res.
	Reset value											0			0	1		1					1	1	1	1	1	1					
0x094	**EXTI_EMR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM53	Res.	Res.	EM50	EM49	Res.	EM47	Res.	Res.	Res.	Res.	EM42	EM41	EM40	EM39	EM38	EM37	Res.	Res.	Res.	Res.	Res.
	Reset value											0			0	0		0					0	0	0	0	0	0					

Refer to *Section 2.2* for the register boundary addresses.

18 Cyclic redundancy check calculation unit (CRC)

18.1 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.

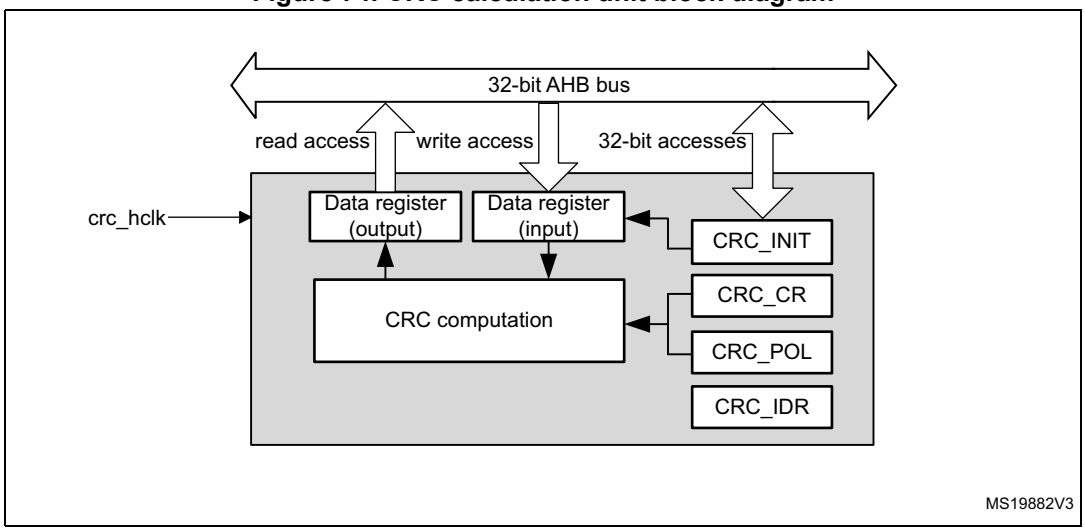
18.2 CRC main features

- 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

18.3 CRC functional description

18.3.1 CRC block diagram

Figure 74. CRC calculation unit block diagram



18.3.2 CRC internal signals

Table 105. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

18.3.3 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 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 calculation. 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.

18.4 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.

18.4.1 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.

18.4.2 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

18.4.3 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

18.4.4 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.

18.4.5 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.

18.4.6 CRC register map

Table 106. 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 2.2* for the register boundary addresses.

19 Analog-to-digital converters (ADC1)

19.1 Introduction

The ADC consists of a 12-bit successive approximation analog-to-digital converter.

The ADC has up to 20 multiplexed channels. 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 ADC is mapped on the AHB bus to allow fast data handling.

The analog watchdog features allow the application to detect if the input voltage goes outside the user-defined high or low thresholds.

A built-in hardware oversampler allows improving analog performances while off-loading the related computational burden from the CPU.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

19.2 ADC main features

- High-performance features
  - 16 external channels and to 4 internal channels
  - 12, 10, 8 or 6-bit configurable resolution
  - ADC conversion time independent from the AHB bus clock frequency
  - Faster conversion time by lowering resolution
  - Manage single-ended or differential inputs
  - AHB slave bus interface to allow fast data handling
  - Self-calibration
  - Channel-wise programmable sampling time
  - Flexible sampling time control
  - Up to 4 injected channels (analog inputs assignment to regular or injected channels is fully configurable)
  - Hardware assistant to prepare the context of the injected channels to allow fast context switching
  - Data alignment with in-built data coherency
  - Data can be managed by DMA for regular channel conversions
  - Four dedicated data registers for the injected channels
- Low-power features
  - Speed adaptive low-power mode to reduce ADC consumption when operating at low frequency
  - Allows slow bus frequency application while keeping optimum ADC performance

  - Provides automatic control to avoid ADC overrun in low AHB bus clock frequency application (auto-delayed mode)
- Oversampler
  - 16-bit data register
  - Oversampling ratio adjustable from 2 to 256x
  - Programmable data shift up to 8 bits
- Data preconditioning
  - Offset compensation
- Analog input channels
  - External analog inputs:

    Up to 5 fast channels from GPIO pads

    Up to 11 slow channels from GPIO pads
  - 1 channel for the internal temperature sensor (V_{SENSE})
  - 1 channel for the internal reference voltage (V_{REFINT})
  - 1 channel for monitoring the external VBAT power supply pin
  - 1 channel for monitoring the internal V_{DDCORE} supply
- Start-of-conversion can be initiated:
  - By software for both regular and injected conversions
  - By hardware triggers with configurable polarity (internal timers events or GPIO input events) for both regular and injected conversions
- Conversion modes
  - The ADC 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 ADC ready, the end of sampling, the end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or overrun events
- 3 analog watchdogs
  - Watchdog can perform filtering to ignore out-of-range data
- ADC input range: $V_{SSA} \leq V_{IN} \leq V_{REF+}$

Figure 75 shows the block diagram of one ADC.

19.3 ADC implementation

Table 107. ADC features

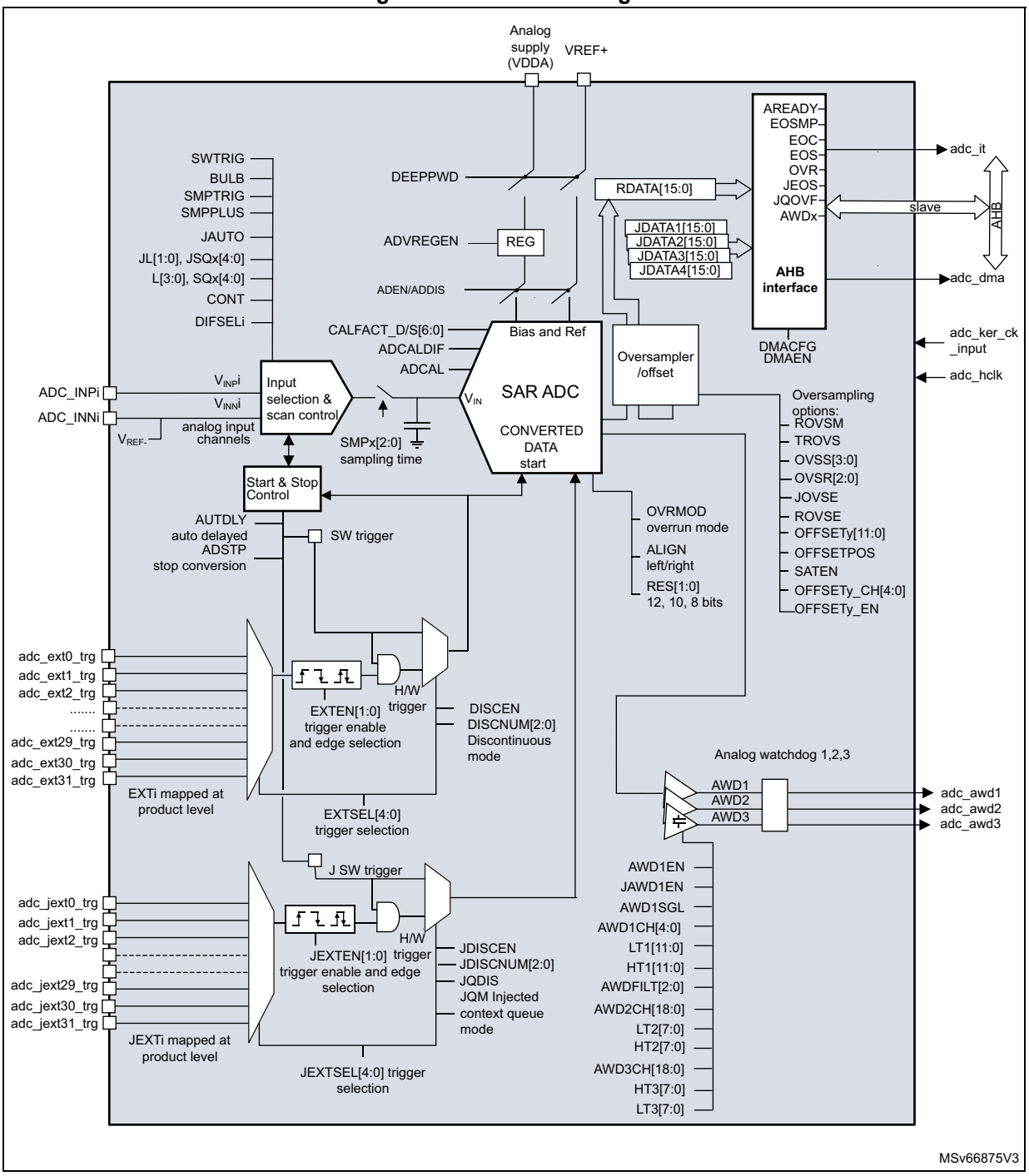
ADC modes/features	ADC1
Resolution	12 bit
Maximum sampling speed	2.5 Msps (12-bit resolution)
Dual mode operation	-
Hardware offset calibration	X
Hardware linearity calibration	-
Single-end input	X
Differential input	X
Injected channel conversion	X
Oversampling	up to x256
Data register	16 bits
Data register FIFO depth	3 stages
DMA support	X
Parallel data output to ADF	-
Offset compensation	X
Gain compensation	-
Number of analog watchdog	3
Option register	X

19.4 ADC functional description

19.4.1 ADC block diagram

Figure 75 shows the ADC block diagram and *Table 108* gives the ADC pin description.

Figure 75. ADC block diagram



19.4.2 ADC pins and internal signals

Table 108. ADC input/output pins

Pin name	Signal type	Description
VDDA	Input, analog supply	Analog power supply and positive reference voltage for the ADC
VSSA	Input, analog supply ground	Ground for analog power supply, equal to V_{SS}.
VREF+	Input, analog reference positive	The higher/positive reference voltage for the ADC.
VREF−	Input, analog reference negative	The lower/negative reference voltage for the ADC. V_{REF-} is internally connected to V_{SSA}
ADC1_INNi/INPi	Negative/positive external analog input signals	20 negative/positive external analog input channels (refer to *Section 19.4.4: ADC connectivity* for details)

Table 109. ADC internal input/output signals

Internal signal name	Signal type	Description
$V_{INP}i$	Positive analog input channels	Positive internal analog input channels connected either to ADC1_INP*i* external channels or to internal channels.
$V_{INN}i$	Negative analog input channels	Negative internal analog input channels connected either to ADC1_INN*i* external channels or to internal channels
adc_ext_trgi	Inputs	ADC external trigger inputs for regular conversions.
adc_jext_trgi	Inputs	ADC external trigger inputs for the injected conversions.
adc_awdx	Output	Internal analog watchdog output signal connected to on-chip timers. (x = Analog watchdog number 1,2,3)
adc_ker_ck_input	Input	ADC kernel clock
adc_hclk	Input	ADC peripheral clock
adc_it	Output	ADC interrupt
adc_dma	Output	ADC DMA request

Table 110. ADC interconnection

Signal name	Source/destination
ADC1 V_{INP}[16]	V_{SENSE} (internal temperature sensor output voltage).
ADC1 V_{INP}[17]	V_{REFINT} (output voltage from internal reference voltage).
ADC1 V_{INP}[2]	V_{BAT}/4 (VBAT pin input voltage divided by 4).
ADC1 V_{INP}[6]	V_{DDCORE} (internal digital core voltage).
adc_ext_trg0	tim1_oc1
adc_ext_trg1	tim1_oc2
adc_ext_trg2	tim1_oc3
adc_ext_trg3	tim2_oc2

Table 110. ADC interconnection (continued)

Signal name	Source/destination
adc_ext_trg4	tim3_trgo
adc_ext_trg5	reserved
adc_ext_trg6	exti11
adc_ext_trg7	reserved
adc_ext_trg8	reserved
adc_ext_trg9	tim1_trgo
adc_ext_trg10	tim1_trgo2
adc_ext_trg11	tim2_trgo
adc_ext_trg12	reserved
adc_ext_trg13	tim6_trgo
adc_ext_trg14	reserved
adc_ext_trg15	tim3_oc4
adc_ext_trg16	exti15
adc_ext_trg17	tim7_trgo
adc_ext_trg18	lptim1_ch1
adc_ext_trg19	lptim2_ch1
adc_ext_trg20	reserved
adc_ext_trg21	reserved
adc_ext_trg22	reserved
adc_ext_trg23	reserved
adc_ext_trg24	reserved
adc_ext_trg25	reserved
adc_ext_trg26	reserved
adc_ext_trg27	reserved
adc_ext_trg28	reserved
adc_ext_trg29	reserved
adc_ext_trg30	reserved
adc_ext_trg31	reserved
adc_jext_trg0	tim1_trgo
adc_jext_trg1	tim1_oc4
adc_jext_trg2	tim2_trgo
adc_jext_trg3	tim2_oc1
adc_jext_trg4	tim3_oc4
adc_jext_trg5	reserved
adc_jext_trg6	exti15

Table 110. ADC interconnection (continued)

Signal name	Source/destination
adc_jext_trg7	reserved
adc_jext_trg8	tim1_trgo2
adc_jext_trg9	reserved
adc_jext_trg10	reserved
adc_jext_trg11	tim3_oc3
adc_jext_trg12	tim3_trgo
adc_jext_trg13	tim3_oc1
adc_jext_trg14	tim6_trgo
adc_jext_trg15	reserved
adc_jext_trg16	reserved
adc_jext_trg17	tim7_trgo
adc_jext_trg18	lptim1_ch1
adc_jext_trg19	lptim2_ch1
adc_jext_trg20	reserved
adc_jext_trg21	reserved
adc_jext_trg22	reserved
adc_jext_trg23	reserved
adc_jext_trg24	reserved
adc_jext_trg25	reserved
adc_jext_trg26	reserved
adc_jext_trg27	reserved
adc_jext_trg28	reserved
adc_jext_trg29	reserved
adc_jext_trg30	reserved
adc_jext_trg31	reserved

19.4.3 ADC clocks

Dual clock domain architecture

The dual clock-domain architecture means that the ADC clock is independent from the AHB bus clock.

The ADC input clock can be selected between two different clock sources (see *Figure 76: ADC clock scheme*):

1. The ADC clock can be a specific clock source (adc_ker_ck_input), independent and asynchronous with the AHB clock.

   Refer to section *Reset and clock control (RCC)* for more information on how to generate the ADC dedicated clock. To select this scheme, CKMODE[1:0] bits of ADC_CCR register must be set to 00.

2. The ADC clock can be derived from the AHB clock interface divided by a programmable factor of 1, 2 or 4. To select this scheme, CKMODE[1:0] bits of ADC_CCR must be different from 00. The programmable divider factor can be configured through to CKMODE[1:0] bits of ADC_CCR.

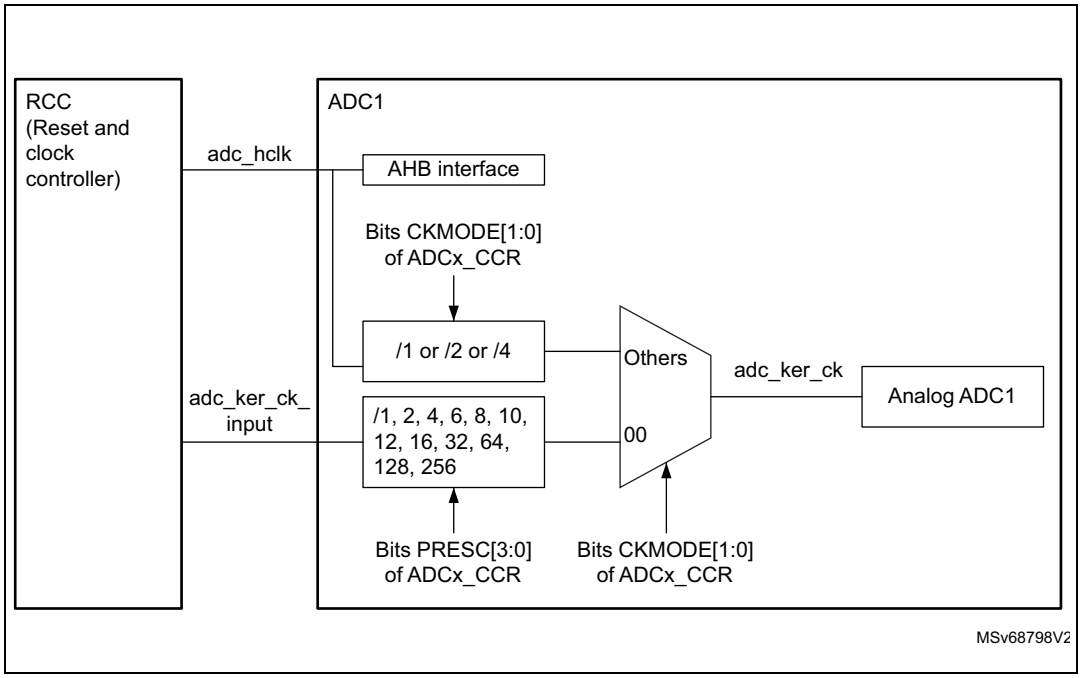
   The prescaling factor of 1 (CKMODE[1:0] = 01) can be used only if the AHB prescaler is set to 1 (HPRE[3:0] = 0xxx in the RCC_CFGR register).

Option **1** has the advantage of achieving the maximum ADC clock frequency whatever the AHB clock scheme selected. The ADC clock can eventually be divided by the following ratio: 1, 2, 4, 6, 8, 12, 16, 32, 64, 128, 256, using the prescaler configured with bits PRESC[3:0] in the ADC_CCR register.

Option **2** 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).

The clock is configured through CKMODE[1:0] bits must be compliant with the operating frequency specified in the device datasheet.

Figure 76. ADC clock scheme



Clock ratio constraint between ADC clock and AHB clock

There are generally no constraints to be respected for the ratio between the ADC clock and the AHB clock except if some injected channels are programmed. In this case, it is mandatory to respect the following ratio:

- $F_{adc\_hclk} \geq F_{ADC} / 4$ if the resolution of all channels are 12-bit or 10-bit
- $F_{adc\_hclk} \geq F_{ADC} / 3$ if there are some channels with resolutions equal to 8-bit (and none with lower resolutions)
- $F_{adc\_hclk} \geq F_{ADC} / 2$ if there are some channels with resolutions equal to 6-bit

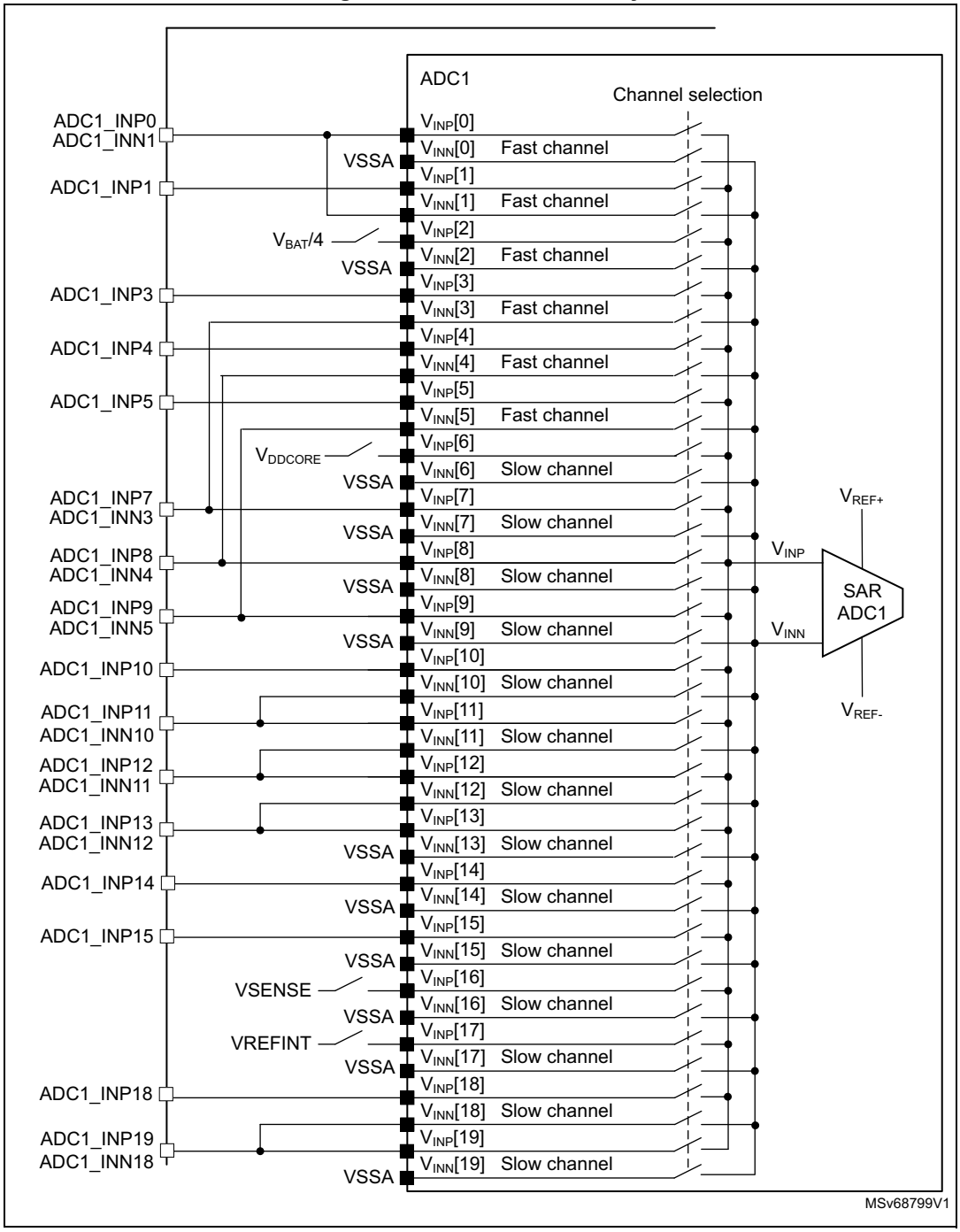
Constraints between ADC clocks

When several ADC interfaces are used simultaneously, it is mandatory to use the same clock source from the RCC block without prescaler ratio for all ADC interfaces.

19.4.4 ADC connectivity

ADC inputs are connected to the external channels as well as internal sources as described below.

Figure 77. ADC1 connectivity



19.4.5 Slave AHB interface

The ADC implements an AHB slave port for control/status register and data access. The features of the AHB interface are listed below:

- Word (32-bit) accesses
- Single cycle response
- Response to all read/write accesses to the registers with zero wait states.

The AHB slave interface does not support split/retry requests, and never generates AHB errors.

19.4.6 ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)

By default, the ADC is in Deep-power-down mode where its supply is internally switched off to reduce the leakage currents (the reset state of bit DEEPPWD is 1 in the ADC_CR register).

To start ADC operations, it is first needed to exit Deep-power-down mode by setting bit DEEPPWD = 0.

Then, it is mandatory to enable the ADC internal voltage regulator by setting the bit ADVREGEN = 1 into ADC_CR register. The software must wait for the startup time of the ADC voltage regulator ($T_{ADCVREG\_STUP}$) before launching a calibration or enabling the ADC. This delay must be implemented by software.

For the startup time of the ADC voltage regulator, refer to device datasheet for $T_{ADCVREG\_STUP}$ parameter.

When ADC operations are complete, the ADC can be disabled (ADEN = 0). It is possible to save power by also disabling the ADC voltage regulator. This is done by writing bit ADVREGEN = 0.

Then, to save more power by reducing the leakage currents, it is also possible to re-enter in ADC Deep-power-down mode by setting bit DEEPPWD = 1 into ADC_CR register. This is particularly interesting before entering Stop mode.

Note: Writing DEEPPWD = 1 automatically disables the ADC voltage regulator and bit ADVREGEN is automatically cleared.

When the internal voltage regulator is disabled (ADVREGEN = 0), the internal analog calibration is kept.

In ADC Deep-power-down mode (DEEPPWD = 1), the internal analog calibration is lost and it is necessary to either relaunch a calibration or re-apply the calibration factor which was previously saved (refer to *Section 19.4.8: Calibration (ADCAL, ADCALDIF, ADC_CALFACT)*).

19.4.7 Single-ended and differential input channels

Channels can be configured to be either single-ended input or differential input by programming DIFSEL[i] bits in the ADC_DIFSEL register. This configuration must be written while the ADC is disabled (ADEN = 0). Note that the DIFSEL[i] bits corresponding to single-ended channels are always programmed at 0.

In single-ended input mode, the analog voltage to be converted for channel “i” is the difference between the external voltage $V_{INP[i]}$ (positive input) and V_{REF-} (negative input).

In differential input mode, the analog voltage to be converted for channel “i” is the difference between the external voltage $V_{INP[i]}$ (positive input) and $V_{INN[i]}$ (negative input).

The output data for the differential mode is an unsigned data. When $V_{INP[i]}$ equals V_{REF-}, $V_{INN[i]}$ equals V_{REF+} and the output data is 0x000 (12-bit resolution mode). When $V_{INP[i]}$ equals V_{REF+}, $V_{INN[i]}$ equals V_{REF-} and the output data is 0xFFF.

$$\text{Converted value} = \frac{\text{ADC\_Full\_Scale}}{2} \times \left[1 + \frac{V_{INP} - V_{INN}}{V_{REF+}}\right]$$

When ADC is configured as differential mode, both inputs should be biased at (V_{REF+}) /2 voltage.

The input signals are supposed to be differential (common mode voltage should be fixed).

Internal channels (such as V_{REFINT} and V_{SENSE}) are used in single-ended mode only.

For a complete description of how the input channels are connected, refer to *Section 19.4.4: ADC connectivity*.

Caution: When configuring the channel “i” in differential input mode, its negative input voltage $V_{INN[i]}$ is connected to another channel. As a consequence, this channel is no longer usable in single-ended mode or in differential mode and must never be configured to be converted.

19.4.8 Calibration (ADCAL, ADCALDIF, ADC_CALFACT)

The ADC provides an automatic calibration procedure which drives all the calibration sequence including the power-on/off sequence of the ADC. During the procedure, the ADC calculates a calibration factor which is 7-bit wide and which is applied internally to the ADC until the next ADC power-off. During the calibration procedure, the application must not use the ADC and must wait until calibration is complete.

Calibration is preliminary to any ADC operation. It removes the offset error which may vary from chip to chip due to process or bandgap variation.

The calibration factor to be applied for single-ended input conversions is different from the factor to be applied for differential input conversions:

- Write ADCALDIF = 0 before launching a calibration which is applied for single-ended input conversions.
- Write ADCALDIF = 1 before launching a calibration which is applied for differential input conversions.

The calibration is then initiated by software by setting bit ADCAL = 1. Calibration can only be initiated when the ADC is disabled (when ADEN = 0). ADCAL bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon the calibration completes. At this time, the associated calibration factor is stored internally in the analog ADC and also in

the bits CALFACT_S[6:0] or CALFACT_D[6:0] of ADC_CALFACT register (depending on single-ended or differential input calibration)

The internal analog calibration is kept if the ADC is disabled (ADEN = 0). However, if the ADC is disabled for extended periods, then it is recommended that a new calibration cycle is run before re-enabling the ADC.

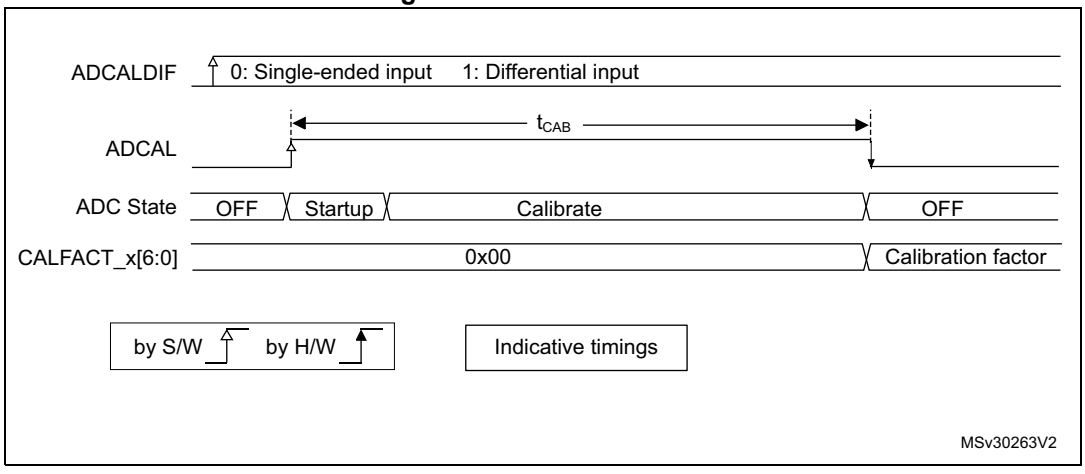
The internal analog calibration is lost each time the power of the ADC is removed (example, when the product enters in Standby or V_{BAT} mode). In this case, to avoid spending time recalibrating the ADC, it is possible to re-write the calibration factor into the ADC_CALFACT register without recalibrating, supposing that the software has previously saved the calibration factor delivered during the previous calibration.

The calibration factor can be written if the ADC is enabled but not converting (ADEN = 1 and ADSTART = 0 and JADSTART = 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. It is recommended to recalibrate when V_{REF+} voltage changed more than 10%.

Software procedure to calibrate the ADC

1. Ensure DEEPPWD = 0, ADVREGEN = 1 and that ADC voltage regulator startup time has elapsed.
2. Ensure that ADEN = 0.
3. Select the input mode for this calibration by setting ADCALDIF = 0 (single-ended input) or ADCALDIF = 1 (differential input).
4. Set ADCAL = 1.
5. Wait until ADCAL = 0.
6. The calibration factor can be read from ADC_CALFACT register.

Figure 78. ADC calibration

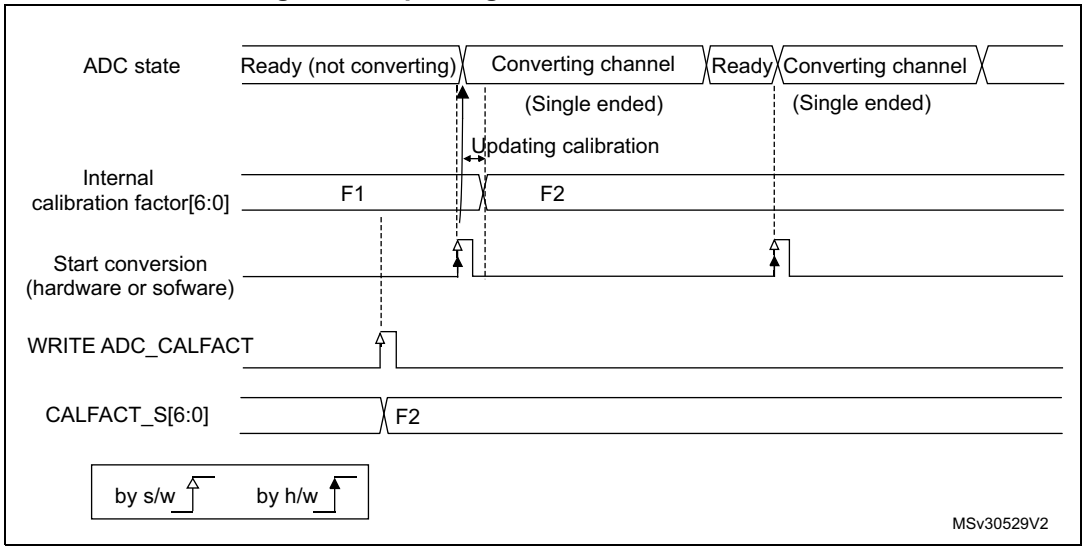


Software procedure to reinject a calibration factor into the ADC

1. Ensure ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing).
2. Write CALFACT_S and CALFACT_D with the new calibration factors.
3. When a conversion is launched, the calibration factor is injected into the analog ADC only if the internal analog calibration factor differs from the one stored in bits

CALFACT_S for single-ended input channel or bits CALFACT_D for differential input channel.

Figure 79. Updating the ADC calibration factor

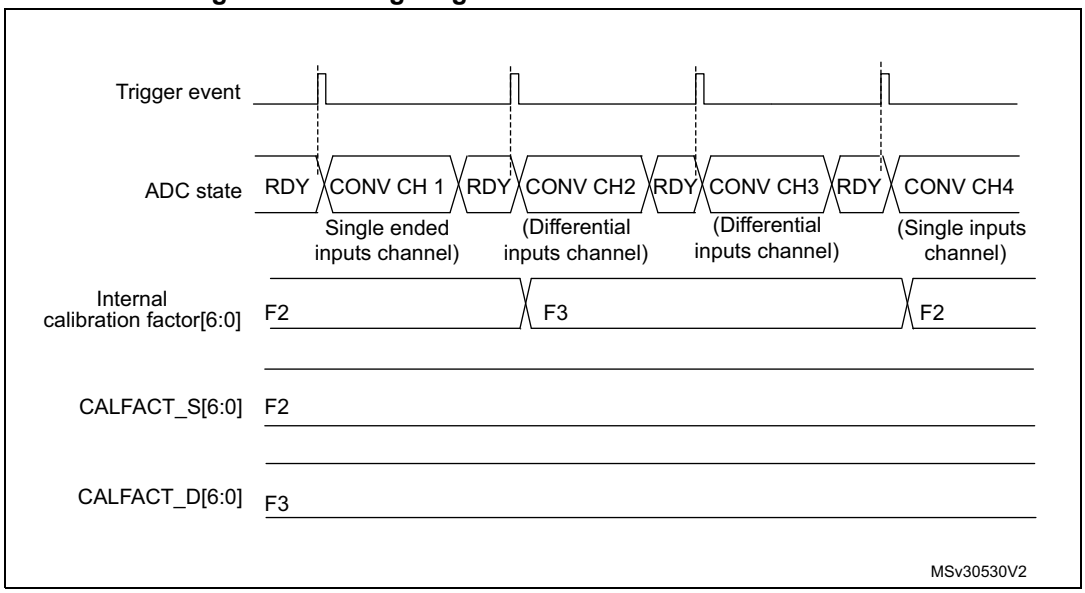


Converting single-ended and differential analog inputs with a single ADC

If the ADC is supposed to convert both differential and single-ended inputs, two calibrations must be performed, one with ADCALDIF = 0 and one with ADCALDIF = 1. The procedure is the following:

1. Disable the ADC.
2. Calibrate the ADC in single-ended input mode (with ADCALDIF = 0). This updates the register CALFACT_S[6:0].
3. Calibrate the ADC in differential input modes (with ADCALDIF = 1). This updates the register CALFACT_D[6:0].
4. Enable the ADC, configure the channels and launch the conversions. Each time there is a switch from a single-ended to a differential inputs channel (and vice-versa), the calibration is automatically injected into the analog ADC.

Figure 80. Mixing single-ended and differential channels



19.4.9 ADC on-off control (ADEN, ADDIS, ADRDY)

First of all, follow the procedure explained in *Section 19.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)*).

Once DEEPPWD = 0 and ADVREGEN = 1, the ADC can be enabled and the ADC needs a stabilization time of t_{STAB} before it starts converting accurately, as shown in *Figure 81*. Two control bits enable or disable the ADC:

- ADEN = 1 enables the ADC. The flag ADRDY is set once the ADC is ready for operation.
- ADDIS = 1 disables the ADC. ADEN and ADDIS are then automatically cleared by hardware as soon as the analog ADC is effectively disabled.

Regular conversion can then start either by setting ADSTART = 1 (refer to *Section 19.4.18: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN,JEXTSEL, JEXTEN)*) or when an external trigger event occurs, if triggers are enabled.

Injected conversions start by setting JADSTART = 1 or when an external injected trigger event occurs, if injected triggers are enabled.

Software procedure to enable the ADC

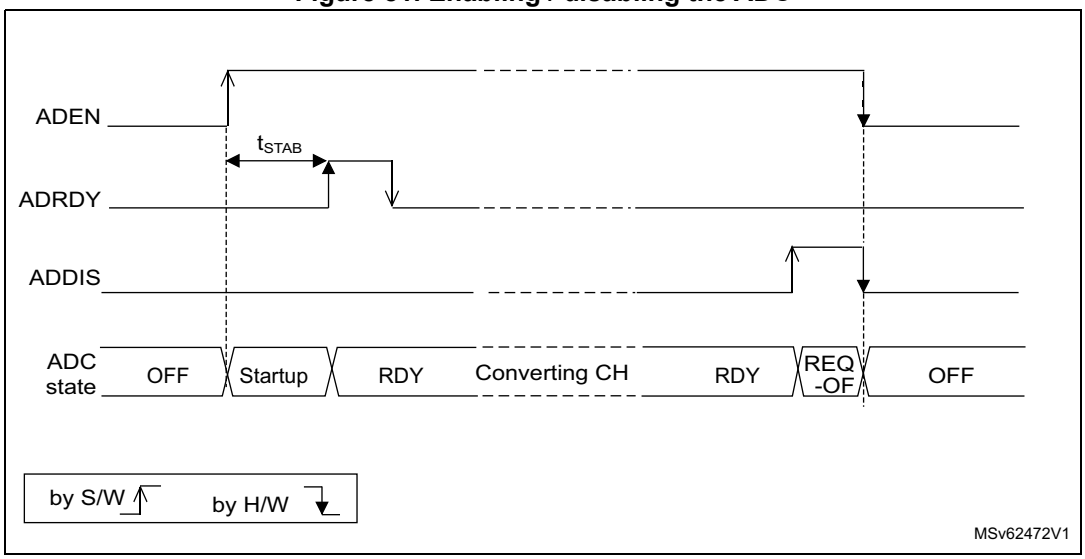
1. Clear the ADRDY bit in the ADC_ISR register by writing ‘1’.
2. Set ADEN = 1.
3. Wait until ADRDY = 1 (ADRDY is set after the ADC startup time). This can be done using the associated interrupt (setting ADRDYIE = 1).
4. Clear the ADRDY bit in the ADC_ISR register by writing ‘1’ (optional).

Caution: ADEN bit cannot be set when ADCAL is set and during four ADC clock cycles after the ADCAL bit is cleared by hardware (end of the calibration).

Software procedure to disable the ADC

1. Check that both ADSTART = 0 and JADSTART = 0 to ensure that no conversion is ongoing. If required, stop any regular and injected conversion ongoing by setting ADSTP = 1 and JADSTP = 1 and then wait until ADSTP = 0 and JADSTP = 0.
2. Set ADDIS = 1.
3. If required by the application, wait until ADEN = 0, until the analog ADC is effectively disabled (ADDIS is automatically reset once ADEN = 0).

Figure 81. Enabling / disabling the ADC



19.4.10 Constraints when writing the ADC control bits

The software is allowed to write the RCC control bits to configure and enable the ADC clock (refer to RCC Section), the DIFSEL[i] control bits in the ADC_DIFSEL register and the control bits ADCAL and ADEN in the ADC_CR register, only if the ADC is disabled (ADEN must be equal to 0).

The software is then allowed to write the control bits ADSTART, JADSTART and ADDIS of the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN must be equal to 1 and ADDIS to 0).

For all the other control bits of the ADC_CFGR, ADC_SMPRx, ADC_TRy, ADC_SQRy, ADC_JDRy, ADC_OFRy, ADC_OFCHRy and ADC_IER registers:

- For control bits related to configuration of regular conversions, the software is allowed to write them only if the ADC is enabled (ADEN = 1) and if there is no regular conversion ongoing (ADSTART must be equal to 0).
- For control bits related to configuration of injected conversions, the software is allowed to write them only if the ADC is enabled (ADEN = 1) and if there is no injected conversion ongoing (JADSTART must be equal to 0).
- ADC_TRy registers can be modified when an analog-to-digital conversion is ongoing (refer to *Section 19.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)* for details).

The software is allowed to write the ADSTP or JADSTP control bits of the ADC_CR register only if the ADC is enabled, possibly converting, and if there is no pending request to disable the ADC (ADSTART or JADSTART must be equal to 1 and ADDIS to 0).

The software can write the register ADC_JSQR at any time, when the ADC is enabled (ADEN = 1). Refer to *Section 19.6.16: ADC injected sequence register (ADC_JSQR)* for additional details.

Note: There is no hardware protection to prevent these forbidden write accesses and ADC behavior may become in an unknown state. To recover from this situation, the ADC must be disabled (clear ADEN = 0 as well as all the bits of ADC_CR register).

19.4.11 Channel selection (SQRx, JSQRx)

The ADC features up to 20 multiplexed channels, out of which:

- Up to 16 analog inputs coming from GPIO pads (ADC_INP/INN[i]) depending on the products, not all of them are available on GPIO pads.
- Up to 4 internal channels (refer to *Section 19.4.4: ADC connectivity* for details)

  To convert one of the internal analog channels, the corresponding analog sources must first be enabled by programming bits VREFEN, VBATEN or TSEN in the ADC_CCR registers.

Refer to *Table ADC interconnection* in *Section 19.4.2: ADC pins and internal signals* for the connection of the above internal analog inputs to external ADC pins or internal signals.

The conversions can be organized in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order: ADC1_INP/INN3, ADC1_INP/INN8, ADC1_INP/INN2, ADC1_INN/INP2, ADC1_INP/INN0, ADC1_INP/INN2, ADC1_INP/INN2, ADC1_INP/INN15.

- A **regular group** is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADC_SQRy registers. The total number of conversions in the regular group must be written in the L[3:0] bits in the ADC_SQR1 register.
- An **injected group** is composed of up to 4 conversions. The injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register. The total number of conversions in the injected group must be written in the L[1:0] bits in the ADC_JSQR register.

ADC_SQRy registers must not be modified while regular conversions can occur. For this, the ADC regular conversions must be first stopped by writing ADSTP = 1 (refer to *Section 19.4.17: Stopping an ongoing conversion (ADSTP, JADSTP)*).

The software is allowed to modify on-the-fly the ADC_JSQR register when JADSTART is set to 1 (injected conversions ongoing) only when the context queue is enabled (JQDIS = 0 in ADC_CFGR register). Refer to *Section 19.4.21: Queue of context for injected conversions*

19.4.12 Channel-wise programmable sampling time (SMPR1, SMPR2)

Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the embedded capacitor to the input voltage level.

Each channel can be sampled with a different sampling time which is programmable using the SMP[2:0] bits in the ADC_SMPR1 and ADC_SMPR2 registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 2.5 ADC clock cycles
- SMP = 001: 6.5 ADC clock cycles
- SMP = 010: 12.5 ADC clock cycles
- SMP = 011: 24.5 ADC clock cycles
- SMP = 100: 47.5 ADC clock cycles
- SMP = 101: 92.5 ADC clock cycles
- SMP = 110: 247.5 ADC clock cycles
- SMP = 111: 640.5 ADC clock cycles

The total conversion time is calculated as follows:

T_{CONV} = Sampling time + 12.5 ADC clock cycles

Example:

With $F_{adc\_ker\_ck}$ = 30 MHz and a sampling time of 2.5 ADC clock cycles:

T_{CONV} = (2.5 + 12.5) ADC clock cycles = 15 ADC clock cycles = 500 ns

The ADC notifies the end of the sampling phase by setting the status bit EOSMP (only for regular conversion).

Note: Depending on the ADC conversion mode, the real sampling time can vary compared to the SMP value programmed above, while the equivalent total conversion time (T_{CONV}) does not change:

- *For the first conversion in scan or continuous mode and all the conversions in discontinuous mode, the real sampling time is 0.5 clock cycle less compared to the value configured above.*
- *For the second and subsequent conversions in scan or continuous mode, 0.5 cycle is added to the configured sampling time. This additional 0.5 clock cycle overlaps with the previous conversion cycle.*

Constraints on the sampling time

For each channel, SMP[2:0] bits must be programmed to respect a minimum sampling time as specified in the ADC characteristics section of the datasheets.

Bulb sampling mode

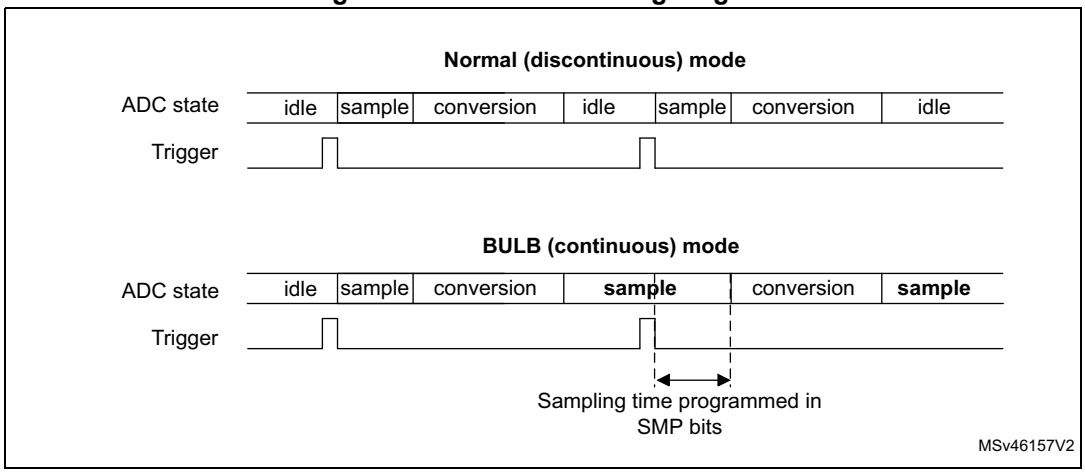
When the BULB bit is set in ADC register, the sampling period starts immediately after the last ADC conversion. A hardware or software trigger starts the conversion after the sampling time has been programmed in ADC_SMPR1 register. The very first ADC conversion, after the ADC is enabled, is performed with the sampling time programmed in SMP bits. The bulb mode is effective starting from the second conversion.

The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

The bulb mode is neither compatible with the continuous conversion mode nor with the injected channel conversion.

When the BULB bit is set, it is not allowed to set SMPTRIG bit in ADC_CFGR2.

Figure 82. Bulb mode timing diagram



Sampling time control trigger mode

When the SMPTRIG bit is set, the sampling time programmed though SMPx bits is not applicable. The sampling time is controlled by the trigger signal edge.

When a hardware trigger is selected, each rising edge of the trigger signal starts the sampling period. A falling edge ends the sampling period and starts the conversion.

When a software trigger is selected, the software trigger is not the ADSTART bit in ADC_CR but the SWTRIG bit. SWTRIG bit has to be set to start the sampling period, and the SWTRIG bit has to be cleared to end the sampling period and start the conversion.

The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

This mode is neither compatible with the continuous conversion mode, nor with the injected channel conversion.

When SMPTRIG bit is set, it is not allowed to set BULB bit.

I/O analog switch voltage booster

The resistance of the I/O analog switches 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}

by enabling an internal voltage booster through BOOSTE bit or by selecting a V_{DD} booster voltage (if V_{DD} > 2.7 V) through the ADV_READY bit of the PWR_PMCR register.

SMPPLUS control bit

The SMPPLUS bit can be used to change the sampling time from 2.5 ADC clock cycles to 3.5 ADC clock cycles.

19.4.13 Single conversion mode (CONT = 0)

In single conversion mode, the ADC performs once all the conversions of the channels. This mode is started with the CONT bit at 0 by either:

- Setting the ADSTART bit in the ADC_CR register (for a regular channel)
- Setting the JADSTART bit in the ADC_CR register (for an injected channel)
- External hardware trigger event (for a regular or injected channel)

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 16-bit ADC_DR register
- The EOC (end of regular conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

Inside the injected sequence, after each conversion is complete:

- The converted data are stored into one of the four 16-bit ADC_JDRy registers
- The JEOC (end of injected conversion) flag is set
- An interrupt is generated if the JEOCIE bit is set

After the regular sequence is complete:

- The EOS (end of regular sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

After the injected sequence is complete:

- The JEOS (end of injected sequence) flag is set
- An interrupt is generated if the JEOSIE bit is set

Then the ADC stops until a new external regular or injected trigger occurs or until bit ADSTART or JADSTART is set again.

Note: To convert a single channel, program a sequence with a length of 1.

19.4.14 Continuous conversion mode (CONT = 1)

This mode applies to regular channels only.

In continuous conversion mode, when a software or hardware regular trigger event occurs, the ADC performs once all the regular conversions of the channels and then automatically restarts and continuously converts each conversions of the sequence. This mode is started with the CONT bit at 1 either by external trigger or by setting the ADSTART bit in the ADC_CR register.

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into 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 DISCEN = 1 and CONT = 1.

Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit), refer to Auto-injection mode section).

19.4.15 Starting conversions (ADSTART, JADSTART)

Software starts ADC regular conversions by setting ADSTART = 1.

When ADSTART is set, the conversion starts:

- Immediately: if EXTEN = 0x0 (software trigger)
- At the next active edge of the selected regular hardware trigger: if EXTEN is not equal to 0x0

Software starts ADC injected conversions by setting JADSTART = 1.

When JADSTART is set, the conversion starts:

- Immediately, if JEXTEN = 0x0 (software trigger)
- At the next active edge of the selected injected hardware trigger: if JEXTEN is not equal to 0x0

Note: In auto-injection mode (JAUTO = 1), use ADSTART bit to start the regular conversions followed by the auto-injected conversions (JADSTART must be kept cleared).

ADSTART and JADSTART also provide information on whether any ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART = 0 and JADSTART = 0 are both true, indicating that the ADC is idle.

ADSTART is cleared by hardware:

- In single mode with software regular trigger (CONT = 0, EXTSEL = 0x0)
  - At any end of regular conversion sequence (EOS assertion) or at any end of subgroup processing if DISCEN = 1
- In all cases (CONT = x, EXTSEL = x)
  - After execution of the ADSTP procedure asserted by the software.

Note: In continuous mode (CONT = 1), ADSTART is not cleared by hardware with the assertion of EOS because the sequence is automatically relaunched.

When a hardware trigger is selected in single mode (CONT = 0 and EXTSEL # 0x00), ADSTART is not cleared by hardware with the assertion of EOS to help the software which does not need to reset ADSTART again for the next hardware trigger event. This ensures that no further hardware triggers are missed.

JADSTART is cleared by hardware:

- In single mode with software injected trigger (JEXTSEL = 0x0)
  - At any end of injected conversion sequence (JEOS assertion) or at any end of subgroup processing if JDISCEN = 1
- in all cases (JEXTSEL = x)
  - After execution of the JADSTP procedure asserted by the software.

Note: When the software trigger is selected, ADSTART bit should not be set if the EOC flag is still high.

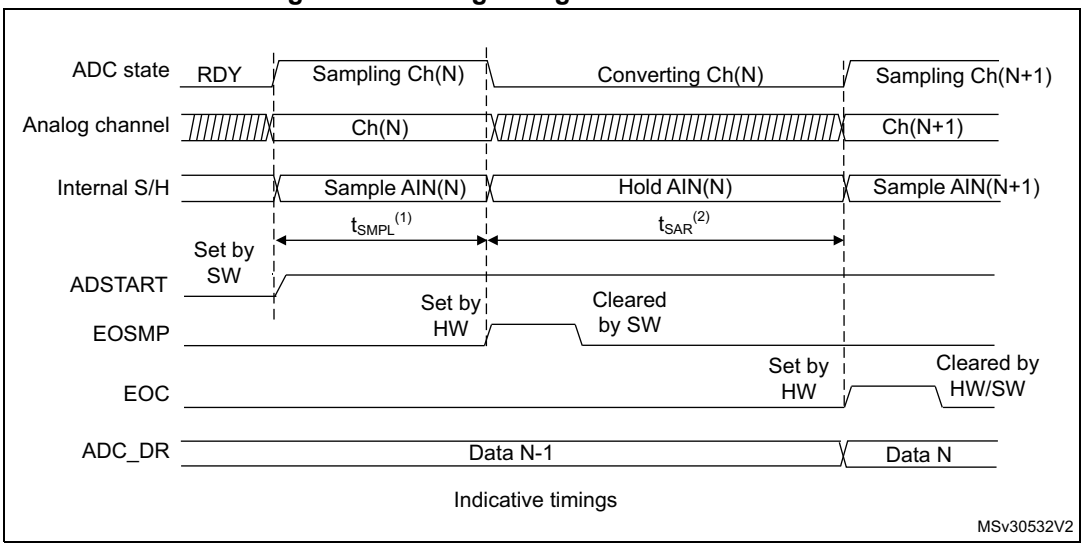
19.4.16 ADC timing

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:

$T_{CONV} = T_{SMPL} + T_{SAR} = [2.5\ _{|min} + 12.5\ _{|12bit}] \times T_{ADC\_CLK}$

$T_{CONV} = T_{SMPL} + T_{SAR} = 83.33\ ns\ _{|min} + 416.67\ ns\ _{|12bit} = 500.0\ ns$ (for $F_{ADC\_CLK} = 30$ MHz)

Figure 83. Analog-to-digital conversion time



1. T_{SMPL} depends on SMP[2:0].
2. T_{SAR} depends on RES[2:0].

19.4.17 Stopping an ongoing conversion (ADSTP, JADSTP)

The software can decide to stop regular conversions ongoing by setting ADSTP = 1 and injected conversions ongoing by setting JADSTP = 1.

Stopping conversions resets the ongoing ADC operation. Then the ADC can be reconfigured (ex: changing the channel selection or the trigger) ready for a new operation.

Note that it is possible to stop injected conversions while regular conversions are still operating and vice-versa. This allows, for instance, re-configuration of the injected conversion sequence and triggers while regular conversions are still operating (and vice-versa).

When the ADSTP bit is set by software, any ongoing regular conversion is aborted with partial result discarded (ADC_DR register is not updated with the current conversion).

When the JADSTP bit is set by software, any ongoing injected conversion is aborted with partial result discarded (ADC_JDRy register is not updated with the current conversion). The scan sequence is also aborted and reset (meaning that relaunching the ADC would restart a new sequence).

Once this procedure is complete, bits ADSTP/ADSTART (in case of regular conversion), or JADSTP/JADSTART (in case of injected conversion) are cleared by hardware and the software must poll ADSTART (or JADSTART) until the bit is reset before assuming the ADC is completely stopped.

Note: In auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (JADSTP must not be used).

Figure 84. Stopping ongoing regular conversions

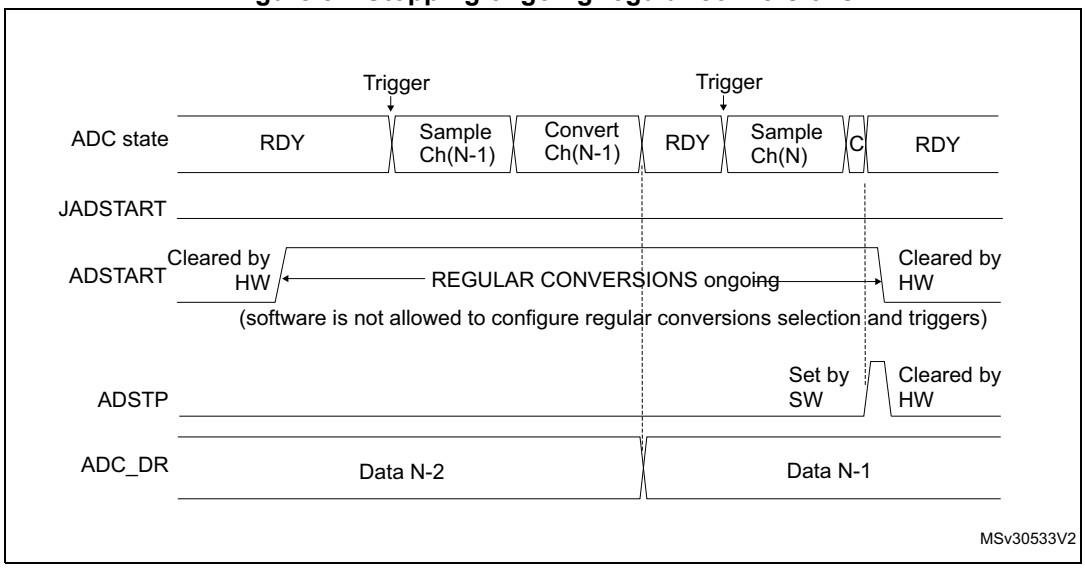
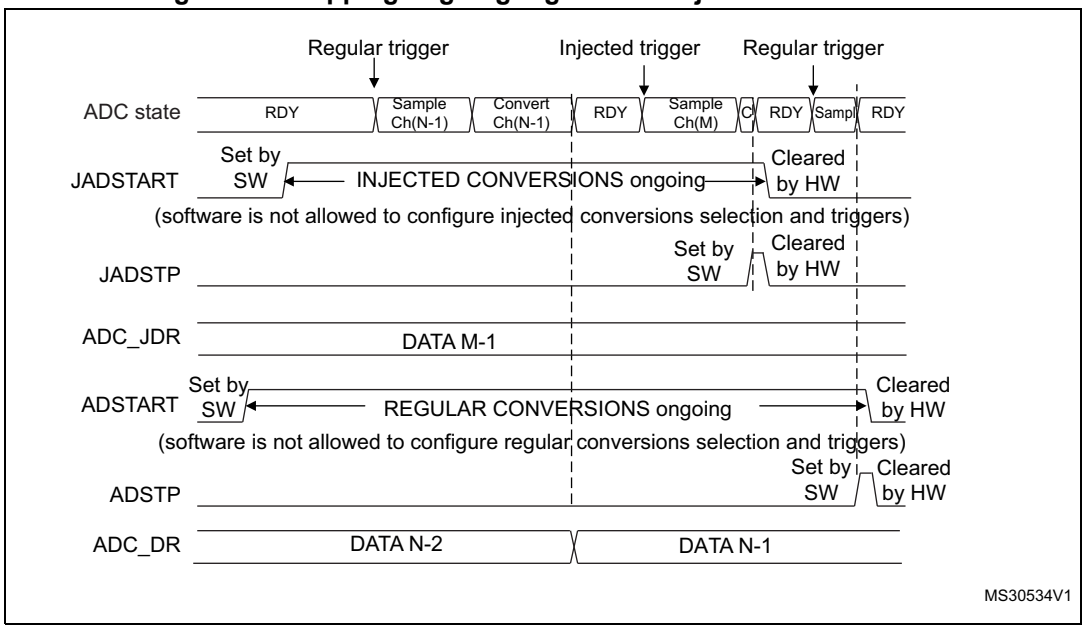


Figure 85. Stopping ongoing regular and injected conversions



19.4.18 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN,JEXTSEL, JEXTEN)

A conversion or a sequence of conversions can be triggered either by software or by an external event (such as timer capture, input pins). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, then external events are able to trigger a conversion with the selected polarity.

When the Injected Queue is enabled (bit JQDIS = 0), injected software triggers are not possible.

The regular trigger selection is effective once software has set bit ADSTART = 1 and the injected trigger selection is effective once software has set bit JADSTART = 1.

Any hardware triggers which occur while a conversion is ongoing are ignored.

- If bit ADSTART = 0, any regular hardware triggers which occur are ignored.
- If bit JADSTART = 0, any injected hardware triggers which occur are ignored.

Table 111 provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

Table 111. Configuring the trigger polarity for regular external triggers

EXTEN[1:0]	Source
00	Hardware Trigger detection disabled, software trigger detection enabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

Note: The polarity of the regular trigger cannot be changed on-the-fly.

Table 112. Configuring the trigger polarity for injected external triggers

JEXTEN[1:0]	Source
00	– If JQDIS = 1 (Queue disabled): Hardware trigger detection disabled, software trigger detection enabled – If JQDIS = 0 (Queue enabled), Hardware and software trigger detection disabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

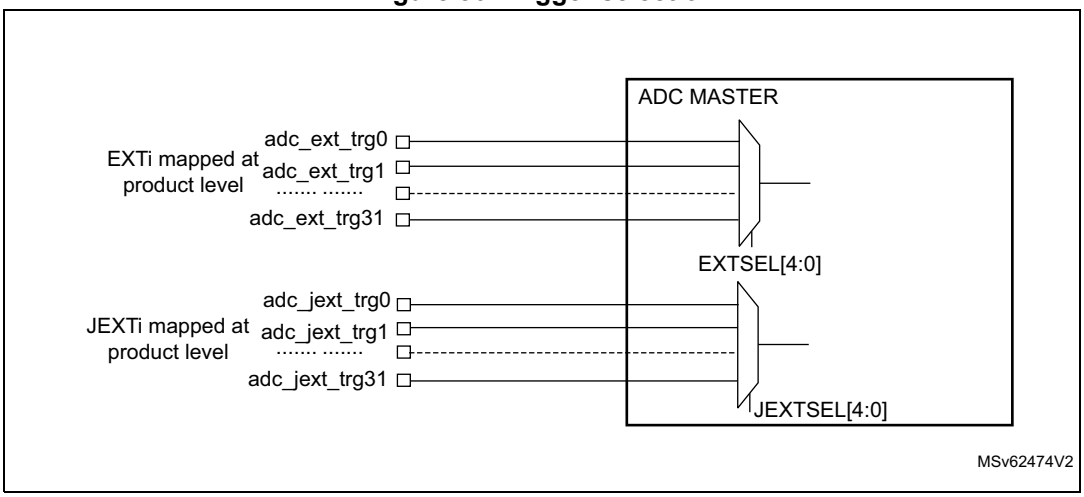
Note: The polarity of the injected trigger can be anticipated and changed on-the-fly when the queue is enabled (JQDIS = 0). Refer to Section 19.4.21: Queue of context for injected conversions.

The EXTSEL and JEXTSEL control bits select which out of 32 possible events can trigger conversion for the regular and injected groups.

A regular group conversion can be interrupted by an injected trigger.

Note: The regular trigger selection cannot be changed on-the-fly. The injected trigger selection can be anticipated and changed on-the-fly. Refer to Section 19.4.21: Queue of context for injected conversions on page 530.

Figure 86. Trigger selection



Refer to Table ADC interconnection in *Section 19.4.2: ADC pins and internal signals* for the list of all the external triggers that can be used for regular conversion.

19.4.19 Injected channel management

Triggered injection mode

To use triggered injection, the JAUTO bit in the ADC_CFGR register must be cleared.

1. Start the conversion of a group of regular channels either by an external trigger or by setting the ADSTART bit in the ADC_CR register.
2. If an external injected trigger occurs, or if the JADSTART bit in the ADC_CR register is set during the conversion of a regular group of channels, the current conversion is reset and the injected channel sequence switches are launched (all the injected channels are converted once).
3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.
4. If a regular event occurs during an injected conversion, the injected conversion is not interrupted but the regular sequence is executed at the end of the injected sequence. *Figure 87* shows the corresponding timing diagram.

Note: When using triggered injection, one must ensure that the interval between trigger events is longer than the injection sequence. For instance, if the sequence length is 30 ADC clock cycles (that is two conversions with a sampling time of 2.5 clock periods), the minimum interval between triggers must be 31 ADC clock cycles.

Auto-injection mode

If the JAUTO bit in the ADC_CFGR register is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADC_SQRy and ADC_JSQR registers.

In this mode, the ADSTART bit in the ADC_CR register must be set to start regular conversions, followed by injected conversions (JADSTART must be kept cleared). Setting the ADSTP bit aborts both regular and injected conversions (JADSTP bit must not be used).

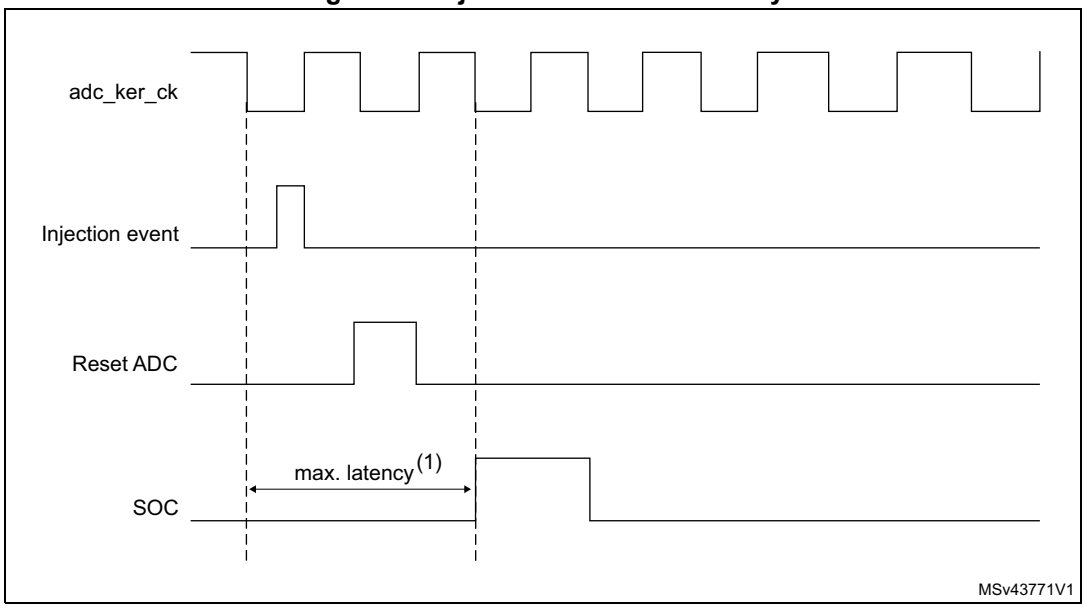
In this mode, external trigger on injected channels must be disabled.

If the CONT bit is also set in addition to the JAUTO bit, regular channels followed by injected channels are continuously converted.

Note: It is not possible to use both the auto-injected and discontinuous modes simultaneously.

When the DMA is used for exporting regular sequencer's data in JAUTO mode, it is necessary to program it in circular mode (CIRC bit set in DMA_CCRx register). If the CIRC bit is reset (single-shot mode), the JAUTO sequence is stopped upon DMA Transfer Complete event.

Figure 87. Injected conversion latency



1. The maximum latency value can be found in the electrical characteristics of the device datasheet.

19.4.20 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

Regular group mode

This mode is enabled by setting the DISCEN bit in the ADC_CFGR register.

It is used to convert a short sequence (subgroup) of n conversions (n ≤ 8) that is part of the sequence of conversions selected in the ADC_SQRy registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CFGR register.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQRy registers until all the conversions in the sequence are done. The total sequence length is defined by the L[3:0] bits in the ADC_SQR1 register.

Example:

- DISCEN = 1, n = 3, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - 1st trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - 2nd trigger: channels converted are 6, 7, 8 (an EOC event is generated at each conversion).
  - 3rd trigger: channels converted are 9, 10, 11 (an EOC event is generated at each conversion) and an EOS event is generated after the conversion of channel 11.
  - 4th trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - ...
- DISCEN = 0, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10,11
  - 1st trigger: the complete sequence is converted: channel 1, then 2, 3, 6, 7, 8, 9, 10 and 11. Each conversion generates an EOC event and the last one also generates an EOS event.
  - All the next trigger events relaunch the complete sequence.

Note: The channel numbers referred to in the above example might not be available on all microcontrollers.

When a regular group is converted in discontinuous mode, no rollover occurs (the last subgroup of the sequence can have less than n conversions).

When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the 4th trigger reconverts the channels 1, 2 and 3 in the 1st subgroup.

It is not possible to have both discontinuous mode and continuous mode enabled. In this case (if DISCEN = 1, CONT = 1), the ADC behaves as if continuous mode was disabled.

Injected group mode

This mode is enabled by setting the JDISCEN bit in the ADC_CFGR register. It converts the sequence selected in the ADC_JSQR register, channel by channel, after an external injected trigger event. This is equivalent to discontinuous mode for regular channels where ‘n’ is fixed to 1.

When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JL[1:0] bits in the ADC_JSQR register.

Example:

- JDISCEN = 1, channels to be converted = 1, 2, 3
  - 1st trigger: channel 1 converted (a JEOC event is generated)
  - 2nd trigger: channel 2 converted (a JEOC event is generated)
  - 3rd trigger: channel 3 converted and a JEOC event + a JEOS event are generated
  - ...

Note: The channel numbers referred to in the above example might not be available on all microcontrollers.

When all injected channels have been converted, the next trigger starts the conversion of the first injected channel. In the example above, the 4th trigger reconverts the 1st injected channel 1.

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

19.4.21 Queue of context for injected conversions

A queue of context is implemented to anticipate up to 2 contexts for the next injected sequence of conversions. JQDIS bit of ADC_CFGR register must be reset to enable this feature. Only hardware-triggered conversions are possible when the context queue is enabled.

This context consists of:

- Configuration of the injected triggers (bits JEXTEN[1:0] and JEXTSEL bits in ADC_JSQR register)
- Definition of the injected sequence (bits JSQx[4:0] and JL[1:0] in ADC_JSQR register)

All the parameters of the context are defined into a single register ADC_JSQR and this register implements a queue of 2 buffers, allowing the bufferization of up to 2 sets of parameters:

- The JSQR register can be written at any moment even when injected conversions are ongoing.
- Each data written into the JSQR register is stored into the Queue of context.
- At the beginning, the Queue is empty and the first write access into the JSQR register immediately changes the context and the ADC is ready to receive injected triggers.
- Once an injected sequence is complete, the Queue is consumed and the context changes according to the next JSQR parameters stored in the Queue. This new context is applied for the next injected sequence of conversions.
- A Queue overflow occurs when writing into register JSQR while the Queue is full. This overflow is signaled by the assertion of the flag JQOVF. When an overflow occurs, the write access of JSQR register which has created the overflow is ignored and the queue of context is unchanged. An interrupt can be generated if bit JQOVFIE is set.
- Two possible behaviors are possible when the Queue becomes empty, depending on the value of the control bit JQM of register ADC_CFGR:
  - If JQM = 0, the Queue is empty just after enabling the ADC, but then it can never be empty during run operations: the Queue always maintains the last active context and any further valid start of injected sequence is served according to the last active context.
  - If JQM = 1, the Queue can be empty after the end of an injected sequence or if the Queue is flushed. When this occurs, there is no more context in the queue and hardware triggers are disabled. Therefore, any further hardware injected triggers are ignored until the software re-writes a new injected context into JSQR register.
- Reading JSQR register returns the current JSQR context which is active at that moment. When the JSQR context is empty, JSQR is read as 0x0000.
- The Queue is flushed when stopping injected conversions by setting JADSTP = 1 or when disabling the ADC by setting ADDIS = 1:
  - If JQM = 0, the Queue is maintained with the last active context.
  - If JQM = 1, the Queue becomes empty and triggers are ignored.

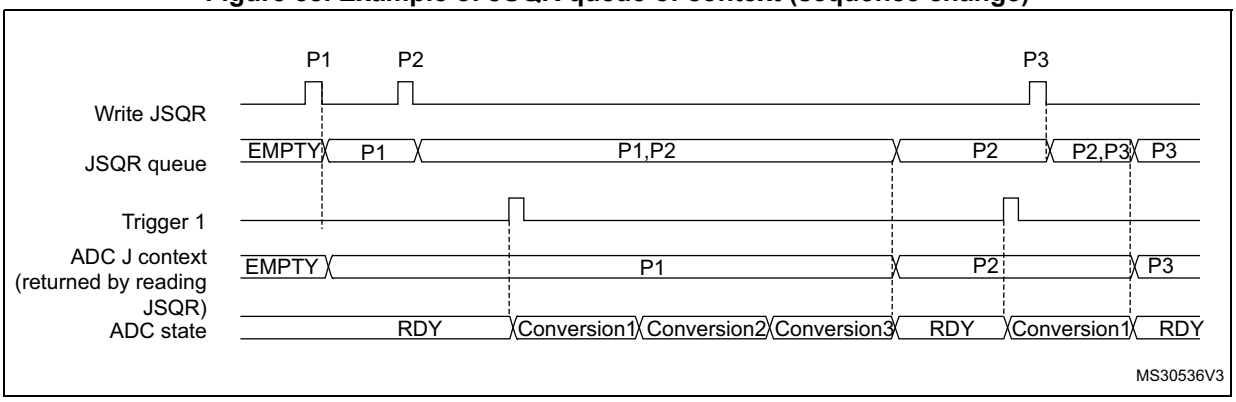
Note: When configured in discontinuous mode (bit JDISCEN = 1), only the last trigger of the injected sequence changes the context and consumes the Queue.The 1st trigger only consumes the queue but others are still valid triggers as shown by the discontinuous mode example below (length = 3 for both contexts):

- *1st trigger, discontinuous. Sequence 1: context 1 consumed, 1st conversion carried out*
- *2nd trigger, disc. Sequence 1: 2nd conversion.*
- *3rd trigger, discontinuous. Sequence 1: 3rd conversion.*
- *4th trigger, discontinuous. Sequence 2: context 2 consumed, 1st conversion carried out.*
- *5th trigger, discontinuous. Sequence 2: 2nd conversion.*
- *6th trigger, discontinuous. Sequence 2: 3rd conversion.*

Behavior when changing the trigger or sequence context

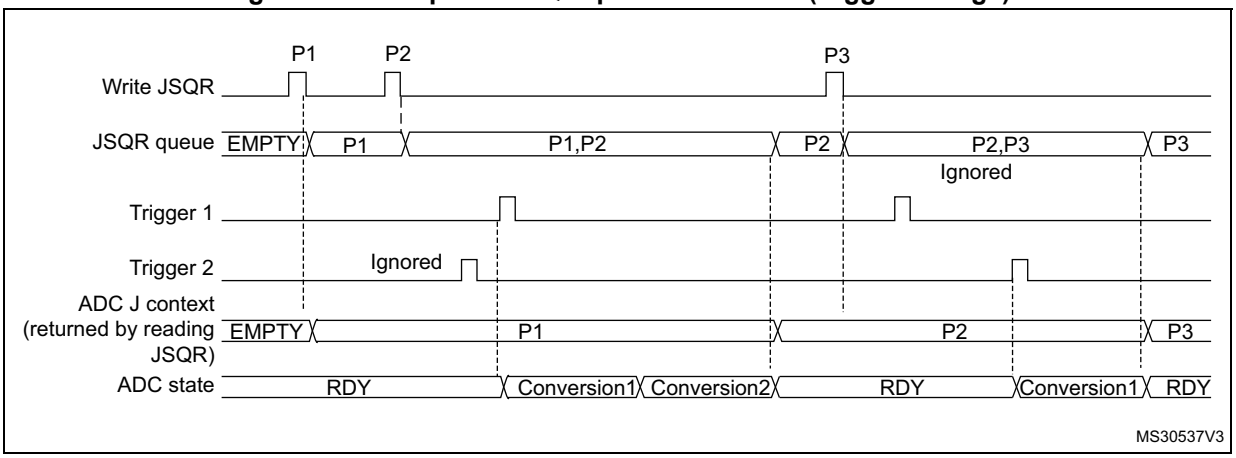
Figure 88 and *Figure 89* show the behavior of the context Queue when changing the sequence or the triggers.

Figure 88. Example of JSQR queue of context (sequence change)



1. Parameters:
   P1: sequence of 3 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 4 conversions, hardware trigger 1

Figure 89. Example of JSQR queue of context (trigger change)

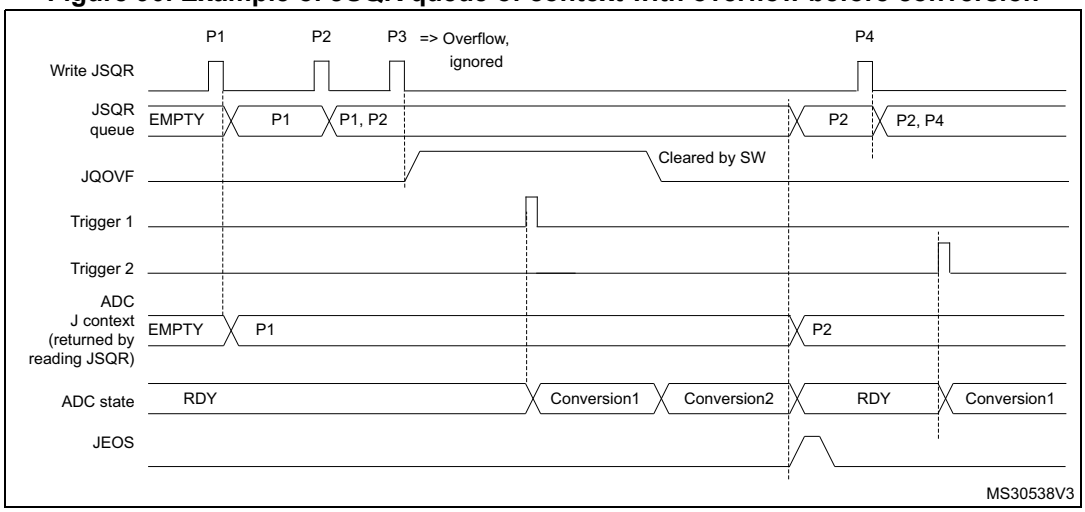


1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 4 conversions, hardware trigger 1

Queue of context: Behavior when a queue overflow occurs

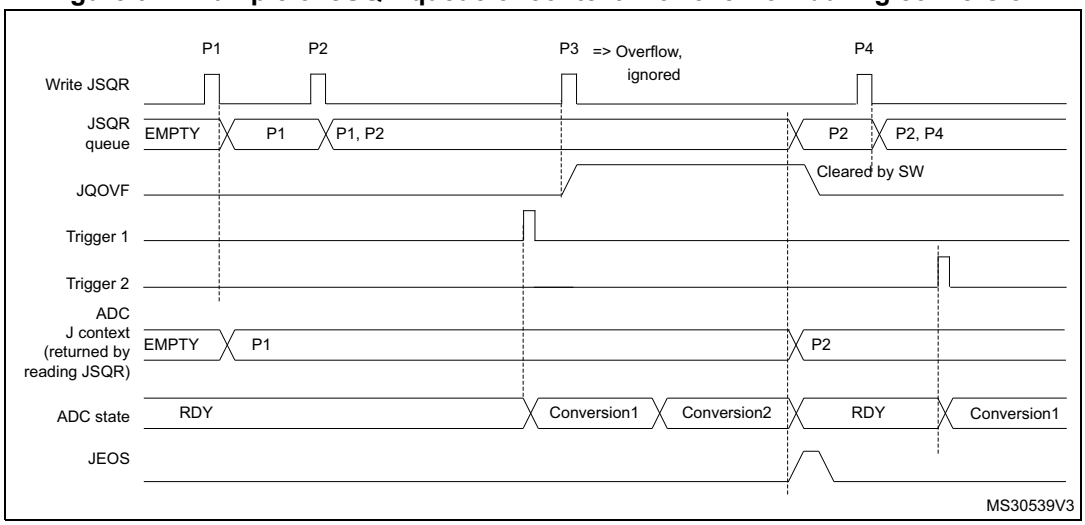
The *Figure 90* and *Figure 91* show the behavior of the context Queue if an overflow occurs before or during a conversion.

Figure 90. Example of JSQR queue of context with overflow before conversion



1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 3 conversions, hardware trigger 1
   P4: sequence of 4 conversions, hardware trigger 1

Figure 91. Example of JSQR queue of context with overflow during conversion



1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 3 conversions, hardware trigger 1
   P4: sequence of 4 conversions, hardware trigger 1

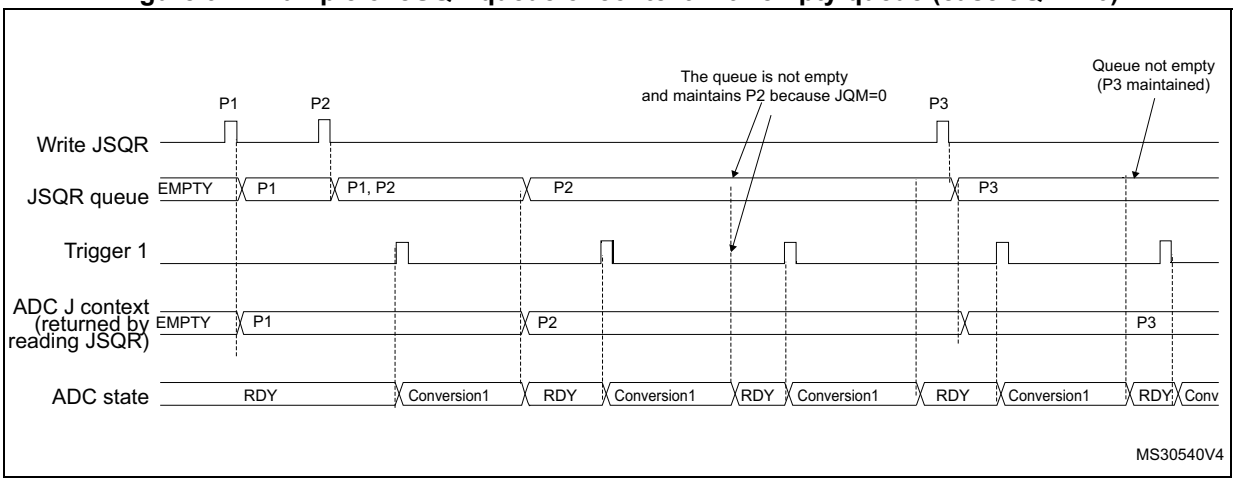
It is recommended to manage the queue overflows as described below:

- After each P context write into JSQR register, flag JQOVF shows if the write has been ignored or not (an interrupt can be generated).
- Avoid Queue overflows by writing the third context (P3) only once the flag JEOS of the previous context P2 has been set. This ensures that the previous context has been consumed and that the queue is not full.

Queue of context: Behavior when the queue becomes empty

Figure 92 and *Figure 93* show the behavior of the context Queue when the Queue becomes empty in both cases JQM = 0 or 1.

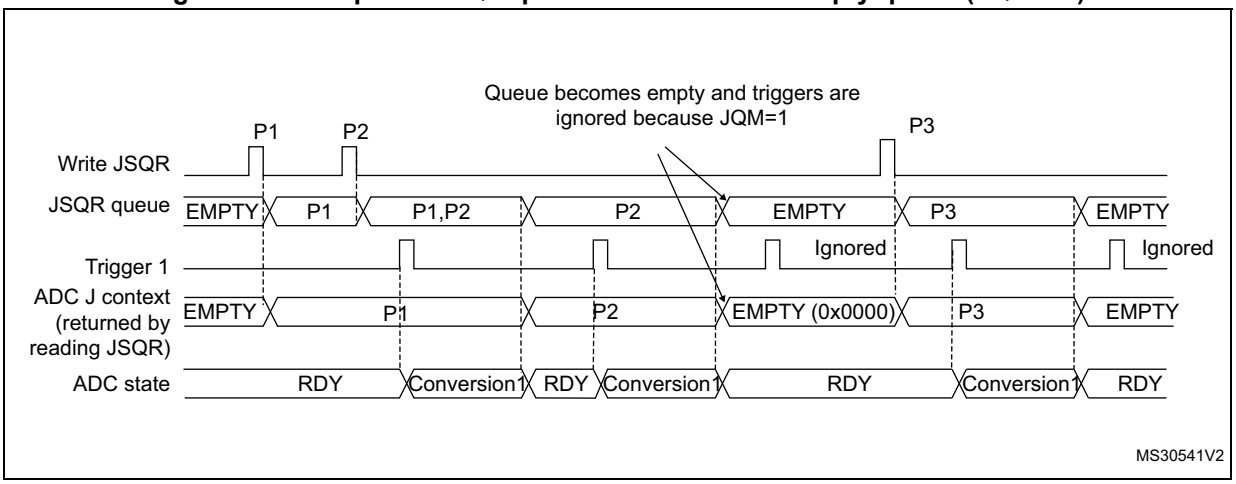
Figure 92. Example of JSQR queue of context with empty queue (case JQM = 0)



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Note: When writing P3, the context changes immediately. However, because of internal resynchronization, there is a latency and if a trigger occurs just after or before writing P3, it can happen that the conversion is launched considering the context P2. To avoid this situation, the user must ensure that there is no ADC trigger happening when writing a new context that applies immediately.

Figure 93. Example of JSQR queue of context with empty queue (JQM = 1)

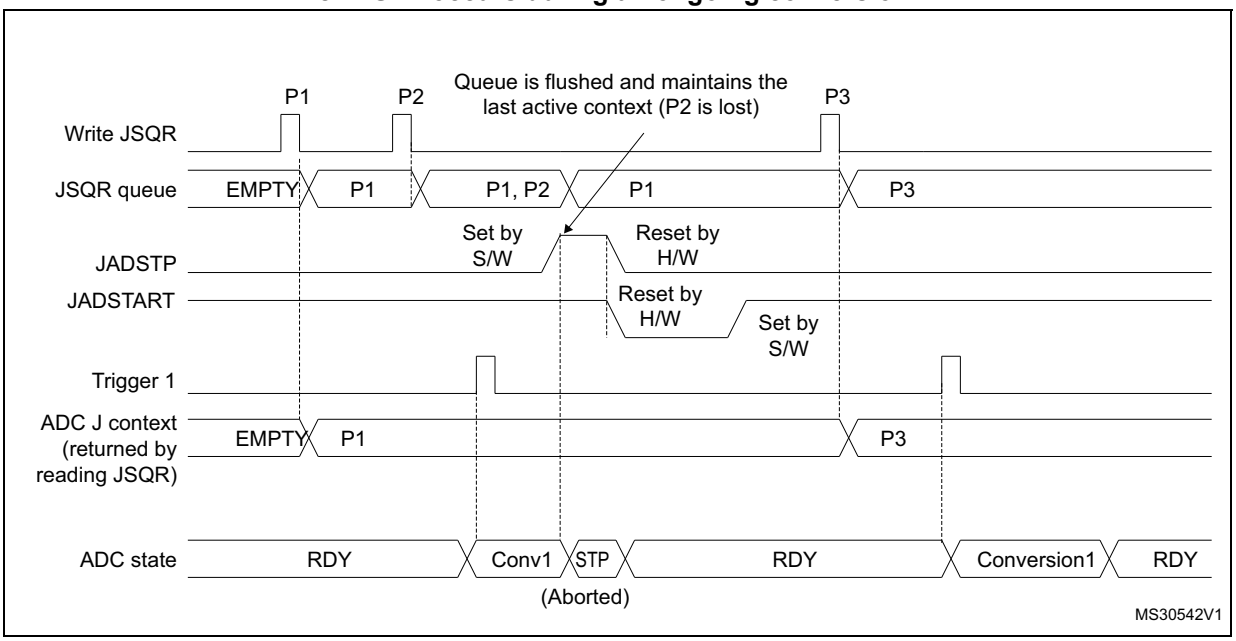


1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Flushing the queue of context

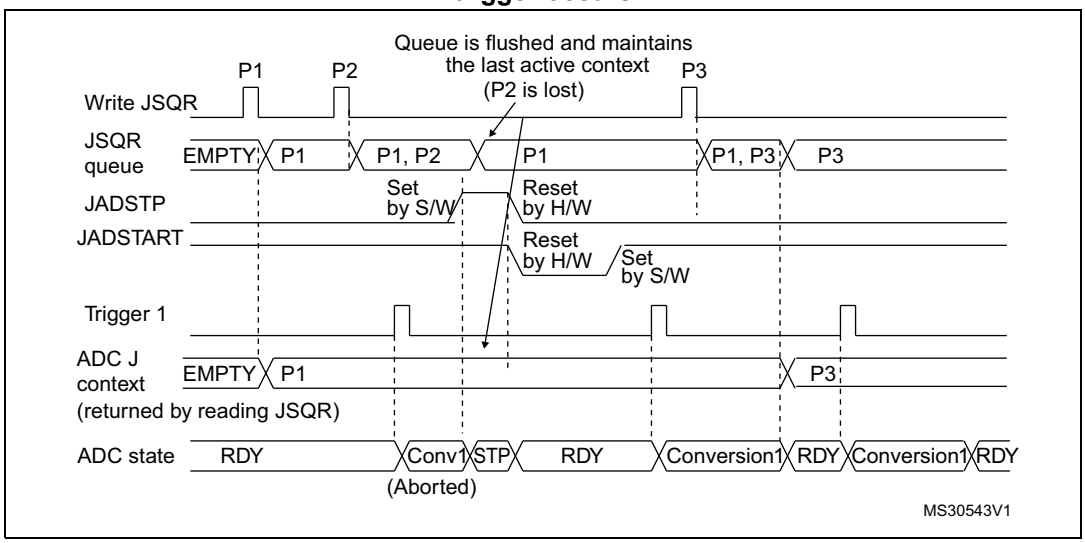
The figures below show the behavior of the context Queue in various situations when the queue is flushed.

Figure 94. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion.



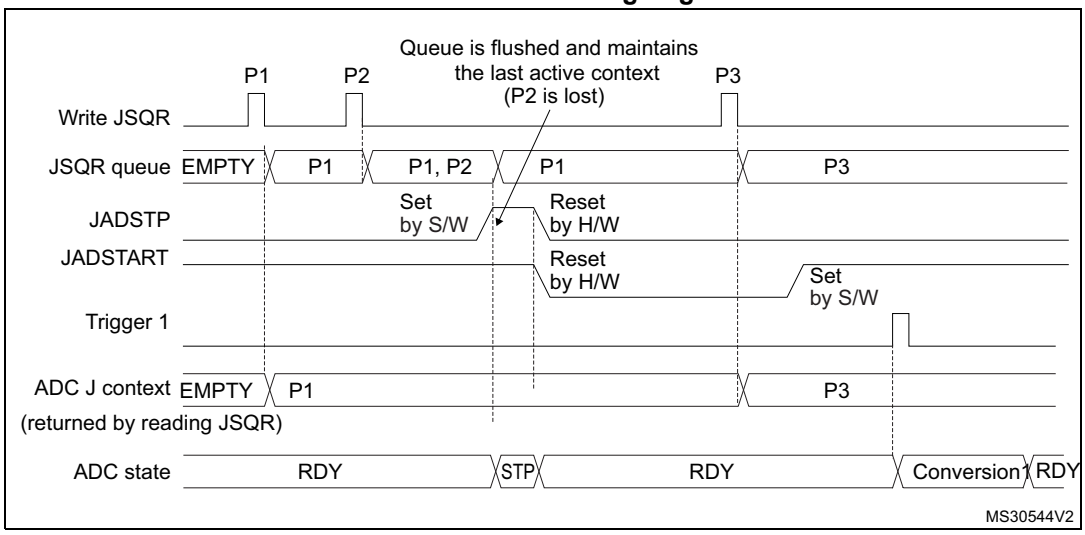
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 95. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion and a new trigger occurs



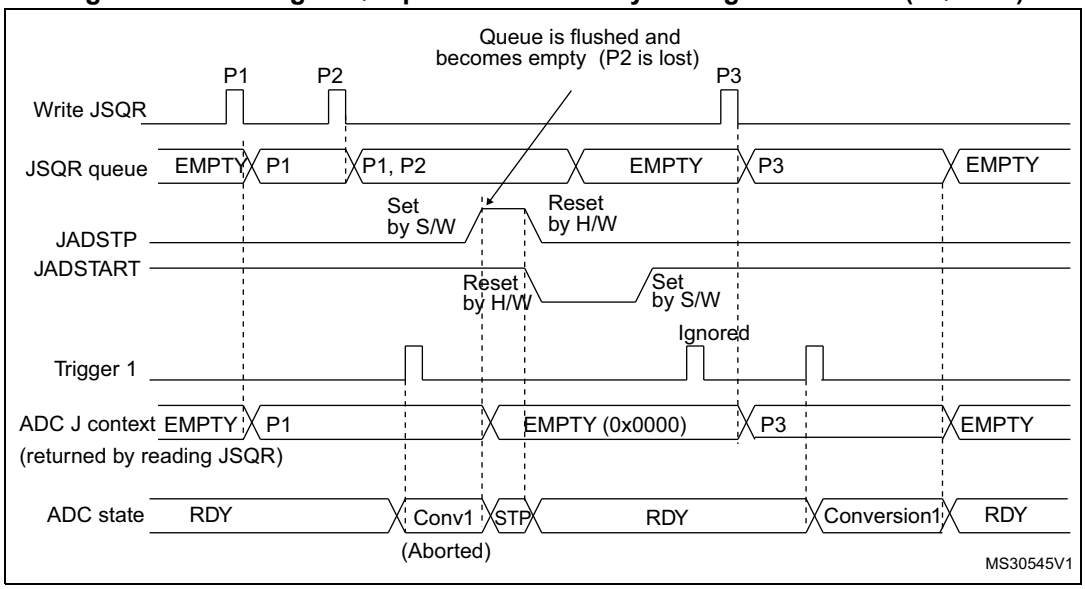
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 96. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs outside an ongoing conversion



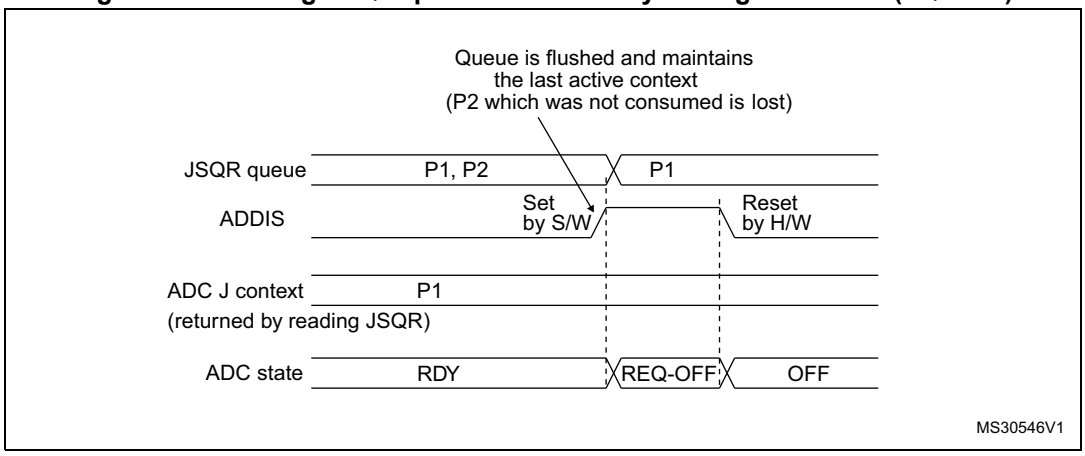
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 97. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 1)



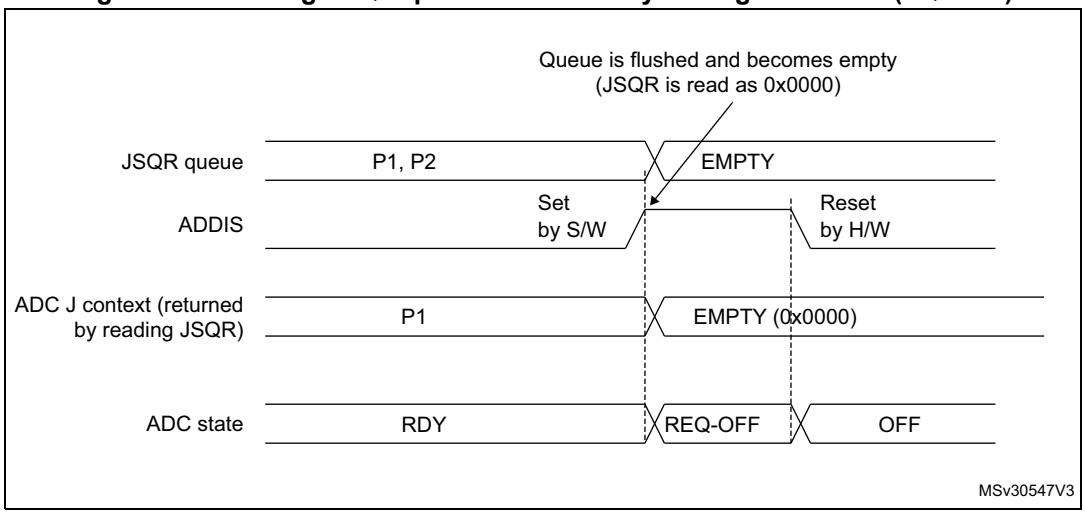
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 98. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 0)



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 99. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 1)



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Queue of context: Starting the ADC with an empty queue

The following procedure must be followed to start ADC operation with an empty queue, in case the first context is not known at the time the ADC is initialized. This procedure is only applicable when JQM bit is reset:

5. Write a dummy JSQR with JEXTEN not equal to 0 (otherwise triggering a software conversion)
6. Set JADSTART
7. Set JADSTP
8. Wait until JADSTART is reset
9. Set JADSTART.

Disabling the queue

It is possible to disable the queue by setting bit JQDIS = 1 into the ADC_CFGR register.

19.4.22 Programmable resolution (RES) - fast conversion mode

It is possible to perform faster conversion by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the control bits RES[1:0]. *Figure 104*, *Figure 105*, *Figure 106* and *Figure 107* show the conversion result format with respect to the resolution as well as to the data alignment.

Lower resolution allows faster conversion time for applications where high-data precision is not required. It reduces the conversion time spent by the successive approximation steps according to *Table 113*.

Table 113. T_{SAR} timings depending on resolution

RES (bits)	T_{SAR} (ADC clock cycles)	T_{SAR} (ns) at F_{ADC} = 30 MHz	T_{CONV} (ADC clock cycles) (with Sampling Time = 2.5 ADC clock cycles)	T_{CONV} (ns) at F_{ADC} = 30 MHz
12	12.5 ADC clock cycles	416.67 ns	15 ADC clock cycles	500.0 ns
10	10.5 ADC clock cycles	350.0 ns	13 ADC clock cycles	433.33 ns
8	8.5 ADC clock cycles	203.33 ns	11 ADC clock cycles	366.67 ns
6	6.5 ADC clock cycles	216.67 ns	9 ADC clock cycles	300.0 ns

19.4.23 End of conversion, end of sampling phase (EOC, JEOC, EOSMP)

The ADC notifies the application for each end of regular conversion (EOC) event and each injected conversion (JEOC) event.

The ADC sets the EOC flag as soon as a new regular conversion data is available in the ADC_DR register. An interrupt can be generated if bit EOCIE is set. EOC flag is cleared by the software either by writing 1 to it or by reading ADC_DR.

The ADC sets the JEOC flag as soon as a new injected conversion data is available in one of the ADC_JDRy register. An interrupt can be generated if bit JEOCIE is set. JEOC flag is cleared by the software either by writing 1 to it or by reading the corresponding ADC_JDRy register.

The ADC also notifies the end of Sampling phase by setting the status bit EOSMP (for regular conversions only). EOSMP flag is cleared by software by writing 1 to it. An interrupt can be generated if bit EOSMPIE is set.

19.4.24 End of conversion sequence (EOS, JEOS)

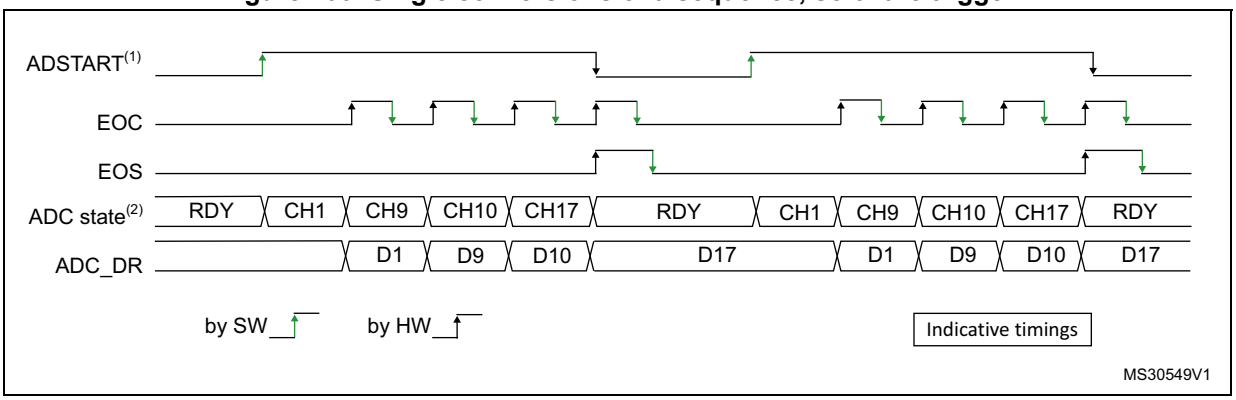
The ADC notifies the application for each end of regular sequence (EOS) and for each end of injected sequence (JEOS) event.

The ADC sets the EOS flag as soon as the last data of the regular conversion sequence is available in the ADC_DR register. An interrupt can be generated if bit EOSIE is set. EOS flag is cleared by the software either by writing 1 to it.

The ADC sets the JEOS flag as soon as the last data of the injected conversion sequence is complete. An interrupt can be generated if bit JEOSIE is set. JEOS flag is cleared by the software either by writing 1 to it.

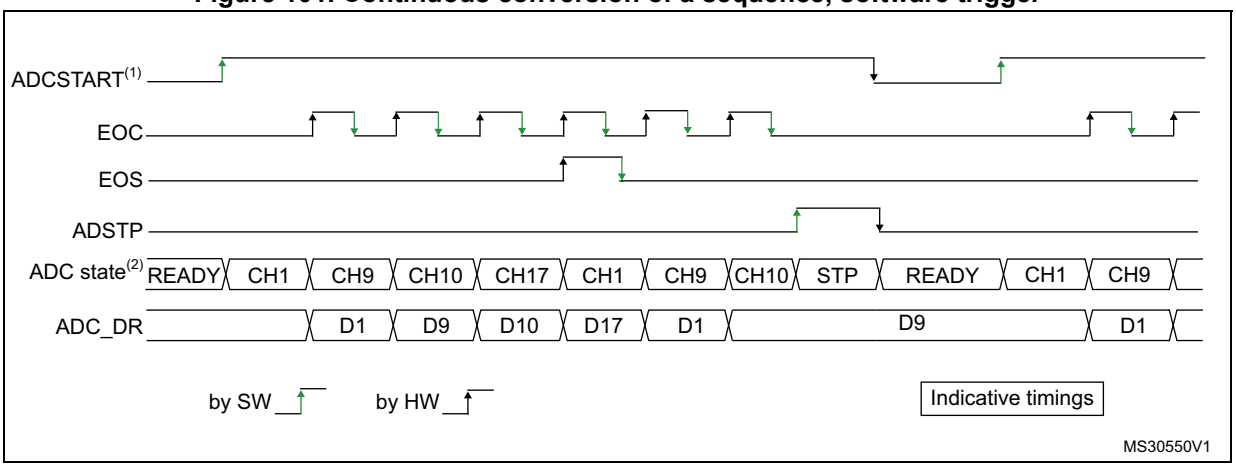
19.4.25 Timing diagrams example (single/continuous modes, hardware/software triggers)

Figure 100. Single conversions of a sequence, software trigger



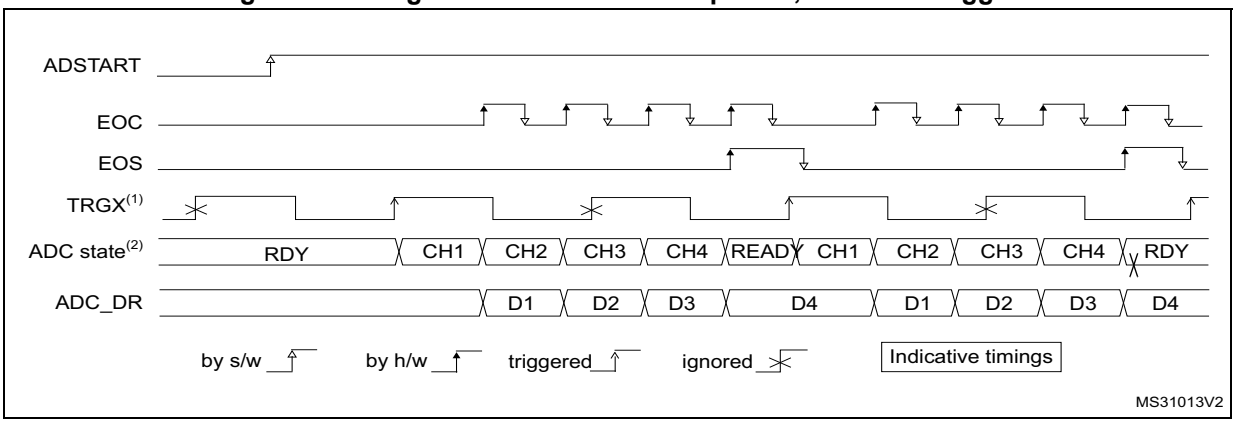
1. EXTEN = 0x0, CONT = 0
2. Channels selected = 1,9, 10, 17; AUTDLY = 0.

Figure 101. Continuous conversion of a sequence, software trigger



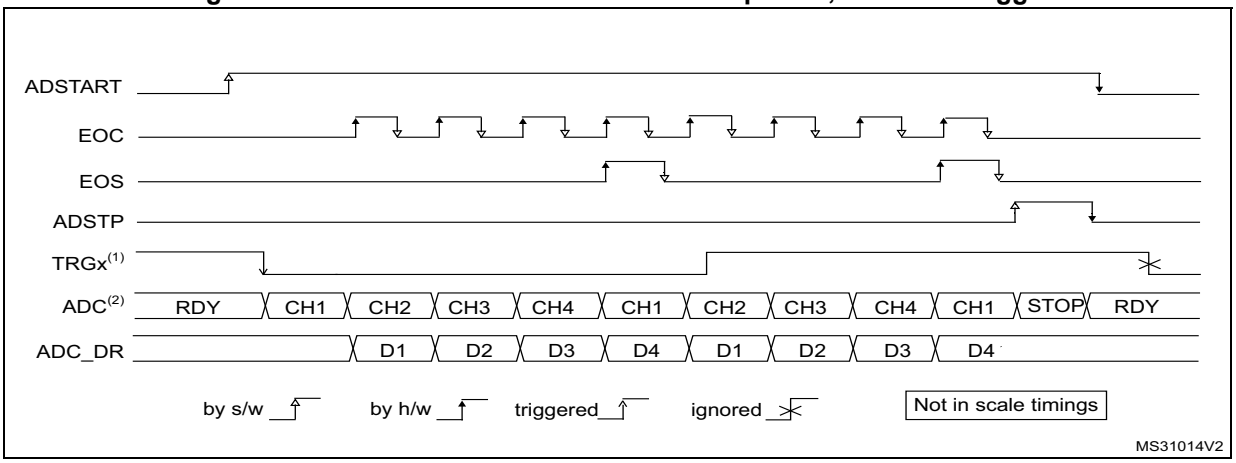
1. EXTEN = 0x0, CONT = 1
2. Channels selected = 1,9, 10, 17; AUTDLY = 0.

Figure 102. Single conversions of a sequence, hardware trigger



1. TRGx (over-frequency) is selected as trigger source, EXTEN = 01, CONT = 0
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

Figure 103. Continuous conversions of a sequence, hardware trigger



1. TRGx is selected as trigger source, EXTEN = 10, CONT = 1
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

19.4.26 Data management

Data register, data alignment and offset (ADC_DR, OFFSET, OFFSET_CH, ALIGN)

Data and alignment

At the end of each regular conversion channel (when EOC event occurs), the result of the converted data is stored into the ADC_DR data register which is 16 bits wide.

At the end of each injected conversion channel (when JEOC event occurs), the result of the converted data is stored into the corresponding ADC_JDRy data register which is 16 bits wide.

The ALIGN bit in the ADC_CFGR register selects the alignment of the data stored after conversion. Data can be right- or left-aligned as shown in *Figure 104*, *Figure 105*, *Figure 106* and *Figure 107*.

Special case: when left-aligned, the data are aligned on a half-word basis except when the resolution is set to 6-bit. In that case, the data are aligned on a byte basis as shown in *Figure 106* and *Figure 107*.

Note: Left-alignment is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the ALIGN bit value is ignored and the ADC only provides right-aligned data.

Offset

An offset y (y = 1,2,3,4) can be applied to a channel by setting the bit OFFSET_EN = 1 into ADC_OFRy register. The channel to which the offset is to be applied is programmed into the bits OFFSET_CH[4:0] of ADC_OFRy register. In this case, the converted value is decreased by the user-defined offset written in the bits OFFSET[11:0]. The result may be a negative value so the read data is signed and the SEXT bit represents the extended sign value.

Note: Offset correction is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the value of the OFFSET_EN bit in ADC_OFRy register is ignored (considered as reset).

Table 116 describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

Table 114. Offset computation versus data resolution

Resolution (bits RES[1:0])	Subtraction between raw converted data and offset		Result	Comments
	Raw converted Data, left aligned	Offset		
00: 12-bit	DATA[11:0]	OFFSET[11:0]	Signed 12-bit data	-
01: 10-bit	DATA[11:2],00	OFFSET[11:0]	Signed 10-bit data	The user must configure OFFSET[1:0] to “00”

Table 114. Offset computation versus data resolution (continued)

Resolution (bits RES[1:0])	Subtraction between raw converted data and offset		Result	Comments
	Raw converted Data, left aligned	Offset		
10: 8-bit	DATA[11:4],0000	OFFSET[11:0]	Signed 8-bit data	The user must configure OFFSET[3:0] to “0000”
11: 6-bit	DATA[11:6],000000	OFFSET[11:0]	Signed 6-bit data	The user must configure OFFSET[5:0] to “000000”

When reading data from ADC_DR (regular channel) or from ADC_JDRy (injected channel, y = 1,2,3,4) corresponding to the channel “i”:

- If one of the offsets is enabled (bit OFFSET_EN = 1) for the corresponding channel, the read data is signed.
- If none of the four offsets is enabled for this channel, the read data is not signed.

Figure 104, *Figure 105*, *Figure 106* and *Figure 107* show alignments for signed and unsigned data.

Figure 104. Right alignment (offset disabled, unsigned value)

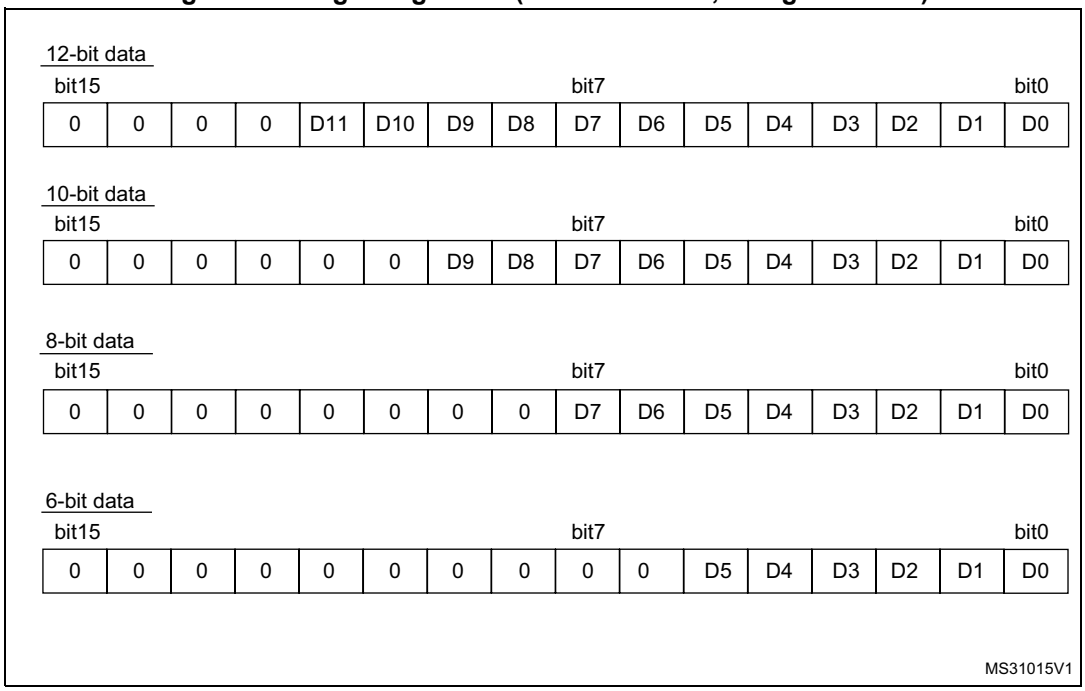


Figure 105. Right alignment (offset enabled, signed value)

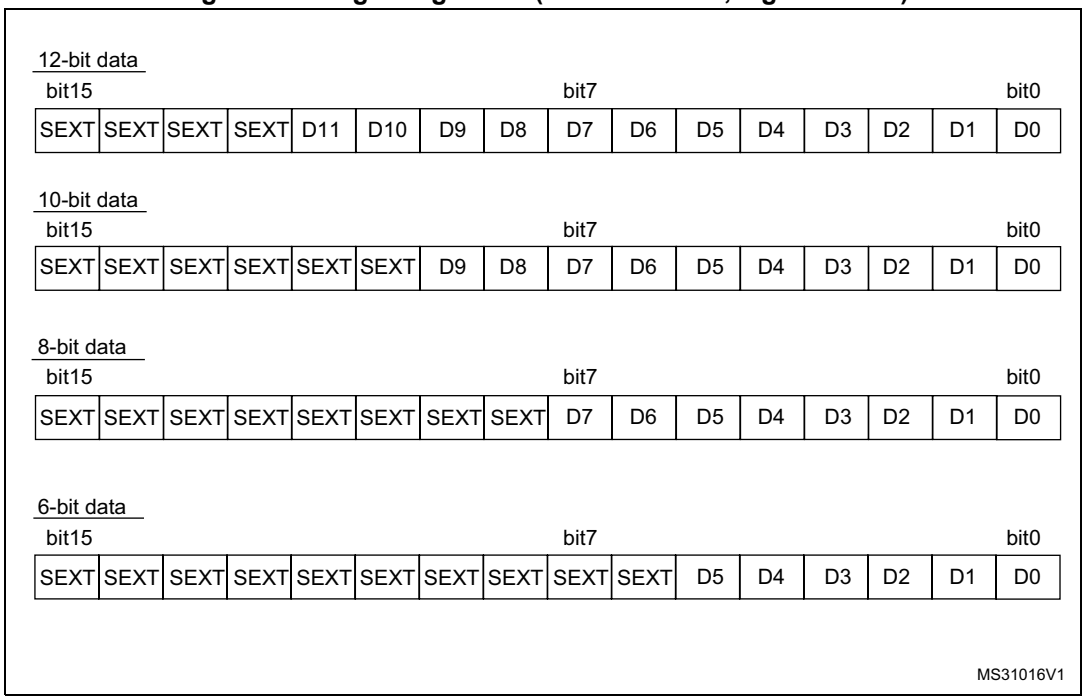


Figure 106. Left alignment (offset disabled, unsigned value)

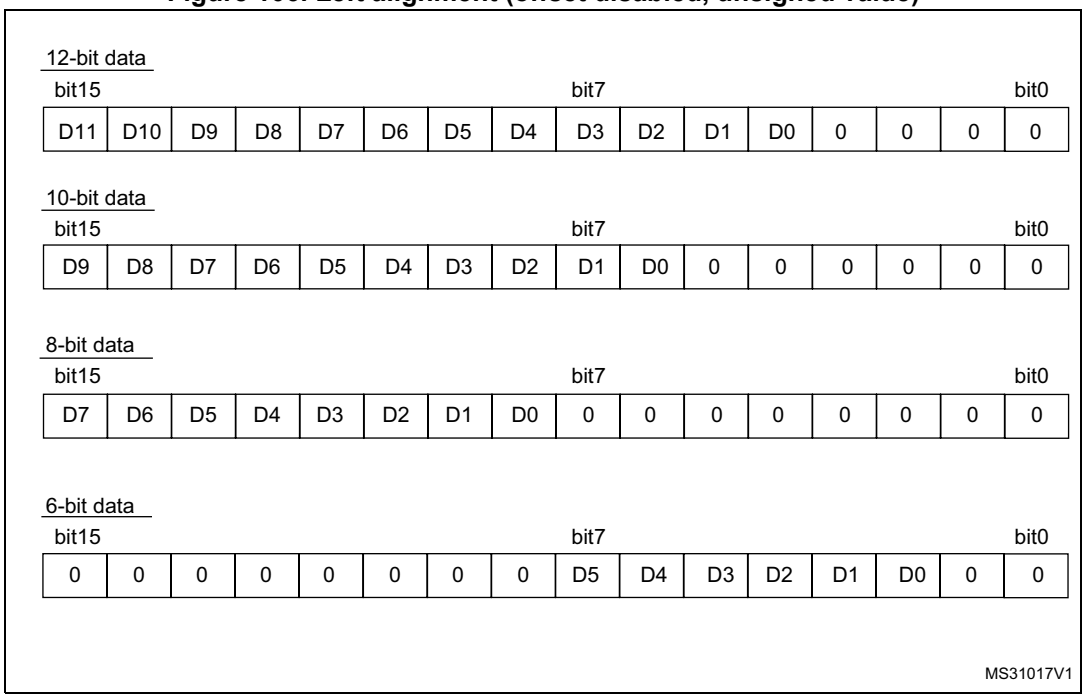
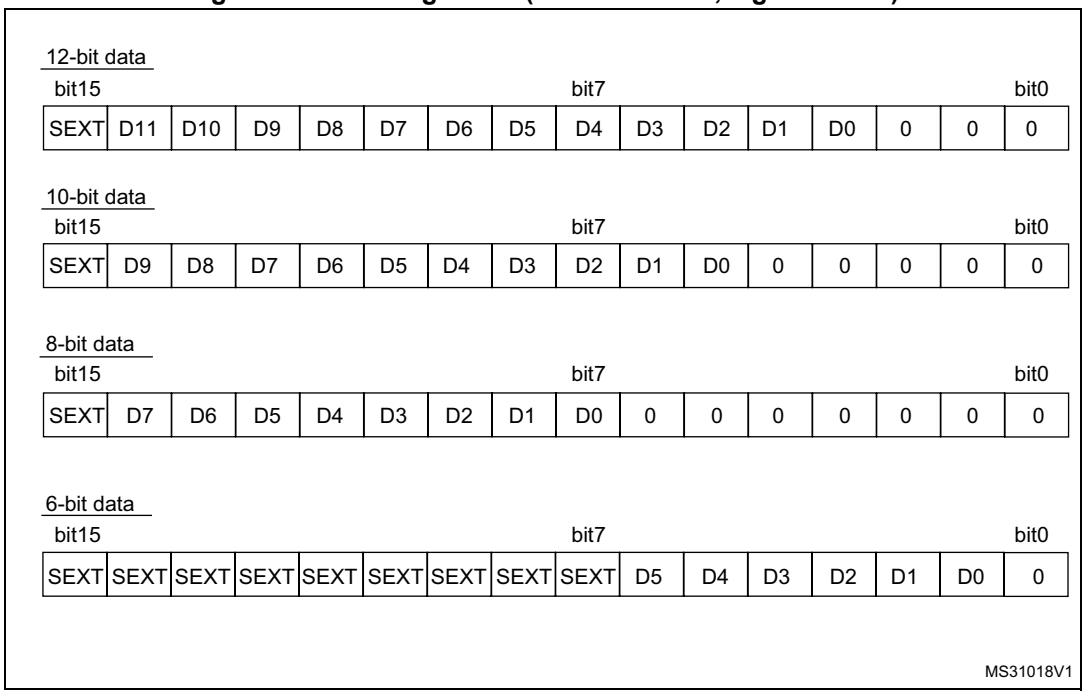


Figure 107. Left alignment (offset enabled, signed value)



Offset compensation

When SATEN bit is set in ADC_OFRy register during offset operation, data are unsigned. All the offset data saturate at 0x000 (in 12-bit mode). When OFFSETPOS bit is set, the offset direction is positive and the data saturate at 0xFFF (in 12-bit mode). In 8-bit mode, data saturate at 0x00 and 0xFF, respectively.

The analog watchdog comparison is performed before the offset compensation.

ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) notifies when the regular converted data has not been read (by the CPU or the DMA) before ADC_DR FIFO (three stages) is overflowed.

The OVR flag is set when a new conversion completes while ADC_CR register FIFO was full. An interrupt is generated if OVRIE bit is set to 1.

When an overrun condition occurs, the ADC is still operating and can continue converting unless the software decides to stop and reset the sequence by setting ADSTP to 1. Since ADC_DR FIFO features three stages, up to three data are stored in the FIFO.

OVR flag is cleared by software by writing 1 to it.

It is possible to configure if data is preserved or overwritten when an overrun event occurs by programming the control bit OVRMOD:

- OVRMOD = 0: The overrun event preserves the data register from being overwritten: the old data is maintained up to ADC_DR FIFO depth (three stages) and the new conversion is discarded and lost. In this mode, ADC_DR FIFO is disabled. If the FIFO is full, any further conversion is performed but the resulting data is also discarded. EOC

is cleared by reading ADC_DR register. However, the FIFO can still contain previously converted data.

- OVRMOD = 1: The data register is overwritten with the last conversion result and the previous unread data is lost. In this mode, ADC_DR FIFO is disabled. If OVR remains at 1, any further conversions is performed normally and the ADC_DR register always contains the latest converted data.

Figure 108. Example of overrun (OVRMOD = 0)

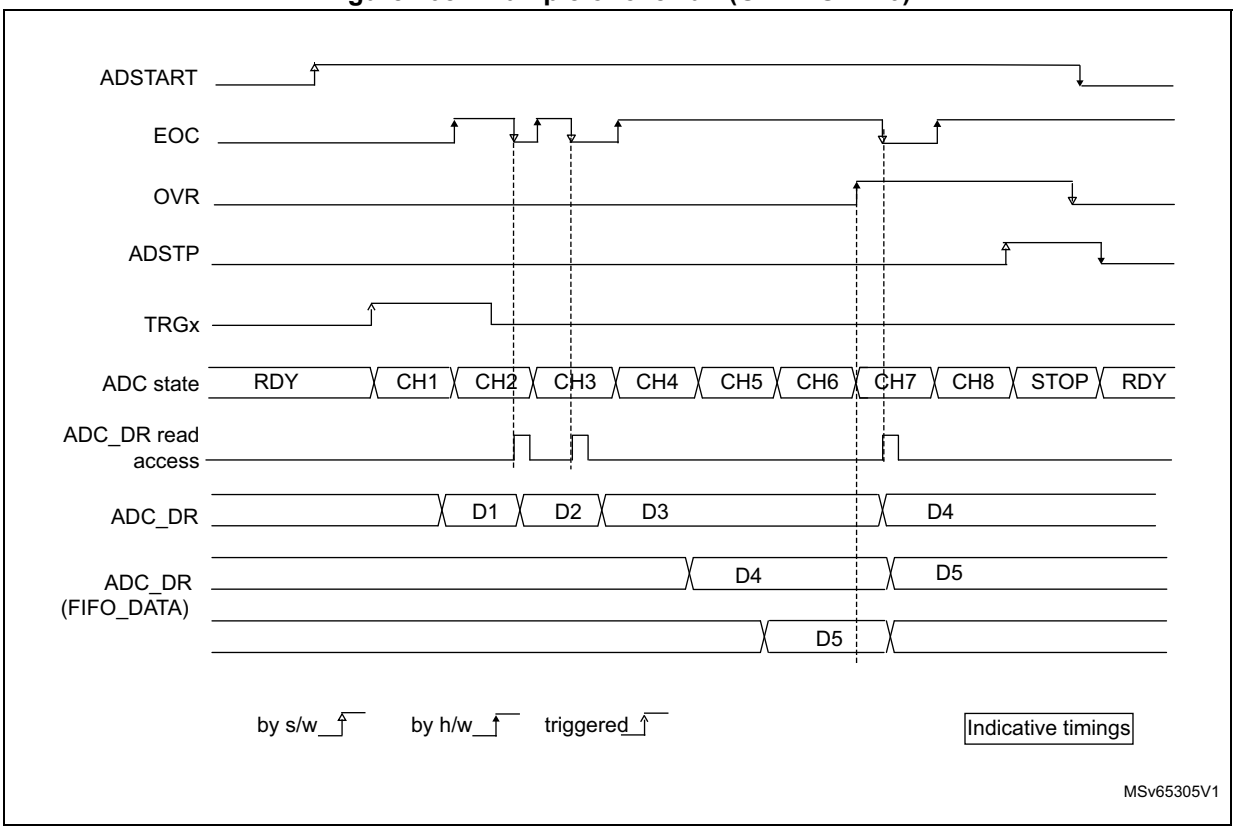
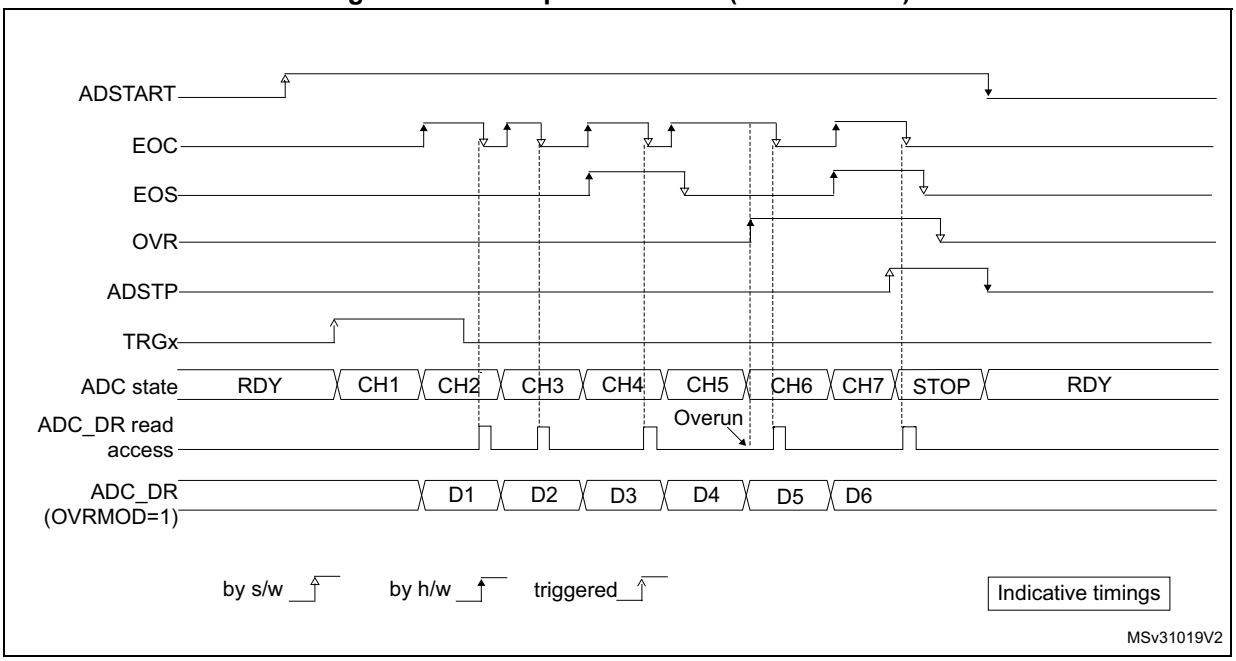


Figure 109. Example of overrun (OVRMOD = 1)



Note: There is no overrun detection on the injected channels since there is a dedicated data register for each of the four injected channels.

Managing a sequence of conversions without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADC_DR register can be read. OVRMOD must be configured to 0 to manage overrun events or FIFO overflow as an error.

Managing conversions without using the DMA and without overrun

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). In this case, the OVRMOD bit must be configured to 1 and OVR flag should be ignored by the software. An overrun event does not prevent the ADC from continuing to convert and the ADC_DR register always contains the latest conversion.

Managing conversions using the DMA

Since converted channel values are stored into a unique data register, it is useful to use DMA for conversion of more than one channel. This avoids the loss of the data already stored in the ADC_DR register.

When the DMA mode is enabled (DMAEN bit set to 1 in the ADC_CFGR register, a DMA request is generated after each conversion of a channel. This allows the transfer of the converted data from the ADC_DR register to the destination location selected by the software.

Despite this, if an overrun occurs (OVR = 1) because the DMA could 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 *Section : 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 of the ADC_CFGR register:

- DMA one shot mode (DMACFG = 0).
  This mode is suitable when the DMA is programmed to transfer a fixed number of data.
- DMA circular mode (DMACFG = 1)
  This mode is suitable when programming the DMA in circular mode.

DMA one shot mode (DMACFG = 0)

In this mode, the ADC generates a DMA transfer request each time a new conversion data 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 with 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.
- 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 is available in the data register, even if the DMA has reached the last DMA transfer. This allows configuring the DMA in circular mode to handle a continuous analog input data stream.

19.4.27 Dynamic low-power features

Auto-delayed conversion mode (AUTDLY)

The ADC implements an auto-delayed conversion mode controlled by the AUTDLY configuration bit. Auto-delayed conversions are useful to simplify the software as well as to optimize performance of an application clocked at low frequency where there would be risk of encountering an ADC overrun.

When AUTDLY = 1, a new conversion can start only if all the previous data of the same group has been treated:

- For a regular conversion: once the ADC_DR register has been read or if the EOC bit has been cleared (see *Figure 110*).
- For an injected conversion: when the JEOS bit has been cleared (see *Figure 111*).

This is a way to automatically adapt the speed of the ADC to the speed of the system which reads the data.

The delay is inserted after each regular conversion (whatever DISCEN = 0 or 1) and after each sequence of injected conversions (whatever JDISCEN = 0 or 1).

Note: There is no delay inserted between each conversions of the injected sequence, except after the last one.

During a conversion, a hardware trigger event (for the same group of conversions) occurring during this delay is ignored.

Note: This is not true for software triggers where it remains possible during this delay to set the bits ADSTART or JADSTART to restart a conversion: it is up to the software to read the data before launching a new conversion.

No delay is inserted between conversions of different groups (a regular conversion followed by an injected conversion or conversely):

- If an injected trigger occurs during the automatic delay of a regular conversion, the injected conversion starts immediately (see *Figure 111*).
- Once the injected sequence is complete, the ADC waits for the delay (if not ended) of the previous regular conversion before launching a new regular conversion (see *Figure 113*).

The behavior is slightly different in auto-injected mode (JAUTO = 1) where a new regular conversion can start only when the automatic delay of the previous injected sequence of conversion has ended (when JEOS has been cleared). This is to ensure that the software can read all the data of a given sequence before starting a new sequence (see *Figure 114*).

To stop a conversion in continuous auto-injection mode combined with autodelay mode (JAUTO = 1, CONT = 1 and AUTDLY = 1), follow the following procedure:

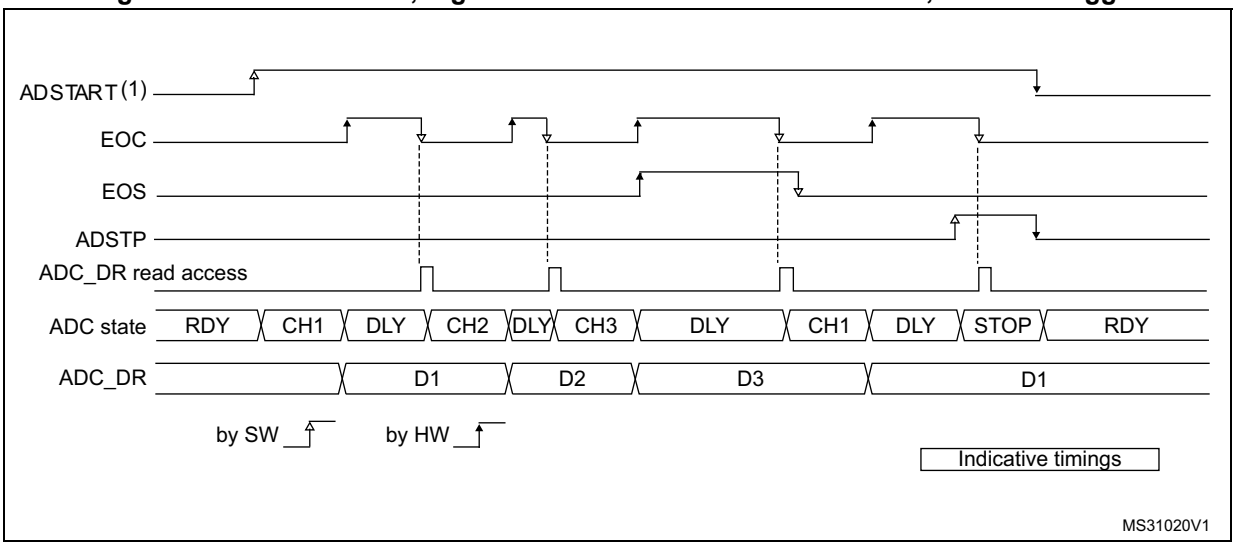
1. Wait until JEOS = 1 (no more conversions are restarted)
2. Clear JEOS,
3. Set ADSTP = 1
4. Read the regular data.

If this procedure is not respected, a new regular sequence can restart if JEOS is cleared after ADSTP has been set.

In AUTDLY mode, a hardware regular trigger event is ignored if it occurs during an already ongoing regular sequence or during the delay that follows the last regular conversion of the sequence. It is however considered pending if it occurs after this delay, even if it occurs during an injected sequence of the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

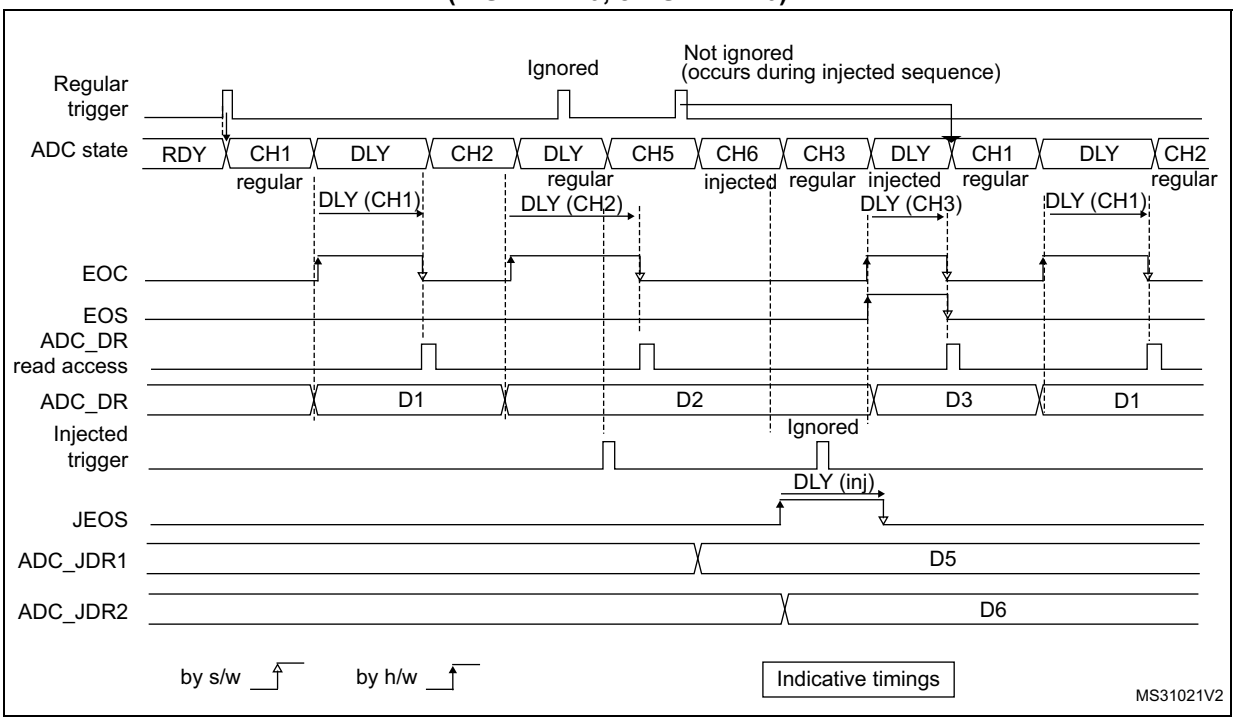
In AUTDLY mode, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.

Figure 110. AUTODLY = 1, regular conversion in continuous mode, software trigger



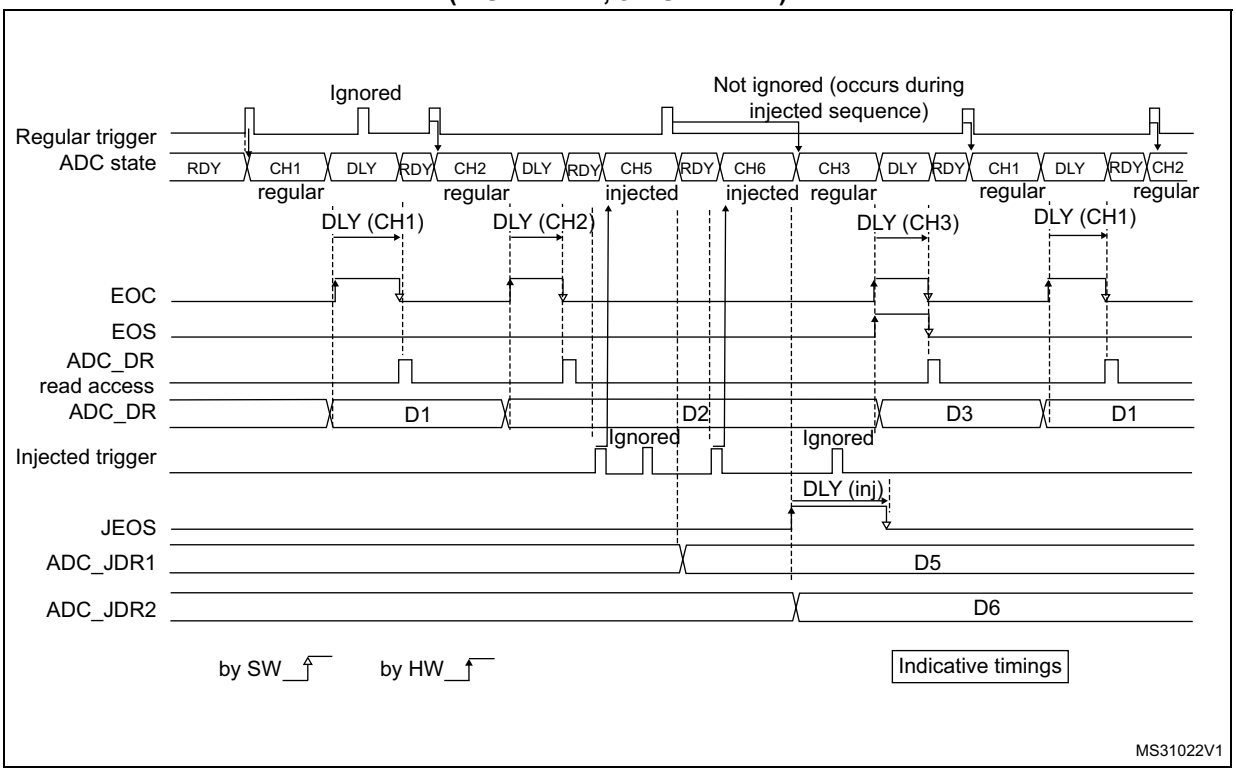
1. AUTDLY = 1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT = 1, CHANNELS = 1,2,3
3. Injected configuration DISABLED

Figure 111. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0)



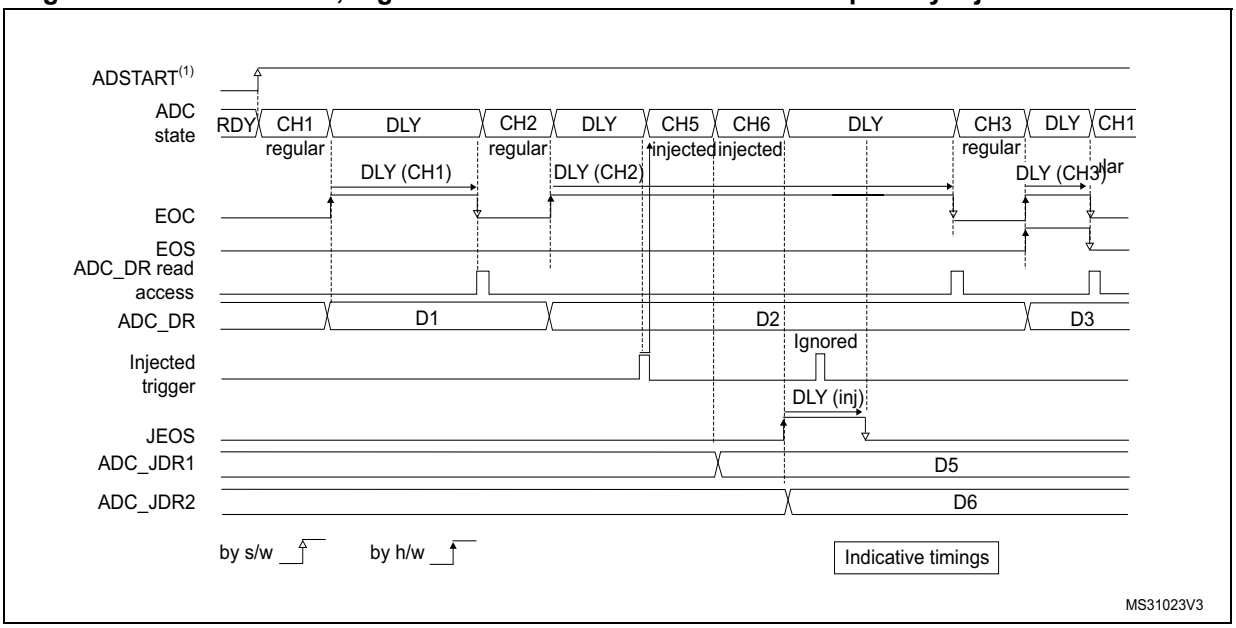
1. AUTDLY = 1
2. Regular configuration: EXTEN=0x1 (HW trigger), CONT = 0, DISCEN = 0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN = 0x1 (HW Trigger), JDISCEN = 0, CHANNELS = 5,6

Figure 112. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1)



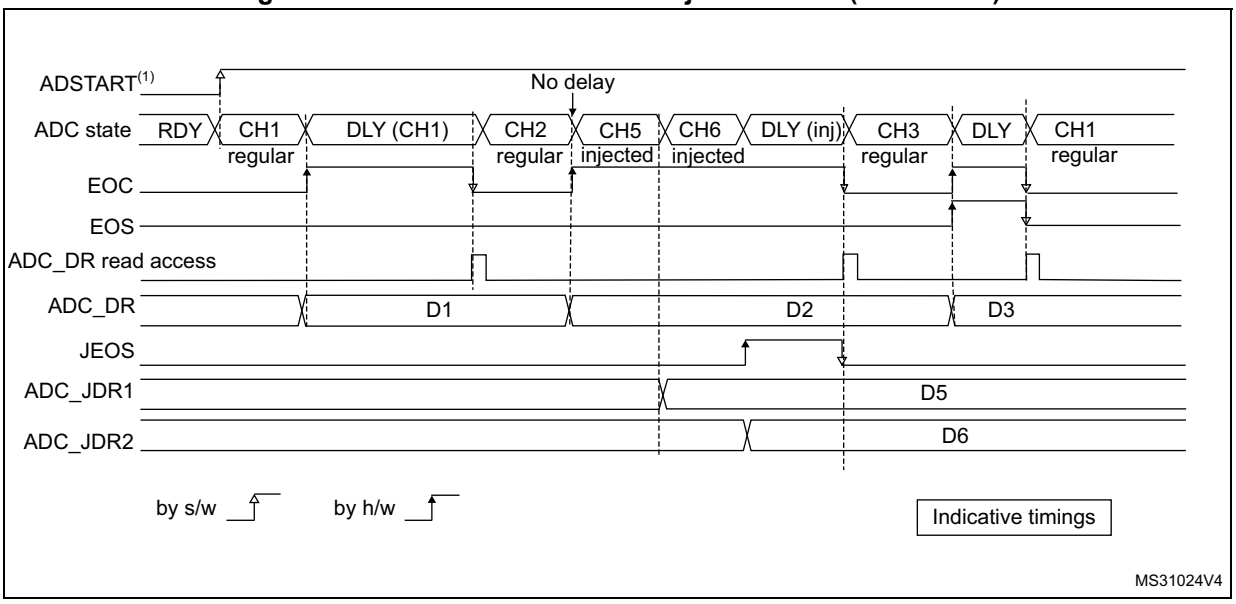
1. AUTDLY = 1
2. Regular configuration: EXTEN = 0x1 (HW trigger), CONT = 0, DISCEN = 1, DISCNUM = 1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN = 0x1 (HW Trigger), JDISCEN = 1, CHANNELS = 5,6

Figure 113. AUTODLY = 1, regular continuous conversions interrupted by injected conversions



1. AUTDLY = 1
2. Regular configuration: EXTEN = 0x0 (SW trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN = 0x1 (HW Trigger), JDISCEN = 0, CHANNELS = 5,6

Figure 114. AUTODLY = 1 in auto- injected mode (JAUTO = 1)

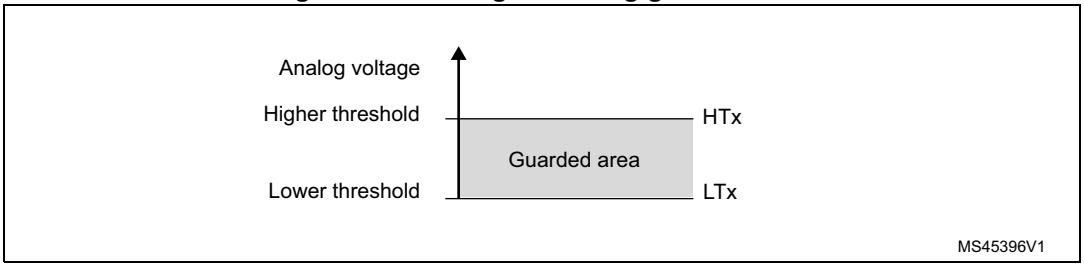


1. AUTDLY = 1
2. Regular configuration: EXTEN = 0x0 (SW trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2
3. Injected configuration: JAUTO = 1, CHANNELS = 5,6

19.4.28 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

Figure 115. Analog watchdog guarded area



AWDx flag and interrupt

An interrupt can be enabled for each of the 3 analog watchdogs by setting AWDxIE in the ADC_IER register (x = 1,2,3).

AWDx (x = 1,2,3) flag is cleared by software by writing 1 to it.

The ADC conversion result is compared to the lower and higher thresholds before alignment.

Description of analog watchdog 1

The AWD analog watchdog 1 is enabled by setting the AWD1EN bit in the ADC_CFGR register. This watchdog monitors whether either one selected channel or all enabled channels(1) remain within a configured voltage range (window).

Table 115 shows how the ADC_CFGR registers should be configured to enable the analog watchdog on one or more channels.

Table 115. Analog watchdog channel selection

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit	JAWD1EN bit
None	x	0	0
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single(1) injected channel	1	0	1
Single(1) regular channel	1	1	0
Single(1) regular or injected channel	1	1	1

1. Selected by the AWD1CH[4:0] bits. The channels must also be programmed to be converted in the appropriate regular or injected sequence.

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 bits HT1[11:0] and LT1[11:0] of the ADC_TR1 register for the analog watchdog 1. When converting data with a resolution of less than 12 bits (according to bits RES[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) before the offset compensation stage.

Table 116 describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

Table 116. Analog watchdog 1 comparison

Resolution(bit RES[1:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned	Thresholds	
00: 12-bit	DATA[11:0]	LT1[11:0] and HT1[11:0]	-
01: 10-bit	DATA[11:2],00	LT1[11:0] and HT1[11:0]	User must configure LT1[1:0] and HT1[1:0] to 00
10: 8-bit	DATA[11:4],0000	LT1[11:0] and HT1[11:0]	User must configure LT1[3:0] and HT1[3:0] to 0000
11: 6-bit	DATA[11:6],000000	LT1[11:0] and HT1[11:0]	User must configure LT1[5:0] and HT1[5:0] to 000000

Analog watchdog filter for watchdog 1

When an ADC is configured with only one input channel (selecting several channels in scan mode not allowed), a valid ADC conversion data interval can be configured through the ADC_TR1 register:

- When converted data belong to the interval defined in ADC_TR1, a DMA request is generated.
- Otherwise, no DMA request is issued. RDATA register is updated at each conversion. If data are out-of-range a number of times higher than the value specified in AWDFILT bit of ADC_TR1, the AWDx flag is set an the corresponding interrupt is issued.

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 corresponding bits in AWDxCH[19:0] (x = 2,3).

The corresponding watchdog is enabled when any bit of AWDxCH[19:0] (x = 2,3) is set.

They are limited to a resolution of 8 bits and only the 8 MSBs of the thresholds can be programmed into HTx[7:0] and LTx[7:0]. *Table 117* describes how the comparison is performed for all the possible resolutions.

Table 117. Analog watchdog 2 and 3 comparison

Resolution (bits RES[1:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned	Thresholds	
00: 12-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	DATA[3:0] are not relevant for the comparison
01: 10-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	DATA[3:2] are not relevant for the comparison
10: 8-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	-
11: 6-bit	DATA[11:6],00	LTx[7:0] and HTx[7:0]	User must configure LTx[1:0] and HTx[1:0] to 00

ADCy_AWDx_OUT signal output generation

Each analog watchdog is associated to an internal hardware signal ADCy_AWDx_OUT (y = ADC number, x = watchdog number) which is directly connected to the ETR input (external trigger) of some on-chip timers. Refer to the on-chip timers section to understand how to select the ADCy_AWDx_OUT signal as ETR.

ADCy_AWDx_OUT is activated when the associated analog watchdog is enabled:

- ADCy_AWDx_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADCy_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).
- ADCy_AWDx_OUT is also reset when disabling the ADC (when setting ADDIS = 1). Note that stopping regular or injected conversions (setting ADSTP = 1 or JADSTP = 1) has no influence on the generation of ADCy_AWDx_OUT.

Note: AWDx flag is set by hardware and reset by software: AWDx flag has no influence on the generation of ADCy_AWDx_OUT (ex: ADCy_AWDx_OUT can toggle while AWDx flag remains at 1 if the software did not clear the flag).

Figure 116. ADCy_AWDx_OUT signal generation (on all regular channels)

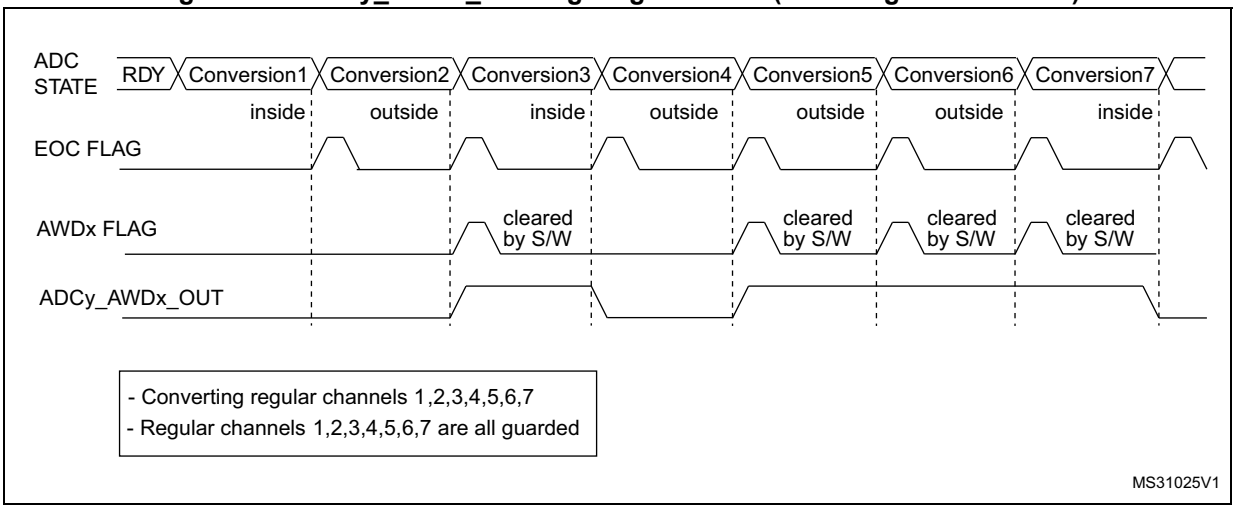


Figure 117. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by software)

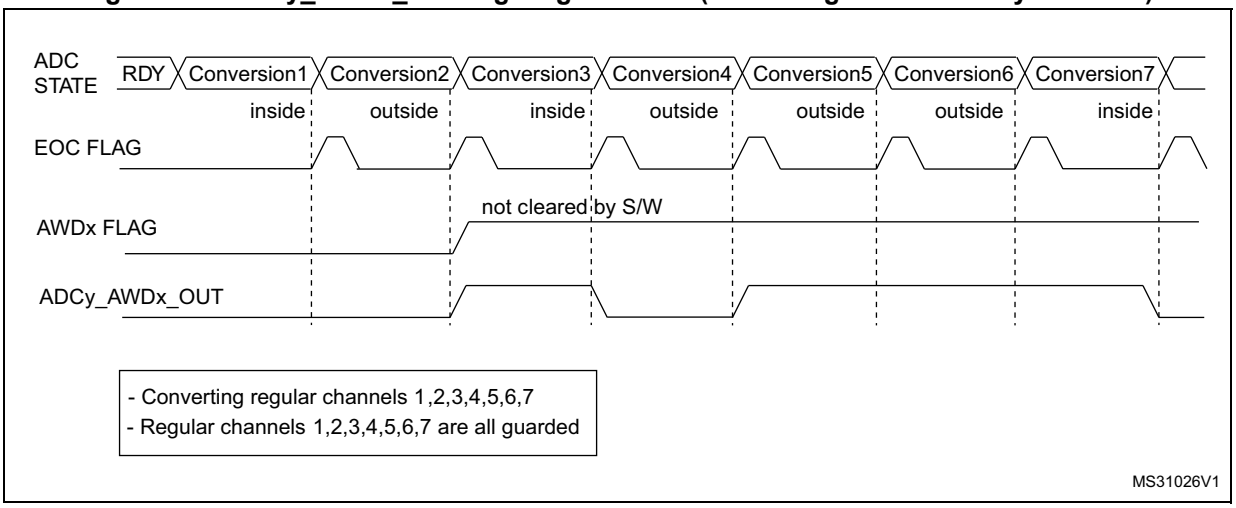


Figure 118. ADCy_AWDx_OUT signal generation (on a single regular channel)

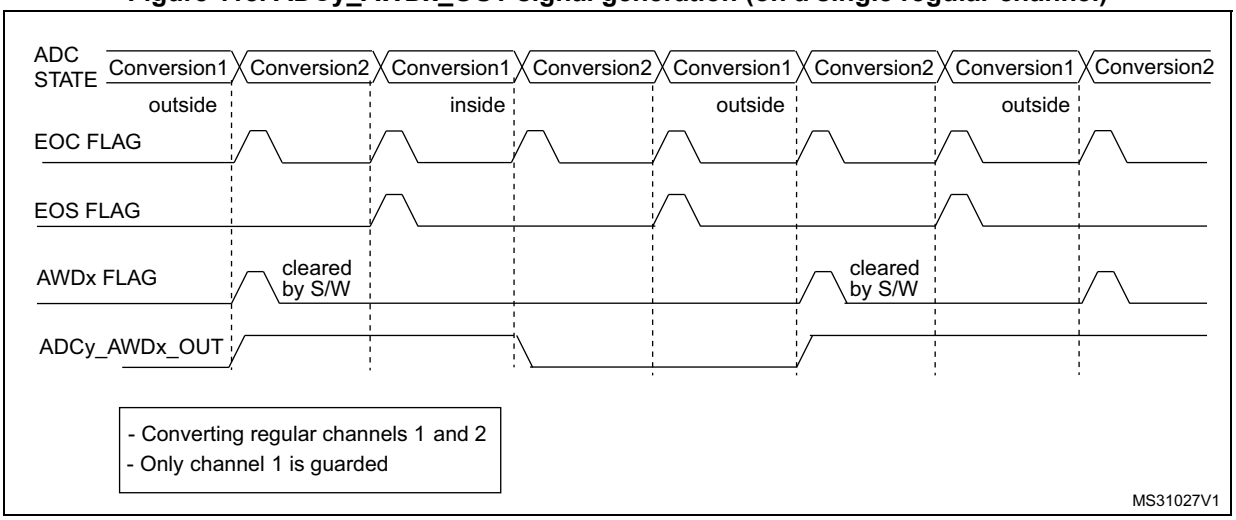
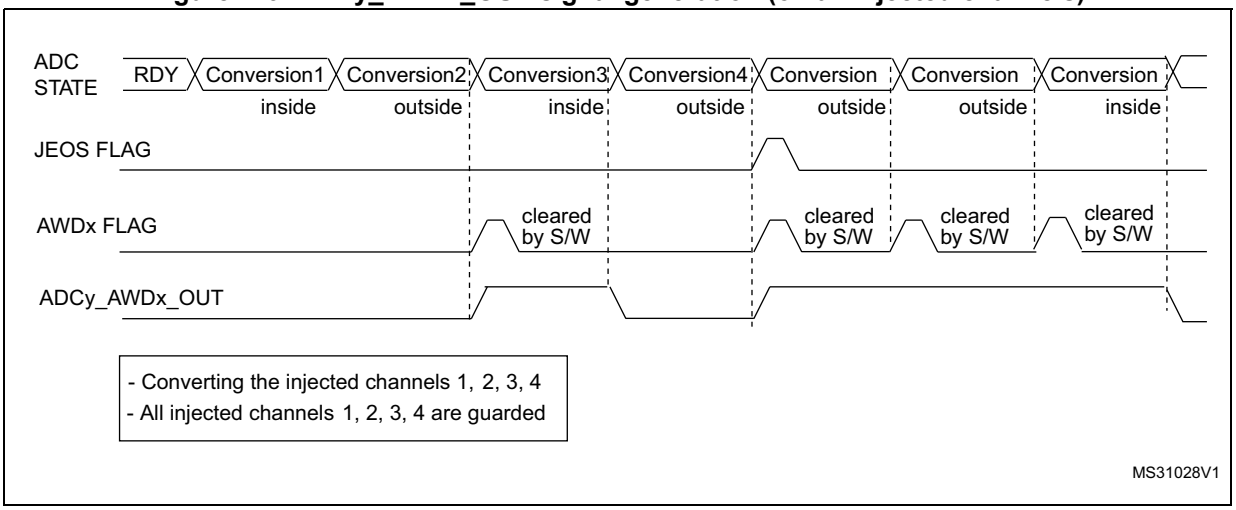


Figure 119. ADCy_AWDx_OUT signal generation (on all injected channels)



Analog watchdog threshold control

LTx[11:0] and HTx[11:0] can be changed when an analog-to-digital conversion is ongoing (that is between the start of conversion and the end of conversion of the ADC internal state). If LTx[11:0] and HTx[11:0] are updated during the ADC conversion of the ADC guarded channel, the watchdog function is masked for this conversion. This masking is removed at the next start of conversion, resulting in a analog watchdog thresholds to be applied from the next ADC conversion. The analog watchdog comparison is performed at each end of conversion. If the current ADC data is out of the new interval, no interrupt and AWDx_OUT signal are issued. The Interrupt and the AWD generation only happen at the end of the conversion which started after the threshold update. If AWD_xOUT is already asserted, programming the new thresholds does not deassert the AWDx_OUT signal.

To update both LTx[11:0] and HTx[11:0] during an ADC conversion, a unique write access to the ADC_TRx register be performed.

Analog watchdog with offset compensation

When the offset compensation is enabled, the analog watchdog compares the threshold before the data compensation.

19.4.29 Oversampler

The oversampling unit performs data pre-processing to offload the CPU. It is able to 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 to perform by hardware the following functions: averaging, data rate reduction, SNR improvement, basic filtering.

The oversampling ratio N is defined using the OVFS[2:0] bits in the ADC_CFGR2 register, and can range from 2x to 256x. The division coefficient M consists of a right bit shift up to 8 bits, and is defined using the OVSS[3:0] bits in the ADC_CFGR2 register.

The summation unit can yield a result up to 20 bits (256x 12-bit results), which is first shifted right. It is then truncated to 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 intermediary result after the shifting exceeds 16-bit, the result is truncated as is, without saturation.

Figure 120. 20-bit to 16-bit result truncation

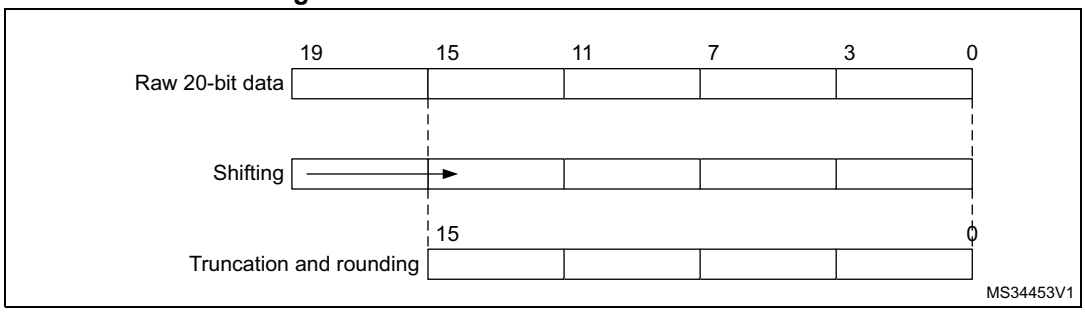


Figure 121 gives a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

Figure 121. Numerical example with 5-bit shift and rounding

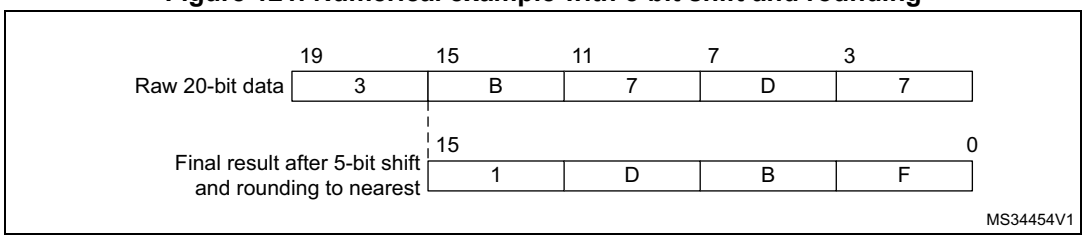


Table 118 gives the data format for the various N and M combinations, for a raw conversion data equal to 0xFFF.

Table 118. Maximum output results versus N and M (gray cells indicate truncation)

Over sampling 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	0x020
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

There are no changes for conversion timings in oversampled mode: the sample time is maintained equal during the whole oversampling sequence. A new data is provided every N

conversions, with an equivalent delay equal to N x T_{CONV} = N x (t_{SMPL} + t_{SAR}). The flags are set as follow:

- 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 (EOS) occurs once the sequence of oversampled data is completed (that is after N x sequence length conversions total)

ADC operating modes supported when oversampling

In oversampling mode, most of the ADC operating modes are maintained:

- Single or continuous conversion modes
- 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 (AUTDLY)
- Programmable resolution: in this case, the reduced conversion values (as per RES[1:0] bits in ADC_CFGR 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_CFGR is ignored and the data are always provided right-aligned.

Offset correction is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the value of the OFFSET_EN bit in ADC_OFRy register is ignored (considered as reset).

Analog watchdog

The analog watchdog functionality is maintained, with the following difference:

- The RES[1:0] bits are ignored, the comparison is always done by using the full 12-bit values HT1[11:0] and LT1[11:0] for AWD1, and the 8-MSB value of HT2/HT3[7:0] and LT2/LT3[7:0] for AWD2 and AWD3.
- The comparison is done on the most significant 12-bit of the 16-bit oversampled results ADC_DR[15:4] for AWD1, and ADC_DR[15:8] for AWD2 and ADW3

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] (AWD1/2/3), with HT1[11:8] and LT1[11:8] kept reset (AWD1 only).

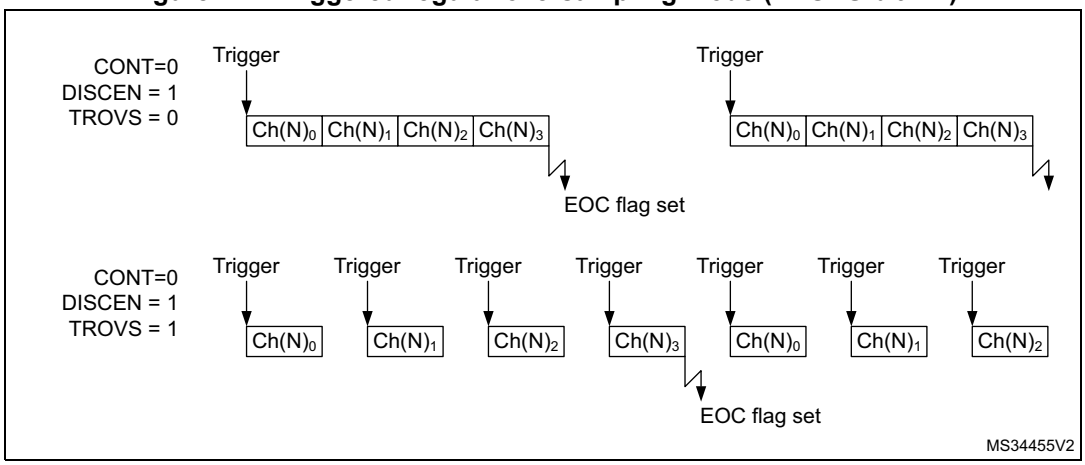
Triggered mode

The averager can also be used for basic filtering purpose. Although not a very powerful 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 TROVS bit in ADC_CFGR2, to be able to have an oversampling frequency defined by a user and independent from the conversion time itself.

The *Figure 122* below shows how conversions are started in response to triggers during discontinuous mode.

If the TROVS bit is set, the content of the DISCEN bit is ignored and considered as 1.

Figure 122. Triggered regular oversampling mode (TROVS bit = 1)



Injected and regular sequencer management when oversampling

In oversampling mode, it is possible to have differentiated behavior for injected and regular sequencers. The oversampling can be enabled for both sequencers with some limitations if they have to be used simultaneously (this is related to a unique accumulation unit).

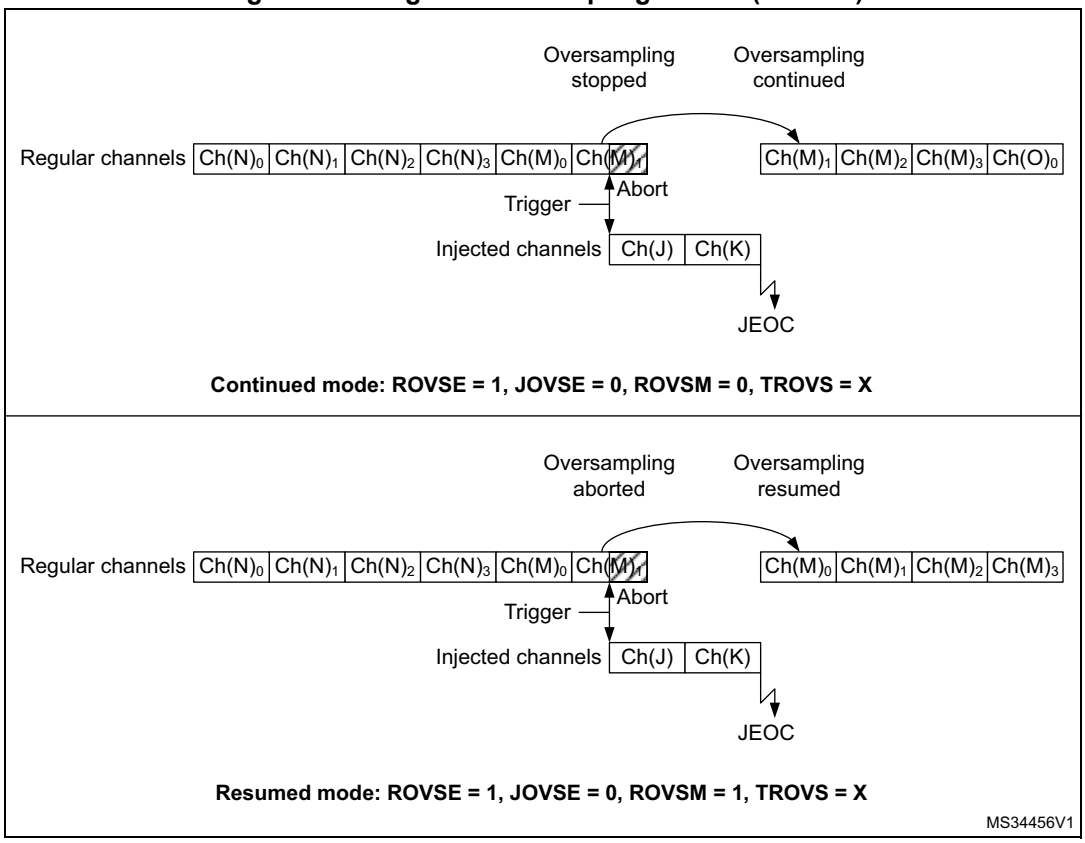
Oversampling regular channels only

The regular oversampling mode bit ROVSM defines how the regular oversampling sequence is resumed if it is interrupted by injected conversion:

- In continued mode, the accumulation restarts from the last valid data (prior to the conversion abort request due to the injected trigger). This ensures that oversampling is complete whatever the injection frequency (providing at least one regular conversion can be completed between triggers);
- In resumed mode, the accumulation restarts from 0 (previous conversions results are ignored). This mode allows to guarantee that all data used for oversampling were converted back-to-back within a single timeslot. Care must be taken to have a injection trigger period above the oversampling period length. If this condition is not respected, the oversampling cannot be completed and the regular sequencer is blocked.

The *Figure 123* gives examples for a 4x oversampling ratio.

Figure 123. Regular oversampling modes (4x ratio)



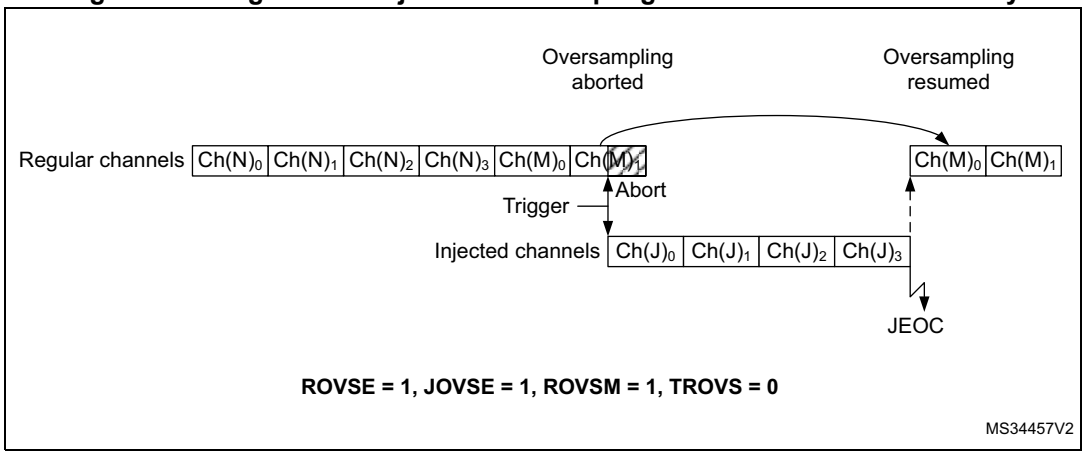
Oversampling Injected channels only

The Injected oversampling mode bit JOVSE enables oversampling solely for conversions in the injected sequencer.

Oversampling regular and Injected channels

It is possible to have both ROVSE and JOVSE bits set. In this case, the regular oversampling mode is forced to resumed mode (ROVSM bit ignored), as represented on *Figure 124* below.

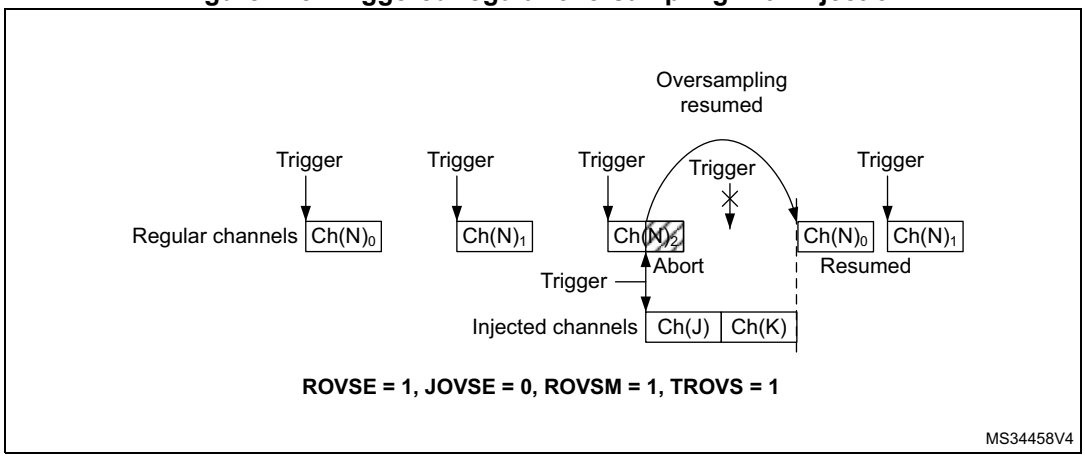
Figure 124. Regular and injected oversampling modes used simultaneously



Triggered regular oversampling with injected conversions

It is possible to have triggered regular mode with injected conversions. In this case, the injected mode oversampling mode must be disabled, and the ROVSM bit is ignored (resumed mode is forced). The JOVSE bit must be reset. The behavior is represented on *Figure 125* below.

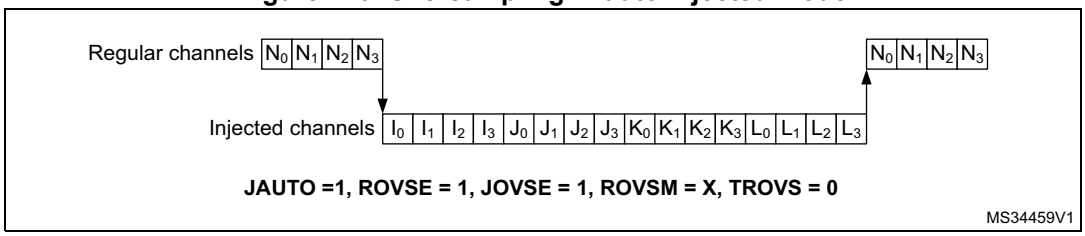
Figure 125. Triggered regular oversampling with injection



Auto-injected mode

It is possible to oversample auto-injected sequences and have all conversions results stored in registers to save a DMA resource. This mode is available only with both regular and injected oversampling active: JAUTO = 1, ROVSE = 1 and JOVSE = 1, other combinations are not supported. The ROVSM bit is ignored in auto-injected mode. The *Figure 126* below shows how the conversions are sequenced.

Figure 126. Oversampling in auto-injected mode



It is possible to have also the triggered mode enabled, using the TROVS bit. In this case, the ADC must be configured as following: JAUTO = 1, DISCEN = 0, JDISCEN = 0, ROVSE = 1, JOVSE = 1 and TROVSE = 1.

19.4.30 Temperature sensor

The temperature sensor can be used to measure the junction temperature (Tj) of the device.

The temperature sensor is internally connected to the ADC input channels which are used to convert the sensor output voltage to a digital value (see *Table: ADC interconnection* in *Section 19.4.2: ADC pins and internal signals* for more details). When not in use, the sensor can be put in power down mode. It support the temperature range –40 to 125 °C.

Figure 127 shows the block diagram of connections between the temperature sensor and the ADC.

The temperature sensor output voltage changes linearly with 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 ST during production.

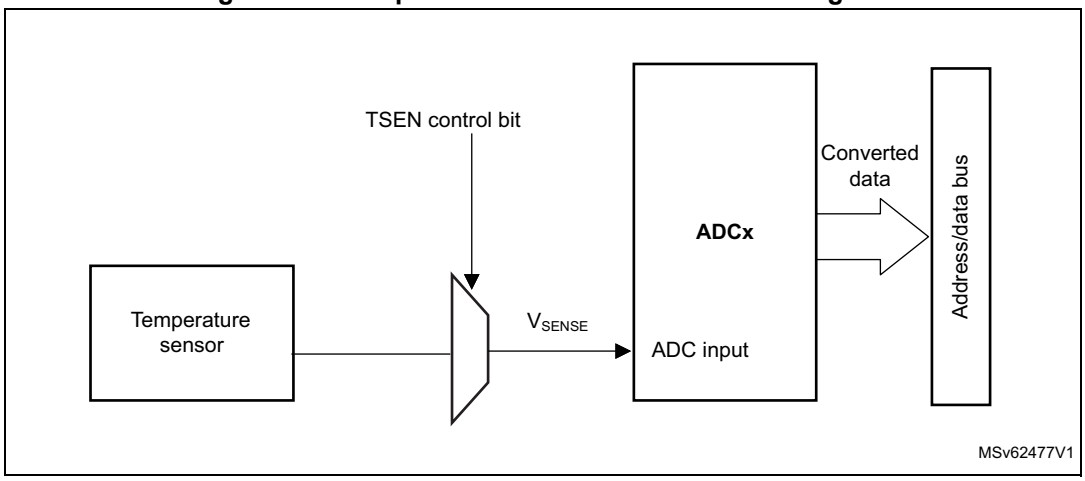
During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference (refer to the datasheet for additional information).

The temperature sensor is internally connected to the ADC input channel which is used to convert the sensor's output voltage to a digital value. Refer to the electrical characteristics section of the device datasheet for the sampling time value to be applied when converting the internal temperature sensor.

When not in use, the sensor can be put in power-down mode.

Figure 127 shows the block diagram of the temperature sensor.

Figure 127. Temperature sensor channel block diagram



Reading the temperature

To use the sensor:

1. Select the ADC input channels that is connected to V_{SENSE}.
2. Program with the appropriate sampling time (refer to electrical characteristics section of the device datasheet).
3. Set the bit in the ADC_CCR register to wake up the temperature sensor from power-down mode.
4. Start the ADC conversion.
5. Read the resulting V_{SENSE} data in the ADC data register.
6. Calculate the actual 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 ADC.

  Refer to the device datasheet for more information about TS_CAL1 and TS_CAL2 calibration points.

Note: The sensor has a startup time after waking 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 bits should be set at the same time.

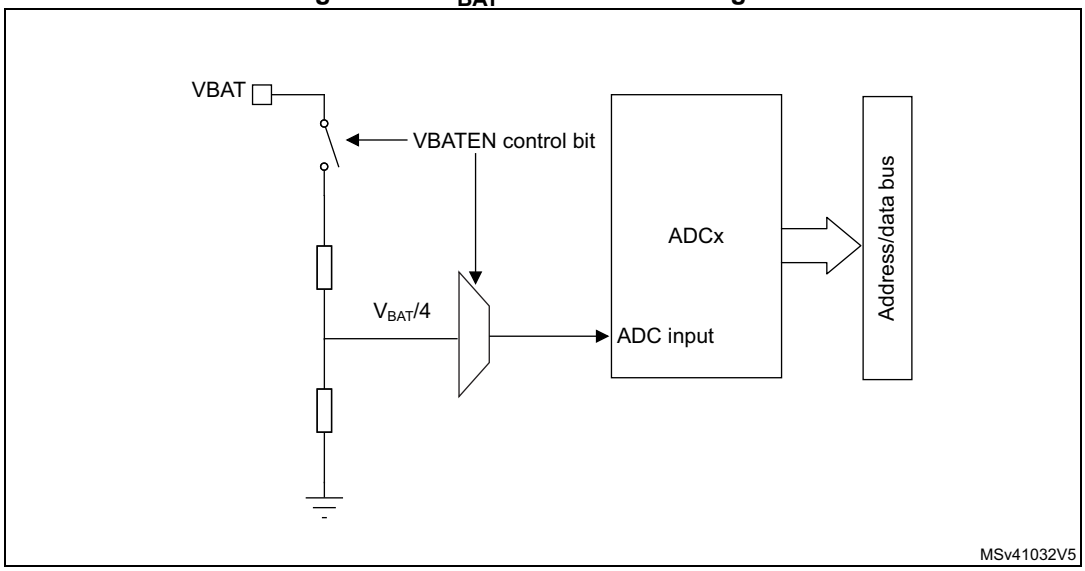
19.4.31 V_{BAT} supply monitoring

The VBATEN bit in the ADC_CCR register is used to switch to the battery voltage. As the V_{BAT} voltage could be higher than V_{DDA}, to ensure the correct operation of the ADC, the V_{BAT} pin is internally connected to a bridge divider by 4. This bridge is automatically enabled when VBATEN is set, to connect $V_{BAT}/4$ to the ADC input channels (see *Table: ADC interconnection* in *Section 19.4.2: ADC pins and internal signals* for more details). As a consequence, the converted digital value is one third of the V_{BAT} voltage. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed, for ADC conversion.

Refer to the electrical characteristics of the device datasheet for the sampling time value to be applied when converting the $V_{BAT}/4$ voltage.

Figure 128 shows the block diagram of the V_{BAT} sensing feature.

Figure 128. V_{BAT} channel block diagram



1. The VBATEN bit must be set to enable the conversion of internal channel for $V_{BAT}/4$.

19.4.32 Monitoring the internal voltage reference

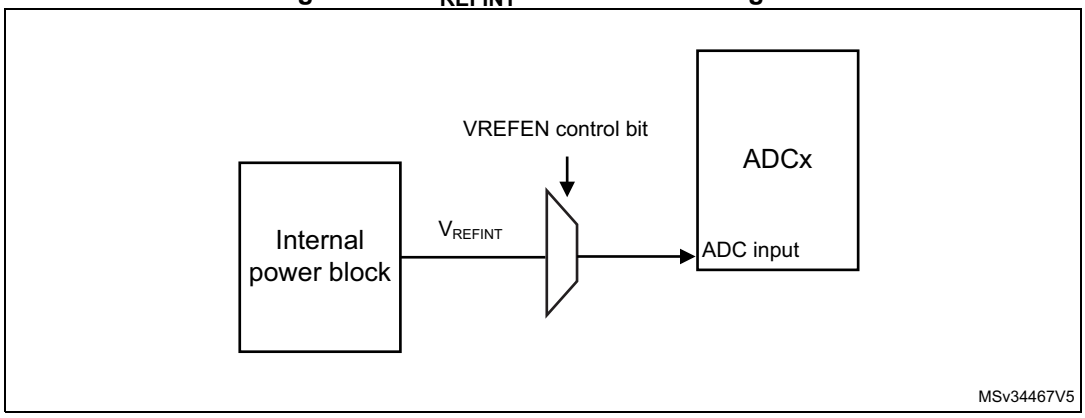
It is possible to monitor the internal voltage reference (V_{REFINT}) to have a reference point for evaluating the ADC V_{REF+} voltage level.

Refer to *Table: ADC interconnection* in *Section 19.4.2: ADC pins and internal signals* for details on the ADC input channels to which the internal voltage reference is internally connected.

Refer to the electrical characteristics section of the product datasheet for the sampling time value to be applied when converting the internal voltage reference voltage.

Figure 129 shows the block diagram of the V_{REFINT} sensing feature.

Figure 129. V_{REFINT} channel block diagram



1. The VREFEN bit into ADC_CCR register must be set to enable the conversion of internal channels (V_{REFINT}).

Calculating the actual V_{REF+} voltage using the internal reference voltage

V_{REF+} voltage may be subject to variations or not precisely known. The embedded internal reference voltage V_{REFINT} and its calibration data acquired by the ADC during the manufacturing process at V_{REF+_charac} can be used to evaluate the actual V_{REF+} voltage level.

The following formula gives the actual V_{REF+} voltage supplying the device:

$$V_{REF+} = V_{REF+\_Charac} \times VREFINT\_CAL / VREFINT\_DATA$$

Where:

- V_{REF+_Charac} is the value of V_{REF+} voltage characterized at V_{REFINT} during the manufacturing process. It is specified in the device datasheet.
- VREFINT_CAL is the VREFINVREFINT_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 V_{REF+} and the voltage applied on the converted channel.

For applications where V_{REF+} value is unknown and ADC converted values are right-aligned. In this case, it is necessary to convert this ratio into a voltage independent from V_{REF+}:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL\_SCALE} \times ADC\_DATA$$

By replacing V_{REF+} by the formula provided above, the absolute voltage value is given by the following formula

$$V_{CHANNELx} = \frac{V_{REF+\_Charac} \times VREFINT\_CAL \times ADC\_DATA}{VREFINT\_DATA \times FULL\_SCALE}$$

Where:

- V_{REF+_Charac} is the value of V_{REF+} voltage characterized at V_{REFINT} during the manufacturing process.
- VREFINT_CAL is the VREFINT calibration value
- ADC_DATA is the value measured by the ADC on channel x (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.

19.4.33 Monitoring the supply voltage

ADC is connected to the internal supply voltage. To use the ADC to measure this voltage, enable the connection through ADC option register.

19.5 ADC interrupts

An interrupt can be generated:

- After ADC power-up, when the ADC is ready (flag ADRDY)
- On the end of any conversion for regular groups (flag EOC)
- On the end of a sequence of conversion for regular groups (flag EOS)
- On the end of any conversion for injected groups (flag JEOC)
- On the end of a sequence of conversion for injected groups (flag JEOS)
- When an analog watchdog detection occurs (flag AWD1, AWD2 and AWD3)
- When the end of sampling phase occurs (flag EOSMP)
- When the data overrun occurs (flag OVR)
- When the injected sequence context queue overflows (flag JQOVF)

Separate interrupt enable bits are available for flexibility.

Table 119. ADC interrupts

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop, Standby mode
ADC	ADC ready	ADRDY	ADRDYIE	Set by hardware and cleared by software	Yes	No
	End of conversion of a regular group	EOC	EOCIE			
	End of conversion sequence of a regular group	EOS	EOSIE			
	End of conversion of an injected group	JEOC	JEOCIE			
	End of conversion sequence of an injected group	JEOS	JEOSIE			
	Analog watchdog 1 status bit is set	AWD1	AWD1IE			
	Analog watchdog 2 status bit is set	AWD2	AWD2IE			
	Analog watchdog 3 status bit is set	AWD3	AWD3IE			
	End of sampling phase	EOSMP	EOSMPIE			
	Overrun	OVR	OVRIE			
	Injected context queue overflows	JQOVF	JQOVFIE			

19.6 ADC registers

Refer to *Section 1.2 on page 65* for a list of abbreviations used in register descriptions.

19.6.1 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.	Res.	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	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	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **JQOVF**: Injected context queue overflow

This bit is set by hardware when an Overflow of the Injected Queue of Context occurs. It is cleared by software writing 1 to it. Refer to *Section 19.4.21: Queue of context for injected conversions* for more information.

0: No injected context queue overflow occurred (or the flag event was already acknowledged and cleared by software)

1: Injected context queue overflow has occurred

Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT3[7:0] and HT3[7:0] of ADC_TR3 register. It is cleared by software writing 1 to it.

0: No analog watchdog 3 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 3 event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT2[7:0] and HT2[7:0] of ADC_TR2 register. It is cleared by software writing 1 to it.

0: No analog watchdog 2 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 2 event occurred

Bit 7 **AWD1**: Analog watchdog 1 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT1[11:0] and HT1[11:0] of ADC_TR1 register. It is cleared by software. writing 1 to it.

0: No analog watchdog 1 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 1 event occurred

Bit 6 **JEOS:** Injected channel end of sequence flag

This bit is set by hardware at the end of the conversions of all injected channels in the group. It is cleared by software writing 1 to it.

0: Injected conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Injected conversions complete

Bit 5 **JEOC:** Injected channel end of conversion flag

This bit is set by hardware at the end of each injected conversion of a channel when a new data is available in the corresponding ADC_JDRy register. It is cleared by software writing 1 to it or by reading the corresponding ADC_JDRy register

0: Injected channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Injected channel conversion complete

Bit 4 **OVR**: ADC overrun

This bit is set by hardware when an overrun occurs on a regular channel, meaning that a new conversion has completed 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 regular sequence flag

This bit is set by hardware at the end of the conversions of a regular sequence of channels. It is cleared by software writing 1 to it.

0: Regular Conversions sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Regular Conversions sequence complete

Bit 2 **EOC**: End of conversion flag

This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADC_DR register. It is cleared by software writing 1 to it or by reading the ADC_DR register

0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Regular channel conversion complete

Bit 1 **EOSMP**: End of sampling flag

This bit is set by hardware during the conversion of any channel (only for regular channels), at the end of the sampling phase.

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

19.6.2 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.	Res.	Res.	JQOVFIE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **JQOVFIE**: Injected context queue overflow interrupt enable

This bit is set and cleared by software to enable/disable the Injected Context Queue Overflow interrupt.

0: Injected Context Queue Overflow interrupt disabled

1: Injected Context Queue Overflow interrupt enabled. An interrupt is generated when the JQOVF bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 3 interrupt disabled

1: Analog watchdog 3 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which 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 2 interrupt.

0: Analog watchdog 2 interrupt disabled

1: Analog watchdog 2 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which 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 1 interrupt.

0: Analog watchdog 1 interrupt disabled

1: Analog watchdog 1 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 6 **JEOSIE**: End of injected sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of injected sequence of conversions interrupt.

0: JEOS interrupt disabled

1: JEOS interrupt enabled. An interrupt is generated when the JEOS bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 5 **JEOCIE**: End of injected conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of an injected conversion interrupt.

0: JEOC interrupt disabled.

1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 4 **OVRIE**: Overrun interrupt enable

This bit is set and cleared by software to enable/disable the Overrun interrupt of a regular conversion.

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 = 0 (which ensures that no regular conversion is ongoing).

Bit 3 **EOSIE**: End of regular sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of regular 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 = 0 (which ensures that no regular conversion is ongoing).

Bit 2 **EOCIE**: End of regular conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of a regular 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 = 0 (which ensures that no regular conversion is ongoing).

Bit 1 **EOSMPIE**: End of sampling flag interrupt enable for regular conversions

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt for regular conversions.

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 = 0 (which ensures that no regular 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 = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

19.6.3 ADC control register (ADC_CR)

Address offset: 0x08

Reset value: 0x2000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	ADCALDIF	DEEPPWD	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs	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.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
										rs	rs	rs	rs	rs	rs

Bit 31 **ADCAL**: ADC calibration

This bit is set by software to start the calibration of the ADC. Program first the bit ADCALDIF to determine if this calibration applies for single-ended or Differential inputs mode.

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 in progress.

Note: The software is allowed to launch a calibration by setting ADCAL only when ADEN = 0.

The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing)

Bit 30 **ADCALDIF**: Differential mode for calibration

This bit is set and cleared by software to configure the single-ended or Differential inputs mode for the calibration.

0: Writing ADCAL launches a calibration in single-ended inputs mode.

1: Writing ADCAL launches a calibration in Differential inputs mode.

Note: The software is allowed to write this bit only when the ADC is disabled and is not calibrating (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bit 29 **DEEPPWD**: Deep-power-down enable

This bit is set and cleared by software to put the ADC in Deep-power-down mode.

0: ADC not in Deep-power down

1: ADC in Deep-power-down (default reset state)

Note: The software is allowed to write this bit only when the ADC is disabled (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bit 28 **ADVREGEN**: ADC voltage regulator enable

This bits is set by software to enable the ADC voltage regulator.

Before performing any operation such as launching a calibration or enabling the ADC, the ADC voltage regulator must first be enabled and the software must wait for the regulator start-up time.

0: ADC Voltage regulator disabled

1: ADC Voltage regulator enabled.

For more details about the ADC voltage regulator enable and disable sequences, refer to *Section 19.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)*.

The software can program this bit field only when the ADC is disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 27:6 Reserved, must be kept at reset value.

Bit 5 **JADSTP**: ADC stop of injected conversion command

This bit is set by software to stop and discard an ongoing injected conversion (JADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC injected sequence and triggers can be re-configured. The ADC is then ready to accept a new start of injected conversions (JADSTART command).

0: No ADC stop injected conversion command ongoing

1: Write 1 to stop injected conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: The software is allowed to set JADSTP only when JADSTART = 1 and ADDIS = 0 (ADC is enabled and eventually converting an injected conversion and there is no pending request to disable the ADC)

In auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP)

Bit 4 **ADSTP**: ADC stop of regular conversion command

This bit is set by software to stop and discard an ongoing regular conversion (ADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC regular sequence and triggers can be re-configured. The ADC is then ready to accept a new start of regular conversions (ADSTART command).

0: No ADC stop regular conversion command ongoing

1: Write 1 to stop regular conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: The software is allowed to set ADSTP only when ADSTART = 1 and ADDIS = 0 (ADC is enabled and eventually converting a regular conversion and there is no pending request to disable the ADC).

In auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP).

Bit 3 **JADSTART**: ADC start of injected conversion

This bit is set by software to start ADC conversion of injected channels. Depending on the configuration bits JEXTEN, a conversion immediately starts (software trigger configuration) or once an injected hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

– in single conversion mode when software trigger is selected (JEXTSEL = 0x0): at the assertion of the End of Injected Conversion Sequence (JEOS) flag.

– in all cases: after the execution of the JADSTP command, at the same time that JADSTP is cleared by hardware.

0: No ADC injected conversion is ongoing.

1: Write 1 to start injected conversions. Read 1 means that the ADC is operating and eventually converting an injected channel.

Note: The software is allowed to set JADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC).

In auto-injection mode (JAUTO = 1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)

Bit 2 **ADSTART**: ADC start of regular conversion

This bit is set by software to start ADC conversion of regular channels. Depending on the configuration bits EXTEN, a conversion immediately starts (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

– in single conversion mode when software trigger is selected (EXTSEL = 0x0): at the assertion of the End of Regular Conversion Sequence (EOS) flag.

– in all cases: after the execution of the ADSTP command, at the same time that ADSTP is cleared by hardware.

0: No ADC regular conversion is ongoing.

1: Write 1 to start regular conversions. Read 1 means that the ADC is operating and eventually converting a regular channel.

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)

In auto-injection mode (JAUTO = 1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)

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: The software is allowed to set ADDIS only when ADEN = 1 and both ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)

Bit 0 **ADEN**: ADC enable control

This bit is set by software to enable the ADC. The ADC is effectively ready to operate once the flag ADRDY 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, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0) except for bit ADVREGEN which must be 1 (and the software must have wait for the startup time of the voltage regulator)

19.6.4 ADC configuration register (ADC_CFGR)

Address offset: 0x0C

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JQDIS	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM[2:0]			DISCEN
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
ALIGN	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL4	EXTSEL3	EXTSEL2	EXTSEL1	EXTSEL0	RES[1:0]		Res.	DMACFG	DMAEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 31 **JQDIS**: Injected queue disable

This bit is set and cleared by software to disable the injected queue mechanism:

0: Injected queue enabled

1: Injected queue disabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no regular nor injected conversion is ongoing).

A set or reset of JQDIS bit causes the injected queue to be flushed and the JSQR register is cleared.

Bits 30:26 **AWD1CH[4:0]**: Analog watchdog 1 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 AWD1

00001: ADC analog input channel 1 monitored by AWD1

.....

10011: ADC analog input channel 19 monitored by AWD1

others: reserved, must not be used

Note: Some channels are not connected physically. Keep the corresponding AWD1CH[4:0] setting to the reset value.

The channel selected by AWD1CH must be also selected into the SQRi or JSQRi registers.

The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 25 **JAUTO:** Automatic injected group conversion

This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.

0: Automatic injected group conversion disabled

1: Automatic injected group conversion enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no regular nor injected conversion is ongoing).

Bit 24 **JAWD1EN**: Analog watchdog 1 enable on injected channels
This bit is set and cleared by software
0: Analog watchdog 1 disabled on injected channels
1: Analog watchdog 1 enabled on injected channels
Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 23 **AWD1EN**: Analog watchdog 1 enable on regular channels
This bit is set and cleared by software
0: Analog watchdog 1 disabled on regular channels
1: Analog watchdog 1 enabled on regular channels
Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 22 **AWD1SGL**: Enable the watchdog 1 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 AWD1CH[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 these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 21 **JQM**: JSQR queue mode
This bit is set and cleared by software.
It defines how an empty Queue is managed.
0: JSQR mode 0: The Queue is never empty and maintains the last written configuration into JSQR.
1: JSQR mode 1: The Queue can be empty and when this occurs, the software and hardware triggers of the injected sequence are both internally disabled just after the completion of the last valid injected sequence.
Refer to *Section 19.4.21: Queue of context for injected conversions* for more information.
Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 20 **JDISCEN:** Discontinuous mode on injected channels
This bit is set and cleared by software to enable/disable discontinuous mode on the injected channels of a group.
0: Discontinuous mode on injected channels disabled
1: Discontinuous mode on injected channels enabled
Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).
It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

Bits 19:17 **DISCNUM[2:0]:** Discontinuous mode channel count
These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.
000: 1 channel
001: 2 channels
...
111: 8 channels
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 16 **DISCEN**: Discontinuous mode for regular channels

This bit is set and cleared by software to enable/disable discontinuous mode for regular channels.

0: Discontinuous mode for regular channels disabled

1: Discontinuous mode for regular channels enabled

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN = 1 and CONT = 1.

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 15 **ALIGN**: Data alignment

This bit is set and cleared by software to select right or left alignment. Refer to *Section : Data register, data alignment and offset (ADC_DR, OFFSET, OFFSET_CH, ALIGN)*.

0: Right alignment

1: Left alignment

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 14 **AUTDLY**: Delayed conversion mode

This bit is set and cleared by software to enable/disable the Auto Delayed Conversion mode.

0: Auto-delayed conversion mode off

1: Auto-delayed conversion mode on

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 13 **CONT**: Single / continuous conversion mode for regular conversions

This bit is set and cleared by software. If it is set, regular 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 DISCEN = 1 and CONT = 1.

The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 12 **OVRMOD**: Overrun mode

This bit is set and cleared by software and configure the way data overrun is 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 = 0 (which ensures that no regular conversion is ongoing).

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection for regular channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.

00: Hardware trigger detection disabled (conversions can be launched 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 these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 9:5 **EXTSEL[4:0]**: External trigger selection for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

00000: adc_ext_trg0
00001: adc_ext_trg1
00010: adc_ext_trg2
00011: adc_ext_trg3
00100: adc_ext_trg4
00101: adc_ext_trg5
00110: adc_ext_trg6
00111: adc_ext_trg7
...
11111: adc_ext_trg31

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 4:3 **RES[1:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

00: 12-bit
01: 10-bit
10: 8-bit
11: 6-bit

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 2 Reserved, must be kept at reset value.

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 *Section : Managing conversions using the DMA*

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which 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 to use the DMA to manage automatically the converted data. For more details, refer to *Section : Managing conversions using the DMA*.

0: DMA disabled
1: DMA enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

19.6.5 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.	Res.	Res.	SMPTRIG	BULB	SWTRIG	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
Res.	Res.	Res.	Res.	Res.	ROVSM	TROVS	OVSS[3:0]				OVSR[2:0]			JOVSE	ROVSE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 27 **SMPTRIG**: Sampling time control trigger mode

This bit is set and cleared by software to enable the sampling time control trigger mode.

0: Sampling time control trigger mode disabled

1: Sampling time control trigger mode enabled

The sampling time starts on the trigger rising edge, and the conversion on the trigger falling edge.

EXTEN bit should be set to 01. BULB bit must not be set when the SMPTRIG bit is set.

When EXTEN bit is set to 00, set SWTRIG to start the sampling and clear SWTRIG bit to start the conversion.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 26 **BULB**: Bulb sampling mode

This bit is set and cleared by software to enable the bulb sampling mode.

0: Bulb sampling mode disabled

1: Bulb sampling mode enabled. The sampling period starts just after the previous end of conversion.

SAMPTRIG bit must not be set when the BULB bit is set.

The very first ADC conversion is performed with the sampling time specified in SMPx bits.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 25 **SWTRIG**: Software trigger bit for sampling time control trigger mode

This bit is set and cleared by software to enable the bulb sampling mode.

0: Software trigger starts the conversion for sampling time control trigger mode

1: Software trigger starts the sampling for sampling time control trigger mode

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 24:17 Reserved, must be kept at reset value.

Bits 16:11 Reserved, must be kept at reset value.

Bit 10 **ROVSM**: Regular oversampling mode

This bit is set and cleared by software to select the regular oversampling mode.

0: Continued mode: When injected conversions are triggered, the oversampling is temporary stopped and continued after the injection sequence (oversampling buffer is maintained during injected sequence)

1: Resumed mode: When injected conversions are triggered, the current oversampling is aborted and resumed from start after the injection sequence (oversampling buffer is zeroed by injected sequence start)

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 9 **TROVS**: Triggered Regular oversampling

This bit is set and cleared by software to enable triggered oversampling

0: All oversampled conversions for a channel are done consecutively following a trigger

1: Each oversampled conversion for a channel needs a new trigger

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 8:5 **OVSS[3:0]**: Oversampling shift

This bitfield is set and cleared by software to define the right shifting applied to the raw oversampling result.

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

Other codes reserved

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 4:2 **OVSR[2:0]**: Oversampling ratio

This bitfield is set and cleared by software to define the 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 these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 1 **JOVSE**: Injected oversampling Enable

This bit is set and cleared by software to enable injected oversampling.

0: Injected oversampling disabled

1: Injected oversampling enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)

Bit 0 **ROVSE**: Regular oversampling Enable

This bit is set and cleared by software to enable regular oversampling.

0: Regular oversampling disabled

1: Regular oversampling enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)

19.6.6 ADC sample time register 1 (ADC_SMPR1)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SMPPLUS	Res.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
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
SMP5[0]	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SMPPLUS:** Addition of one clock cycle to the sampling time.

1: 2.5 ADC clock cycle sampling time becomes 3.5 ADC clock cycles for the ADC_SMPR1 and ADC_SMPR2 registers.

0: The sampling time remains set to 2.5 ADC clock cycles remains

To make sure no conversion is ongoing, the software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0.

Bit 30 Reserved, must be kept at reset value.

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection (x = 9 to 0)

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

000: 2.5 ADC clock cycles

001: 6.5 ADC clock cycles

010: 12.5 ADC clock cycles

011: 24.5 ADC clock cycles

100: 47.5 ADC clock cycles

101: 92.5 ADC clock cycles

110: 247.5 ADC clock cycles

111: 640.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

19.6.7 ADC sample time register 2 (ADC_SMPR2)

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.	SMP19[2:0]			SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
		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
SMP15[0]	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
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.

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection (x = 19 to 10)

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

000: 2.5 ADC clock cycles
001: 6.5 ADC clock cycles
010: 12.5 ADC clock cycles
011: 24.5 ADC clock cycles
100: 47.5 ADC clock cycles
101: 92.5 ADC clock cycles
110: 247.5 ADC clock cycles
111: 640.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

19.6.8 ADC watchdog threshold register 1 (ADC_TR1)

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.	AWDFILT[2:0]			LT1[11:0]											
	rw	rw	rw	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 1.

Refer to *Section 19.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **AWDFILT[2:0]**: Analog watchdog filtering parameter

This bit is set and cleared by software.

000: No filtering

001: two consecutive detection generates an AWDx flag or an interrupt

...

111: Eight consecutive detection generates an AWDx flag or an interrupt

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

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 1.

Refer to *Section 19.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

19.6.9 ADC watchdog threshold register 2 (ADC_TR2)

Address offset: 0x24

Reset value: 0x00FF 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.	HT2[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.	LT2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **HT2[7:0]**: Analog watchdog 2 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 2.

Refer to *Section 19.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **LT2[7:0]**: Analog watchdog 2 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 2.

Refer to *Section 19.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

19.6.10 ADC watchdog threshold register 3 (ADC_TR3)

Address offset: 0x28

Reset value: 0x00FF 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.	HT3[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.	LT3[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **HT3[7:0]**: Analog watchdog 3 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 3.

Refer to *Section 19.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **LT3[7:0]**: Analog watchdog 3 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 3.

This watchdog compares the 8-bit of LT3 with the 8 MSB of the converted data.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

19.6.11 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4]
			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
SQ2[3:0]				Res.	SQ1[4:0]					Res.	Res.	L[3:0]			
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ4[4:0]:** 4th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 4th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ3[4:0]:** 3rd conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 3rd in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ2[4:0]:** 2nd conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 2nd in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ1[4:0]:** 1st conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 1st in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

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

Bits 3:0 **L[3:0]:** Regular channel sequence length
These bits are written by software to define the total number of conversions in the regular channel conversion sequence.
0000: 1 conversion
0001: 2 conversions
...
1111: 16 conversions
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

19.6.12 ADC regular sequence register 2 (ADC_SQR2)

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.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4]
			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
SQ7[3:0]				Res.	SQ6[4:0]					Res.	SQ5[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ9[4:0]:** 9th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 9th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ8[4:0]:** 8th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 8th in the regular conversion sequence

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ7[4:0]:** 7th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 7th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ6[4:0]:** 6th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 6th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ5[4:0]:** 5th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 5th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

19.6.13 ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4]
			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
SQ12[3:0]				Res.	SQ11[4:0]					Res.	SQ10[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ14[4:0]:** 14th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 14th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ13[4:0]:** 13th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 13th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ12[4:0]:** 12th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 12th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ11[4:0]:** 11th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 11th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ10[4:0]:** 10th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 10th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

19.6.14 ADC regular sequence register 4 (ADC_SQR4)

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.	SQ16[4:0]					Res.	SQ15[4:0]				
					rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:6 **SQ16[4:0]:** 16th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 16th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ15[4:0]:** 15th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 15th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

19.6.15 ADC regular 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
RDATA[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 **RDATA[15:0]**: Regular data converted

These bits are read-only. They contain the conversion result from the last converted regular channel. The data are left- or right-aligned as described in *Section 19.4.26: Data management*.

19.6.16 ADC injected sequence register (ADC_JSQR)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:1]			
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
JSQ2[0]	Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[4:0]					JL[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 **JSQ4[4:0]:** 4th conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 4th in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 26 Reserved, must be kept at reset value.

Bits 25:21 **JSQ3[4:0]:** 3rd conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 3rd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 20 Reserved, must be kept at reset value.

Bits 19:15 **JSQ2[4:0]:** 2nd conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 2nd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 **JSQ1[4:0]:** 1st conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 1st in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bits 8:7 **JEXTEN[1:0]**: External trigger enable and polarity selection for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

00: If JQDIS = 0 (queue enabled), hardware and software trigger detection disabled. Otherwise, the queue is disabled as well as hardware trigger detection (conversions can be launched 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 these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

If JQM = 1 and if the Queue of Context becomes empty, the software and hardware triggers of the injected sequence are both internally disabled (refer to Section 19.4.21: Queue of context for injected conversions)

Bits 6:2 **JEXTSEL[4:0]**: External Trigger Selection for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

00000: adc_jext_trg0

00001: adc_jext_trg1

00010: adc_jext_trg2

00011: adc_jext_trg3

00100: adc_jext_trg4

00101: adc_jext_trg5

00110: adc_jext_trg6

00111: adc_jext_trg7

...

11111: adc_jext_trg31

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bits 1:0 **JL[1:0]:** Injected channel sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion

01: 2 conversions

10: 3 conversions

11: 4 conversions

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

19.6.17 ADC offset y register (ADC_OFRy)

Address offset: 0x60 + 0x04 * (y -1), (y = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OFFSET_EN	OFFSET_CH[4:0]					SATEN	OFFSETPOS	Res.	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
Res.	Res.	Res.	Res.	OFFSET[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OFFSET_EN:** Offset y enable

This bit is written by software to enable or disable the offset programmed into bits OFFSET[11:0].

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 30:26 **OFFSET_CH[4:0]:** Channel selection for the data offset y

These bits are written by software to define the channel to which the offset programmed into bits OFFSET[11:0] applies.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Some channels are not connected physically and must not be selected for the data offset y.

If OFFSET_EN is set, it is not allowed to select the same channel for different ADC_OFRy registers.

Bit 25 **SATEN**: Saturation enable

This bit is set and cleared by software to enable the saturation at 0x000 and 0xFFF for the offset function.

0: No saturation control, offset result can be signed

1: Saturation enabled, offset result unsigned and saturated at 0x000 and 0xFFF

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 24 **OFFSETPOS**: Positive offset

This bit is set and cleared by software to enable the positive offset.

0: Negative offset

1: Positive offset

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:0 **OFFSET[11:0]:** Data offset y for the channel programmed into bits OFFSET_CH[4:0]

These bits are written by software to define the offset to be subtracted from the raw converted data when converting a channel (can be regular or injected). The channel to which applies the data offset must be programmed in the bits OFFSET_CH[4:0]. The conversion result can be read from in the ADC_DR (regular conversion) or from in the ADC_JDRyi registers (injected conversion).

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

If several offset (OFFSET) point to the same channel, only the offset with the lowest x value is considered for the subtraction.
Ex: if OFFSET1_CH[4:0] = 4 and OFFSET2_CH[4:0] = 4, this is OFFSET1[11:0] which is subtracted when converting channel 4.

19.6.18 ADC injected channel y data register (ADC_JDRy)

Address offset: 0x80 + 0x04 * (y - 1), (y = 1 to 4)

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
JDATA[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 **JDATA[15:0]:** Injected data

These bits are read-only. They contain the conversion result from injected channel y. The data are left -or right-aligned as described in *Section 19.4.26: Data management*.

19.6.19 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.	AWD2CH[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD2CH[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 **AWD2CH[19:0]**: Analog watchdog 2 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2.

AWD2CH[i] = 0: ADC analog input channel i is not monitored by AWD2

AWD2CH[i] = 1: ADC analog input channel i is monitored by AWD2

When AWD2CH[19:0] = 000..0, the analog Watchdog 2 is disabled

Note: The channels selected by AWD2CH must be also selected into the SQRi or JSQRi registers.

The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Some channels are not connected physically and must not be selected for the analog watchdog.

19.6.20 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.	AWD3CH[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD3CH[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 **AWD3CH[19:0]**: Analog watchdog 3 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 3.

AWD3CH[i] = 0: ADC analog input channel i is not monitored by AWD3

AWD3CH[i] = 1: ADC analog input channel i is monitored by AWD3

When AWD3CH[19:0] = 000..0, the analog Watchdog 3 is disabled

Note: The channels selected by AWD3CH must be also selected into the SQRi or JSQRi registers.

The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Some channels are not connected physically and must not be selected for the analog watchdog.

19.6.21 ADC Differential mode selection register (ADC_DIFSEL)

Address offset: 0xB0

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.	DIFSEL[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIFSEL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **DIFSEL[19:0]:** Differential mode for channels 19 to 0.

These bits are set and cleared by software. They allow to select if a channel is configured as single-ended or Differential mode.

DIFSEL[i] = 0: ADC analog input channel is configured in single-ended mode

DIFSEL[i] = 1: ADC analog input channel i is configured in Differential mode

Note: The DIFSEL bits corresponding to channels that are either connected to a single-ended I/O port or to an internal channel must be kept their reset value (single-ended input mode).

The software is allowed to write these bits only when the ADC is disabled (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

19.6.22 ADC calibration factors (ADC_CALFACT)

Address offset: 0xB4

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.	CALFACT_D[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.	CALFACT_S[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **CALFACT_D[6:0]**: Calibration Factors in differential mode

These bits are written by hardware or by software.

Once a differential inputs 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 differential calibration is launched.

Note: The software is allowed to write these bits only when ADEN = 1, ADSTART = 0 and JADSTART = 0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **CALFACT_S[6:0]**: Calibration Factors In single-ended mode

These bits are written by hardware or by software.

Once a single-ended inputs 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 single-ended calibration is launched.

Note: The software is allowed to write these bits only when ADEN = 1, ADSTART = 0 and JADSTART = 0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

19.6.23 ADC option register (ADC_OR)

Address offset: 0xC8

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.	OP1	OP0
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OP1**: Option bit 1
0: V_{DDCORE} channel disabled
1: V_{DDCORE} channel enabled

Bit 0 **OP0**: Option bit 0
0: INP0/INN1 GPIO switch control disabled
1: INP0/INN1 GPIO switch control enabled
Note: This option bit must be set to 1 when ADC1_INP0 or ADC1_INN1 channel is selected.

Note: ADC_OR register might be reserved on some ADC instances. Refer to Section 19.3: ADC implementation.

19.7 ADC common registers

19.7.1 ADC common control 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.	VBATEN	TSEN	VREFEN	PRESC[3:0]				CKMODE[1:0]	
							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
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 **VBATEN**: VBAT enable
This bit is set and cleared by software to control.
0: V_{BAT} channel disabled
1: V_{BAT} channel enabled

Bit 23 **TSEN**: V_{SENSE} enable
This bit is set and cleared by software to control V_{SENSE}.
0: Temperature sensor channel disabled
1: Temperature sensor channel enabled

Bit 22 **VREFEN**: V_{REFINT} enable
This bit is set and cleared by software to enable/disable the V_{REFINT} channel.
0: V_{REFINT} channel disabled
1: V_{REFINT} channel enabled

Bits 21:18 **PRESC[3:0]**: ADC prescaler

These bits are set and cleared by software to select the frequency of the clock to the ADC. The clock is common for all the ADCs.

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: The software is allowed to write these bits only when the ADC is disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0). The ADC prescaler value is applied only when CKMODE[1:0] = 0b00.

Bits 17:16 **CKMODE[1:0]:** ADC clock mode

These bits are set and cleared by software to define the ADC clock scheme (which is common to both master and slave ADCs):

00: adc_ker_ck (x = 1) (Asynchronous clock mode), generated at product level (refer to *Section 6: Reset and clock control (RCC)*)

01: adc_hclk/1 (Synchronous clock mode). This configuration must be enabled only if the AHB clock prescaler is set to 1 (HPRE[3:0] = 0XXX in RCC_CFGR register) and if the system clock has a 50% duty cycle.

10: adc_hclk/2 (Synchronous clock mode)

11: adc_hclk/4 (Synchronous clock mode)

In all synchronous clock modes, there is no jitter in the delay from a timer trigger to the start of a conversion.

Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 15:0 Reserved, must be kept at reset value.

19.7.2 ADC hardware configuration register (ADC_HWCFGR0)

Address offset: 0x3F0

Reset value: 0x0000 1211

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDLEVALUE[3:0]				OPBITS[3:0]				MULPIPE[3:0]				ADCNUM[3:0]			
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 15:12 **IDLEVALUE[3:0]**: Idle value for non-selected channels
0000: Dummy channel selection is 0x13
0001: Dummy channel selection is 0x1F

Bits 11:8 **OPBITS[3:0]**: Number of option bits
0000: No option register implemented
0002: 2 option bits implemented in the ADC option register (ADC_OR) at address offset 0xC8

Bits 7:4 **MULPIPE[3:0]**: Number of pipeline stages
0001: One-stage pipeline

Bits 3:0 **ADCNUM[3:0]**: Number of ADCs implemented
0001: One ADC instance implemented
0010: Two ADC instances implemented
0011: Three ADCs instances implemented

19.7.3 ADC version register (ADC_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0012

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision
These bits returns the ADC IP major revision
0001: Major revision = 1.X

Bits 3:0 **MINREV[3:0]**: Minor revision
These bits returns the ADC IP minor revision
0001: Minor revision = X.1
0002: Minor revision = X.2
0003: Minor revision = X.3

19.7.4 ADC identification register (ADC_IPDR)

Address offset: 0x3F8

Reset value: 0x0011 0006

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Peripheral identifier

These bits returns the ADC identifier.

ID[31:0] = 0x0011 0006: c7amba_aditf5_90_v1

19.7.5 ADC size identification register (ADC_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[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
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size Identification

SID[31:8]: fixed code that characterizes the ADC_SIDR register. This field is always read at 0xA3C5DD.

SID[7:0]: read-only numeric field that returns the address offset (in Kbytes) of the identification registers from the IP base address:

0x01: 1 Kbytes address offset

0x02: 2 Kbytes address offset

0x04: 4 Kbytes address offset

0x08: 8 Kbytes address offset

19.8 ADC register map

Table 120. ADC global register map

Offset	Register
0x000 - 0x0B4	Master ADC1/2
0x0B8 - 0x2FC	Reserved
0x300 - 0x30C	Master and slave ADCs common registers

Table 121. ADC register map and reset values

Offset	Register name reset value	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.	Res.	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
	Reset value																						0	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.	Res.	Res.	JQOVFIE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x08	**ADC_CR**	ADCAL	ADCALDIF	DEEPPWD	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
	Reset value	0	0	1	0																							0	0	0	0	0	0
0x0C	**ADC_CFGR**	JQDIS	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM[2:0]		DISCEN	ALIGN	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL4	EXTSEL3	EXTSEL2	EXTSEL1	EXTSEL0	RES[1:0]		Res.	DMACFG	DMAEN
	Reset value	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0
0x0C	**ADC_CFGR2**	Res.	Res.	Res.	Res.	SMPTRIG	BULB	SWTRIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ROVSM	TROVS	OVSS[3:0]				OVSR[2:0]			JOVSE	ROVSE
	Reset value					0	0	0															0	0	0	0	0	0	0	0	0	0	0
0x14	**ADC_SMPR1**	SMPPLUS	Res.	SMP9 [2:0]			SMP8 [2:0]			SMP7 [2:0]			SMP6 [2:0]			SMP5 [2:0]			SMP4 [2:0]			SMP3 [2:0]			SMP2 [2:0]			SMP1 [2:0]			SMP0 [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
0x18	**ADC_SMPR2**	Res.	Res.	SMP19 [2:0]			SMP18 [2:0]			SMP17 [2:0]			SMP16 [2:0]			SMP15 [2:0]			SMP14 [2:0]			SMP13 [2:0]			SMP12 [2:0]			SMP11 [2:0]			SMP10 [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
0x1C	Reserved	Res.																															
0x20	**ADC_TR1**	Res.	Res.	Res.	Res.	HT1[11:0]												Res.	AWDFILT [2:0]			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	0	0	0
0x24	**ADC_TR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT2[[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LT2[7:0]							
	Reset value									1	1	1	1	1	1	1	1									0	0	0	0	0	0	0	0
0x28	**ADC_TR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT3[[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LT3[7:0]							
	Reset value									1	1	1	1	1	1	1	1									0	0	0	0	0	0	0	0

Table 121. ADC register map and reset values (continued)

Offset	Register name reset value	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	Reserved	Res.																															
0x30	**ADC_SQR1**	Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4:0]					Res.	SQ1[4:0]					Res.	Res.	L[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
0x34	**ADC_SQR2**	Res.	Res.	Res.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4:0]					Res.	SQ6[4:0]					Res.	SQ5[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
0x38	**ADC_SQR3**	Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4:0]					Res.	SQ11[4:0]					Res.	SQ10[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
0x3C	**ADC_SQR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
	Reset value																						0	0	0	0	0		0	0	0	0	0
0x40	**ADC_DR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	regular RDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44-0x48	Reserved	Res.																															
0x4C	**ADC_JSQR**	JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:0]				Res.	JSQ1[4:0]				JEXTEN[1:0]		JEXTSEL [4:0]				JL[1:0]			
	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
0x50-0x5C	Reserved	Res.																															
0x60	**ADC_OFR1**	OFFSET_EN	OFFSET_ CH[4:0]					SATEN	OFFSETPOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[11:0]											
	Reset value	0	0	0	0	0	0	0	0													0	0	0	0	0	0	0	0	0	0	0	0
0x64	**ADC_OFR2**	OFFSET_EN	OFFSET_ CH[4:0]					SATEN	OFFSETPOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[11:0]											
	Reset value	0	0	0	0	0	0	0	0													0	0	0	0	0	0	0	0	0	0	0	0
0x68	**ADC_OFR3**	OFFSET_EN	OFFSET_ CH[4:0]					SATEN	OFFSETPOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[11:0]											
	Reset value	0	0	0	0	0	0	0	0													0	0	0	0	0	0	0	0	0	0	0	0
0x6C	**ADC_OFR4**	OFFSET_EN	OFFSET_ CH[4:0]					SATEN	OFFSETPOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[11:0]											
	Reset value	0	0	0	0	0	0	0	0													0	0	0	0	0	0	0	0	0	0	0	0
0x70-0x7C	Reserved	Res.																															
0x80	**ADC_JDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x84	**ADC_JDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 121. ADC register map and reset values (continued)

Offset	Register name reset value	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
0x88	**ADC_JDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C	**ADC_JDR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x90-0x9C	Reserved	Res.																															
0xA0	**ADC_AWD2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD2CH[19:0]																			
	Reset value													0	0	0	0	0	0	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.	AWD3CH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA8-0xAC	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.
0xB0	**ADC_DIFSEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIFSEL[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB4	**ADC_CALFACT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_D[6:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_S[6:0]							
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0xB8-0xC4	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.
0xC8	**ADC_OR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OP1	OP0
	Reset value																															0	0
0xCC-0xFC	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.

Table 122. ADC register map and reset values (master and slave ADC common registers)

Offset	Register name reset value	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
0x304	Reserved	Res.																															
0x308	**ADC_CCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBATEN	TSEN	VREFEN	PRESC[3:0]				CKMODE[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0	0	0	0	0	0	0	0	0																
x30C-0x3EC	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.
0x3F0	**ADC_HWCFGR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLEVALUE [3:0]				OPBITS[3:0]				MULPIPE[3:0]				ADCNUM[3:0]			
	Reset value	0x0000 1211																															
0x3F4	**ADC_VERR**		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value	0x0000 0012																															
0x3F8	**ADC_IPDR**	ID[31:0]																															
	Reset value	0	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	1	1	0
0x3FC	**ADC_SIDR**	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.2 on page 70* for the register boundary addresses.

20 Digital temperature sensor (DTS)

20.1 Introduction

The device embeds a sensor that converts the temperature into a square wave which frequency is proportional to the temperature. The frequency is measured either with the PCLK or the LSE clock.

20.2 DTS main features

The temperature sensor block main features are the following:

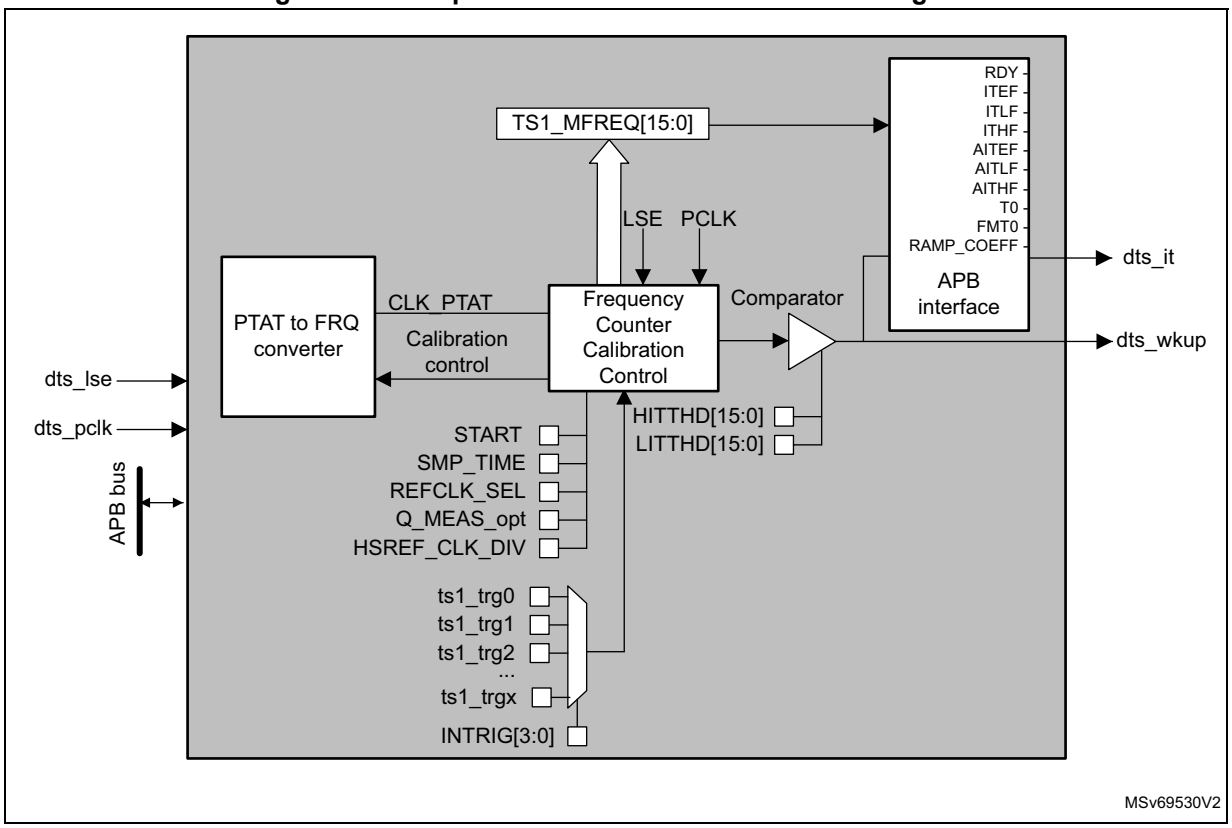
- Start of measurement triggered by software or 4 hardware sources
- Programmable sampling time to increase temperature measurement precision
- Counter synchronized on LSE or PCLK clock
- Temperature watchdog on low and high threshold
- Interrupt generation when the temperature is lower or higher than predefined thresholds and at the end of measurement.
- Asynchronous wakeup signal generation when the temperature is higher/lower than a predefined threshold (LSE mode only)
- Quick measurement using LSE clock

20.3 DTS functional description

20.3.1 DTS block diagram

The temperature sensor block diagram is shown in *Figure 130*.

Figure 130. Temperature sensor functional block diagram



20.3.2 DTS internal signals

Table 123. DTS internal input/output signals

Signal name	Signal type	Description
dts_lse	Digital input	LSE clock
dts_pclk	Digital input	APB clock
dts_it	Digital output	Temperature sensor interrupt
dts_wkup	Digital output	Temperature sensor wakeup

20.3.3 DTS block operation

The analog part of the temperature sensor outputs a frequency that is proportional to the absolute temperature (CLK_PTAT). The frequency measurement is based on the PCLK or the LSE clock.

Before each measurement, the temperature sensor performs a calibration of the frequency generation blocks.

20.3.4 Operating modes

Several operating modes can be selected by setting the REFCLK_SEL bit in *Temperature sensor configuration register 1 (DTS_CFGR1)*:

- PCLK only (REFCLK_SEL = 0)

  The temperature sensor registers can be accessed. The interface can consequently be reconfigured and the measurement sequence is performed using PCLK clock
- PCLK and LSE (REFCLK_SEL = 1)

  The temperature sensor registers can be accessed. The interface can consequently be reconfigured and the measurement sequence is performed using the LSE clock.
- LSE only (REFCLK_SEL = 1) and PCLK OFF

  The registers cannot be accessed. The measurement can be performed using the LSE clock. This mode is used to exit from Sleep mode by using hardware triggers and the asynchronous interrupt line.

20.3.5 Calibration

The temperature sensor must run the calibration prior to any frequency measurement. The calibration is performed automatically when the temperature measurement is triggered except for quick measurement mode (Q_MEAS_OPT set to 1 in DTS_CFGR1).

20.3.6 Prescaler

When a calibration is ongoing, the counter clock must be slower than 1 MHz. This is achieved by the PCLK clock prescaler embedded in the temperature sensor.

During the temperature measurement period, the prescaler is bypassed.

- When PCLK is used as reference clock (REFCLK_SEL set to 0 in DTS_CFGR1), a prescaler is used. Its division ratio must be configured up to 127 (refer to the HSREF_CLK_DIV[6:0] register definition for the divider setting).
- When LSE is used as reference clock (REFCLK_SEL set to 1 in DTS_CFGR1), the timebase is equal to 2 LSE periods. In this case, no prescaler is used.

20.3.7 Temperature measurement principles

The analog part of temperature sensor outputs a signal (CLK_PTAT) which FM(T) frequency is temperature-dependent.

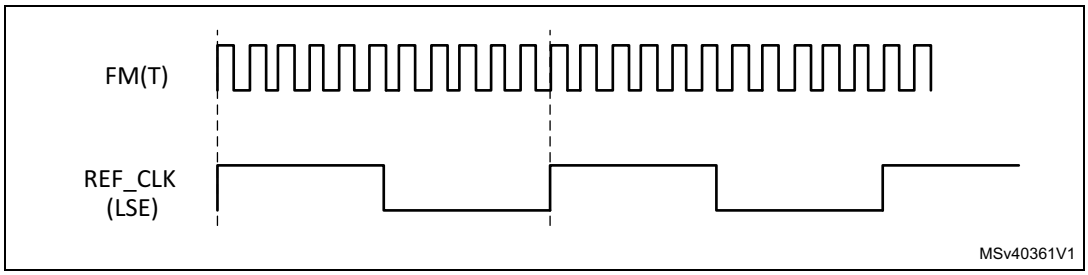
Either PCLK or LSE can be selected as reference clock (REF_CLK) through the REFCLK_SEL bit in DTS_CFGR1.

The counting method depends on the REF_CLK frequency. This is due to the fact that two counters are implemented in the temperature sensor block:

- For low REF_CLK frequencies, a counting of FM(T) cycles is performed during one or several REF_CLK cycles.
- For high REF_CLK frequencies, a counting of REF_CLK cycles is performed during one or several FM(T) cycles.

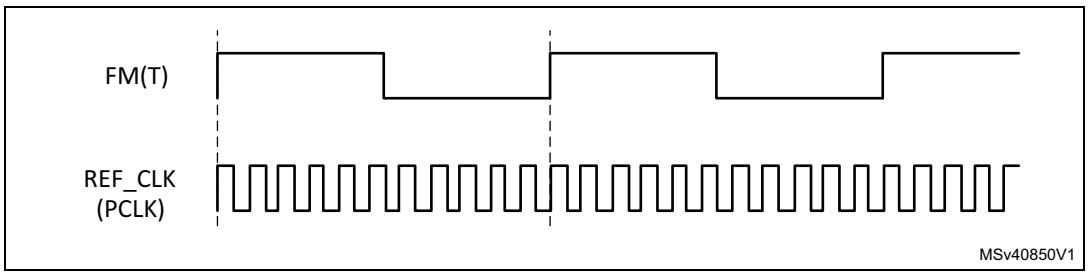
This counter behavior is shown in *Figure 131* and *Figure 132*.

Figure 131. Method for low REF_CLK frequencies



1. To increase the precision, FM(T) measurement can be done on several LSE periods.

Figure 132. Method for high REF_CLK frequencies



1. To increase the precision, PCLK measurement can be done on several FM(T) periods.

The counting result is stored in the DTS_DR register (see *Temperature sensor data register (DTS_DR)*.

Once the FM(T) frequency has been obtained, the corresponding temperature can be calculated by software using the following formula:

- When PCLK is used:

$$T = T_0 + ((F_{PCLK} / TS1\_MFREQ) \times TS1\_SMP\_TIME - 100 \times TS1\_FMT0) / TS1\_RAMP\_COEFF$$

where

T0 (factory calibration temperature) is equal to 30 °C.

TS1_FMT0 is measured and stored in the DTS_T0VALR1 register. It is expressed in hundreds of Hertz.

TS1_RAMP_COEFF is measured during tests in factory and stored in DTS_RAMPVALR register. This value is expressed in Hz/°C.

- When the LSE clock is used

$$T = T_0 + ((F_{LSE} \times TS1\_MFREQ / TS1\_SMP\_TIME) - (100 \times TS1\_FMT0)) / TS1\_RAMP\_COEFF$$

20.3.8 Sampling time

The sampling period can be increased to improve measurement accuracy. This is useful when the reference frequency (REF_CLK) is close to the FM(T) frequency. The default value is one REF_CLK cycle in LSE mode, and one FM(T) cycle in PCLK mode.

The sampling time is configured through TS1_SMP_TIME bits in DTS_CFGR1 register (see *Table 124*).

Table 124. Sampling time configuration

TS1_SMP_TIME[3:0]	LSE or FM(T) clock cycle(s)
0000	1
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

20.3.9 Quick measurement mode

If a high precision is not required, the calibration step included in each measurement sequence can be skipped by setting Q_MEAS_OPT to 1 in the DTS_CFGR1 register. This method must used only when the LSE clock is selected as reference clock (LSREF_CLK set to 1). This mode can reduce the measurement time.

20.3.10 Trigger input

The temperature measurement can be triggered either by software or by an external event. The trigger source can be selected through TS1_INTRIG[3:0] bits in DTS_CFGR1.

Table 125. Trigger configuration

Name	TS1_INTRIG[3:0]				Comment
N.A	0	0	0	0	No hardware trigger
ts1_trg0	0	0	0	1	lptim1_ch1
ts1_trg1	0	0	1	0	lptim2_ch1
ts1_trg2	0	0	1	1	Reserved
ts1_trg3	0	1	0	0	exti13
ts1_trg4	0	1	0	1	Reserved
ts1_trg5	0	1	1	0	
ts1_trg6	0	1	1	1	
ts1_trg7	1	0	0	0	
ts1_trg8	1	0	0	1	
ts1_trg9	1	0	1	0	
ts1_trg10	1	0	1	1	
ts1_trg11	1	1	0	0	
ts1_trg12	1	1	0	1	
ts1_trg13	1	1	1	0	
ts1_trg14	1	1	1	1	

Note: Hardware triggers are active only on the rising edge.

The temperature sensor can only capture a hardware trigger rising edge when TS1_RDY bit is set (see Section 20.3.11: On-off control and ready flag), otherwise the trigger is ignored.

If a trigger source changes on-the-fly, the new trigger source signal should be low. If the new source signal is high, the temperature sensor detects a rising edge and start the measurement sequence.

20.3.11 On-off control and ready flag

The DTS block can be enabled by setting TS1_EN bit in DTS_CFGR1 register. The TS1_RDY flag in the *Temperature sensor status register (DTS_SR)* indicate that the DTS block is ready for temperature measurement: when TS1_RDY bit is set to 1, the measurement can be started. Once a measurement has started, TS1_RDY bit is reset. New measurement requests are then ignored. Once the measurement is finished, TS1_RDY bit is set again to indicate the sensor is ready to start a new measurement.

20.3.12 Temperature measurement sequence

Start of measurement can be triggered by software or hardware.

Software trigger

The software trigger is selected when TS1_INTRIG_SEL[3:0] is set to ‘0000’ in DTS_CFGR1.

If TS1_RDY is set to 1, writing TS1_START bit to 1 in DTS_CFGR1 starts the measurement.

If TS1_RDY equals 0, the software trigger does not start until TS1_RDY is set.

If TS1_START bit is kept at 1 once the measurement is finished, then the TS1_RDY flag become 1 and the measurement restarts.

Hardware trigger

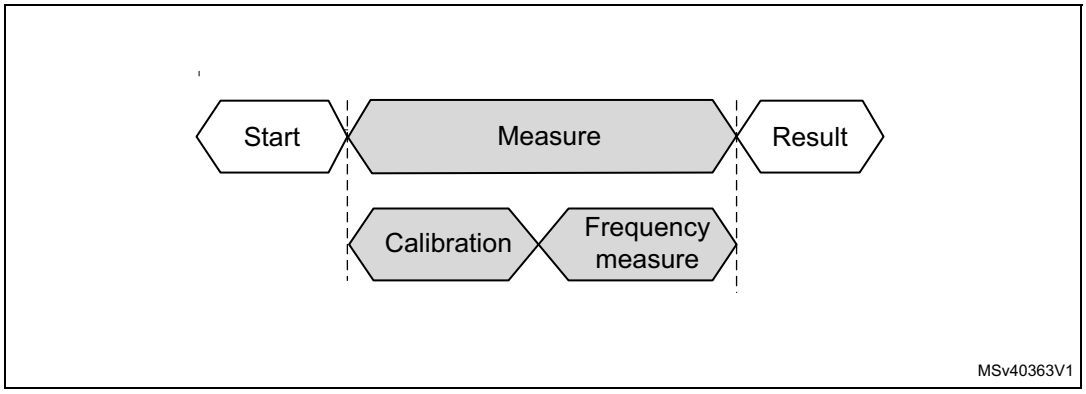
TS1_INTRIG_SEL[3:0] bits allow selecting one hardware trigger out of 4. If TS1_RDY is set to 1, a rising edge on the trigger signal starts the measurement. When TS1_RDY is 0, the rising edge is ignored.

Temperature measurement sequence

One measurement contains two steps: the calibration of the analog blocks and the measurement. The calibration automatically starts when the measurement is triggered (see *Section 20.3.5: Calibration*). The measurement period depends on the following DTS_CFGR1 bits:

- the reference clock selected through REFCLK_SEL bit
- the divider ratio configured by HSREF_CLK_DIV bits
- the sampling time defined by TS1_SMP_TIME bits.

Figure 133. Temperature sensor sequence



20.4 DTS low-power modes

Table 126. Temperature sensor behavior in low-power modes

Mode	Description
Sleep	Only works in LSE mode. DTS interrupt causes the device to exit from Sleep mode.
Stop	Only works in LSE mode. DTS interrupt cause the device to exit from Stop mode.

20.5 DTS interrupts

There are two ways to use the DTS block as an interrupt source.The DTS interrupt line can be connected to the EXTI controller (see *Section 20.5.3: Asynchronous wakeup*) or to the CPU NVIC (see *Section 20.5.2: Synchronous interrupt*).

20.5.1 Temperature window comparator

The DTS_ITR1 register allows defining the high and low threshold that are used for temperature comparison. If the temperature data is equal or higher than TS1_HITTHD, or equal or lower than TS1_LITTHD bit, an interrupt is generated and the corresponding flag, TS1_ ITLF, TS1_ITHF, TS1_AITLF and TS1_AITHF, is set in the DTS_SR register (see *Section 20.6.6*).

20.5.2 Synchronous interrupt

A global interrupt output line is available on the DTS block. The interrupt can be generated at the end of measurement and/or when the measurement result is equal/higher or equal/lower than a predefined threshold (see *Section 20.5.1: Temperature window comparator*).

Three interrupt events can be select via 3 bits in DTS_ITENR register (see *Section 20.6.7*). All combinations of interrupts are allowed.

The TS1_ITEF, TS1_ITLF and TS1_ITHF flags in the DTS_SR register reflect the interrupt event. They can be reset with the correspond bits of the DTS_ICIFR register (see *Section 20.6.8*).

20.5.3 Asynchronous wakeup

The DTS block also provides an asynchronous interrupt line. It is used only when the LSE is selected as reference clock (REFCLK_SEL=1).

This line can generate a signal that wakes up the system from Sleep mode at the end of measurement and/or when the measurement result is equal/higher or equal/lower than a predefined threshold (see *Section 20.5.1: Temperature window comparator*).

Three asynchronous wakeup events can be selected via 3 bits in DTS_ITENR register. All combination of interrupts are allowed.

The TS1_AITEF, TS1_AITLF and TS1_AITHF flags in the DTS_SR register reflect the interrupt status. They can be reset with the correspond bits of the DTS_ICIFR register.

The following table shows the interrupt bits an the their description.

Table 127. Interrupt control bits

Interrupt event	Interrupt flag	Enable control bit	Interrupt clear bit	Exit from Sleep mode	Synchronous/ Asynchronous
At the end of measurement	TS1_ITEF in DTS_SR	TS1_ITEEN in DTS_ITENR	TS1_CITEF in DTS_ICIFR	NO	Synchronous on PCLK
When the measure is equal or exceeds the low threshold	TS1_ITLF in DTS_SR	TS1_ITLEN in DTS_ITENR	TS1_CITLF in DTS_ICIFR	NO	
When the measure is equal or exceeds the high threshold	TS1_ITHF in DTS_SR	TS1_ITHEN in DTS_ITENR	TS1_CITHF in DTS_ICIFR	NO	
At the end of measurement	TS1_AITEF in DTS_SR	TS1_AITEEN in DTS_ITENR	TS1_CAITEF in DTS_ICIFR	YES	Asynchronous
When the measure is equal or exceeds the low threshold	TS1_AITLF in DTS_SR	TS1_AITLEN in DTS_ITENR	TS1_CAITLF in DTS_ICIFR	YES	
When the measure is equal or exceeds the high threshold	TS1_AITHF in DTS_SR	TS1_AITHEN in DTS_ITENR	TS1_CAITHF in DTS_ICIFR	YES	

20.6 DTS registers

The registers of this peripheral can only be accessed by-word (32-bit).

20.6.1 Temperature sensor configuration register 1 (DTS_CFGR1)

DTS_CFGR1 is the configuration register for temperature sensor 1.

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	HSREF_CLK_DIV[6:0]							Res.	Res.	Q_MEAS_OPT	REFCLK_SEL	TS1_SMP_TIME[3:0]			
	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.	Res.	Res.	TS1_INTRIG_SEL[3:0]				Res.	Res.	Res.	TS1_START	Res.	Res.	Res.	TS1_EN
				rw	rw	rw	rw				rw				rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **HSREF_CLK_DIV[6:0]**: High speed clock division ratio

These bits are set and cleared by software. They can be used to define the division ratio for the main clock in order to obtain the internal frequency lower than 1 MHz required for the calibration. They are applicable only for calibration when PCLK is selected as reference clock (REFCLK_SEL=0).

0000000: No divider

0000001: No divider

0000010: 1/2 division ratio

...

1111111: 1/127 division ratio

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **Q_MEAS_OPT**: Quick measurement option bit

This bit is set and cleared by software. It is used to increase the measurement speed by suppressing the calibration step. It is effective only when the LSE clock is used as reference clock (REFCLK_SEL=1).

0: Measurement with calibration

1: Measurement without calibration

Bit 20 **REFCLK_SEL**: Reference clock selection bit

This bit is set and cleared by software. It indicates whether the reference clock is the high speed clock (PCLK) or the low speed clock (LSE).

0: High speed reference clock (PCLK)

1: Low speed reference clock (LSE)

Bits 19:16 **TS1_SMP_TIME[3:0]**: Sampling time for temperature sensor 1

These bits allow increasing the sampling time to improve measurement precision.

When the PCLK clock is selected as reference clock (REFCLK_SEL = 0), the measurement is performed at TS1_SMP_TIME period of CLK_PTAT.

When the LSE is selected as reference clock (REFCLK_SEL = 1), the measurement is performed at TS1_SMP_TIME period of LSE.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TS1_INTRIG_SEL[3:0]**: Input trigger selection bit for temperature sensor 1

These bits are set and cleared by software. They select which input triggers a temperature measurement. Refer to *Section 20.3.10: Trigger input*.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **TS1_START:** Start frequency measurement on temperature sensor 1

This bit is set and cleared by software.

0: No software trigger.

1: Software trigger for a frequency measurement. (only if TS1 is ready).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **TS1_EN:** Temperature sensor 1 enable bit

This bit is set and cleared by software.

0: Temperature sensor 1 disabled

1: Temperature sensor 1 enabled

Note: Once enabled, the temperature sensor is active after a specific delay time. The TS1_RDY flag is set when the sensor is ready.

20.6.2 Temperature sensor T0 value register 1 (DTS_T0VALR1)

DTS_T0VALR1 contains the value of the factory calibration temperature (T0) for temperature sensor 1. The reset value is factory trimmed.

Address offset: 0x08

Reset value: 0x000X 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.	TS1_T0[1:0]	
														r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS1_FMT0[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 17:16 **TS1_T0[1:0]**: Engineering value of the T0 temperature for temperature sensor 1.
00: 30 °C
01: 130 °C
Others: Reserved, must not be used.

Bits 15:0 **TS1_FMT0[15:0]**: Engineering value of the frequency measured at T0 for temperature sensor 1
This value is expressed in 0.1 kHz.

20.6.3 Temperature sensor ramp value register (DTS_RAMPVALR)

The DTS_RAMPVALR is the ramp coefficient for the temperature sensor. The reset value is factory trimmed.

Address offset: 0x10

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
TS1_RAMP_COEFF[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 **TS1_RAMP_COEFF[15:0]**: Engineering value of the ramp coefficient for the temperature sensor 1.
This value is expressed in Hz/°C.

20.6.4 Temperature sensor interrupt threshold register 1 (DTS_ITR1)

DTS_ITR1 contains the threshold values for sensor 1.

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TS1_HITTHD[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
TS1_LITTHD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **TS1_HITTHD[15:0]:** High interrupt threshold for temperature sensor 1
These bits are set and cleared by software. They indicate the highest value than can be reached before raising an interrupt signal.

Bits 15:0 **TS1_LITTHD[15:0]:** Low interrupt threshold for temperature sensor 1
These bits are set and cleared by software. They indicate the lowest value than can be reached before raising an interrupt signal.

20.6.5 Temperature sensor data register (DTS_DR)

The DTS_DR contains the number of REF_CLK cycles used to compute the FM(T) frequency.

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
TS1_MFREQ[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 **TS1_MFREQ[15:0]**: Value of the counter output value for temperature sensor 1

20.6.6 Temperature sensor status register (DTS_SR)

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.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS1_RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_AITHF	TS1_AITLF	TS1_AITEF	Res.	TS1_ITHF	TS1_ITLF	TS1_ITEF
r									r	r	r		r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **TS1_RDY:** Temperature sensor 1 ready flag
This bit is set and reset by hardware.
It indicates that a measurement is ongoing.
0: Temperature sensor 1 busy
1: Temperature sensor 1 ready

Bits 14:7 Reserved, must be kept at reset value.

Bit 6 **TS1_AITHF:** Asynchronous interrupt flag for high threshold on temperature sensor 1
This bit is set by hardware when the high threshold is reached.
It is cleared by software by writing 1 to the TS1_CAITHF bit in the DTS_ICIFR register.
0: High threshold not reached on temperature sensor 1
1: High threshold reached on temperature sensor 1
Note: This bit is active only when the TS1_AITHFEN bit is set

Bit 5 **TS1_AITLF:** Asynchronous interrupt flag for low threshold on temperature sensor 1
This bit is set by hardware when the low threshold is reached.
It is cleared by software by writing 1 to the TS1_CAITLF bit in the DTS_ICIFR register.
0: Low threshold not reached on temperature sensor 1
1: Low threshold reached on temperature sensor 1
Note: This bit is active only when the TS1_AITLFEN bit is set

Bit 4 **TS1_AITEF:** Asynchronous interrupt flag for end of measure on temperature sensor 1
This bit is set by hardware when a temperature measure is done.
It is cleared by software by writing 1 to the TS1_CAITEF bit in the DTS_ICIFR register.
0: End of measure not detected on temperature sensor 1
1: End of measure detected on temperature sensor 1
Note: This bit is active only when the TS1_AITEFEN bit is set

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TS1_ITHF:** Interrupt flag for high threshold on temperature sensor 1, synchronized on PCLK
This bit is set by hardware when the high threshold is set and reached.
It is cleared by software by writing 1 to the TS1_CITHF bit in the DTS_ICIFR register.
0: High threshold not reached on temperature sensor 1
1: High threshold reached on temperature sensor 1
Note: This bit is active only when the TS1_ITHFEN bit is set

Bit 1 **TS1_ITLF:** Interrupt flag for low threshold on temperature sensor 1, synchronized on PCLK.
This bit is set by hardware when the low threshold is set and reached.
It is cleared by software by writing 1 to the TS1_CITLF bit in the DTS_ICIFR register.
0: Low threshold not reached on temperature sensor 1
1: Low threshold reached on temperature sensor 1
Note: This bit is active only when the TS1_ITLFEN bit is set

Bit 0 **TS1_ITEF:** Interrupt flag for end of measurement on temperature sensor 1, synchronized on PCLK.
This bit is set by hardware when a temperature measure is done.
It is cleared by software by writing 1 to the TS2_CITEF bit in the DTS_ICIFR register.
0: No end of measurement detected on temperature sensor 1
1: End of measure detected on temperature sensor 1
Note: This bit is active only when the TS1_ITEFEN bit is set

20.6.7 Temperature sensor interrupt enable register (DTS_ITENR)

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.	Res.	TS1_ AITHEN	TS1_ AITLEN	TS1_ AITEEN	Res.	TS1_ ITHEN	TS1_ ITLEN	TS1_ ITEEN
									rw	rw	rw		rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **TS1_AITHEN:** Asynchronous interrupt enable flag on high threshold for temperature sensor 1.
This bit are set and cleared by software.
It enables the asynchronous interrupt when the temperature is above the high threshold (only when REFCLK_SEL= 1")
0: Asynchronous interrupt on high threshold disabled for temperature sensor 1
1: Asynchronous interrupt on high threshold enabled for temperature sensor 1

Bit 5 **TS1_AITLEN:** Asynchronous interrupt enable flag for low threshold on temperature sensor 1.
This bit are set and cleared by software.
It enables the asynchronous interrupt when the temperature is below the low threshold (only when REFCLK_SEL= 1)
0: Asynchronous interrupt on low threshold disabled for temperature sensor 1
1: Asynchronous interrupt on low threshold enabled for temperature sensor 1

Bit 4 **TS1_AITEEN:** Asynchronous interrupt enable flag for end of measurement on temperature sensor 1

This bit are set and cleared by software.
It enables the asynchronous interrupt for end of measurement (only when REFCLK_SEL = 1).
0: Asynchronous interrupt for end of measurement disabled on temperature sensor 1
1: Asynchronous interrupt for end of measurement enabled on temperature sensor 1

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TS1_ITHEN:** Interrupt enable flag for high threshold on temperature sensor 1, synchronized on PCLK.

This bit are set and cleared by software.
It enables the interrupt when the measure reaches or is above the high threshold.
0: Synchronous interrupt for high threshold disabled on temperature sensor 1
1: Synchronous interrupt for high threshold enabled on temperature sensor 1

Bit 1 **TS1_ITLEN:** Interrupt enable flag for low threshold on temperature sensor 1, synchronized on PCLK.

This bit are set and cleared by software.
It enables the synchronous interrupt when the measure reaches or is below the low threshold.
0: Synchronous interrupt for low threshold disabled on temperature sensor 1
1: Synchronous interrupt for low threshold enabled on temperature sensor 1

Bit 0 **TS1_ITEEN:** Interrupt enable flag for end of measurement on temperature sensor 1, synchronized on PCLK.

This bit are set and cleared by software.
It enables the synchronous interrupt for end of measurement.
0: Synchronous interrupt for end of measurement disabled on temperature sensor 1
1: Synchronous interrupt for end of measurement enabled on temperature sensor 1

20.6.8 Temperature sensor clear interrupt flag register (DTS_ICIFR)

DTS_ICIFR is the control register for the interrupt flags.

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.	Res.	Res.	TS1_ CAITHF	TS1_ CAITLF	TS1_ CAITEF	Res.	TS1_ CITHF	TS1_ CITLF	TS1_ CITEF
									rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **TS1_CAITHF:** Asynchronous interrupt clear flag for high threshold on temperature sensor 1
Writing 1 to this bit clears the TS1_AITHF flag in the DTS_SR register.

Bit 5 **TS1_CAITLF:** Asynchronous interrupt clear flag for low threshold on temperature sensor 1
Writing 1 to this bit clears the TS1_AITLF flag in the DTS_SR register.

Bit 4 **TS1_CAITEF:** Write once bit. Clear the asynchronous IT flag for End Of Measure for thermal sensor 1.
Writing 1 clears the TS1_AITEF flag of the DTS_SR register.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TS1_CITHF:** Interrupt clear flag for high threshold on temperature sensor 1
Writing this bit to 1 clears the TS1_ITHF flag in the DTS_SR register.

Bit 1 **TS1_CITLF:** Interrupt clear flag for low threshold on temperature sensor 1
Writing 1 to this bit clears the TS1_ITLF flag in the DTS_SR register.

Bit 0 **TS1_CITEF:** Interrupt clear flag for end of measurement on temperature sensor 1
Writing 1 to this bit clears the TS1_ITEF flag in the DTS_SR register.

20.6.9 Temperature sensor option register (DTS_OR)

The DTS_OR contains general-purpose option bits.

Address offset: 0x2C

Reset value: 0x0000 0000

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

Bits 31:0 **TS_OP[31:0]:** general purpose option bits

20.6.10 DTS register map

The following table summarizes the temperature sensor registers.

Table 128. DTS register map and reset values

Offset	Register name reset value	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	**DTS_ CFGR1**	Res.	HSREF_CLK_DIV [6:0]							Res.	Res.	Q_MEAS_OPT	REFCLK_SEL	TS1_ SMP_TIME [3:0]				Res.	Res.	Res.	Res.	TS1_ INTRIG_SEL [3:0]				Res.	Res.	Res.	TS1_START	Res.	Res.	Res.	TS1_EN
	Reset value		0	0	0	0	0	0	0			0	0	0	0	0	0					0	0	0	0				0				0
0x04	Reserved																																
0x08	**DTS_ T0VALR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_T0[1:0]		TS1_FMT0[15:0]															
	Reset value															X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0C	Reserved																																
0x10	**DTS_ RAMPVALR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_RAMP_COEFF[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x14	**DTS_ITR1**	TS1_HITTHD[15:0]																TS1_LITTHD[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
0x18	Reserved																																
0x1C	**DTS_DR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_MFREQ[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**DTS_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_AITHF	TS1_AITLF	TS1_AITEF	Res.	TS1_ITHF	TS1_ITLF	TS1_ITEF
	Reset value																	0									0	0	0		0	0	0
0x24	**DTS_ITENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_AITHEN	TS1_AITLEN	TS1_AITEEN	Res.	TS1_ITHEN	TS1_ITLEN	TS1_ITEEN
	Reset value																										0	0	0		0	0	0
0x28	**DTS_ICIFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_CAITHF	TS1_CAITLF	TS1_CAITEF	Res.	TS1_CITHF	TS1_CITLF	TS1_CITEF
	Reset value																										0	0	0		0	0	0
0x2C	**DTS_OR**	TS_OP31	TS_OP30	TS_OP29	TS_OP28	TS_OP27	TS_OP26	TS_OP25	TS_OP24	TS_OP23	TS_OP22	TS_OP21	TS_OP20	TS_OP19	TS_OP18	TS_OP17	TS_OP16	TS_OP15	TS_OP14	TS_OP13	TS_OP12	TS_OP11	TS_OP10	TS_OP9	TS_OP8	TS_OP7	TS_OP6	TS_OP5	TS_OP4	TS_OP3	TS_OP2	TS_OP1	TS_OP0
	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 2.2 on page 70* for the register boundary addresses.

21 Digital-to-analog converter (DAC)

21.1 Introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data can be left- or right-aligned. The DAC features two output channels, each with its own converter. In dual DAC channel mode, conversions can be done independently or simultaneously when both channels are grouped together for synchronous update operations.

The DACx_OUTy pin can be used as general purpose input/output (GPIO) when the DAC output is disconnected from output pad and connected to on chip peripheral. The DAC output buffer can be optionally enabled to obtain a high drive output current. An individual calibration can be applied on each DAC output channel. The DAC output channels support a low power mode, the sample and hold mode.

21.2 DAC main features

The DAC main features are the following (see *Figure 134: Dual-channel DAC block diagram*)

- One DAC interface, maximum two output channels
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave and Triangular-wave generation
- Dual DAC channel for independent or simultaneous conversions
- DMA capability for each channel including DMA underrun error detection
- Double data DMA capability to reduce the bus activity
- External triggers for conversion
- DAC output channel buffered/unbuffered modes
- Buffer offset calibration
- Each DAC output can be disconnected from the DACx_OUTy output pin
- DAC output connection to on-chip peripherals
- Sample and hold mode for low power operation in Stop mode

Figure 134 shows the block diagram of a DAC channel and *Table 130* gives the pin description.

21.3 DAC implementation

Table 129. DAC features

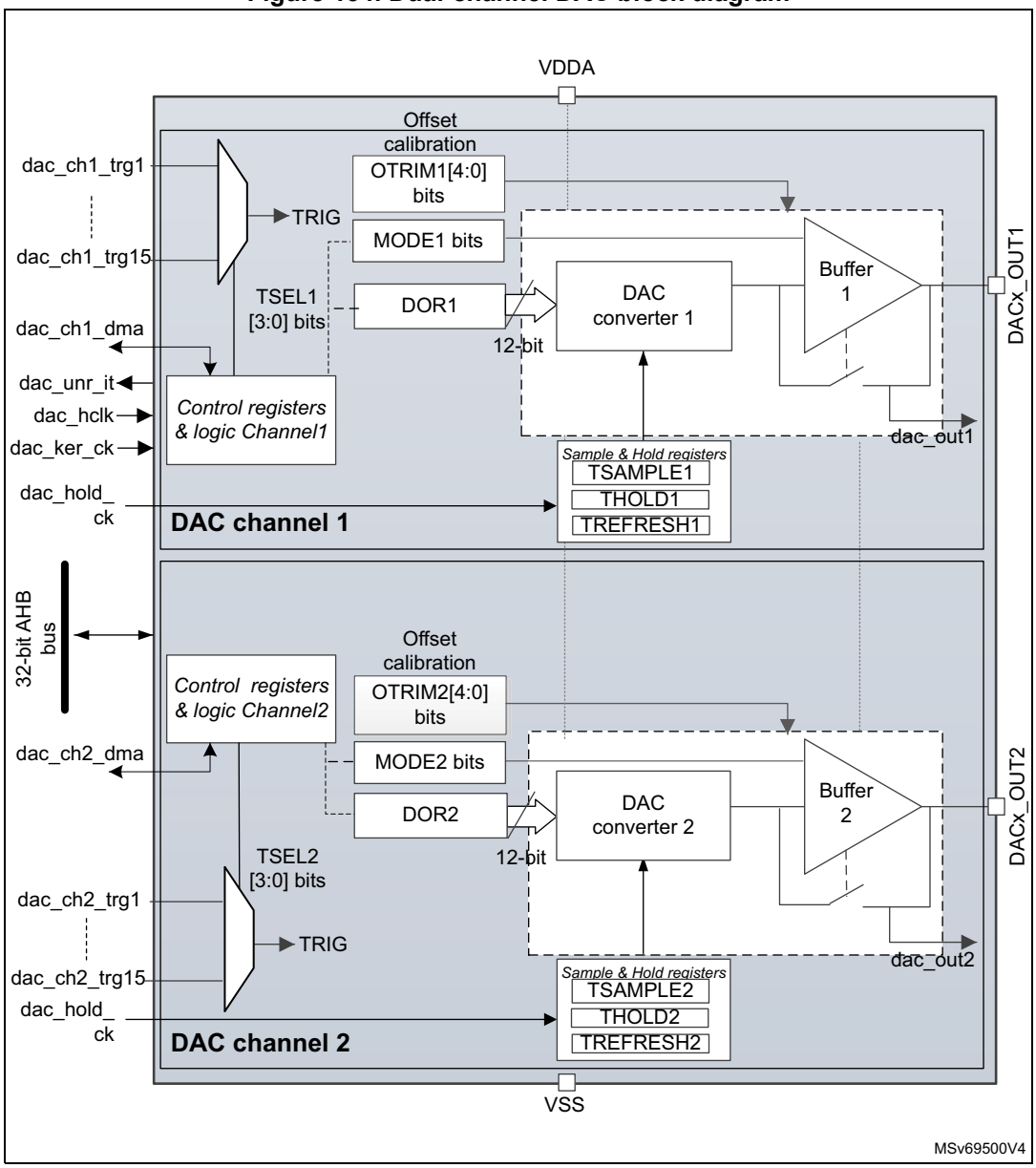
DAC features	DAC1
Dual channel	X
Output buffer	X
I/O connection	DAC1_OUT1 on PA4, DAC1_OUT2 on PA5
Maximum sampling time	1 Msps
Autonomous mode	-
VREF+ pin	-(1)

1. VREF+ is internally connected to VDDA.

21.4 DAC functional description

21.4.1 DAC block diagram

Figure 134. Dual-channel DAC block diagram



1. MODEx bits in the DAC_MCR control the output mode and allow switching between the normal mode in buffer/unbuffered configuration and the sample and hold mode.
2. Refer to *Section 21.3: DAC implementation* for channel2 availability.

21.4.2 DAC pins and internal signals

The DAC includes:

- Up to two output channels
- The DACx_OUTy can be disconnected from the output pin and used as an ordinary GPIO
- The dac_outx can use an internal pin connection to on-chip peripherals such as comparator, operational amplifier and ADC (if available).
- DAC output channel buffered or non buffered
- Sample and hold block and registers operational in Stop mode, using the LSI/LSE clock source (dac_hold_ck) for static conversion.

The DAC includes up to two separate output channels. Each output channel can be connected to on-chip peripherals such as comparator, operational amplifier and ADC (if available). In this case, the DAC output channel can be disconnected from the DACx_OUTy output pin and the corresponding GPIO can be used for another purpose.

The DAC output can be buffered or not. The sample and hold block and its associated registers can run in Stop mode using the LSI/LSE clock source (dac_hold_ck).

Table 130. DAC input/output pins

Pin name	Signal type	Remarks
VDDA	Input, analog supply	Analog power supply
VSSA	Input, analog supply ground	Ground for analog power supply
DACx_OUTy	Analog output signal	DACx channely analog output

Table 131. DAC internal input/output signals

Internal signal name	Signal type	Description
dac_ch1_dma	Bidirectional	DAC channel1 DMA request/acknowledge
dac_ch2_dma	Bidirectional	DAC channel2 DMA request/acknowledge
dac_ch1_trgx (x = 1 to 15)	Inputs	DAC channel1 trigger inputs
dac_ch2_trgx (x = 1 to 15)	Inputs	DAC channel2 trigger inputs
dac_unr_it	Output	DAC underrun interrupt
dac_hclk	Input	DAC peripheral clock
dac_ker_ck	Input	DAC kernel clock
dac_hold_ck	Input	DAC low-power clock used in sample and hold mode
dac_out1	Analog output	DAC channel1 output for on-chip peripherals
dac_out2	Analog output	DAC channel2 output for on-chip peripherals

Table 132. DAC interconnection

Signal name	Source	Source type
dac_hold_ck	ck_lsi or ck_lse	LSI or LSE clock selected in the RCC
dac_chx_trg1 (x = 1, 2)	tim1_trgo	Internal signal from on-chip timers
dac_chx_trg2 (x = 1, 2)	tim2_trgo	Internal signal from on-chip timers
dac_chx_trg3 (x = 1, 2)	tim3_trgo	Internal signal from on-chip timers
dac_chx_trg5 (x = 1, 2)	tim6_trgo	Internal signal from on-chip timers
dac_chx_trg6 (x = 1, 2)	tim7_trgo	Internal signal from on-chip timers
dac_chx_trg11 (x = 1, 2)	lptim1_ch1	Internal signal from on-chip timers
dac_chx_trg12 (x = 1, 2)	lptim2_ch1	Internal signal from on-chip timers
dac_chx_trg13 (x = 1, 2)	exti9	External pin

21.4.3 DAC clocks

Two clock sources can be used to update the DAC:

- dac_hclk: DAC peripheral clock (AHB clock)
- dac_ker_ck: DAC kernel clock: this clock can be used to synchronize DAC and ADC.
- dac_hold_ck: low-power clock used in sample and hold mode

The DAC clock is selected in the RCC.

21.4.4 DAC channel enable

Each DAC channel can be powered on by setting its corresponding ENx bit in the DAC_CR register. The DAC channel is then enabled after a t_{WAKEUP} startup time.

DACxRDY bit is set in the DAC_SR register when the DAC interface is ready to accept data. Writing new data or asserting the trigger is not allowed when ENx bit is set while DACxRDY signal is reset.

Note: The ENx bit enables the analog DAC channelx only. The DAC channelx digital interface is enabled even if the ENx bit is reset.

21.4.5 DAC data format

Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- Single DAC channel

  There are three possibilities:

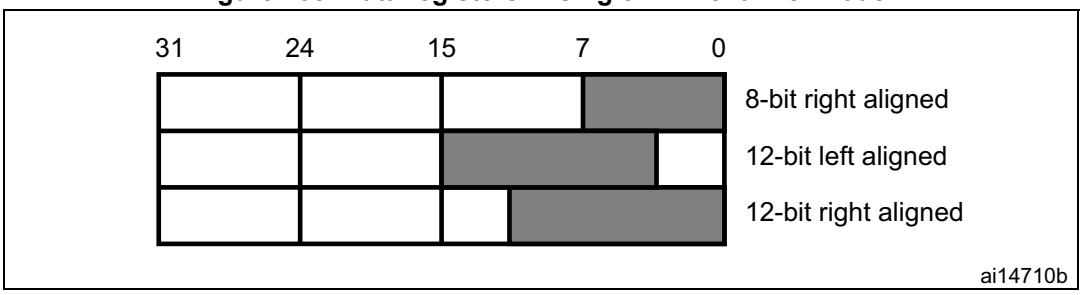
  – 8-bit right alignment: the software has to load data into the DAC_DHR8Rx[7:0] bits (stored into the DHRx[11:4] bits)

  – 12-bit left alignment: the software has to load data into the DAC_DHR12Lx [15:4] bits (stored into the DHRx[11:0] bits)

  – 12-bit right alignment: the software has to load data into the DAC_DHR12Rx [11:0] bits (stored into the DHRx[11:0] bits)

Depending on the loaded DAC_DHRyyyx register, the data written by the user is shifted and stored into the corresponding DAC_DHRx (data holding registerx, which are internal non-memory-mapped registers). The DAC_DHRx register is then loaded into the DAC_DORx register either automatically, by software trigger or by an external event trigger.

Figure 135. Data registers in single DAC channel mode



- Dual DAC channels (when available)

  There are three possibilities:

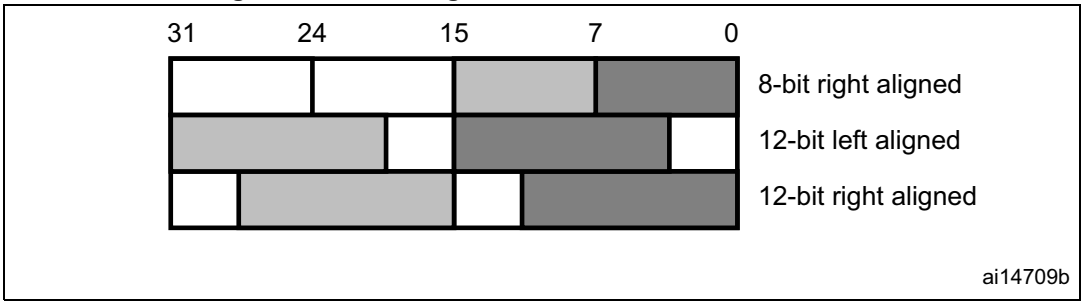
  – 8-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR8RD [7:0] bits (stored into the DHR1[11:4] bits) and data for DAC channel2 to be loaded into the DAC_DHR8RD [15:8] bits (stored into the DHR2[11:4] bits)

  – 12-bit left alignment: data for DAC channel1 to be loaded into the DAC_DHR12LD [15:4] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12LD [31:20] bits (stored into the DHR2[11:0] bits)

  – 12-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR12RD [11:0] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12RD [27:16] bits (stored into the DHR2[11:0] bits)

Depending on the loaded DAC_DHRyyyD register, the data written by the user is shifted and stored into DHR1 and DHR2 (data holding registers, which are internal non-memory-mapped registers). The DHR1 and DHR2 registers are then loaded into the DAC_DOR1 and DOR2 registers, respectively, either automatically, by software trigger or by an external event trigger.

Figure 136. Data registers in dual DAC channel mode



Signed/unsigned data

DAC input data are unsigned: 0x000 corresponds to the minimum value and 0xFFF to the maximum value for 12-bit mode.

The DAC can also handle signed input data in 2’s complement format. This is done by setting SINFORMATx bit in the DAC_MCR register.

When SINFORMATx bit is set, the MSB of the data written to DAC_DHRx registers is inverted when it is copied to the DAC_DORx register, and the DAC interface can accept signed data (Q1.15, Q1.11 or Q1.7 format). DAC_DHR12Lx register can be used to store 16-bit signed data in the data holding registers. The 12 MSBs of 16-bit data are used for the DAC output data and the MSB is inverted. The four LSBs are simply ignored.

Table 133. Data format (case of 12-bit data)

SINFORMATx bit	DATA written to DAC_DHRx register	DATA transfered to DAC_DORx register
0	0x000	0x000
0	0xFFF	0xFFF
1	0x7FF	0xFFF
1	0x000	0x800
1	0xFFF	0x7FF
1	0x800	0x000

21.4.6 DAC conversion

The DAC_DORx cannot be written directly and any data transfer to the DAC channelx must be performed by loading the DAC_DHRx register (write operation to DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12RD or DAC_DHR12LD).

Data stored in the DAC_DHRx register are automatically transferred to the DAC_DORx register after one dac_hclk clock cycle, if no hardware trigger is selected (TENx bit in DAC_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three dac_hclk clock cycles after the trigger signal.

When DAC_DORx is loaded with the DAC_DHRx contents, the analog output voltage becomes available after a time $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

To synchronize DAC and ADC, the same clock source can be used for both peripherals. This is done by selecting the dac_ker_ck clock instead of the dac_hclk clock (AHB clock) in the RCC.

HFSEL bits of DAC_MCR must be set when dac_ker_ck clock speed is faster than 80 MHz.

Refer to Table *HFSEL description* below for the limitation of the DAC_DORx update rate depending on HFSEL bits and dac_hclk or dac_ker_ck clock frequency.

If the data is updated or a software/hardware trigger event occurs during the non-allowed period, the peripheral behavior is unpredictable.

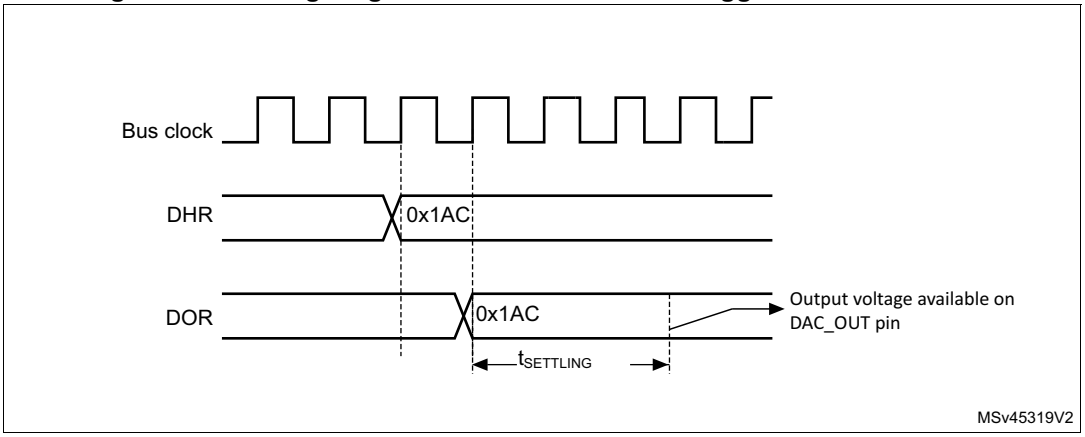
The above timing is only related to the limitation of the DAC interface. Refer also to the $t_{SETTLING}$ parameter value in the product datasheet.

Table 134. HFSEL description

HFSEL [1:0]	Clock frequency	Latency using AHB clock (dac_hclk)	Latency using dac_ker_ck clock	Function
00	< 80 MHz	3	4	DAC_DOR update rate up to 3 AHB clock cycles or 4 dac_ker_ck cycles.
01	≥ 80 MHz(1)	5	5	DAC_DOR update rate up to 5 AHB clock or dac_ker_ck cycles.
10	≥ 160 MHz	7	6	DAC_DOR update rate up to 7 AHB clock cycles or 6 dac_ker_ck cycles.
11	Reserved	-	-	-

1. Refer to the device datasheet for the value of the maximum dac_hclk or dac_ker_ck frequency.

Figure 137. Timing diagram for conversion with trigger disabled TEN = 0



21.4.7 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and V_{DDA}.

The analog output voltages on each DAC channel pin are determined by the following equation:

$$\text{DACoutput} = V_{REF} \times \frac{DOR}{4096}$$

where all voltages are expressed in Volt.

21.4.8 DAC trigger selection

If the TENx control bit is set, the conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[3:0] control bits determine which out of 16 possible events triggers the conversion as shown in TSELx[3:0] bits of the DAC_CR register. These events can be either the software trigger or hardware triggers. Refer to the interconnection table in *Section 21.4.2*.

Each time a DAC interface detects a rising edge on the selected trigger source (refer to the table below), the last data stored into the DAC_DHRx register are transferred into the DAC_DORx register. The DAC_DORx register is updated three dac_hclk cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DORx register has been loaded with the DAC_DHRx register contents.

Note: TSELx[3:0] bit cannot be changed when the ENx bit is set.

When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DORx register takes only one dac_hclk clock cycle.

21.4.9 DMA requests

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

When an external trigger (but not a software trigger) occurs while the DMAENx bit is set, the value of the DAC_DHRx register is transferred into the DAC_DORx register when the transfer is complete, and a DMA request is generated.

In dual mode, if both DMAENx bits are set, two DMA requests are generated. If only one DMA request is needed, only the corresponding DMAENx bit must be set. In this way, the application can manage both DAC channels in dual mode by using one DMA request and a unique DMA channel.

As DAC_DHRx to DAC_DORx data transfer occurred before the DMA request, the very first data has to be written to the DAC_DHRx before the first trigger event occurs.

DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgment for the first external trigger is received (first request), then no new request is issued and the DMA channelx underrun flag DMAUDRx in the DAC_SR register is set, reporting the error condition. The DAC channelx continues to convert old data.

The software must clear the DMAUDRx flag by writing 1, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channelx to restart the transfer correctly. The software must modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, the DAC conversion can be resumed by enabling both DMA data transfer and conversion trigger.

For each DAC channelx, an interrupt is also generated if its corresponding DMAUDRIEx bit in the DAC_CR register is enabled.

DMA double data mode

When the DMA controller is used in normal mode, only 12-bit (or 8-bit) data are transferred by a DMA request. As the AHB width is 32 bits, two 12-bit data may be transferred simultaneously. To use this mode, set the DMADOUBLEx bit of DAC_MCR register.

A DAC DMA request is generated every two external triggers (except for software triggers) when the DMAENx bit is set:

1. When the first trigger is detected, the value of the DAC_DHRx and DAC_DHRBx registers are transferred into the DAC_DORx and DAC_DORBx registers. The actual DAC data is loaded into the DAC_DORx register. A DMA request is then generated. The DMA writes the new data to the DAC_DHRx and DAC_DHRBx data registers.
2. When the next trigger is detected, the actual DAC data is loaded into the DAC_DHRBx register. This second trigger does not generate any DMA request. The DORSTATx bit indicates which DOR data is actually loaded into the analog DAC input.

DMA underrun function is also supported in DMA double data mode.

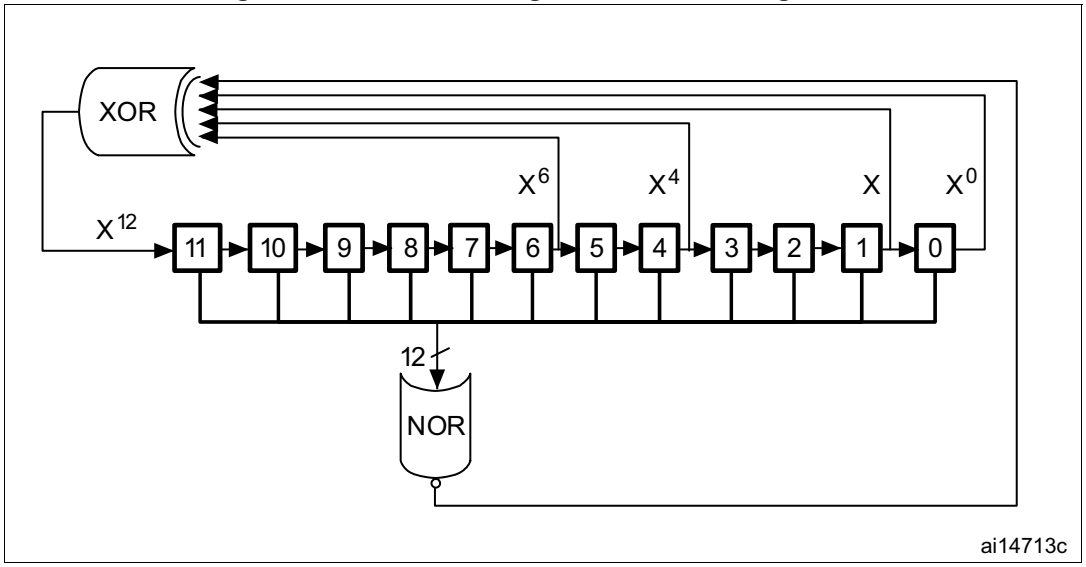
The following conditions must be met to change from double data to single data mode or vice versa:

- The DAC must be disabled.
- DMAEN bit must be cleared (ENx = 0 and DMAEN = 0).

21.4.10 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVEx[1:0] to 01. The preloaded value in LFSR is 0xAAA. This register is updated three dac_hclk clock cycles after each trigger event, following a specific calculation algorithm.

Figure 138. DAC LFSR register calculation algorithm

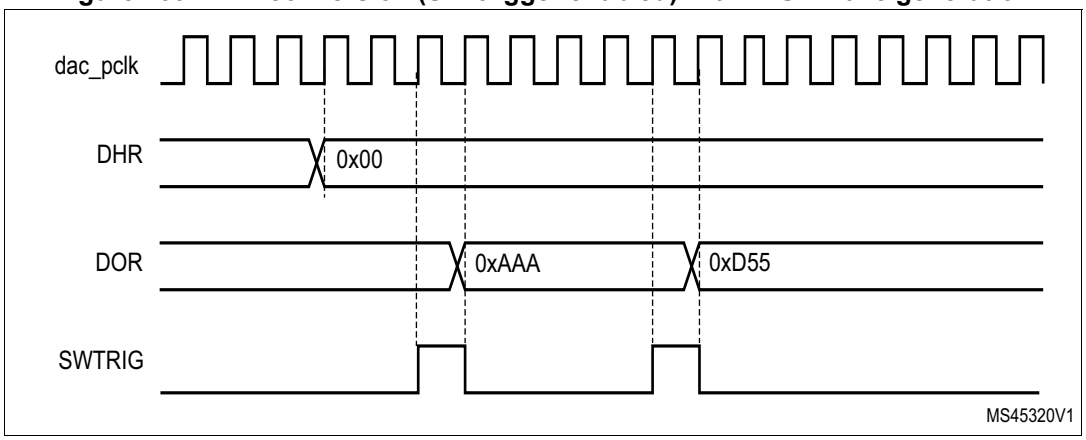


The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC_CR register, is added up to the DAC_DHRx contents without overflow and this value is then transferred into the DAC_DORx register.

If LFSR is 0x0000, a 1 is injected into it (antilock-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVEx[1:0] bits.

Figure 139. DAC conversion (SW trigger enabled) with LFSR wave generation



Note: The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC_CR register.

21.4.11 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVEx[1:0] to 10. The amplitude is configured through the MAMPx[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three dac_hclk clock cycles after each trigger event. The value of this counter is then added to the DAC_DHRx register without overflow and the sum is transferred into the DAC_DORx register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMPx[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVEx[1:0] bits.

Figure 140. DAC triangle wave generation

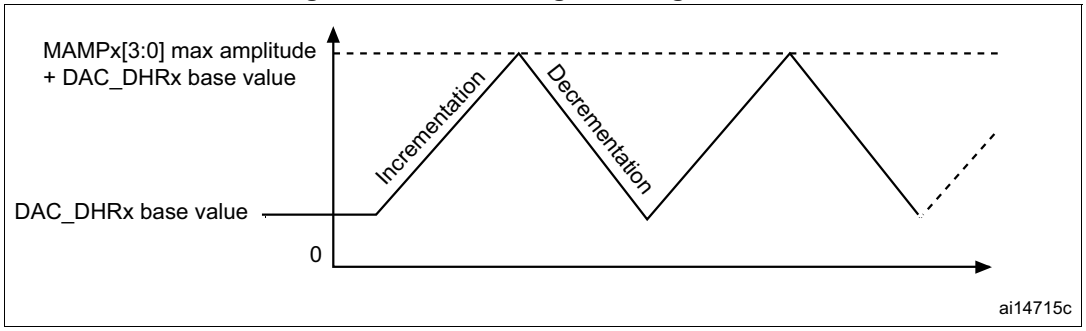
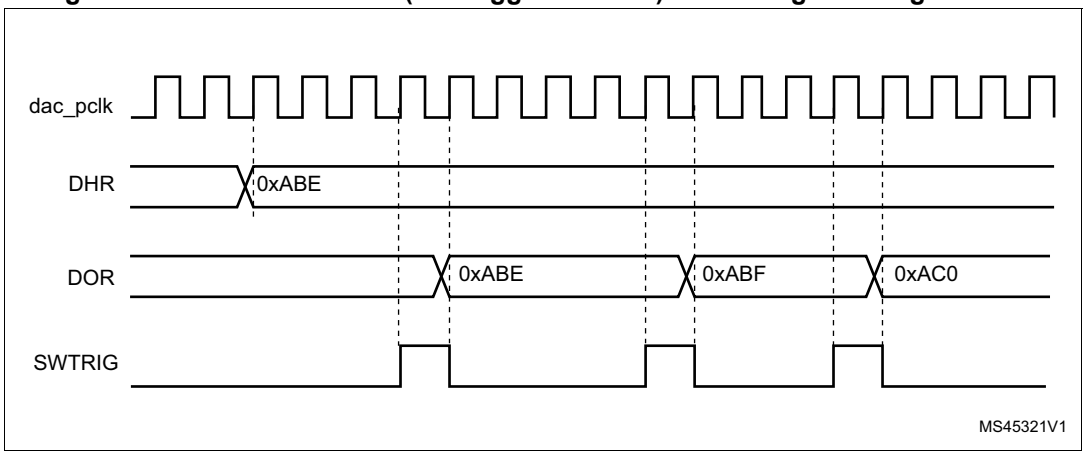


Figure 141. DAC conversion (SW trigger enabled) with triangle wave generation



Note: The DAC trigger must be enabled for triangle wave generation by setting the TENx bit in the DAC_CR register.

The MAMPx[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.

21.4.12 DAC channel modes

Each DAC channel can be configured in normal mode or sample and hold mode. The output buffer can be enabled to obtain a high drive capability. Before enabling output buffer, the voltage offset needs to be calibrated. This calibration is performed at the factory (loaded after reset) and can be adjusted by software during application operation.

Normal mode

In normal mode, there are four combinations, by changing the buffer state and by changing the DACx_OUTy pin interconnections.

To enable the output buffer, the MODEx[2:0] bits in DAC_MCR register must be:

- 000: DAC is connected to the external pin
- 001: DAC is connected to external pin and to on-chip peripherals

To disable the output buffer, the MODEx[2:0] bits in DAC_MCR register must be:

- 010: DAC is connected to the external pin
- 011: DAC is connected to on-chip peripherals

Sample and hold mode

In sample and hold mode, the DAC core converts data on a triggered conversion, and then holds the converted voltage on a capacitor. When not converting, the DAC cores and buffer are completely turned off between samples and the DAC output is tri-stated, therefore reducing the overall power consumption. A stabilization period, which value depends on the buffer state, is required before each new conversion.

In this mode, the DAC core and all corresponding logic and registers are driven by the LSI/LSE low-speed clock (dac_hold_ck) in addition to the dac_hclk clock, allowing using the DAC channels in deep low power modes such as Stop mode.

The LSI/LSE low-speed clock (dac_hold_ck) must not be stopped when the sample and hold mode is enabled.

The sample/hold mode operations can be divided into three phases:

1. Sample phase: the sample/hold element is charged to the desired voltage. The charging time depends on capacitor value (internal or external, selected by the user). The sampling time is configured with the TSAMPLEx[9:0] bits in DAC_SHSRx register. During the write of the TSAMPLEx[9:0] bits, the BWSTx bit in DAC_SR register is set to 1 to synchronize between both clocks domains (AHB and low speed clock) and allowing the software to change the value of sample phase during the DAC channel operation
2. Hold phase: the DAC output channel is tri-stated, the DAC core and the buffer are turned off, to reduce the current consumption. The hold time is configured with the THOLDx[9:0] bits in DAC_SHHR register
3. Refresh phase: the refresh time is configured with the TREFRESHx[7:0] bits in DAC_SHRR register

The timings for the three phases above are in units of LSI/LSE clock periods. As an example, to configure a sample time of 350 µs, a hold time of 2 ms and a refresh time of 100 µs assuming LSI/LSE ~32 KHz is selected:

- 12 cycles are required for sample phase: TSAMPLEx[9:0] = 11.
- 62 cycles are required for hold phase: THOLDx[9:0] = 62.
- and 4 cycles are required for refresh period: TREFRESHx[7:0] = 4.

In this example, the power consumption is reduced by almost a factor of 15 versus normal modes.

The formulas to compute the right sample and refresh timings are described in the table below, the Hold time depends on the leakage current.

Table 135. Sample and refresh timings

Buffer State	t_{SAMP} (1)(2)	$t_{REFRESH}$ (2)(3)
Enable	7 µs + (10*R_{BON}*C_{SH})	7 µs + (R_{BON}*C_{SH})*ln(2*N_{LSB})
Disable	3 µs + (10*R_{BOFF}*C_{SH})	3 µs + (R_{BOFF}*C_{SH})*ln(2*N_{LSB})

1. In the above formula, the settling to the desired code value with ½ LSB or accuracy requires 10 constant time for 12 bits resolution. For 8-bit resolution, the settling time is 7 constant time.
2. C_{SH} is the capacitor in sample and hold mode.
3. The tolerated voltage drop during the hold phase “Vd” is represented by the number of LSBs after the capacitor discharging with the output leakage current. The settling back to the desired value with ½ LSB error accuracy requires ln(2*Nlsb) constant time of the DAC.

Example of the sample and refresh time calculation with output buffer on

The values used in the example below are provided as indication only. Refer to the product datasheet for product data.

C_{SH} = 100 nF

V_{REF+} = 3.0 V

Sampling phase:

t_{SAMP} = 7 µs + (10 * 2000 * 100 * 10^{-9}) = 2.007 ms

(where R_{BON} = 2 kΩ)

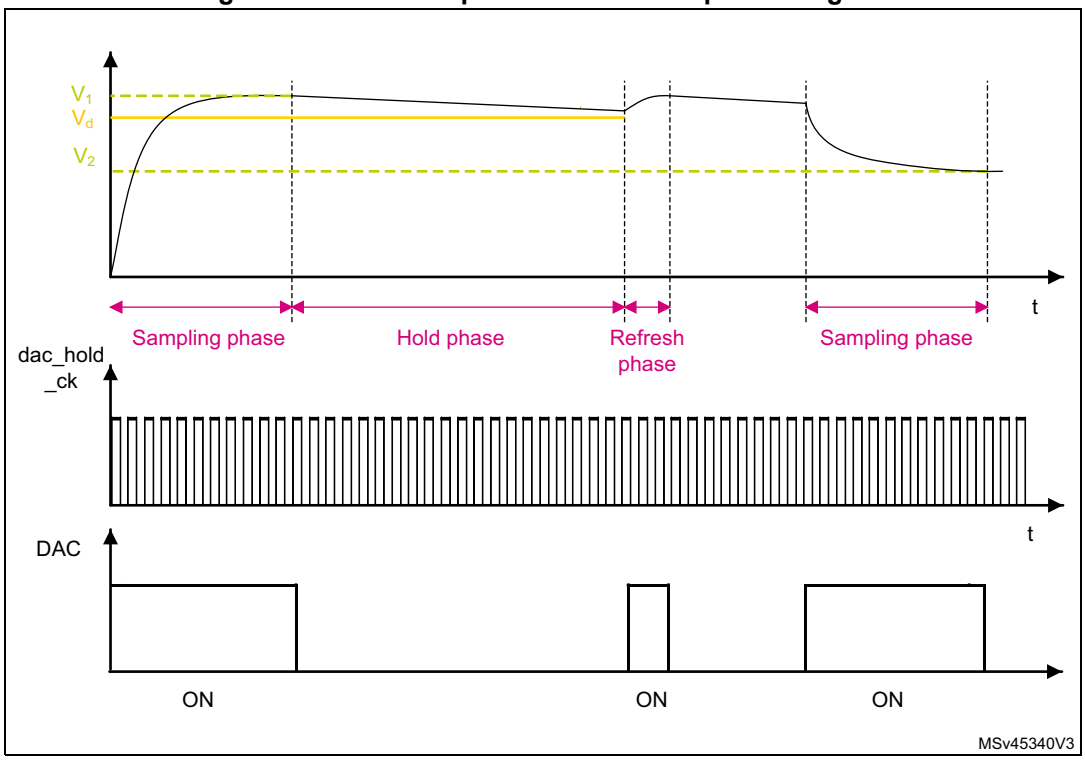
Refresh phase:

$t_{REFRESH}$ = 7 µs + (2000 * 100 * 10^{-9}) * ln(2*10) = 606.1 µs
(where N_{LSB} = 10 (10 LSB drop during the hold phase)

Hold phase:

$D_V = i_{leak} * t_{hold} / C_{SH}$ = 0.0073 V (10 LSB of 12bit at 3 V)
i_{leak} = 150 nA (worst case on the IO leakage on all the temperature range)
t_{hold} = 0.0073 * 100 * 10^{-9} / (150 * 10^{-9}) = 4.867 ms

Figure 142. DAC sample and hold mode phase diagram



Like in normal mode, the sample and hold mode has different configurations.

To enable the output buffer, MODEx[2:0] bits in DAC_MCR register must be set to:

- 100: DAC is connected to the external pin
- 101: DAC is connected to external pin and to on chip peripherals

To disabled the output buffer, MODEx[2:0] bits in DAC_MCR register must be set to:

- 110: DAC is connected to external pin and to on chip peripherals
- 111: DAC is connected to on chip peripherals

  When MODEx[2:0] bits are equal to 111, an internal capacitor, C_{Lint}, holds the voltage output of the DAC core and then drive it to on-chip peripherals.

All sample and hold phases are interruptible, and any change in DAC_DHRx immediately triggers a new sample phase.

Table 136. Channel output modes summary

MODEx[2:0]			Mode	Buffer	Output connections
0	0	0	Normal mode	Enabled	Connected to external pin
0	0	1			Connected to external pin and to on chip-peripherals (such as comparators)
0	1	0		Disabled	Connected to external pin
0	1	1			Connected to on chip peripherals (such as comparators)
1	0	0	Sample and hold mode	Enabled	Connected to external pin
1	0	1			Connected to external pin and to on chip peripherals (such as comparators)
1	1	0		Disabled	Connected to external pin and to on chip peripherals (such as comparators)
1	1	1			Connected to on chip peripherals (such as comparators)

Note: Some packages do not feature any DACx_OUTy pin. In this case, the application must not use a channel output mode where the buffer is connected to an external pin that does not exist on the package. For more detail, refer to the product datasheet.

21.4.13 DAC channel buffer calibration

The transfer function for an N-bit digital-to-analog converter (DAC) is:

$$V_{out} = \left((D / 2^N) \times G \times V_{REF}\right) + V_{OS}$$

Where V_{OUT} is the analog output, D is the digital input, G is the gain, V_{REF} is the nominal full-scale voltage, and V_{OS} is the offset voltage. For an ideal DAC channel, G = 1 and V_{OS} = 0.

Due to output buffer characteristics, the voltage offset may differ from part-to-part and introduce an absolute offset error on the analog output. To compensate the V_{OS}, a calibration is required by a trimming technique.

The calibration is only valid when the DAC channelx is operating with buffer enabled (MODEx[2:0] = 0b000 or 0b001 or 0b100 or 0b101). if applied in other modes when the buffer is off, it has no effect. During the calibration:

- The buffer output is disconnected from the pin internal/external connections and put in tristate mode (HiZ).
- The buffer acts as a comparator to sense the middle-code value 0x800 and compare it to V_{DDA}/2 signal through an internal bridge, then toggle its output signal to 0 or 1 depending on the comparison result (CAL_FLAGx bit).

Two calibration techniques are provided:

- Factory trimming (default setting)

  The DAC buffer offset is factory trimmed. The default value of OTRIMx[4:0] bits in DAC_CCR register is the factory trimming value and it is loaded once DAC digital interface is reset.

- User trimming

  The user trimming can be done when the operating conditions differs from nominal factory trimming conditions and in particular when V_{DDA} voltage, temperature, V_{DDA} values change and can be done at any point during application by software.

Note: Refer to the datasheet for more details of the nominal factory trimming conditions.

In addition, when V_{DD} is removed (example the device enters in Standby or VBAT modes) the calibration is required.

The steps to perform a user trimming calibration are as below:

1. If the DAC channel is active, write 0 to ENx bit in DAC_CR to disable the channel.
2. Select a mode where the buffer is enabled, by writing to DAC_MCR register, MODEx[2:0] = 0b000 or 0b001 or 0b100 or 0b101.
3. Start the DAC channelx calibration, by setting the CENx bit in DAC_CR register to 1.
4. Apply a trimming algorithm:
   a) Write a code into OTRIMx[4:0] bits, starting by 0b00000.
   b) Wait for t_{TRIM} delay.
   c) Check if CAL_FLAGx bit in DAC_SR is set to 1.
   d) If CAL_FLAGx is set to 1, the OTRIMx[4:0] trimming code is found and can be used during *device* operation to compensate the output value, else increment OTRIMx[4:0] and repeat sub-steps from (a) to (d) again.
   e) If CAL_FLAGx remains cleared, set OTRIMx[4:0] to 0b11111.

The software algorithm may use either a successive approximation or dichotomy techniques to compute and set the content of OTRIMx[4:0] bits in a faster way.

The commutation/toggle of CAL_FLAGx bit indicates that the offset is correctly compensated and the corresponding trim code must be kept in the OTRIMx[4:0] bits in DAC_CCR register.

Note: A t_{TRIM} delay must be respected between the write to the OTRIMx[4:0] bits and the read of the CAL_FLAGx bit in DAC_SR register in order to get a correct value.This parameter is specified into datasheet electrical characteristics section.

If V_{DDA} and temperature conditions do not change during device operation while it enters more often in Standby and VBAT modes, the software may store the OTRIMx[4:0] bits found in the first user calibration in the flash or in back-up registers. then to load/write them directly when the device power is back again thus avoiding to wait for a new calibration time.

When CENx bit is set, it is not allowed to set ENx bit.

21.4.14 Dual DAC channel conversion modes (if dual channels are available)

To efficiently use the bus bandwidth in applications that require the two DAC channels at the same time, three dual registers are implemented: DHR8RD, DHR12RD and DHR12LD. A unique register access is then required to drive both DAC channels at the same time. For the wave generation, no accesses to DHRxxxD registers are required. As a result, two output channels can be used either independently or simultaneously.

15 conversion modes are possible using the two DAC channels and these dual registers. All the conversion modes can nevertheless be obtained using separate DAC_DHRx registers if needed.

All modes are described in the paragraphs below.

Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three dac_hclk clock cycles later).

When a DAC channel2 trigger arrives, the DHR2 register is transferred into DAC_DOR2 (three dac_hclk clock cycles later).

Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the LFSR2 counter is updated.

Independent trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR masks values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the LFSR2 counter is updated.

Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

Independent trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bits.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is

transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

Simultaneous software start

To configure the DAC in this conversion mode, the following sequence is required:

- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

In this configuration, one dac_hclk clock cycle later, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively.

Simultaneous trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively (after three dac_hclk clock cycles).

Simultaneous trigger with single LFSR generation

1. To configure the DAC in this conversion mode, the following sequence is required:
2. Set the two DAC channel trigger enable bits TEN1 and TEN2.
3. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
4. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
5. Load the dual DAC channel data to the desired DHR register (DHR12RD, DHR12LD or DHR8RD).

When a trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR mask values using the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The LFSR1 counter is then updated.
At the same time, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value using the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

At the same time, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

Simultaneous trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three AHB clock cycles later). Then the DAC channel1 triangle counter is updated.

At the same time, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the DAC channel2 triangle counter is updated.

21.5 DAC in low-power modes

Table 137. Effect of low-power modes on DAC

Mode	Description
Sleep	No effect, DAC used with DMA.
Stop(1)	The DAC remains active with a static value. The sample and hold mode can be selected using LSE/LSI clock.
Standby	The DAC peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 21.3: DAC implementation* for information on the Stop modes supported by the DAC peripheral.

21.6 DAC interrupts

Table 138. DAC interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop mode	Exit Standby mode
DAC	DMA underrun	DMAUDRx	DMAUDRIEx	Write DMAUDRx = 1	Yes	No	No

21.7 DAC registers

Refer to *Section 1 on page 65* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

21.7.1 DAC control register (DAC_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.	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[3]	TSEL2[2]	TSEL2[1]	TSEL2[0]	TEN2	EN2
	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.	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[3]	TSEL1[2]	TSEL1[1]	TSEL1[0]	TEN1	EN1
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **CEN2**: DAC channel2 calibration enable

This bit is set and cleared by software to enable/disable DAC channel2 calibration, it can be written only if EN2 bit is set to 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel2 in normal operating mode

1: DAC channel2 in calibration mode

Note: This bit is available only on dual-channel DACs. Refer to Section 21.3: DAC implementation.

Bit 29 **DMAUDRIE2**: DAC channel2 DMA underrun interrupt enable

This bit is set and cleared by software.

0: DAC channel2 DMA underrun interrupt disabled

1: DAC channel2 DMA underrun interrupt enabled

Note: This bit is available only on dual-channel DACs. Refer to Section 21.3: DAC implementation.

Bit 28 **DMAEN2**: DAC channel2 DMA enable

This bit is set and cleared by software.

0: DAC channel2 DMA mode disabled

1: DAC channel2 DMA mode enabled

Note: This bit is available only on dual-channel DACs. Refer to Section 21.3: DAC implementation.

Bits 27:24 **MAMP2[3:0]**: DAC channel2 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1

0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3

0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7

0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15

0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31

0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63

0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127

0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255

1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511

1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023

1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047

≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Note: These bits are available only on dual-channel DACs. Refer to Section 21.3: DAC implementation.

Bits 23:22 **WAVE2[1:0]**: DAC channel2 noise/triangle wave generation enable

These bits are set/reset by software.

00: wave generation disabled

01: Noise wave generation enabled

1x: Triangle wave generation enabled

Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled)

These bits are available only on dual-channel DACs. Refer to Section 21.3: DAC implementation.

Bits 21:18 **TSEL2[3:0]**: DAC channel2 trigger selection
These bits select the external event used to trigger DAC channel2
0000: SWTRIG2
0001: dac_ch2_trg1
0010: dac_ch2_trg2
...
1111: dac_ch2_trg15
Refer to the trigger selection tables in *Section 21.4.2: DAC pins and internal signals* for details on trigger configuration and mapping.

Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled).
These bits are available only on dual-channel DACs. Refer to Section 21.3: DAC implementation.

Bit 17 **TEN2**: DAC channel2 trigger enable
This bit is set and cleared by software to enable/disable DAC channel2 trigger
0: DAC channel2 trigger disabled and data written into the DAC_DHR2 register are transferred one dac_hclk clock cycle later to the DAC_DOR2 register
1: DAC channel2 trigger enabled and data from the DAC_DHR2 register are transferred three dac_hclk clock cycles later to the DAC_DOR2 register

Note: When software trigger is selected, the transfer from the DAC_DHR2 register to the DAC_DOR2 register takes only one dac_hclk clock cycle.
These bits are available only on dual-channel DACs. Refer to Section 21.3: DAC implementation.

Bit 16 **EN2**: DAC channel2 enable
This bit is set and cleared by software to enable/disable DAC channel2.
0: DAC channel2 disabled
1: DAC channel2 enabled

Note: These bits are available only on dual-channel DACs. Refer to Section 21.3: DAC implementation.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CEN1**: DAC channel1 calibration enable
This bit is set and cleared by software to enable/disable DAC channel1 calibration, it can be written only if bit EN1 = 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.
0: DAC channel1 in normal operating mode
1: DAC channel1 in calibration mode

Bit 13 **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable
This bit is set and cleared by software.
0: DAC channel1 DMA Underrun Interrupt disabled
1: DAC channel1 DMA Underrun Interrupt enabled

Bit 12 **DMAEN1**: DAC channel1 DMA enable
This bit is set and cleared by software.
0: DAC channel1 DMA mode disabled
1: DAC channel1 DMA mode enabled

Bits 11:8 **MAMP1[3:0]**: DAC channel1 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6 **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable

These bits are set and cleared by software.
00: wave generation disabled
01: Noise wave generation enabled
1x: Triangle wave generation enabled
Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bits 5:2 **TSEL1[3:0]**: DAC channel1 trigger selection

These bits select the external event used to trigger DAC channel1
0000: SWTRIG1
0001: dac_ch1_trg1
0010: dac_ch1_trg2
...
1111: dac_ch1_trg15
Refer to the trigger selection tables in *Section 21.4.2: DAC pins and internal signals* for details on trigger configuration and mapping.

Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bit 1 **TEN1**: DAC channel1 trigger enable

This bit is set and cleared by software to enable/disable DAC channel1 trigger.
0: DAC channel1 trigger disabled and data written into the DAC_DHR1 register are transferred one dac_hclk clock cycle later to the DAC_DOR1 register
1: DAC channel1 trigger enabled and data from the DAC_DHR1 register are transferred three dac_hclk clock cycles later to the DAC_DOR1 register

Note: When software trigger is selected, the transfer from the DAC_DHR1 *register to the DAC_DOR1 register takes only one dac_hclk clock cycle.*

Bit 0 **EN1**: DAC channel1 enable

This bit is set and cleared by software to enable/disable DAC channel1.
0: DAC channel1 disabled
1: DAC channel1 enabled

21.7.2 DAC software trigger register (DAC_SWTRGR)

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.	Res.	Res.	SWTRIG2	SWTRIG1
														w	w

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SWTRIG2**: DAC channel2 software trigger

This bit is set by software to trigger the DAC in software trigger mode.

0: No trigger

1: Trigger

Note: This bit is cleared by hardware (one dac_hclk clock cycle later) once the DAC_DHR2 register value has been loaded into the DAC_DOR2 register.

This bit is available only on dual-channel DACs. Refer to Section 21.3: DAC implementation.

Bit 0 **SWTRIG1**: DAC channel1 software trigger

This bit is set by software to trigger the DAC in software trigger mode.

0: No trigger

1: Trigger

Note: This bit is cleared by hardware (one dac_hclk clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.

21.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)

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.	DACC1DHRB[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.	DACC1DHR[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 **DACC1DHRB[11:0]**: DAC channel1 12-bit right-aligned data B

These bits are written by software. They specify 12-bit data for DAC channel1 when the DAC operates in double data mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel1.

21.7.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC1DHRB[11:0]												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
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC1DHRB[11:0]**: DAC channel1 12-bit left-aligned data B
These bits are written by software. They specify 12-bit data for DAC channel1 when the DAC operates in double data mode.

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

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software.
They specify 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

21.7.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)

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
DACC1DHRB[7:0]								DACC1DHR[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 **DACC1DHRB[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1 when the DAC operates in double data mode.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1.

21.7.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2)

This register is available only on dual-channel DACs. Refer to *Section 21.3: DAC implementation*.

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.	DACC2DHRB[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.	DACC2DHR[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 **DACC2DHRB[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel2 when the DAC operates in DMA double data mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel2.

21.7.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2)

This register is available only on dual-channel DACs. Refer to *Section 21.3: DAC implementation*.

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHRB[11:0]												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
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC2DHRB[11:0]**: DAC channel2 12-bit left-aligned data B
These bits are written by software. They specify 12-bit data for DAC channel2 when the DAC operates in double data mode.

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

Bits 15:4 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specify 12-bit data for DAC channel2.

Bits 3:0 Reserved, must be kept at reset value.

21.7.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2)

This register is available only on dual-channel DACs. Refer to *Section 21.3: DAC implementation*.

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
DACC2DHRB[7:0]								DACC2DHR[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 **DACC2DHRB[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel2 when the DAC operates in double data mode.

Bits 7:0 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

21.7.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD)

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.	DACC2DHR[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.	DACC1DHR[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 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

21.7.10 Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHR[11:0]												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
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

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

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

21.7.11 Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD)

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
DACC2DHR[7:0]								DACC1DHR[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 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel1.

21.7.12 DAC channel1 data output register (DAC_DOR1)

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.	DACC1DORB[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.	DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 27:16 **DACC1DORB[11:0]**: DAC channel1 data output
These bits are read-only. They contain data output for DAC channel1 B.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DOR[11:0]**: DAC channel1 data output
These bits are read-only, they contain data output for DAC channel1.

21.7.13 DAC channel2 data output register (DAC_DOR2)

This register is available only on dual-channel DACs. Refer to *Section 21.3: DAC implementation*.

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DORB[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.	DACC2DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 27:16 **DACC2DORB[11:0]**: DAC channel2 data output
These bits are read-only. They contain data output for DAC channel2 B.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DOR[11:0]**: DAC channel2 data output
These bits are read-only, they contain data output for DAC channel2.

21.7.14 DAC status register (DAC_SR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BWST2	CAL_ FLAG2	DMAU DR2	DORST AT2	DAC2R DY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1	r	r											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BWST1	CAL_ FLAG1	DMAU DR1	DORST AT1	DAC1R DY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1	r	r											

Bit 31 **BWST2**: DAC channel2 busy writing sample time flag

This bit is systematically set just after sample and hold mode enable. It is set each time the software writes the register DAC_SHSR2, It is cleared by hardware when the write operation of DAC_SHSR2 is complete. (It takes about 3 LSI/LSE periods of synchronization).

0:There is no write operation of DAC_SHSR2 ongoing: DAC_SHSR2 can be written

1:There is a write operation of DAC_SHSR2 ongoing: DAC_SHSR2 cannot be written

Note: This bit is available only on dual-channel DACs. Refer to Section 21.3: DAC implementation.

Bit 30 **CAL_FLAG2**: DAC channel2 calibration offset status

This bit is set and cleared by hardware

0: calibration trimming value is lower than the offset correction value

1: calibration trimming value is equal or greater than the offset correction value

Note: This bit is available only on dual-channel DACs. Refer to Section 21.3: DAC implementation.

Bit 29 **DMAUDR2**: DAC channel2 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel2

1: DMA underrun error condition occurred for DAC channel2 (the currently selected trigger is driving DAC channel2 conversion at a frequency higher than the DMA service capability rate).

Note: This bit is available only on dual-channel DACs. Refer to Section 21.3: DAC implementation.

Bit 28 **DORSTAT2**: DAC channel2 output register status bit

This bit is set and cleared by hardware. It is applicable only when the DAC operates in double data mode.

0: DOR[11:0] is used actual DAC output

1: DORB[11:0] is used actual DAC output

Note: This bit is available only on dual-channel DACs. Refer to Section 21.3: DAC implementation.

Bit 27 **DAC2RDY**: DAC channel2 ready status bit

This bit is set and cleared by hardware.

0: DAC channel2 is not yet ready to accept the trigger nor output data

1: DAC channel2 is ready to accept the trigger or output data

Note: This bit is available only on dual-channel DACs. Refer to Section 21.3: DAC implementation.

Bits 26:16 Reserved, must be kept at reset value.

Bit 15 **BWST1**: DAC channel1 busy writing sample time flag

This bit is systematically set just after sample and hold mode enable and is set each time the software writes the register DAC_SHSR1, It is cleared by hardware when the write operation of DAC_SHSR1 is complete. (It takes about 3 LSI/LSE periods of synchronization).

0:There is no write operation of DAC_SHSR1 ongoing: DAC_SHSR1 can be written

1:There is a write operation of DAC_SHSR1 ongoing: DAC_SHSR1 cannot be written

Bit 14 **CAL_FLAG1**: DAC channel1 calibration offset status

This bit is set and cleared by hardware

0: calibration trimming value is lower than the offset correction value

1: calibration trimming value is equal or greater than the offset correction value

Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel1

1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bit 12 **DORSTAT1**: DAC channel1 output register status bit

This bit is set and cleared by hardware. It is applicable only when the DAC operates in double data mode.

0: DOR[11:0] is used actual DAC output

1: DORB[11:0] is used actual DAC output

Bit 11 **DAC1RDY**: DAC channel1 ready status bit

This bit is set and cleared by hardware.

0: DAC channel1 is not yet ready to accept the trigger nor output data

1: DAC channel1 is ready to accept the trigger or output data

Bits 10:0 Reserved, must be kept at reset value.

21.7.15 DAC calibration control register (DAC_CCR)

Address offset: 0x38

Reset value: 0x00XX 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM2[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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4:0]				
											rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **OTRIM2[4:0]**: DAC channel2 offset trimming value

These bits are available only on dual-channel DACs. Refer to *Section 21.3: DAC implementation*.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **OTRIM1[4:0]**: DAC channel1 offset trimming value

21.7.16 DAC mode control register (DAC_MCR)

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.	SINFORMAT2	DMADOUBLE2	Res.	Res.	Res.	Res.	Res.	MODE2[2:0]		
						rw	rw						rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HFSEL[1:0]		Res.	Res.	Res.	Res.	SINFORMAT1	DMADOUBLE1	Res.	Res.	Res.	Res.	Res.	MODE1[2:0]		
rw	rw					rw	rw						rw	rw	rw

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

Bit 25 **SINFORMAT2**: Enable signed format for DAC channel2

This bit is set and cleared by software.

0: Input data is in unsigned format

1: Input data is in signed format (2's complement). The MSB bit represents the sign.

Note: This bit is available only on dual-channel DACs. Refer to Section 21.3: DAC implementation.

Bit 24 **DMADOUBLE2**: DAC channel2 DMA double data mode

This bit is set and cleared by software.

0: DMA normal mode selected

1: DMA double data mode selected

Note: This bit is available only on dual-channel DACs. Refer to Section 21.3: DAC implementation.

Bits 23:19 Reserved, must be kept at reset value.

Bits 18:16 **MODE2[2:0]:** DAC channel2 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN2 = 0 and bit CEN2 = 0 in the DAC_CR register). If EN2 = 1 or CEN2 = 1 the write operation is ignored.

They can be set and cleared by software to select the DAC channel2 mode:

– DAC channel2 in normal mode

000: DAC channel2 is connected to external pin with Buffer enabled

001: DAC channel2 is connected to external pin and to on chip peripherals with buffer enabled

010: DAC channel2 is connected to external pin with buffer disabled

011: DAC channel2 is connected to on chip peripherals with Buffer disabled

– DAC channel2 in sample and hold mode

100: DAC channel2 is connected to external pin with Buffer enabled

101: DAC channel2 is connected to external pin and to on chip peripherals with Buffer enabled

110: DAC channel2 is connected to external pin and to on chip peripherals with Buffer disabled

111: DAC channel2 is connected to on chip peripherals with Buffer disabled

Note: This register can be modified only when EN2 = 0.

Refer to Section 21.3: DAC implementation for the availability of DAC channel2.

Bits 15:14 **HFSEL[1:0]**: High frequency interface mode selection
00: High frequency interface mode disabled
01: High frequency interface mode enabled for AHB clock frequency > 80 MHz
10: High frequency interface mode enabled for AHB clock frequency >160 MHz
11: Reserved

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **SINFORMAT1**: Enable signed format for DAC channel1
This bit is set and cleared by software.
0: Input data is in unsigned format
1: Input data is in signed format (2's complement). The MSB bit represents the sign.

Bit 8 **DMADOUBLE1**: DAC channel1 DMA double data mode
This bit is set and cleared by software.
0: DMA normal mode selected
1: DMA double data mode selected

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **MODE1[2:0]:** DAC channel1 mode
These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN1 = 0 and bit CEN1 = 0 in the DAC_CR register). If EN1 = 1 or CEN1 = 1 the write operation is ignored.
They can be set and cleared by software to select the DAC channel1 mode:
– DAC channel1 in normal mode
000: DAC channel1 is connected to external pin with Buffer enabled
001: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled
010: DAC channel1 is connected to external pin with Buffer disabled
011: DAC channel1 is connected to on chip peripherals with Buffer disabled
– DAC channel1 in sample & hold mode
100: DAC channel1 is connected to external pin with Buffer enabled
101: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled
110: DAC channel1 is connected to external pin and to on chip peripherals with Buffer disabled
111: DAC channel1 is connected to on chip peripherals with Buffer disabled
Note: This register can be modified only when EN1 = 0.

21.7.17 DAC channel1 sample and hold sample time register (DAC_SHSR1)

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
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 9:0 **TSAMPLE1[9:0]:** DAC channel1 sample time (only valid in sample and hold mode)

These bits can be written when the DAC channel1 is disabled or also during normal operation. in the latter case, the write can be done only when BWST1 of DAC_SR register is low, If BWST1 = 1, the write operation is ignored.

*Note: It represents the number of LSI/LSE clocks to perform a sample phase. Sampling time = (*TSAMPLE1[9:0] + 1) x *LSI/LSE clock period.*

21.7.18 DAC channel2 sample and hold sample time register (DAC_SHSR2)

This register is available only on dual-channel DACs. Refer to *Section 21.3: DAC implementation*.

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.	TSAMPLE2[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 9:0 **TSAMPLE2[9:0]:** DAC channel2 sample time (only valid in sample and hold mode)
These bits can be written when the DAC channel2 is disabled or also during normal operation. in the latter case, the write can be done only when BWST2 of DAC_SR register is low, if BWST2 = 1, the write operation is ignored.

*Note: It represents the number of LSI/LSE clocks to perform a sample phase. Sampling time = (*TSAMPLE1[9:0] + 1) x *LSI/LSE clock period.*

21.7.19 DAC sample and hold time register (DAC_SHHR)

Address offset: 0x48

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	THOLD2[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
Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 25:16 **THOLD2[9:0]:** DAC channel2 hold time (only valid in sample and hold mode).
Hold time = (THOLD[9:0]) x LSI/LSE clock period

Note: This register can be modified only when EN2 = 0.
These bits are available only on dual-channel DACs. Refer to Section 21.3: DAC implementation.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **THOLD1[9:0]:** DAC channel1 hold time (only valid in sample and hold mode)
Hold time = (THOLD[9:0]) x LSI/LSE clock period

Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in normal operating mode (when bit ENx = 0 and bit CENx = 0 in the DAC_CR register). If ENx = 1 or CENx = 1 the write operation is ignored.

21.7.20 DAC sample and hold refresh time register (DAC_SHRR)

Address offset: 0x4C

Reset value: 0x0001 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.	TREFRESH2[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.	TREFRESH1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **TREFRESH2[7:0]:** DAC channel2 refresh time (only valid in sample and hold mode)
Refresh time = (TREFRESH[7:0]) x LSI/LSE clock period

Note: This register can be modified only when EN2 = 0.
These bits are available only on dual-channel DACs. Refer to Section 21.3: DAC implementation.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **TREFRESH1[7:0]:** DAC channel1 refresh time (only valid in sample and hold mode)
Refresh time = (TREFRESH[7:0]) x LSI/LSE clock period

Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in normal operating mode (when bit ENx = 0 and bit CENx = 0 in the DAC_CR register). If ENx = 1 or CENx = 1 the write operation is ignored.

21.7.21 DAC register map

Table 139 summarizes the DAC registers.

Table 139. DAC register map and reset values

Offset	Register name reset value	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	**DAC_CR**	Res.	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[2:0]		TSEL2[3:1]			TSEL2[0]	TEN2	EN2	Res.	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[3:1]			TSEL1[0]	TEN1	EN1
	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	**DAC_SWTRGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG2	SWTRIG1
	Reset value																															0	0
0x08	**DAC_DHR12R1**	Res.	Res.	Res.	Res.	DACC1DHRB[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[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
0x0C	**DAC_DHR12L1**	DACC1DHRB[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]												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	0	0	0	0				
0x10	**DAC_DHR8R1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHRB[7:0]								DACC1DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**DAC_DHR12R2**	Res.	Res.	Res.	Res.	DACC2DHRB[11:0]												Res.	Res.	Res.	Res.	DACC2DHR[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
0x18	**DAC_DHR12L2**	DACC2DHRB[11:0]												Res.	Res.	Res.	Res.	DACC2DHR[11:0]												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	0	0	0	0				
0x1C	**DAC_DHR8R2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHRB[7:0]								DACC2DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**DAC_DHR12RD**	Res.	Res.	Res.	Res.	DACC2DHR[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[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
0x24	**DAC_DHR12LD**	DACC2DHR[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]												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	0	0	0	0				

Table 139. DAC register map and reset values (continued)

Offset	Register name reset value	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
0x28	**DAC_ DHR8RD**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[7:0]								DACC1DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**DAC_ DOR1**	Res.	Res.	Res.	Res.	DACC1DORB[11:0]												Res.	Res.	Res.	Res.	DACC1DOR[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
0x30	**DAC_ DOR2**	Res.	Res.	Res.	Res.	DACC2DORB[11:0]												Res.	Res.	Res.	Res.	DACC2DOR[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
0x34	**DAC_SR**	BWST2	CAL_FLAG2	DMAUDR2	DORSTAT2	DAC2RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BWST1	CAL_FLAG1	DMAUDR1	DORSTAT1	DAC1RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0												0	0	0	0	0											
0x38	**DAC_CCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM2[4]	OTRIM2[3]	OTRIM2[2]	OTRIM2[1]	OTRIM2[0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4]	OTRIM1[3]	OTRIM1[2]	OTRIM1[1]	OTRIM1[0]
	Reset value												X	X	X	X	X												X	X	X	X	X
0x3C	**DAC_MCR**	Res.	Res.	Res.	Res.	Res.	Res.	SINFORMAT2	DMADOUBLE2	Res.	Res.	Res.	Res.	Res.	MODE2 [2:0]			HFSEL[1]	HFSEL[0]	Res.	Res.	Res.	Res.	SINFORMAT1	DMADOUBLE1	Res.	Res.	Res.	Res.	Res.	MODE1 [2:0]		
	Reset value							0	0						0	0	0	0	0					0	0						0	0	0
0x40	**DAC_ SHSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x44	**DAC_ SHSR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE2[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x48	**DAC_ SHHR**	Res.	Res.	Res.	Res.	Res.	Res.	THOLD2[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
	Reset value							0	0	0	0	0	0	0	0	0	1							0	0	0	0	0	0	0	0	0	1
0x4C	**DAC_ SHRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH2[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
	Reset value									0	0	0	0	0	0	0	1									0	0	0	0	0	0	0	1
0x50-0x54	Reserved	Res.																															
0x58-0x60	Reserved	Res.																															
0x64-0x68	Reserved	Res.																															

Refer to *Section 2.2* for the register boundary addresses.

22 Comparator (COMP)

22.1 Introduction

The device embeds an ultra-low-power comparator. It can be used for a variety of functions including:

- Wake-up from low-power mode triggered by an analog signal
- Analog signal conditioning
- Cycle-by-cycle current control loop when combined with a PWM output from a timer

22.2 COMP main features

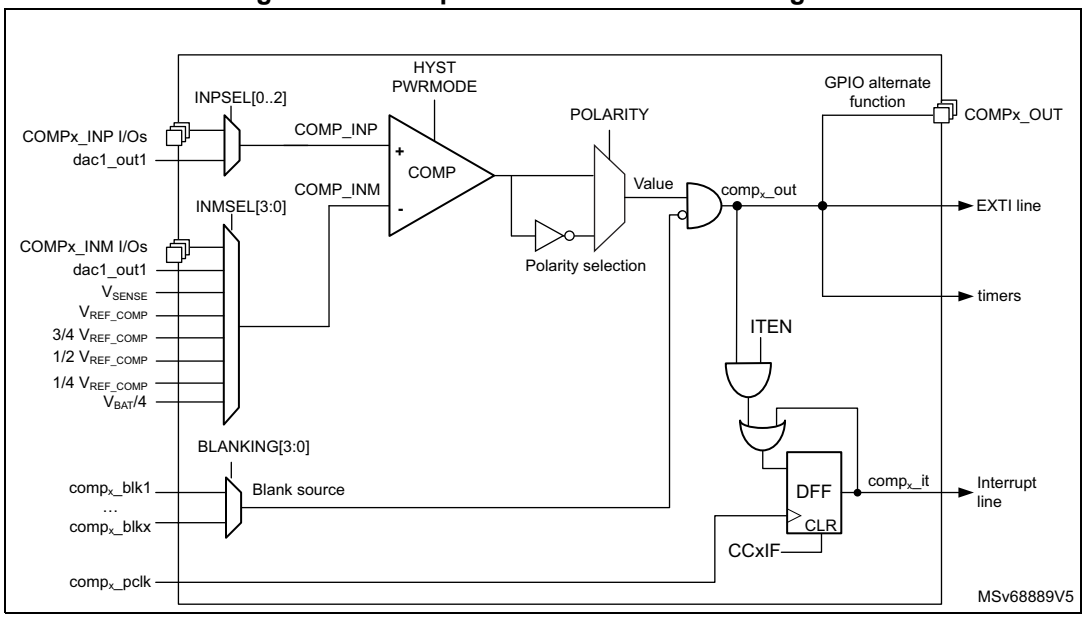
- Selectable inverting analog inputs:
  - I/O pins
  - DAC Channel1 outputs
  - Internal reference voltage and three sub multiple values (1/4, 1/2, 3/4) provided by the scaler (buffered voltage divider)
- Three I/O pins selectable as noninverting analog inputs
- Programmable hysteresis
- Programmable speed/consumption
- Mapping of outputs to I/Os
- Redirection of outputs to timer inputs for triggering:
  - Capture events
  - OCREF_CLR events (for cycle-by-cycle current control)
  - Break events for fast PWM shutdowns
- Blanking of comparator outputs
- Interrupt generation capability with wake-up from sleep mode and stop modes (through the EXTI controller)
- Direct interrupt output to the CPU

22.3 COMP functional description

22.3.1 COMP block diagram

The block diagram of the comparators is shown in the figure below.

Figure 143. Comparator functional block diagram



22.3.2 COMP pins and internal signals

The I/Os used as comparator inputs must be configured in analog mode in the GPIO registers.

The comparator outputs can be connected to the I/Os through their alternate functions. Refer to the product datasheet for more information.

The outputs can also be internally redirected to a variety of timer inputs for the following purposes:

- emergency shut-down of PWM signals, using BKIN and BKIN2 inputs
- cycle-by-cycle current control, using ETR inputs of timers
- input capture for timing measurements

The comparator output can be routed in a simultaneous way internally and to the I/O pins.

Table 140. COMP1 input/output internal signals

Signal name	Signal type	Description
dac1_out1	Analog input	Analog signal from DAC peripheral
comp1_blkx	Digital output	See *Table 141*
comp1_out	Digital output	COMP1 output
comp1_it	Digital output	COMP1 interrupt out

Table 141. COMP1 blanking source selection

Signal name	Signal type	Blanking input source
comp1_blk1	Digital input	tim1_oc5
comp1_blk2	Digital input	tim2_oc3
comp1_blk3	Digital input	tim3_oc3
comp1_blk4	Digital input	tim3_oc4
comp1_blk5	Digital input	lptim1_ch2
comp1_blk6	Digital input	lptim2_ch2

Table 142. COMP1 noninverting input assignment

COMP1_INP	COMP1_INPSEL2	COMP1_INPSEL1	COMP1_INPSEL0
COMP1_INP1	X	X	1
COMP1_INP2	0	0	0
COMP1_INP3	0	1	0
dac1_out1	1	X	0

Note: Other combinations are reserved.

Table 143. COMP1 inverting input assignment

COMP1_INM	COMP1_INMSEL[3:0]
¼ V_{REFINT}	0000
½ V_{REFINT}	0001
¾ V_{REFINT}	0010
V_{REFINT}	0011
dac1_out1	0100
COMP1_INM1	0101
COMP1_INM2	0110
COMP1_INM3	0111
V_{SENSE}	1000
V_{BAT}/4	1001

Note: Other combinations are reserved.

Table 144. COMP1 blanking source assignment

Banking source	COMP1_BLANKING[3:0]
No blanking	0000
tim1_oc5	0001
tim2_oc3	0010

Table 144. COMP1 blanking source assignment (continued)

Banking source	COMP1_BLANKING[3:0]
tim3_oc3	0011
tim3_oc4	0100
lptim1_ch2	0101
lptim2_ch2	0110

22.3.3 COMP reset and clocks

The clock comp_pclk provided by the clock controller is synchronous with the APB clock.

Note: ***Important:*** *The polarity selection logic and the output redirection to the port work independently from the APB clock. This allows the comparator to work even in stop mode. The interrupt line, connected to the NVIC of the CPU, requires the APB clock (comp_pclk) to work. In the absence of the APB clock, the interrupt signal comp_it cannot be generated.*

22.3.4 Comparator LOCK mechanism

The comparators can be used for safety purposes, such as over-current or thermal protection. For applications with specific functional safety requirements, the comparator configuration can be protected against undesired alteration that could happen, for example, at program counter corruption.

For this purpose, the comparator configuration registers can be write-protected (read-only).

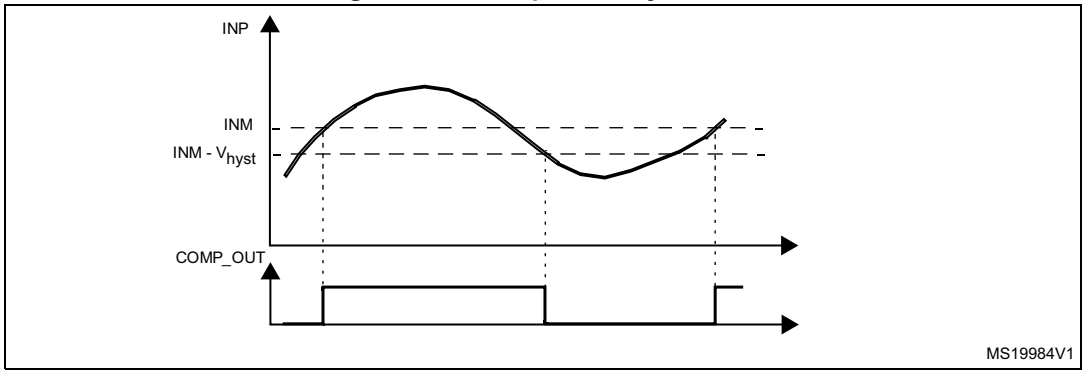
Upon configuring a comparator channel, its LOCK bit is set to 1. This causes the whole register set of the comparator channel to become read-only, the LOCK bit inclusive.

The write protection can only be removed through the MCU reset.

22.3.5 Hysteresis

The comparator includes a programmable hysteresis to avoid spurious output transitions in case of noisy signals. The hysteresis can be disabled if it is not needed, for instance when exiting from low-power mode. This enables the possibility to force the hysteresis value using external components.

Figure 144. Comparator hysteresis

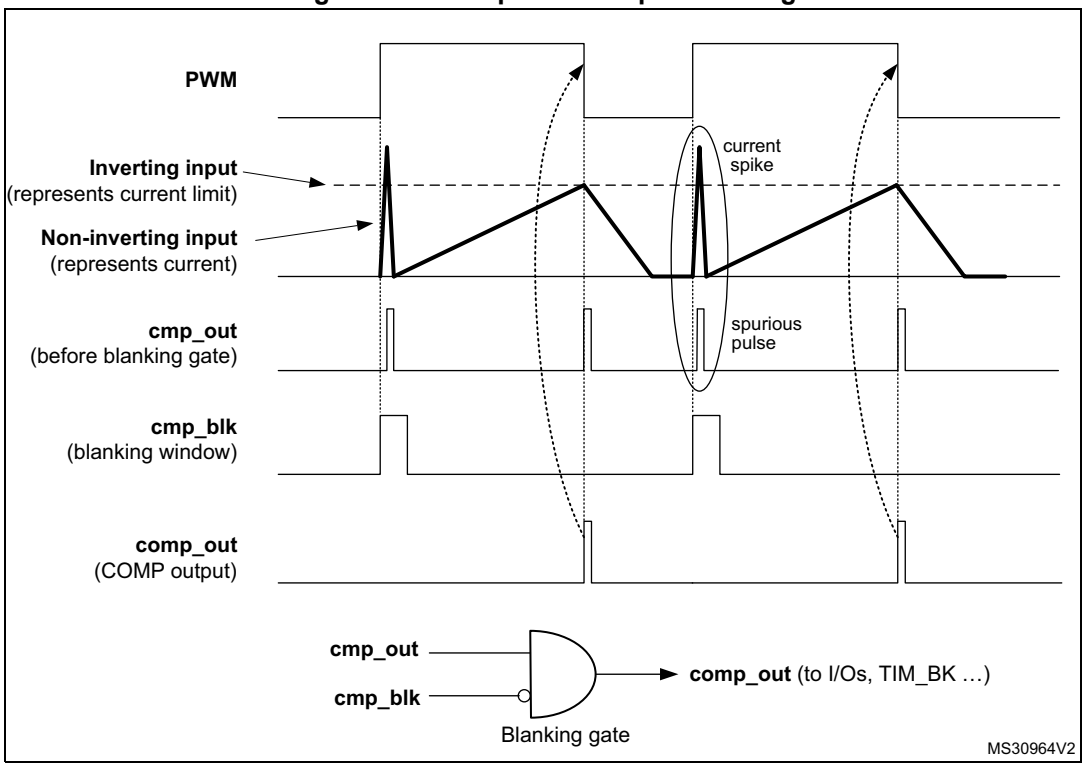


22.3.6 Comparator output blanking function

The purpose of the blanking function is to prevent the current regulation to trip upon short current spikes at the beginning of the PWM period (typically the recovery current in power switches antiparallel diodes). It uses a blanking window defined with a timer output compare signal. Refer to the register description for selectable blanking signals.

The blanking signal gates the internal comparator output such as to clean the comp_out from spurious pulses due to current spikes, as depicted in the following figure (the COMP channel number is not represented).

Figure 145. Comparator output blanking



22.3.7 COMP startup stabilization time

The comparator parameters and its output are undefined after the COMP enable (power-on) for a period defined as startup stabilization time, see the product datasheet for more details. If the comparator output is driving any critical stage of the design, it is recommended to redirect the comparator output to the device I/O through the GPIO alternate function or through the IP configuration (such as timers) after the stabilization time elapses.

22.3.8 COMP power and speed modes

The power consumption of the COMP channels versus propagation delay can be adjusted to have the optimum trade-off for a given application.

The bits PWRMODE[1:0] in the COMP_CFGRx registers can be programmed as follows:

- 00: High speed/high power
- 01: Medium speed/medium power
- 10: Medium speed/medium power
- 11: Very-low speed/ultra-low-power

22.4 COMP low-power modes

Table 145. Comparator behavior in low-power modes

Mode	Description
Sleep	No effect on the comparators. Comparator interrupts cause the device to exit the Sleep mode.
Stop	No effect on the comparators. Comparator interrupts cause the device to exit the Stop mode.

Note: The comparators cannot be used to exit the device from sleep mode or stop mode when the internal reference voltage is switched off.

22.5 COMP interrupts

There are two ways to use the comparator as an interrupt source.

The comparator outputs are internally connected to the extended interrupt and event controller (EXTI). Each comparator has its own EXTI line and can generate either interrupts or events to make the device exit low-power modes.

The comparator also provides an interrupt line to the NVIC of the CPU. This functionality is used when the CPU is active to handle low latency interrupt. It requires the APB clock running.

22.5.1 Interrupt through EXTI

Refer to the interrupt and events section for more details.

Sequence to enable the COMP interrupt through the EXTI block:

1. Enable the COMP and wait for stabilization.
2. Configure the EXTI line, receiving the comp_wkup signal, in interrupt mode, select the rising, falling, or either-edge sensitivity and enable the EXTI line.
3. Configure and enable the NVIC IRQ channel mapped to the corresponding EXTI lines.

Table 146. Interrupt through EXTI control bits

Interrupt event	Event flag	Enable control bit	Exit Sleep mode	Exit Stop modes	Exit Standby mode
comp1_wkup	Through EXTI	Through EXTI	Yes	Yes	N/A

22.5.2 Interrupt through the NVIC of the CPU

Sequence to enable the COMP interrupt through the NVIC of the CPU:

1. Enable the COMP and wait for stabilization.
2. Configure and enable the NVIC IRQ channel mapped to the comp_it line.
3. Configure and enable the ITEN in COMP_CFGRx.

Table 147. Interrupt through NVIC control bits

Interrupt event	Interrupt flag	Enable control bit	Interrupt clear bit	Exit Sleep mode	Exit Stop modes
comp1_it	C1IF in COMP_CFGR1	ITEN in COMP_CFGR1	CC1IF	Yes (with APB clock)	No

Note: It is mandatory to enable the APB clock to use this interrupt. If the clock is not enabled, an interrupt is not generated.
The interrupt is active on level and not automatically clear by hardware. It must be cleared in the comparator interrupt handler by firmware with ITEN=0 (COMP_CFGR1 register).

The interrupt signal comp_it is generated on comparator level, for this reason: upon interruption occurrence, the comp_it signal must be set to low as soon as possible to avoid comparator interrupt handler reentrance. This can be done by setting ITEN=0 or changing the comparator POLARITY bit.

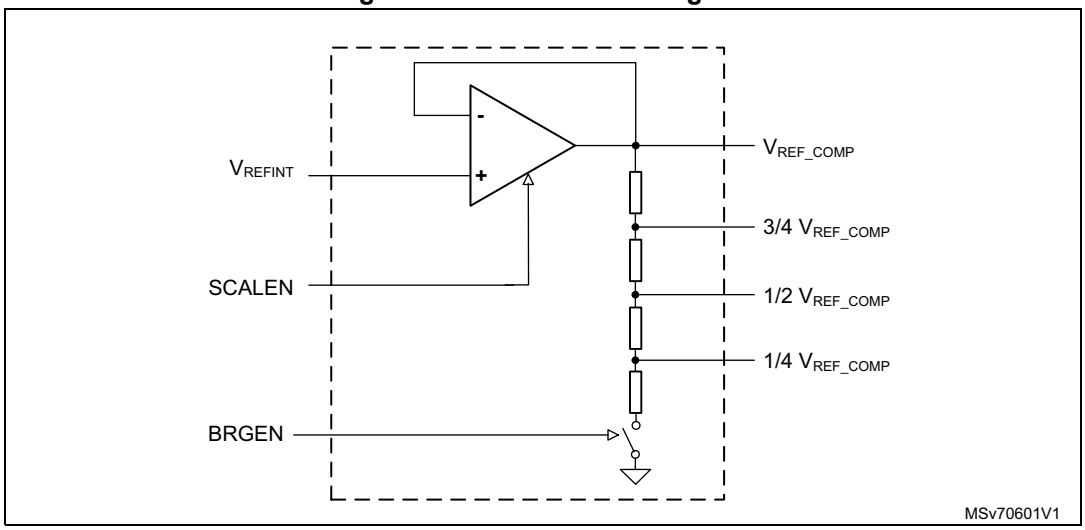
22.6 SCALER function

The scaler block is available to provide the different voltage reference levels to the comparator inputs. It is based on an amplifier driving a resistor bridge. The amplifier input is connected to the internal voltage reference.

The amplifier and the resistor bridge can be enabled separately. The amplifier is enabled by the SCALEN bits of the COMP_CFGRx registers. The resistor bridge is enabled by the BRGEN bits of the COMP_CFGRx registers.

When the resistor divided voltage is not used, the resistor bridge can be disconnected in order to reduce the consumption. When it is disconnected, the 1/4 V_{REF_COMP}, 1/2 V_{REF_COMP}, and 3/4 V_{REF_COMP} levels are equal to V_{REF_COMP}.

Figure 146. Scaler block diagram



22.7 COMP registers

22.7.1 Comparator status register (COMP_SR)

The COMP_SR is the comparator status register.

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.	C1IF
															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.	C1VAL
															r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **C1IF**: COMP Channel1 interrupt flag
This bit is set by hardware when the COMP Channel1 output is set
This bit is cleared by software writing 1 the CC1IF bit in the COMP_ICFR register.

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **C1VAL:** COMP Channel1 output status bit
This bit is read-only. It reflects the current COMP Channel1 output taking into account POLARITY and BLANKING bits effect.

22.7.2 Comparator interrupt clear flag register (COMP_ICFR)

The COMP_ICFR is the comparator interrupt clear flag register.

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.	CC1IF
															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.	Res.

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **CC1IF**: Clear COMP Channel1 interrupt flag
Writing 1 clears the C1IF flag in the COMP_SR register.

Bits 15:0 Reserved, must be kept at reset value.

22.7.3 Comparator configuration register 1 (COMP_CFGR1)

The COMP_CFGR1 is the COMP configuration register.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	Res.	Res.	BLANKING[3:0]				Res.	INPSEL2	Res.	INPSEL1	INMSEL[3:0]			
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
Res.	Res.	PWRMODE[1:0]		Res.	Res.	HYST[1:0]		Res.	ITEN	Res.	Res.	POLARITY	SCALEN	BRGEN	EN
		rw	rw			rw	rw		rw			rw	rw	rw	rw

Bit 31 **LOCK:** Lock
This bit is set by software and cleared by a hardware system reset. It locks the whole content of the COMP Channel1 configuration register COMP_CFGR1[31:0]
0: COMP_CFGR1[31:0] register is read/write
1: COMP_CFGR1[31:0] is read-only

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:24 **BLANKING[3:0]:** COMP Channel1 blanking source selection
Bits of this field are set and cleared by software (only if LOCK not set).
The field selects the input source for COMP Channel1 output blanking:
0000: No blanking
0001: comp_blk1
0010: comp_blk2
0011: comp_blk3
0100: comp_blk4
0101: comp_blk5
0110: comp_blk6
All other values: reserved

Bit 23 Reserved, must be kept at reset value.

Bit 22 **INPSEL2**: COMP noninverting input selection
This bit is set and cleared by software (only if LOCK not set). They select which input is connected to the positive input of the COMP channel.
See *Table 142: COMP1 noninverting input assignment* for more details.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **INPSEL1**: COMP noninverting input selection
This bit is set and cleared by software (only if LOCK not set). They select which input is connected to the positive input of COMP channel.
Note: See Table 142: COMP1 noninverting input assignment for more details.

Bits 19:16 **INMSEL[3:0]**: COMP channel1 inverting input selection
These bits are set and cleared by software (only if LOCK not set). They select which input is connected to the input minus of the COMP channel.
Note: See Table 143: COMP1 inverting input assignment for more details.

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

Bits 13:12 **PWRMODE[1:0]:** Power mode of the COMP channel1
These bits are set and cleared by software (only if LOCK not set). They control the power/speed of the COMP channel1.
00: High speed/high power
01: Medium speed/medium power
10: Medium speed/medium power
11: Ultra low power/ultra-low-power

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **HYST[1:0]:** COMP channel1 hysteresis selection
These bits are set and cleared by software (only if LOCK not set). They select the hysteresis voltage of the COMP channel1.
00: No hysteresis
01: Low hysteresis
10: Medium hysteresis
11: High hysteresis

Bit 7 Reserved, must be kept at reset value.

Bit 6 **ITEN:** COMP channel1 interrupt enable
This bit is set and cleared by software (only if LOCK not set). This bit enable the interrupt generation of the COMP channel1.
0: Interrupt generation disabled for COMP channel1
1: Interrupt generation enabled for COMP channel1

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

Bit 3 **POLARITY:** COMP channel1 polarity selection

This bit is set and cleared by software (only if LOCK not set). It inverts COMP channel1 polarity.

0: COMP channel1 output is not inverted

1: COMP Channel1 output is inverted

Bit 2 **SCALEN**: Voltage scaler enable

This bit is set and cleared by software (only if LOCK not set). This bit enables the V_{REFINT} scaler for the COMP channels.

0: V_{REFINT} scaler disabled

1: V_{REFINT} scaler enabled

Bit 1 **BRGEN:** Scaler bridge enable

This bit is set and cleared by software (only if LOCK not set). This bit enables the bridge of the scaler.

0: Scaler resistor bridge disabled

1: Scaler resistor bridge enabled

If SCALEN is set and BRGEN is reset, all four scaler outputs provide the same level V_{REF_COMP} (similar to V_{REFINT}).

If SCALEN and BRGEN are set, the four scaler outputs provide V_{REF_COMP}, 3/4 V_{REF_COMP}, 1/2 V_{REF_COMP} and 1/4 V_{REF_COMP} levels, respectively.

Bit 0 **EN:** COMP Channel1 enable

This bit is set and cleared by software (only if LOCK not set). It enables the COMP Channel1.

0: Disable

1: Enable

22.7.4 Comparator configuration register 2 (COMP_CFGR2)

The COMP_CFGR2 is the COMP configuration register.

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs															
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.	INPSEL0	Res.	Res.	Res.	Res.
											rw				

Bit 31 **LOCK**: Lock

This bit is set by software and cleared by a hardware system reset. It locks the whole content of the COMP Channel1 configuration register COMP_CFGR2[31:0]

0: COMP_CFGR2[31:0] register is read/write

1: COMP_CFGR2[31:0] is read-only

Bits 30:5 Reserved, must be kept at reset value.

Bit 4 **INPSEL0**: COMP non-inverting input selection
This bit is set and cleared by software (only if LOCK not set). They select which input is connected to the positive input of COMP channel.
See *Table 142: COMP1 noninverting input assignment* for more details.

Bits 3:0 Reserved, must be kept at reset value.

22.7.5 COMP register map

Table 148. COMP 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	**COMP_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C1IF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C1VAL
	Reset value																0																0
0x04	**COMP_ICFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																0																
0x0C	**COMP_ CFGR1**	LOCK	Res.	Res.	Res.	BLANKING [3:0]				Res.	.INPSEL2	Res.	INPSEL1	INMSEL [3:0]				Res.	Res.	PWRMODE [1:0]		Res.	Res.	HYST[1:0]		Res.	ITEN	Res.	Res.	POLARITY	SCALEN	BRGEN	EN
	Reset value	0				0	0	0	0		0		0	0	0	0	0			0	0			0	0		0			0	0	0	0
0x10	**COMP_ CFGR2**	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INPSEL0	Res.	Res.	Res.	Res.
	Reset value	0																											0				

Refer to *Section 2.2* for the register boundary addresses.

23 Operational amplifiers (OPAMP)

23.1 Introduction

The devices embed one operational amplifier with two inputs and one output. The three I/Os can be connected to the external pins, thus enabling any type of external interconnections. The operational amplifiers can be configured internally in three possible ways:

- as a follower
- as an amplifier with a non-inverting gain ranging from 2 to 16
- as an amplifier with inverting gain ranging from -1 to -15.

Refer to *Section 23.3.3: Signal routing* for detailed information on OPAMP input and output connection to internal peripherals.

23.2 OPAMP main features

- Rail-to-rail input voltage range
- Low input offset voltage (1.5 mV after calibration, 10 mV with factory calibration)
- 7 MHz gain bandwidth
- High-speed mode to achieve a better slew rate

Note: Refer to the product datasheet for detailed OPAMP characteristics.

23.3 OPAMP functional description

The OPAMP has several modes.

Each OPAMP can be individually enabled, when disabled the output is high-impedance.

When enabled, it can be in calibration mode, all input, and output of the OPAMP are then disconnected, or in functional mode.

There are two functional modes, the high-speed mode and the normal mode. In functional mode, the inputs and output of the OPAMP are connected as described in *Section 23.3.3: Signal routing*.

23.3.1 OPAMP reset and clocks

The operational amplifier clock is necessary for accessing the registers. When the application does not need to have a read or write access to those registers, the clock can be switched off using the peripheral clock enable register. See OPAMPEN bit in *Section 10.8.24: RCC APB1 peripheral clock register (RCC_APB1LENR)*.

The bit OPAEN enables and disables the OPAMP operation. The OPAMP registers configurations should be changed before enabling the OPAEN bit in order to avoid spurious effects on the output.

When the output of the operational amplifier is no more needed, the operational amplifier can be disabled to save power. All the configurations previously set (including the calibration) are maintained while OPAMP is disabled.

23.3.2 Initial configuration

The default configuration of the operational amplifier is a functional mode where the three input/outputs are connected to external pins. In the default mode, the operational amplifier uses the factory trimming values for its offset calibration. See electrical characteristics section of the datasheet for factory trimming conditions, usually the temperature is 30 °C and the voltage is 3 V. The trimming values can be adjusted. See *Section 23.3.5: Calibration* for changing the trimming values. The default configuration uses the normal mode, which provides the standard performance. The bit OPAHSM can be set in order to switch the operational amplifier to high-speed mode for a better slew rate. Both normal and high-speed mode characteristics are defined in *section: Electrical characteristics* of the datasheet.

As soon as the OPAEN bit in the OPAMPx_CSR register is set, the operational amplifier is functional. The two input pins and the output pin are connected as defined in *Section 23.3.3: Signal routing* and the default connection settings can be changed.

Note: The inputs and output pins must be configured in analog mode (default state) in the corresponding GPIOx_MODER register.

23.3.3 Signal routing

The OPAMPx_CSR register determines the routing for the operational amplifier pins.

The connections of the operational amplifier (OPAMP1) are described in the table below.

Table 149. Possible connections for OPAMP

Signal	Pin	Internal	comment
OPAMP1_VINM	PC5(INM0) PB1(INM1)	ADC1_INP8 ADC1_INM4 ADC1_INP5 COMP1_INM6 OPAMP1_VOUT or PGA	The bits PGA_GAIN and VM_SEL control this pin. ADC controls the ADC input.
OPAMP1_VINP	PB0(INP0) PA0(INP2)	dac1_out1 (INP1) ADC1_INM5 ADC1_INP9 ADC1_INP0 ADC1_INM1 COMP1_INP1	The bit VP_SEL controls this pin. ADC controls the ADC input.
OPAMP1_VOUT	PA7	ADC1_INM3 ADC1_INP7	The pin is connected when the OPAMP is enabled. The ADC input is controlled by ADC.

23.3.4 OPAMP modes

The operational amplifier inputs and outputs are all accessible on terminals. The amplifiers can be used in multiple configuration environments:

- Standalone mode (external gain setting mode)
- Follower configuration mode
- PGA modes

Note: The amplifier output pin is directly connected to the output pad to minimize the output impedance. When the amplifier is enabled, it cannot be used as a general-purpose I/O, even if the amplifier is configured as a PGA and only connected to the internal channel.

The impedance of the signal must be maintained below a level that avoids the input leakage to create significant artifacts (due to a resistive drop in the source). Please refer to the electrical characteristics section in the datasheet for further details.

Standalone mode (external gain setting mode)

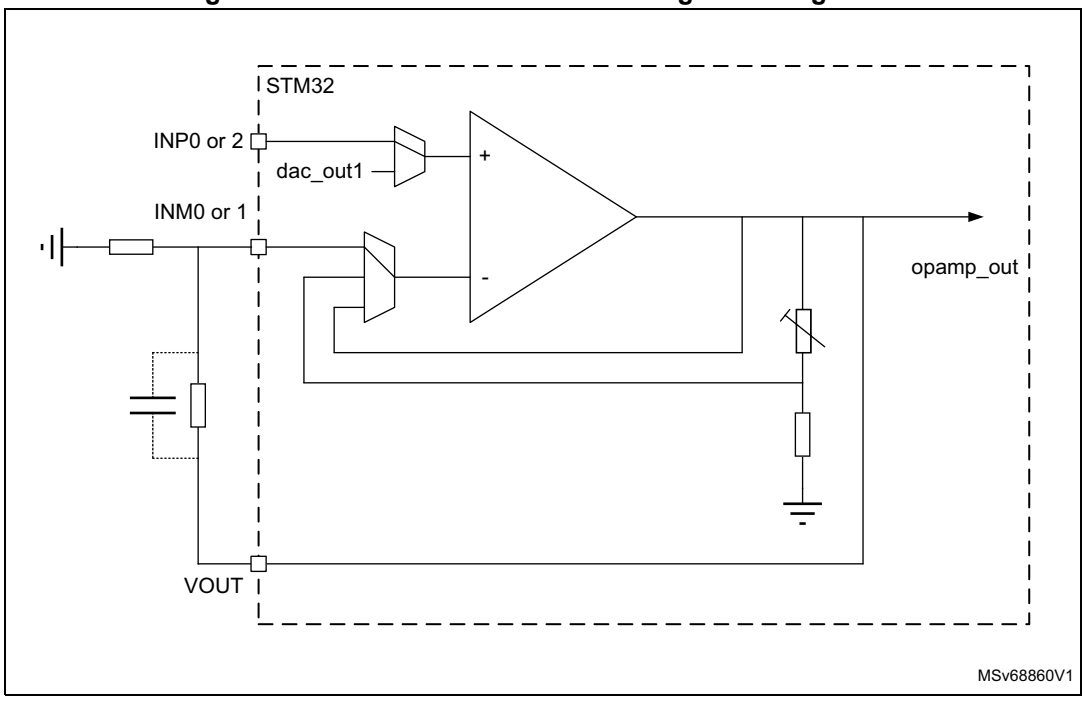
The procedure to use the OPAMP in standalone mode is presented below.

Starting from the default value of OPAMPx_CSR, and the default state of GPIOx_MODER, as soon as the OPAEN bit is set, the two input pins and the output pin are connected to the operational amplifier.

This default configuration uses the factory trimming values and operates in normal mode (highest performance). The behavior of the OPAMP can be changed as follows:

- OPAHSM can be set to “operational amplifier high speed” mode in order to have high slew rate.
- USERTRIM can be set to modify the trimming values for input offsets.

Figure 147. Standalone mode: external gain setting mode



Follower configuration mode

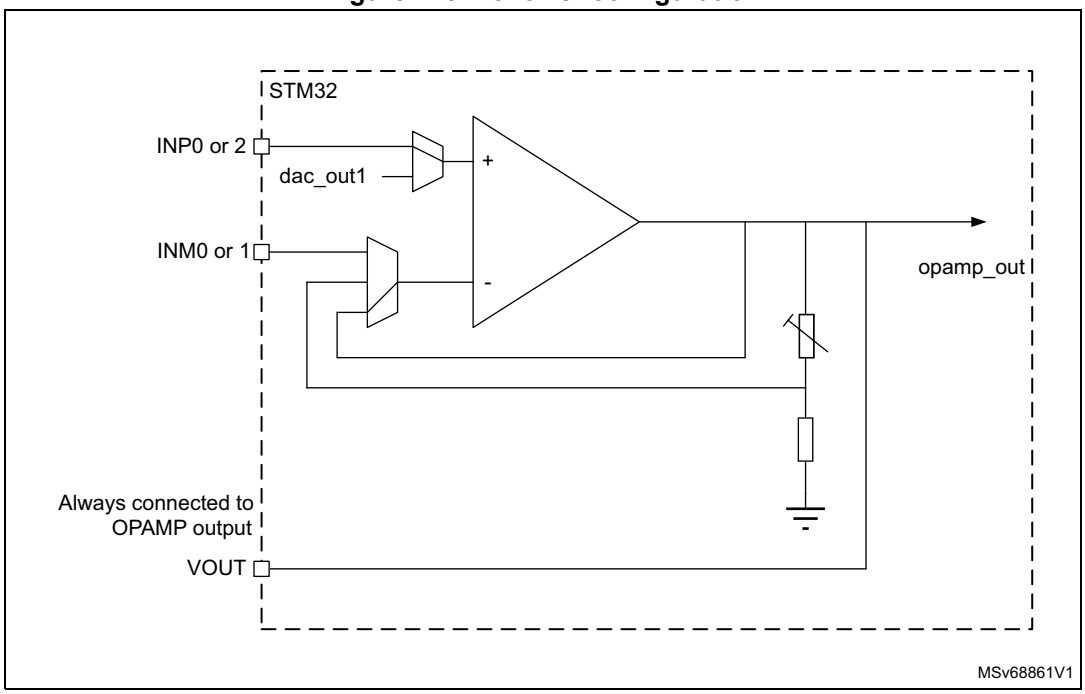
The procedure to use the OPAMP in follower mode is presented below.

- configure VM_SEL bits as “opamp_out connected to OPAMPx_VINM input”, 11
- configure VP_SEL bits as “GPIO connected to OPAMPx_VINP”, 00 or 10
- As soon as the OPAEN bit is set, the voltage on the none-inverting input is buffered to pin OPAMP1_VOUT.

Note: The pin corresponding to OPAMPx_VINM is free for another usage.

The signal on the OPAMP1 output is also seen as an ADC input. As a consequence, the OPAMP configured in follower mode can be used to perform impedance adaptation on input signals before feeding them to the ADC input, assuming the input signal frequency is compatible with the operational amplifier gain bandwidth specification.

Figure 148. Follower configuration



Programmable gain amplifier mode

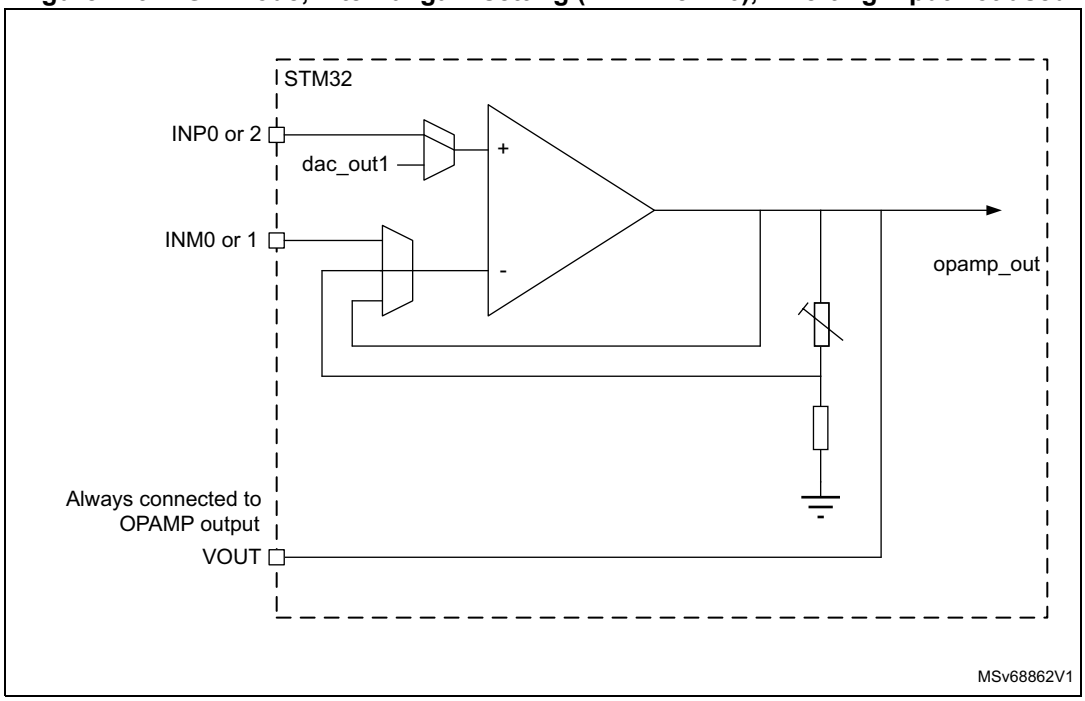
The procedure to use the OPAMP as programmable gain amplifier is presented below.

- configure VM_SEL bits as “Feedback resistor is connected to OPAMPx_VINM input”, 10
- configure PGA_GAIN bits as “internal gain 2, 4, 8 or 16”, 0000 to 0011
- configure VP_SEL bits as “GPIO connected to OPAMPx_VINP”, 00 or 10

  As soon as the OPAEN bit is set, the selected gain amplifies the none-inverting input voltage, and it is visible on pin OPAMPx_VOUT.

Note: To avoid saturation, the input voltage should stay below V_{DDA} divided by the selected gain.

Figure 149. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input not used



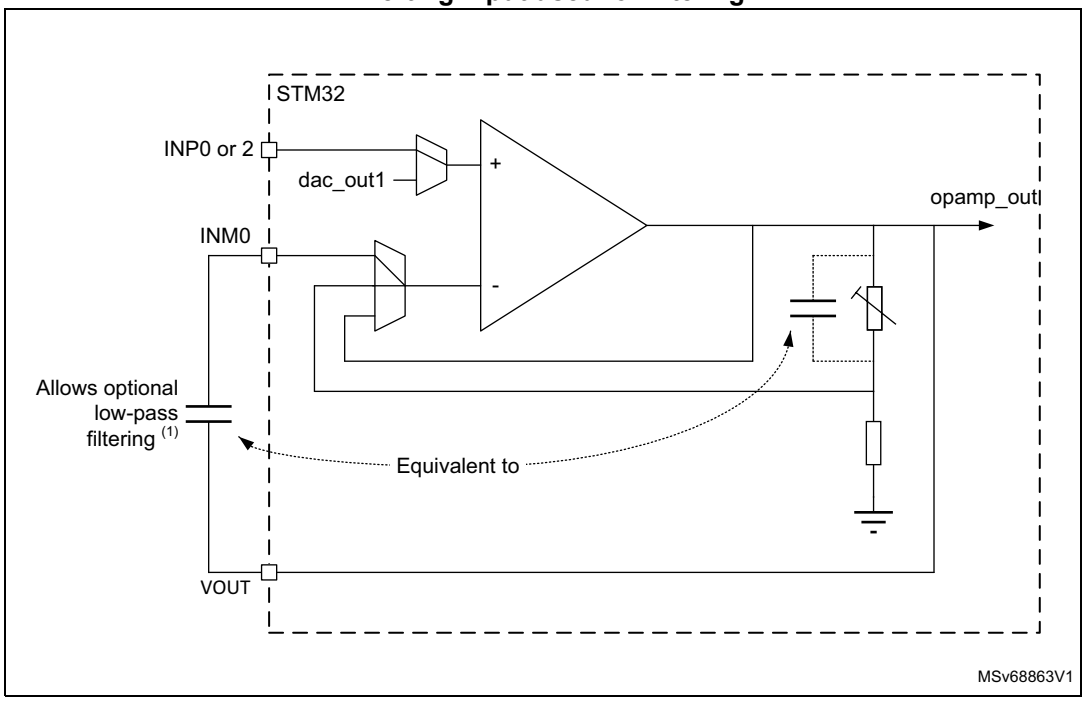
Programmable gain amplifier mode with external filtering

The procedure to use the OPAMP to amplify the amplitude of an input signal, with an external filtering, is presented below.

- configure VM_SEL bits as “Feedback resistor is connected to OPAMPx_VINM input”, 10
- configure PGA_GAIN bits as “internal gain 2, 4, 8 or 16 with filtering on INM0”, 0100 to 0111
- configure VP_SEL bits as “GPIO connected to OPAMPx_VINP”.

  Any external connection on INM can be used in parallel with the internal PGA. For example, a capacitor can be connected between opamp_out and INM for filtering purpose. Refer to the datasheet for the value of resistors used in the PGA resistor network.

Figure 150. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input used for filtering



1. The gain depends on the cut-off frequency.

Programmable gain amplifier, non-inverting with external bias or inverting mode

The procedure to use the OPAMP to amplify the amplitude of an input signal with bias voltage for non-inverting mode or inverting mode.

- configure VM_SEL bits as “Feedback resistor is connected to OPAMPx_VINM input”, 10
- configure PGA_GAIN bits as “Inverting gain=-1,-3,-7,-15/ Non-inverting gain =2,4,8,16 with INM0”, 1000 to 1011
- configure VP_SEL bits as “GPIO connected to OPAMPx_VINP”.

Figure 151. PGA mode, non-inverting gain setting (x2/x4/x8/x16) or inverting gain setting (x-1/x-3/x-7/x-15)

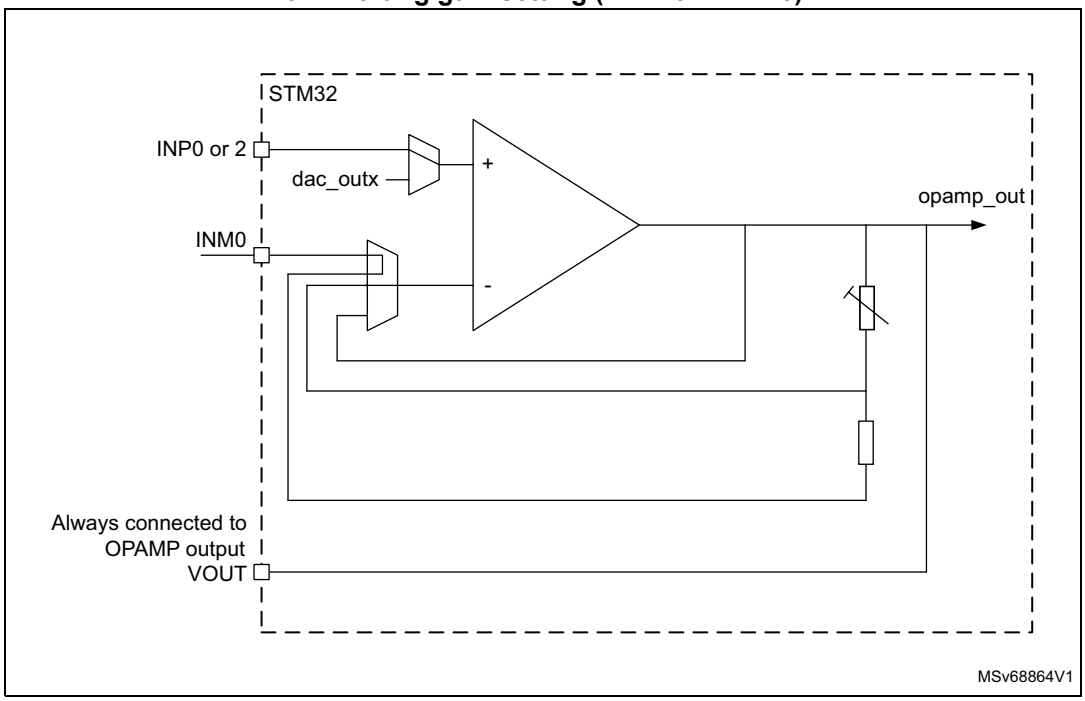
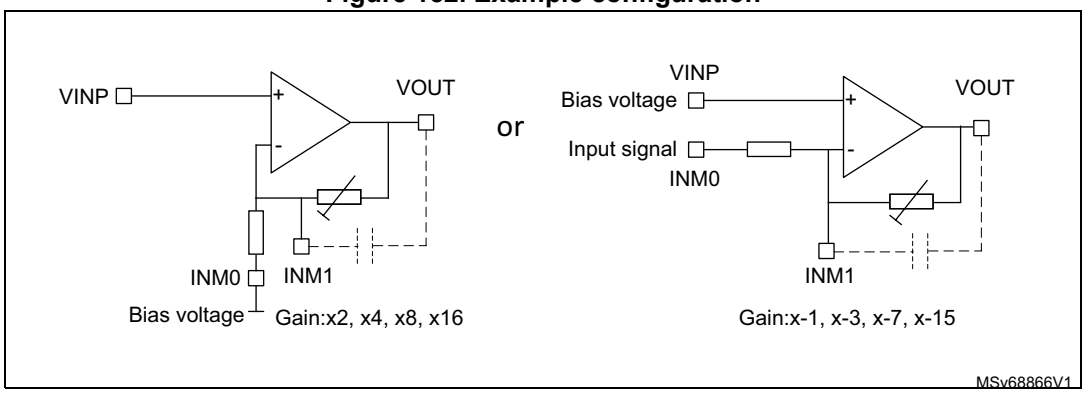


Figure 152. Example configuration



Programmable gain amplifier, non-inverting with external bias or inverting mode with filtering

The procedure to use the OPAMP to amplify the amplitude of an input signal with bias voltage for non-inverting mode or inverting mode with filtering

- configure VM_SEL bits as “Feedback resistor is connected to OPAMPx_VINM input”, 10
- configure PGA_GAIN bits as “Inverting gain=-1,-3,-7,-15/ Non-inverting gain =2,4,8,16 with INM0, INM1 node for filtering”, 1100 to 1111
- configure VP_SEL bits as “GPIO connected to OPAMPx_VINP”.

  Any external connection on VM1 can be used in parallel with the internal PGA. For example, a capacitor can be connected between opamp_out and VM1 for filtering purposes. See datasheet for the value of resistors used in the PGA resistor network.

Figure 153. PGA mode, non-inverting gain setting (x2/x4/x8/x16) or inverting gain setting (x-1/x-3/x-7/x-15) with filtering

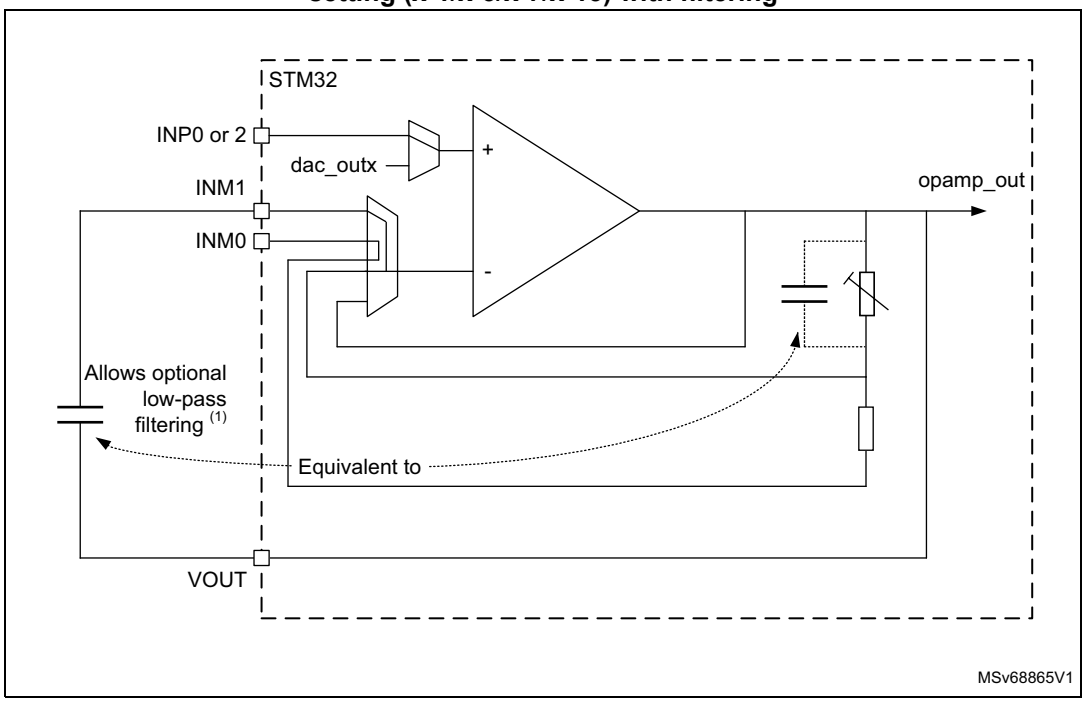
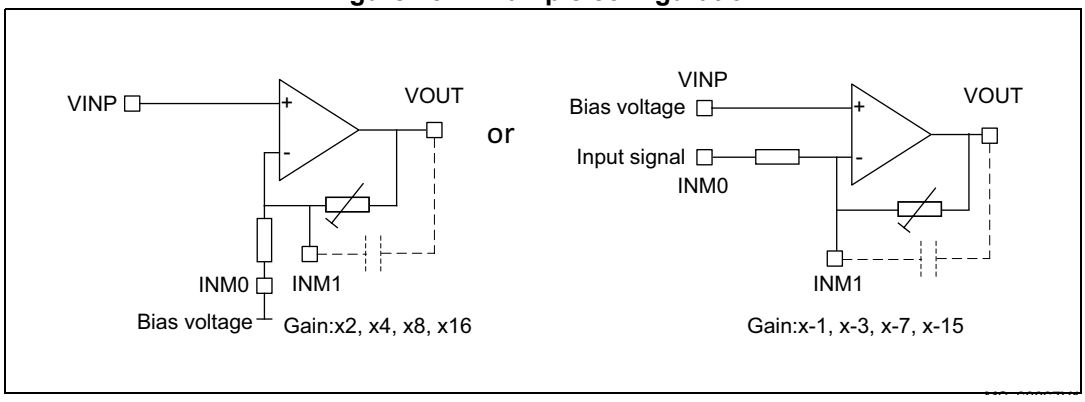


Figure 154. Example configuration



23.3.5 Calibration

The OPAMP interface continuously sends trimmed offset values to the operational amplifiers. At startup, the trimming values are initialized with the preset ‘factory’ trimming value.

The user can trim each operational amplifier. Specific registers allow the user to have different trimming values for normal mode and for high-speed mode.

The aim of the calibration is to cancel as much as possible the OPAMP inputs offset voltage. The calibration circuitry allows the user to reduce the input offset voltage to less than +/- 1.5 mV within stable voltage and temperature conditions.

For each operational amplifier and each mode two trimming value needs to be trimmed, one for N differential pair and one for P differential pair.

There are two registers for trimming the offsets for each operational amplifier, one for normal mode (OPAMPx_OTR) and one high-speed mode (OPAMPx_HSOTR). Each register is composed of five bits for P differential pair trimming and five bits for N differential pair trimming. These are the ‘user’ values.

The user is able to switch from ‘factory’ values to ‘user’ trimmed values using the USERTRIM bit in the OPAMPx_CSR register. This bit is reset at startup and so the ‘factory’ value are applied by default to the OPAMP option registers.

User is liable to change the trimming values in calibration or in functional mode.

The offset trimming registers are typically configured after the calibration operation is initialized by setting bit CALON to 1. When CALON = 1 the inputs of the operational amplifier are disconnected from the functional environment.

- Setting CALSEL to 01 initializes the offset calibration for the P differential pair (low voltage reference used).
- Resetting CALSEL to 11 initializes the offset calibration for the N differential pair (high-voltage reference used).

When CALON = 1, the bit CALOUT reflects the influence of the trimming value selected by CALSEL and OPAHSM. The software should increment the TRIMOFFSETN bits in the OPAMP control register from 0x00 to the first value that causes the CALOUT bit to change from 1 to 0 in the OPAMP register. If the CALOUT bit is reset, the offset is calibrated correctly and the corresponding trimming value must be stored. The CALOUT flag needs up to 1 ms after the trimming value is changed to become steady (see $t_{OFFTRIM}$max delay specification in the electrical characteristics section of the datasheet).

Note: The closer the trimming value is to the optimum trimming value, the longer it takes to stabilize. The maximum stabilization time remains below 1 ms in any case.

Table 150. Operating modes and calibration

Mode	Control bits				Output	
	OPAEN	OPAHSM	CALON	CALSEL	V_{OUT}	CALOUT flag
Normal operating mode	1	0	0	X	analog	0
High-speed mode	1	1	0	X	analog	0
Power down	0	X	X	X	Z	0

Table 150. Operating modes and calibration (continued)

Mode	Control bits				Output	
	OPAEN	OPAHSM	CALON	CALSEL	V_{OUT}	CALOUT flag
Offset calibration N difference for normal mode	1	0	1	11	analog	X
Offset calibration P difference for normal mode	1	0	1	01	analog	X
Offset calibration N difference for high-speed mode	1	1	1	11	analog	X
Offset calibration P difference for high-speed mode	1	1	1	01	analog	X

Calibration procedure

Here are the steps to perform a full calibration of either one of the operational amplifiers:

1. Set the OPAEN bit in OPAMPx_CSR to 1 to enable the operational amplifier.
2. Set the USERTRIM bit in the OPAMPx_CSR register to 1.
3. Choose a calibration mode (refer to *Table 150: Operating modes and calibration*). The steps 3 to 4 must be repeated four times. For the first iteration, select the following:
   – Normal mode and N differential pair

   The above calibration mode corresponds to OPAHSM=0 and CALSEL=11 in the OPAMPx_CSR register.
4. Increment TRIMOFFSETN[4:0] in OPAMPx_OTR starting from 00000b until CALOUT changes to 0 in OPAMPx_CSR.

Note: Between the write to the OPAMPx_OTR register and the read of the CALOUT value, make sure to wait for the $t_{OFFTRIM}$max delay specified in the electrical characteristics section of the datasheet, to get the correct CALOUT value.

Repeat steps 3 to 4 for:

- Normal_mode and P differential pair, CALSEL=01
- High-speed mode and N differential pair
- High-speed mode and P differential pair

If a mode is not used, it is not necessary to perform the corresponding calibration.

Note: During the whole calibration phase the external connection of the operational amplifier output must not pull up or down currents higher than 500 µA.

23.4 OPAMP low-power modes

Table 151. Effect of low-power modes on the OPAMP

Mode	Description
Sleep	No effect.
Stop	No effect, OPAMP registers content is kept.
Standby	The OPAMP registers are powered down and must be reinitialized after exiting Standby.

23.5 OPAMP PGA gain

When OPAMP is configured as PGA mode, it can select the gain of x2, x4, x8, x16 for non-inverting mode and x-1, x-3, x-7, x-15 for inverting mode.

When OPAMP is configured as non-inverting mode, the gain error is specified in the product datasheet. When it is configured as inverting mode, the gain factor is defined not only the on chip feedback resistor but also the signal source output impedance. If the signal source output impedance is not negligible compare to the input feedback resistance of PGA, it creates the gain error. Please refer to the PGA resistance value in the product datasheet.

23.6 OPAMP registers

The registers of this peripheral can only be accessed by-word (32-bit).

23.6.1 OPAMP1 control/status register (OPAMP1_CSR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CAL OUT	TST REF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USER TRIM	PGA_GAIN[3:2]	
	r	rw											rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PGA_GAIN[1:0]		CALSEL[1:0]		CALON	Res.	Res.	OPA HSM	Res.	VM_SEL[1:0]		Res.	VP_SEL[1:0]		FORCE _VP	OPAEN
rw	rw	rw	rw	rw			rw		rw	rw		rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **CALOUT:** Operational amplifier calibration output
OPAMP output status flag. During the calibration mode, OPAMP is used as comparator.
0: Non-inverting < inverting
1: Non-inverting > inverting

Bit 29 **TSTREF:** OPAMP calibration reference voltage output control (reserved for test)
0: INTVREF of OPAMP is not output
1: INTVREF of OPAMP is output

Bits 28:19 Reserved, must be kept at reset value.

Bit 18 **USERTRIM:** User trimming enable

This bit allows to switch from ‘factory’ AOP offset trimmed values to ‘user’ AOP offset trimmed values

This bit is active for both mode normal and high-power.

0: ‘factory’ trim code used

1: ‘user’ trim code used

Bits 17:14 **PGA_GAIN[3:0]:** Programmable gain amplifier gain setting

0000: Non-inverting internal gain 2, VREF-referenced

0001: Non-inverting internal gain 4, VREF-referenced

0010: Non-inverting internal gain 8, VREF-referenced

0011: Non-inverting internal gain 16, VREF-referenced

0100: Non-inverting internal gain 2 with filtering on INM0, VREF-referenced

0101: Non-inverting internal gain 4 with filtering on INM0, VREF-referenced

0110: Non-inverting internal gain 8 with filtering on INM0, VREF-referenced

0111: Non-inverting internal gain 16 with filtering on INM0, VREF-referenced

1000: Inverting gain=-1/ Non-inverting gain =2 with INM0 node for input or bias

1001: Inverting gain=-3/ Non-inverting gain =4 with INM0 node for input or bias

1010: Inverting gain=-7/ Non-inverting gain =8 with INM0 node for input or bias

1011: Inverting gain=-15/ Non-inverting gain =16 with INM0 node for input or bias

1100: Inverting gain=-1/ Non-inverting gain =2 with INM0 node for input or bias, INM1 node for filtering

1101: Inverting gain=-3/ Non-inverting gain =4 with INM0 node for input or bias, INM1 node for filtering

1110: Inverting gain=-7/ Non-inverting gain =8 with INM0 node for input or bias, INM1 node for filtering

1111: Inverting gain=-15/ Non-inverting gain =16 with INM0 node for input or bias, INM1 node for filtering

Bits 13:12 **CALSEL[1:0]:** Calibration selection

It is used to select the offset calibration bus used to generate the internal reference voltage when CALON = 1 or FORCE_VP= 1.

00: 0.033*VDDA applied on OPAMP inputs

01: 0.1*VDDA applied on OPAMP inputs (for PMOS calibration)

10: 0.5*VDDA applied on OPAMP inputs

11: 0.9*VDDA applied on OPAMP inputs (for NMOS calibration)

Bit 11 **CALON:** Calibration mode enabled

0: Normal mode

1: Calibration mode (all switches opened by HW)

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **OPAHSM:** Operational amplifier high-speed mode

The operational amplifier must be disable to change this configuration.

0: operational amplifier in normal mode

1: operational amplifier in high-speed mode

Bit 7 Reserved, must be kept at reset value.

Bits 6:5 **VM_SEL[1:0]:** Inverting input selection
00:INM0 connected to OPAMP_VINM input
01: INM1 connected to OPAMP_VINM input
10:Feedback resistor is connected to the OPAMP_VINM input (PGA mode), Inverting input selection depends on the PGA_GAIN setting
11:opamp_out connected to OPAMP_VINM input (Follower mode)

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **VP_SEL[1:0]:** Non inverted input selection
00: GPIO INP0 connected to OPAMP_VINP
01: dac_out1 connected to OPAMP_VINP
10: GPIO INP2 is connected to OPAMP_VINP
11: Reserved

Bit 1 **FORCE_VP:** Force internal reference on VP (reserved for test)
0: Normal operating mode. Non-inverting input connected to inputs.
1: Calibration verification mode: Non-inverting input connected to calibration reference voltage.

Bit 0 **OPAEN:** Operational amplifier enable
0: operational amplifier disabled
1: operational amplifier enabled
Note: If OPAMP1 is unconnected in a specific package, it must remain disabled (keep OPAMP1_CSR register default value).

23.6.2 OPAMP1 trimming register in normal mode (OPAMP1_OTR)

Address offset: 0x04

Reset value: 0x0000 XXXX (factory trimmed values)

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

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **TRIMOFFSETP[4:0]:** Trim for PMOS differential pairs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **TRIMOFFSETN[4:0]:** Trim for NMOS differential pairs

23.6.3 OPAMP1 trimming register in high-speed mode (OPAMP1_HSOTR)

Address offset: 0x08

Reset value: 0x0000 XXXX (factory trimmed values)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TRIMHSOFFSETP[4:0]					Res.	Res.	Res.	TRIMHSOFFSETN[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **TRIMHSOFFSETP[4:0]:** High-speed mode trim for PMOS differential pairs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **TRIMHSOFFSETN[4:0]:** High-speed mode trim for NMOS differential pairs

23.6.4 OPAMP register map

Table 152. OPAMP 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	**OPAMP1_CSR**	Res.	CALOUT	TSTREF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USERTRIM	PGA_GAIN				CALSEL		CALON	Res.	Res.	OPAHSM	Res.	VM_SEL		Res.	VP_SEL		FORCE_VP	OPAEN
	Reset value		0	0											0	0	0	0	0	0	0	0			0		0	0		0	0	0	0
0x04	**OPAMP1_OTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM OFFSETP[4:0]					Res.	Res.	Res.	TRIM OFFSETN[4:0]				
	Reset value																				(1)								(1)				
0x08	**OPAMP1_ HSOTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIMHS OFFSETP[4:0]					Res.	Res.	Res.	TRIMHS OFFSETN[4:0]				
	Reset value																				(1)								(1)				

1. Factory trimmed values.

Refer to *Section 2.2 on page 70* for the register boundary addresses.

24 True random number generator (RNG)

24.1 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 a 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 SP800-90B. It can also be tested using the German BSI statistical tests of AIS-31 (T0 to T8).

24.2 RNG main features

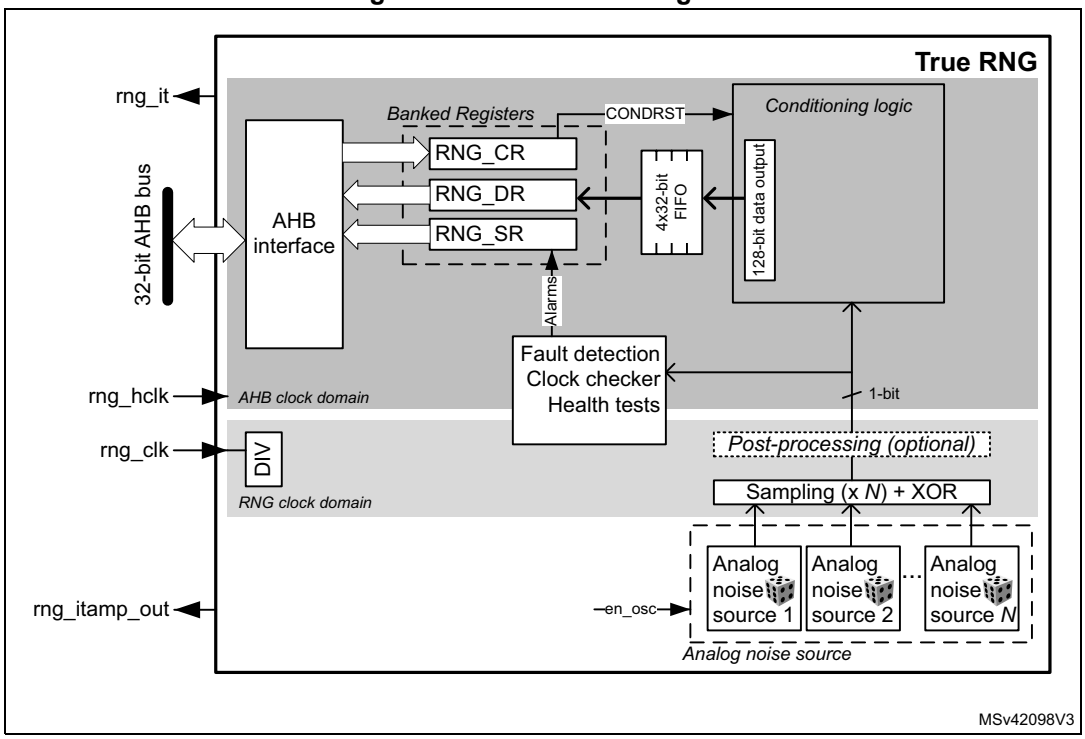
- The RNG delivers 32-bit true random numbers, produced by an analog entropy source conditioned by a NIST SP800-90B approved conditioning stage.
- It can be used as the entropy source to construct a nondeterministic random bit generator (NDRBG).
- In the NIST configuration, it produces four 32-bit random samples every 412 AHB clock cycles if $f_{AHB} < f_{threshold}$ (256 RNG clock cycles otherwise).
- It embeds startup and NIST SP800-90B approved continuous health tests (repetition count and adaptive proportion tests), associated with specific error management
- It can be disabled to reduce power consumption, or enabled with an automatic low power mode (default configuration).
- It 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).

24.3 RNG functional description

24.3.1 RNG block diagram

Figure 155 shows the RNG block diagram.

Figure 155. RNG block diagram



24.3.2 RNG internal signals

Table 153 describes a list of useful-to-know internal signals available at the RNG level, not at the STM32 product level (on pads).

Table 153. RNG internal input/output signals

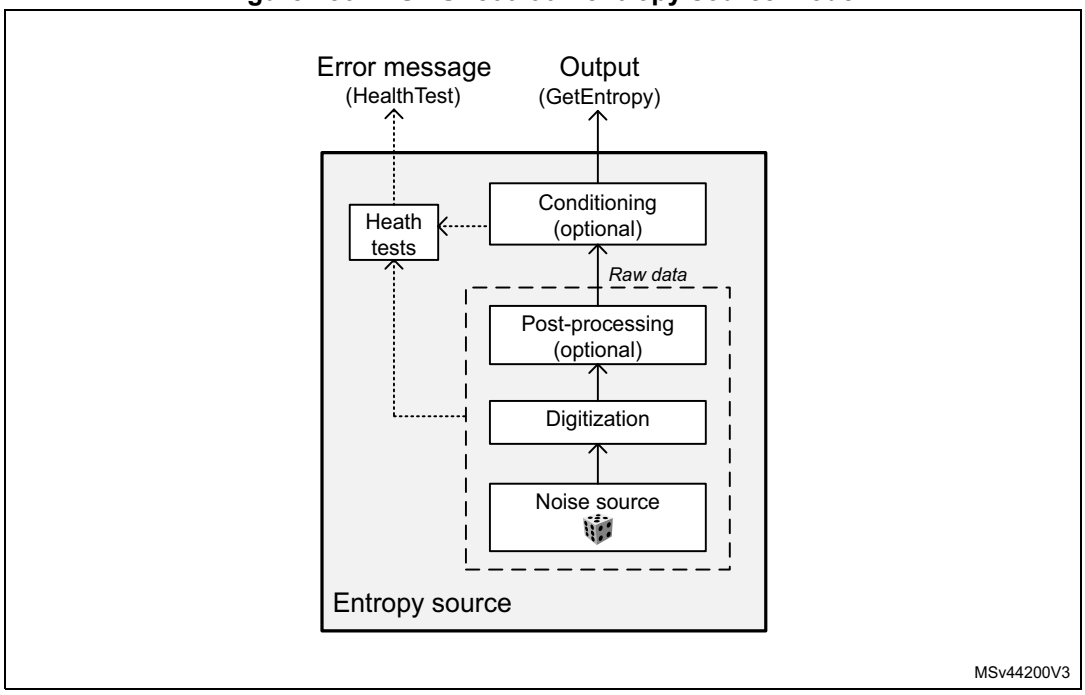
Signal name	Signal type	Description
rng_it	Digital output	RNG global interrupt request
rng_hclk	Digital input	AHB clock
rng_clk	Digital input	RNG dedicated clock, asynchronous to rng_hclk
rng_itamp_out	digital output	RNG internal tamper event signal to TAMP (XORed), triggered when an unexpected hardware fault occurs. When this signal is triggered, RNG stops delivering random samples, requiring a reset and a new initialization to be usable again.

24.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals.

Within its boundary RNG integrates all the required NIST components depicted on *Figure 156*. Those components are an analog noise source, a digitization stage, a conditioning algorithm, a health monitoring block and two interfaces that are used to interact with the entropy source: GetEntropy and HealthTest.

Figure 156. NIST SP800-90B entropy source model



The components pictured above are detailed hereafter.

Noise source

The noise source is the component that contains the non-deterministic, entropy-providing activity that is ultimately responsible for the uncertainty associated with the bitstring output by the entropy source. This noise source provides 1-bit samples. It is composed of:

- Multiple analog noise sources (x6), each based on three XORed free-running ring oscillator outputs. It is possible to disable those analog oscillators to save power, as described in *Section 24.3.8*. Multiple oscillators are also disabled for the configuration A (see *Table 156: RNG configurations*).
- The XORing of all the noise sources into a single analog output.
- A sampling stage of this output clocked by a dedicated clock input (rng_clk with integrated divider), delivering a 1-bit raw data output.

This noise source sampling is independent to the AHB interface clock frequency (rng_hclk), with a possibility for the software to decrease the sampling frequency by using the integrated divider.

Note: In Section 24.6 the recommended RNG clock frequencies and associated divider value are given.

Post processing

In the NIST configuration no post-processing is applied to the sampled noise source. In non-NIST configuration B (as defined in *Section 24.6.2*) a normalization debiasing is applied,

that is half of the bits are taken from the sampled noise source, half of the bits are taken 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 bitstrings output (128-bit). The NIST SP800-90B target is full entropy on the output (128-bit).

The times required between two random number generations, and between the RNG initialization and availability of first sample are described in *Section 24.5*.

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 24.5*.

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 interrupts are enabled an interrupt is generated when this data ready flag transitions from 0 to 1. 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 assurance that failures are caught quickly and with a high probability and reliability.

The RNG implements the following health check features in accordance with NIST SP800-90B. The described thresholds correspond to the value recommended for register RNG_HTCR (configuration A in *Section 24.6.2*).

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 42 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 711 times.
   - Known-answer tests, to verify the conditioning stage.
2. Continuous health tests, running indefinitely on the outputs of the noise source:
   - 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 tests
   - Transition count test, flagging an error when the noise source has delivered more than 32 consecutive occurrences of 2-bit patterns (01 or 10).
   - Real-time “too slow” sampling clock detector, flagging an error when one RNG clock cycle (before divider) is smaller than 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 24.3.4: RNG initialization*). 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 24.3.7: Error management*.

Note: An interrupt can be generated when an error is detected.

Above the health test thresholds are modified by changing the value in the RNG_HTCR register. See Section 24.6: RNG entropy source validation for details.

24.3.4 RNG initialization

The RNG simplified state machine is pictured on *Figure 157*.

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 cycles of 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 startup behavior test on the raw data samples and known-answer tests.
3. When startup health tests are completed. During this time, three 128-bit noise source samples are used.
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 24.5: RNG processing time*.

Figure 157. RNG initialization overview

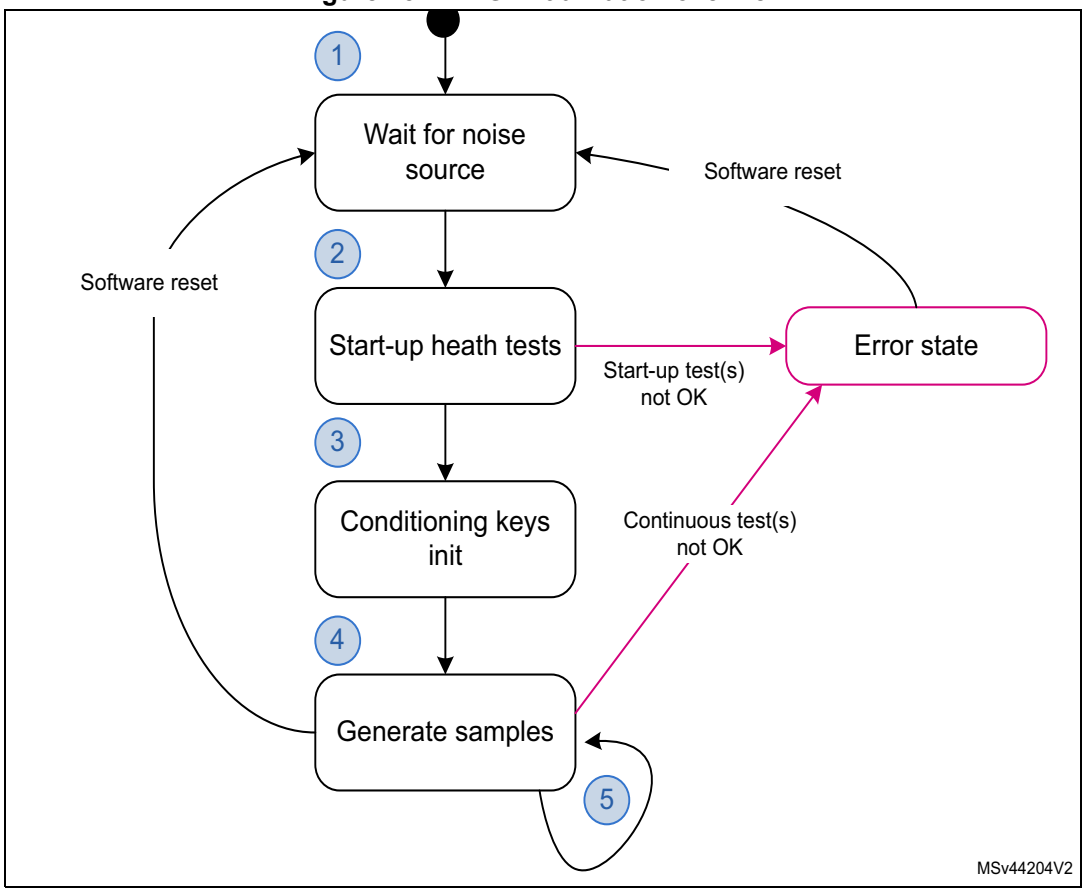


Figure 157 also highlights a possible software reset sequence, implemented by:

1. Writing bits RNGEN = 0 and CONDRST = 1 in the RNG_CR register with the same RNG configuration and a new CLKDIV if needed.
2. Then writing RNGEN = 1 and CONDRST = 0 in the RNG_CR register.
3. Wait for random number to be ready, after initialization completes.

Note: When the RNG peripheral is reset through RCC (hardware reset), the RNG configuration for optimal randomness is lost in the RNG registers. Software reset with CONFIGLOCK set preserves the RNG configuration.

24.3.5 RNG operation

Normal operations

To run the RNG using interrupts, the following steps are recommended:

1. Consult *Section 24.6: RNG entropy source validation* and verify if a specific RNG configuration is required for the application.
   – If it is the case, write in the RNG_CR register the bit CONDRST = 1 together with the correct RNG configuration. Then perform a second write to the RNG_CR

register with the bit CONDRST = 0, the interrupt enable bit IE = 1 and the RNG enable bit RNGEN = 1.

– If it is not the case perform a write to the RNG_CR register with the interrupt enable bit IE = 1 and the 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 is 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 24.3.7* must be used.

To run the RNG in polling mode following steps are recommended:

1. Consult *Section 24.6: RNG entropy source validation* and verify if a specific RNG configuration is required for the application.
   – If it is the case write in the RNG_CR register the bit CONDRST = 1 together with the correction RNG configuration. Then perform a second write to the RNG_CR register with the bit CONDRST = 0 and the RNG enable bit RNGEN = 1.
   – If it 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 above conditions are true read the content of the RNG_DR register up to four consecutive times. If valid data is 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 24.3.7* must be used.

Note: When data is not ready (DRDY = 0) RNG_DR returns zero.
It is recommended to always verify that RNG_DR is different from zero. Because when it is the case a seed error occurred between RNG_SR polling and RND_DR output reading (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 RNG configuration by using the RNG configuration “B”, described in *Section 24.6: RNG entropy source validation*. The gain in random number generation speed is summarized in *Section 24.5: RNG processing time*.

Low-power operations

If the power consumption is a concern to the application, low-power strategies can be used, as described in *Section 24.3.8: RNG low-power use*.

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 24.3.3: Random number generation*.

24.3.6 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 24.6: RNG entropy source validation*.

*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 always flags a clock error (CECS = 1 in the RNG_SR register).*

See *Section 24.3.1: RNG block diagram* for details (AHB and RNG clock domains).

24.3.7 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 24.3.6: RNG clocking*) and then it must clear the CEIS bit interrupt flag. The CECS bit is automatically cleared when the clocking condition is normal.

Note: The clock error has no impact on generated random numbers that is the application can still read the RNG_DR register.

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. If a value is available in the RNG_DR register, it must not be used as it may not have enough entropy.

The following sequence must be used to fully recover from a seed error:

1. Software reset by writing CONDRST at 1 and at 0 (see bitfield description for details). This step is needed only if SECS is set. Indeed, when SEIS is set and SECS is cleared

it means RNG performed the reset automatically (auto-reset). In this case application must clear the SEIS bit interrupt flag.

2. If SECS was set in step 1 (no auto-reset) wait for CONDRST to be cleared in the RNG_CR register, then confirm that SEIS is cleared in the RNG_SR register. Otherwise, just clear the SEIS bit in the RNG_SR register.
3. If SECS was set in step 1 (no auto-reset), wait for SECS to be cleared by RNG. The random number generation is now back to normal.

Note: After a seed error RNG restarts generating random numbers when SECS is cleared.

When the application sets the ARDIS bit in the RNG_CR register, the auto-reset is disabled. CONDRST must be used in step 1.

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 either using the central reset management or the global SoC reset. Then a proper initialization of the RNG is required, again.

24.3.8 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 RNGEN = 0 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 reenabled 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 24.5: RNG processing time on page 698*.

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 RNGEN bit is set to 1, unless the application resets the conditioning logic using CONDRST bit in the RNG_CR register.

When the application wants to disable the RNG clock it is recommended to wait two RNG kernel clock cycles between clearing the RNGEN bit and disabling the RNG kernel clock using the RCC.

Also, when application needs to enter a power mode where RNG is de-activated, it is recommended to wait two RNG kernel clock cycles between clearing the RNGEN bit and entering the low power mode using the PWR.

In the two cases above, to avoid unexpected consumption when RNG analog oscillators stay active, application can set the bit 13 in RNG_CR register. Setting this bit adds some marginal power consumption while RNGEN bit is set (RNG activated).

Note: The power modes where RNG is deactivated (that is retained or not available) can be found in the PWR section.

24.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see *Section 24.3.7: Error management*
- Clock error, see *Section 24.3.7: Error management*

Dedicated interrupt enable control bits are available as shown in *Table 154*.

Table 154. 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 24.3.7: Error management*)
	Clock error flag	CEIS	IE	Write 0 to CEIS

The user can enable or disable the above interrupt sources individually by changing the mask bits or 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.

24.5 RNG processing time

In recommended configuration A or C described in *Table 156*, the time between two sets of four 32-bit data is either:

- 203 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).

With $f_{threshold} = 1.6 \times f_{RNG}$, for instance 77 MHz if f_{RNG}= 48 MHz. Value N is defined in *Section 24.6: RNG entropy source validation*.

Note: When CLKDIV is different from zero, f_{RNG} must take into account the internal divider ratio.

If configuration B is selected the performance figures become:

- 203 AHB cycles if $f_{AHB} < f_{threshold}$ or
- 32 RNG cycles $f_{AHB} \geq f_{threshold}$

with $f_{threshold} = 6.5 \times f_{RNG}$.

Initialization time

More time is needed for the first set of random numbers after the device exits reset (see Section 1.3.4: RNG initialization). Table below gives details on how to compute the time spent in each initialization step.

Table 155. RNG initialization times

Initialization step	Configuration A or C, reset value	Configuraton B
Wait for noise source, then startup health tests	Max((wait_noise(1) + 11 + 1024 x CLKDIV) RNG_cycles, 13 x 203 AHB_cycles)	Max(16 + 1035 RNG_cycles, 13 x 203 AHB_cycles)
Conditioning keys initialization	Max((1+2xN) x 128 x CLKDIV) RNG_cycles + 203 AHB_cycles, (128 x CLKDIV) RNG_cycles + (2xN+2) x 203 AHB_cycles)	Max (3 x 32 RNG_cycles + 203 AHB_cycles, 32 RNG_cycles + 4 x 203 AHB_cycles)

1. 192 RNG_cycles (configuration A or C), 16 RNG_cycles (reset value)

As an example, if AHB clock= 250 MHz and RNG clock= 48 MHz then the initialization time is around 40 µs in the configuration C (CLKDIV=1, N=2) and 26 µs in the configuration B.

24.6 RNG entropy source validation

24.6.1 Introduction

In order to assess the amount of entropy available from the RNG, STMicroelectronics has tested the peripheral using the German BSI AIS-31 statistical tests (T0 to T8), and NIST SP800-90B test suite.

24.6.2 Validation conditions

STMicroelectronics has tested the RNG true random number generator in the following conditions:

- RNG clock rng_clk = 48 MHz
- RNG configurations are described in *Table 156*. Note that only configuration A can be certified NIST SP800-90B. Refer to *Table 157* to select the best configuration for the application.
- Configuration C can be used when configuration A is not flagged as certified.

Table 156. RNG configurations

Configuration	RNG_CR bits						Loop number (N)	RNG_ HTCR register	RNG_ NCSR register
	NISTC bit	RNG_ CONFIG1 [5:0]	CLKDIV [3:0]	RNG_ CONFIG2 [2:0](1)	RNG_ CONFIG3 [3:0](2)	CED bit			
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*.								

Table 156. RNG configurations (continued)

Configuration	RNG_CR bits						Loop number (N)	RNG_ HTCR register	RNG_ NCSR register
	NISTC bit	RNG_ CONFIG1 [5:0]	CLKDIV [3:0]	RNG_ CONFIG2 [2:0](1)	RNG_ CONFIG3 [3:0](2)	CED bit			
B	1	0x18	0x0(3)	0x0	0x0	0	1	0x0000 AAC7(4)	default
C	0	0x0F		0x0	0xD	0	2		default

1. 0x1 value is recommended instead of 0x0 for RNG_CONFIG2[2:0], when RNG power consumption is critical. See the end of *Section 24.3.8: RNG low-power use* for details.
2. RNG_CONFIG3[1:0] defines the loop number N: 0x0 corresponds to N=1, 0x1 to N=2, 0x2 to N=3, 0x3 to N=4
3. The noise source sampling must be 48 MHz or less. Hence, if the RNG clock is different from 48 MHz, this value of CLKDIV must be adapted. See the CLKDIV bitfield description in *Section 24.7.1* for details.
4. This value can be fixed in the RNG driver (it doesn't depend on the STM32 family).

Table 157. Configuration selection

Section criteria	Config A	Config B	Config C
Suitable to generate NIST compliant cryptographic keys	Yes	No	
Entropy(1)	Certified	Good	Very good
Speed(2)	Baseline	Faster	Baseline

1. For configurations B and C entropy is verified using AIS-31 test suite (T0 to T8).
2. When speed is not enough for application a NIST compliant DRBG can be used to increase throughput.

For details on data collection and the running of statistical test suites refer to "STM32 microcontrollers random number generation validation using NIST statistical test suite" application note (AN4230) available from *www.st.com*.

24.7 RNG registers

The RNG is associated with a control register, a data register and a status register.

24.7.1 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.	Res.	Res.	RNG_CONFIG1[5:0]						CLKDIV[3:0]			
rs	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 Config lock

0: Writes to the RNG_NSCR, RNG_HTCR and RNG_CR configuration bits [29:4] are allowed.

1: Writes to the RNG_NSCR, RNG_HTCR and 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_CR, RNG_NSCR and RNG_HTCR are not changed by CONDRST.

This bit must be set to 1 in the same access that set 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.

When CONDRST is set to 0 by the software, its value goes to 0 when the reset process is done. It takes about 2 AHB clock cycles + 2 RNG clock cycles.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:20 **RNG_CONFIG1[5:0]:** RNG configuration 1

Reserved to the RNG configuration (bitfield 1). Must be initialized using the recommended value documented in *Section 24.6: RNG entropy source validation*.

Writing any bit of RNG_CONFIG1 is taken into account only if the CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

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 these bits is taken into account only if the CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bits 15:13 **RNG_CONFIG2[2:0]:** RNG configuration 2

Reserved to the RNG configuration (bitfield 2). Bit 13 can be set when RNG power consumption is critical. See *Section 24.3.8: RNG low-power use*. Refer to the RNG_CONFIG1 bitfield for details.

Bit 12 **NISTC**: NIST custom

0: Hardware default values for NIST compliant RNG. In this configuration per 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 24.6: RNG entropy source validation* for recommended configuration.

Writing this bit is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

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 enabled

1: Auto-reset disabled

Keeping the auto-reset enabled (automatic clearance of the SECS bit) simplifies the management of noise source errors, as described in *Section 24.3.7: Error management*.

Writing this bit is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CED:** Clock error detection

0: Clock error detection 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 this bit is taken into account only if the CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

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 is set in the RNG_SR register.

Bit 2 **RNGEN:** True random number generator enable

0: True random number generator is disabled. Analog noise sources are powered off and logic clocked by the RNG clock is gated.

1: True random number generator is enabled.

Bits 1:0 Reserved, must be kept at reset value.

24.7.2 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	Res.	Res.	SECS	CECS	DRDY
									rc_w0	rc_w0			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.

Bits 4:3 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 sequences has been detected:

- Runtime repetition count test failed (noise source has provided more than 24 consecutive bits at a constant value 0 or 1, or more than 32 consecutive occurrence of two bits patterns 01 or 10)
- Startup or continuous adaptive proportion test on noise source failed.
- Startup post-processing/conditioning sanity check failed.

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 ($f_{RNGCLK} < f_{HCLK}/32$).

Note: CECS bit is valid only if the CED bit in the RNG_CR register is set to 0.

Bit 0 **DRDY:** Data ready

0: The RNG_DR register is not yet valid, no random data is available.

1: The RNG_DR register contains valid random data.

Once the output buffer becomes empty (after reading the RNG_DR register), this bit returns to 0 until a new random value is generated.

Note: The DRDY bit can rise when the peripheral is disabled (RNGEN = 0 in the RNG_CR register).

If IE=1 in the RNG_CR register, an interrupt is generated when DRDY = 1.

24.7.3 RNG data register (RNG_DR)

Address offset: 0x008

Reset value: 0x0000 0000

The RNG_DR 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, which are valid when DRDY = 1. When DRDY = 0, the RNDATA value is zero.
When DRDY is set, it is recommended to always verify that RNG_DR is different from zero. The zero value means that a seed error occurred between RNG_SR polling and RND_DR output reading (a rare event).

24.7.4 RNG noise source control register (RNG_NSCR)

Address offset: 0x00C

Reset value: 0x0003 FFFF

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.	EN_OSC6[2:1]	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN_OSC6[0]	EN_OSC5[2:0]			EN_OSC4[2:0]			EN_OSC3[2:0]			EN_OSC2[2:0]			EN_OSC1[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 17:15 **EN_OSC6[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 6. The bitfield has no effect otherwise.

Bits 14:12 **EN_OSC5[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 5. The bitfield has no effect otherwise.

Bits 11:9 **EN_OSC4[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 4. The bitfield has no effect otherwise.

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.

24.7.5 RNG health test control register (RNG_HTCR)

Address offset: 0x010

Reset value: 0x0000 72AC

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. See *Section 24.6: RNG entropy source validation* for the recommended value.

Note: The RNG behavior, including the read to this register, is not guaranteed if a different value from the recommended value is written.

24.7.6 RNG register map

Table 158. 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.	Res.	Res.	RNG_CONFIG1[5:0]						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	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	Res.	Res.	SECS	CECS	DRDY
	Reset value																										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.	EN_OSC6[2:0]			EN_OSC5[2:0]			EN_OSC4[2:0]			EN_OSC3[2:0]			EN_OSC2[2:0]			EN_OSC1[2:0]		
	Reset value															1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x010	RNG_HTCR	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

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

25 Hash processor (HASH)

25.1 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, corresponding to a 64-byte block size (16 x 32 bits)
- Automatic padding to complete the input bit string to fit digest minimum block size
- AHB slave peripheral, accessible by 32-bit words only (else an AHB error is generated)
- 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

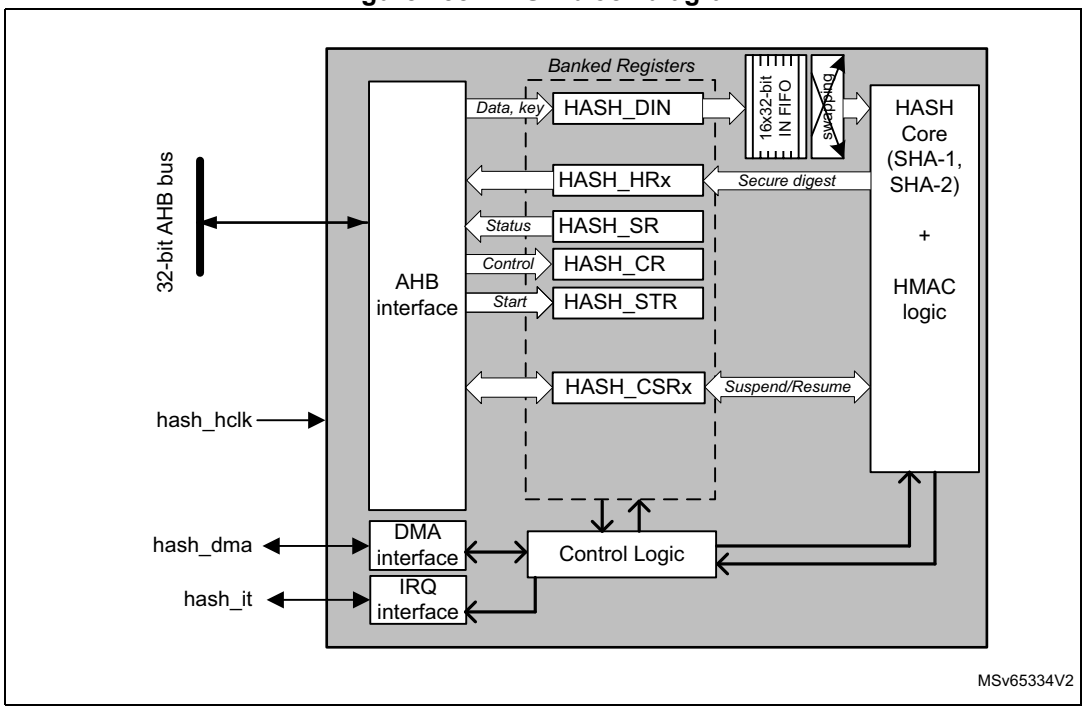
The devices have a single instance of HASH peripheral.

25.4 HASH functional description

25.4.1 HASH block diagram

Figure 158 shows the block diagram of the hash processor.

Figure 158. HASH block diagram



25.4.2 HASH internal signals

Table 159 describes a list of useful to know internal signals available at HASH level, not at product level (on pads).

Table 159. 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 160*).

Table 160. 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**.

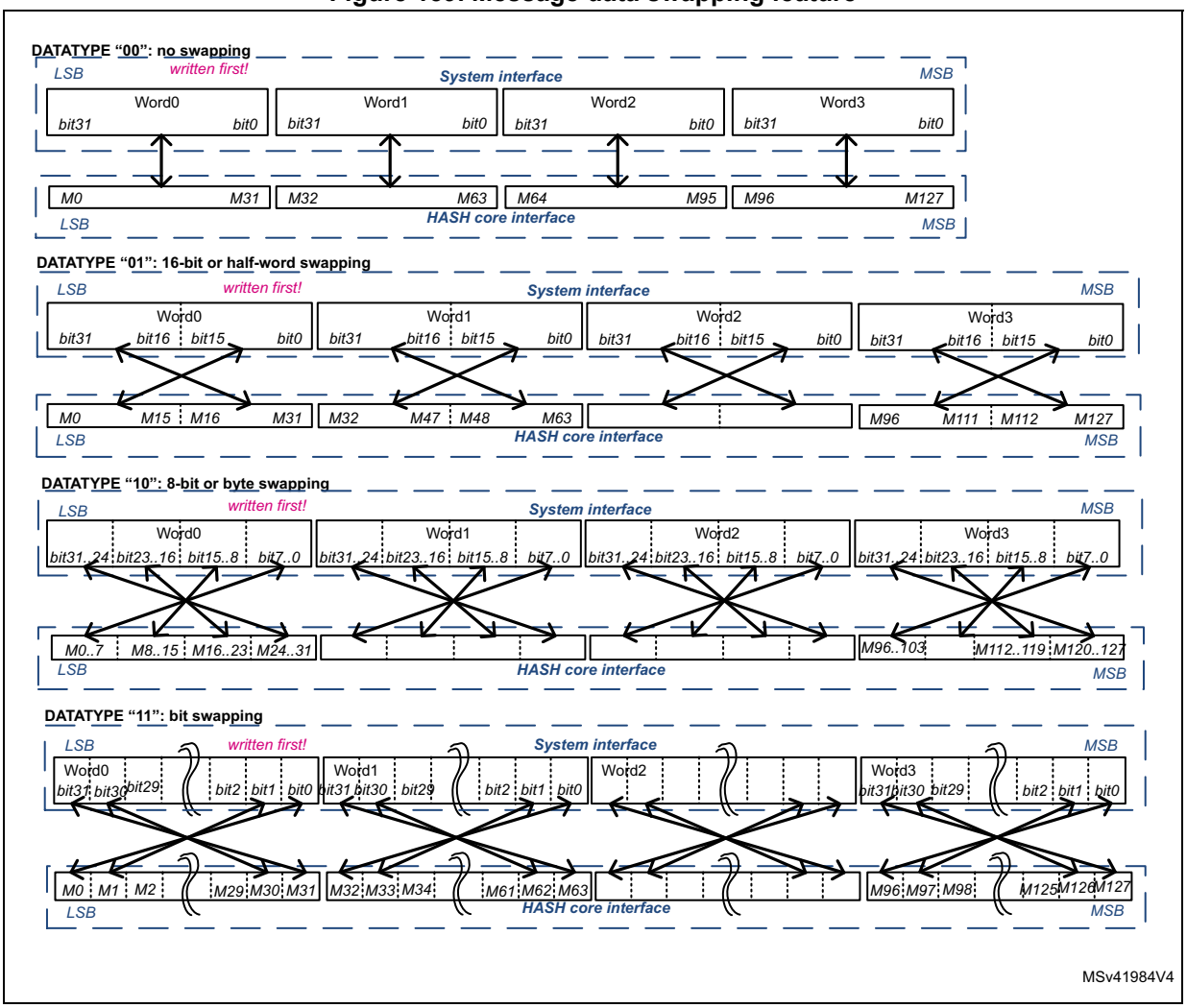
Data are entered into the HASH one 32-bit word at a time, by writing them into the HASH_DIN register. The current contents of the HASH_DIN register are transferred to the 16 words input FIFO each time the register is written with new data. Hence, HASH_DIN and the FIFO form a seventeen 32-bit words length 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 159* 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 159. 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 160: 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 161*. When set, the DMAEN bit is automatically cleared by hardware.

Table 161. 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*.

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:

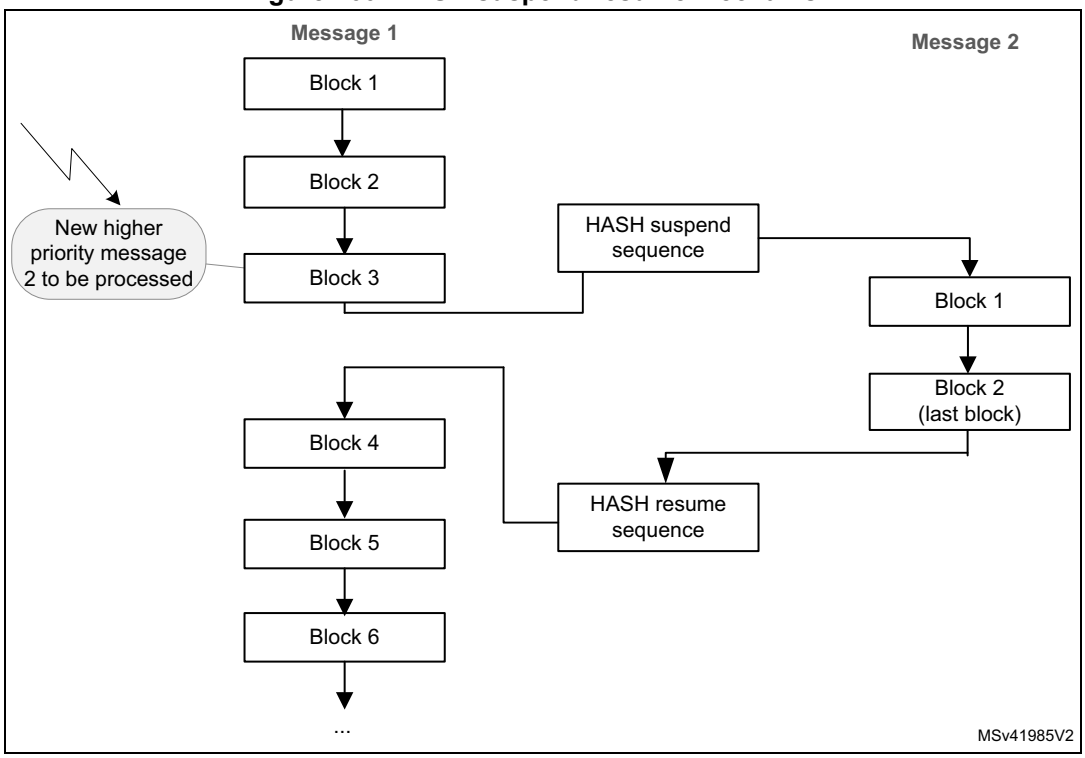
```
HASH_HR0 = 0x5FD596EE
HASH_HR1 = 0x78D5553C
HASH_HR2 = 0x8FF4E72D
HASH_HR3 = 0x266DFD19
HASH_HR4 = 0x2366DA29
```

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 160*.

Figure 160. 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 160*).

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 162 summarizes the time required to process an intermediate block for each mode of operation.

Table 162. 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 163* gives a summary of the available features.

Table 163. 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	rw		

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 bit 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

Setting this bit resets the hash processor core, so that the HASH is ready to compute the message digest of a new message.
Clearing this bit has no effect. Reading this bit always returns 0.

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]				
							rw				rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **DCAL:** Digest calculation

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.

Reading this bit returns zero.

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]

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 164. HASH1 register map and reset values

Offset	Register name / Reset value	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																																	

Table 164. HASH1 register map and reset values (continued)

Offset	Register name Reset value	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
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
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 4) Last address: 0x320	**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
0x310 + 0x4 * x, (x = 5 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 2.2* for the register boundary addresses.

26 Advanced-control timers (TIM1)

In this section, “TIMx” should be understood as “TIM1” since there is only one instance of this type of timer for the products to which this reference manual applies.

26.1 TIM1 introduction

The advanced-control timer (TIM1) consists of a 16-bit autoreload counter driven by a programmable prescaler.

It 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 advanced-control (TIM1) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 26.3.30: Timer synchronization*.

26.2 TIM1 main features

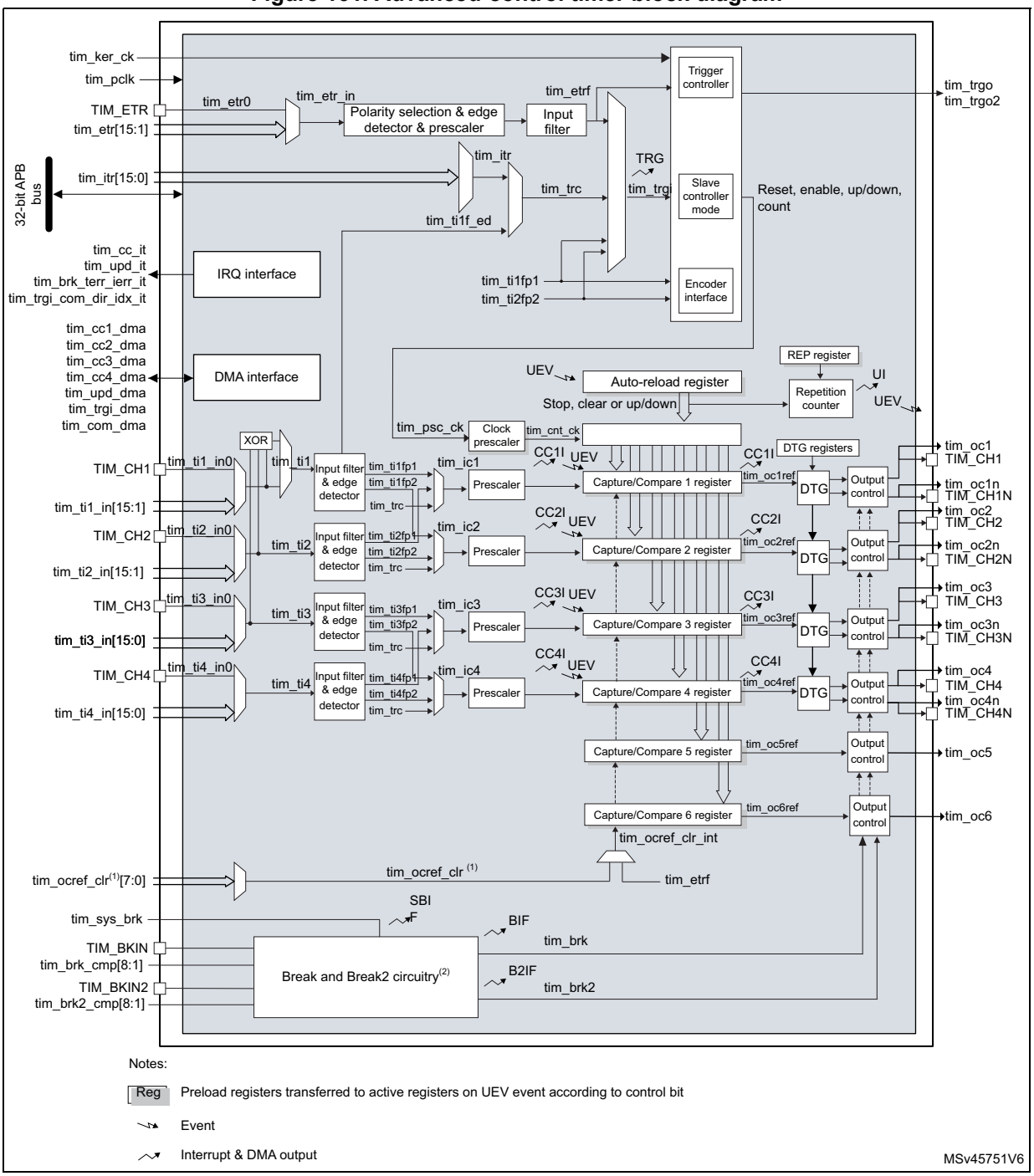
TIM1 timer features include:

- 16-bit up, down, up/down autoreload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency by any factor from 1 to 65536.
- Up to six independent channels for:
  - Input capture (but channels 5 and 6)
  - Output compare
  - PWM generation (edge and center-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- 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

26.3 TIM1 functional description

26.3.1 Block diagram

Figure 161. Advanced-control timer block diagram



1. This feature is not available on all timers, refer to *Section 26.3.2: TIM1 pins and internal signals*.
2. See *Figure 208: Break and Break2 circuitry overview* for details.

26.3.2 TIM1 pins and internal signals

The tables in this section summarize the TIM inputs and outputs

Table 165. 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 can 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_CH1N TIM_CH2N TIM_CH3N TIM_CH4N	Output	Timer complementary outputs, derived from TIM_CHx outputs with the possibility to have deadtime insertion.
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.
TIM_BKIN TIM_BKIN2	Input/output	Break and Break2 inputs. These inputs can also be configured in bidirectional mode.

Table 166. 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 pulsewidth 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/tim_trgo2	Output	Internal trigger outputs. These triggers are used by other timers and /or other peripherals.

Table 166. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
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_brk2_cmp[8:1]	Input	Break2 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
tim_cc_it	Output	Timer capture/compare interrupt
tim_upd_it	Output	Timer update event interrupt
tim_brk_terr_ierr_it	Output	Timer break, break2, transition error and index error interrupt
tim_trgi_com_dir_idx_it	Output	Timer trigger, commutation, direction and index interrupt
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture / compare 1..4 dma requests
tim_upd_dma	Output	Timer update dma request
tim_trgi_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

Table 167, *Table 168*, *Table 169* and *Table 170* list the sources connected to the tim_ti[4:1] input multiplexers.

Table 167. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources
	TIM1
tim_ti1_in0	TIM1_CH1
tim_ti1_in1	COMP1_OUT
tim_ti1_in[15:2]	Reserved

Table 168. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources
	TIM1
tim_ti2_in0	TIM1_CH2
tim_ti2_in[15:1]	Reserved

Table 169. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources
	TIM1
tim_ti3_in0	TIM1_CH3
tim_ti2_in[15:1]	Reserved

Table 170. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources
	TIM1
tim_ti4_in0	TIM1_CH4
tim_ti4_in[15:1]	Reserved

Table 171 lists the internal sources connected to the tim_itr input multiplexer.

Table 171. Internal trigger connection

Timer internal trigger input signal	TIM1
tim_itr0	Reserved
tim_itr1	tim2_trgo
tim_itr2	tim3_trgo
tim_itr[15:3]	Reserved

Table 172 lists the internal sources connected to the tim_etr input multiplexer.

Table 172. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signals assignment
	TIM1
tim_etr0	TIM1_ETR
tim_etr1	COMP1_OUT
tim_etr2	Reserved
tim_etr3	adc1_awd1
tim_etr4	adc1_awd2
tim_etr5	adc1_awd3
tim_etr[15:6]	Reserved

Table 173, *Table 174* and *Table 175* list the sources connected to the tim_brk and tim_brk2inputs.

Table 173. Timer break interconnect

tim_brk inputs	TIM1
TIM_BKIN	TIM1_BKIN pin
tim_brk_cmp1	COMP1_OUT
tim_brk_cmp[8:2]	Reserved

Table 174. Timer break2 interconnect

tim_brk2 inputs	TIM1
TIM_BKIN2	TIM1_BKIN2 pin
tim_brk2_cmp1	COMP1_OUT
tim_brk2_cmp[8:2]	Reserved

Table 175. System break interconnect

tim_sys_brk inputs	TIM1	Enable bit in SBS_CFGR2 register
tim_sys_brk0	Flash memory double ECC error	ECCL
tim_sys_brk1	Programmable Voltage Detector (PVD)	PVDL
tim_sys_brk2	SRAM double ECC error	SEL
tim_sys_brk3	Cortex-M33 LOCKUP	CLL
CSS	Clock Security System	None (always enabled)

26.3.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-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, 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 are 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 downcounting) 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: The counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler divides the counter clock frequency by any factor from 1 to 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.

Figure 162 and *Figure 163* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 162. Counter timing diagram with prescaler division change from 1 to 2

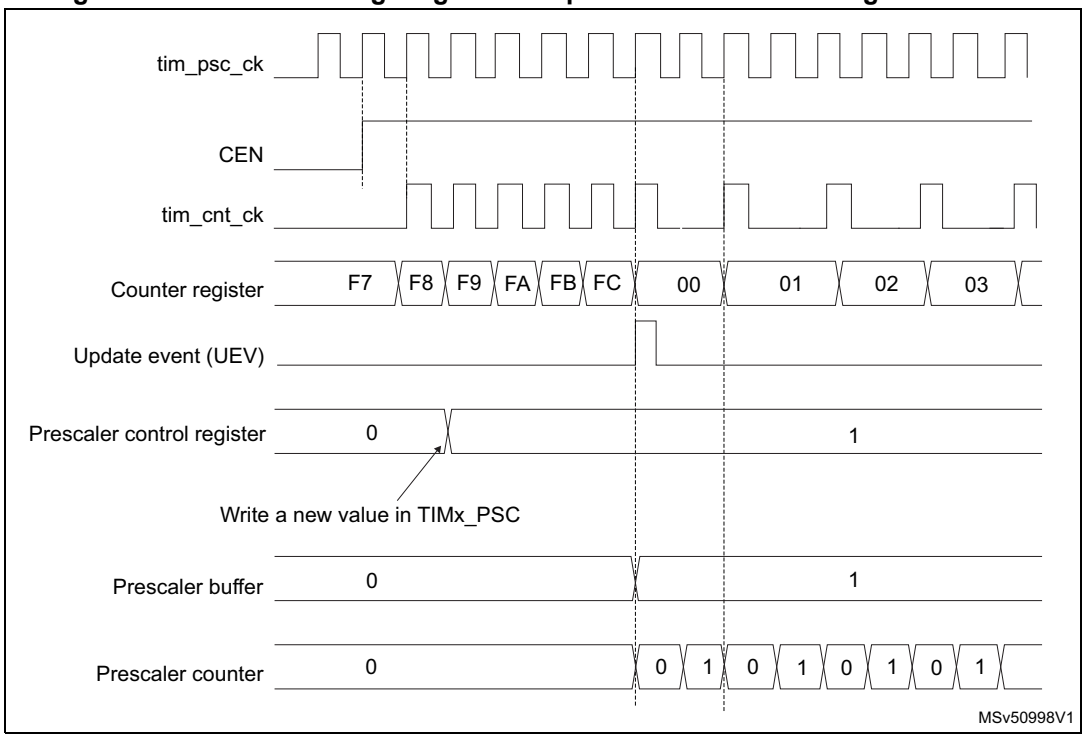
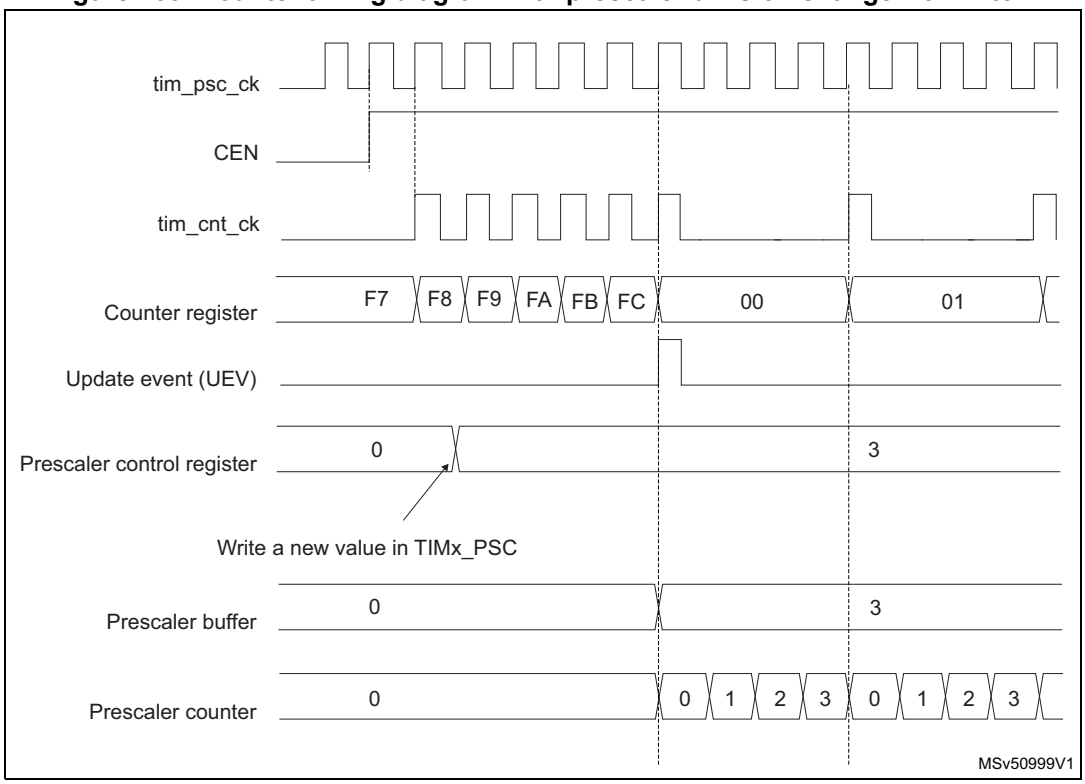


Figure 163. Counter timing diagram with prescaler division change from 1 to 4



26.3.4 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) + 1. 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 164. Counter timing diagram, internal clock divided by 1

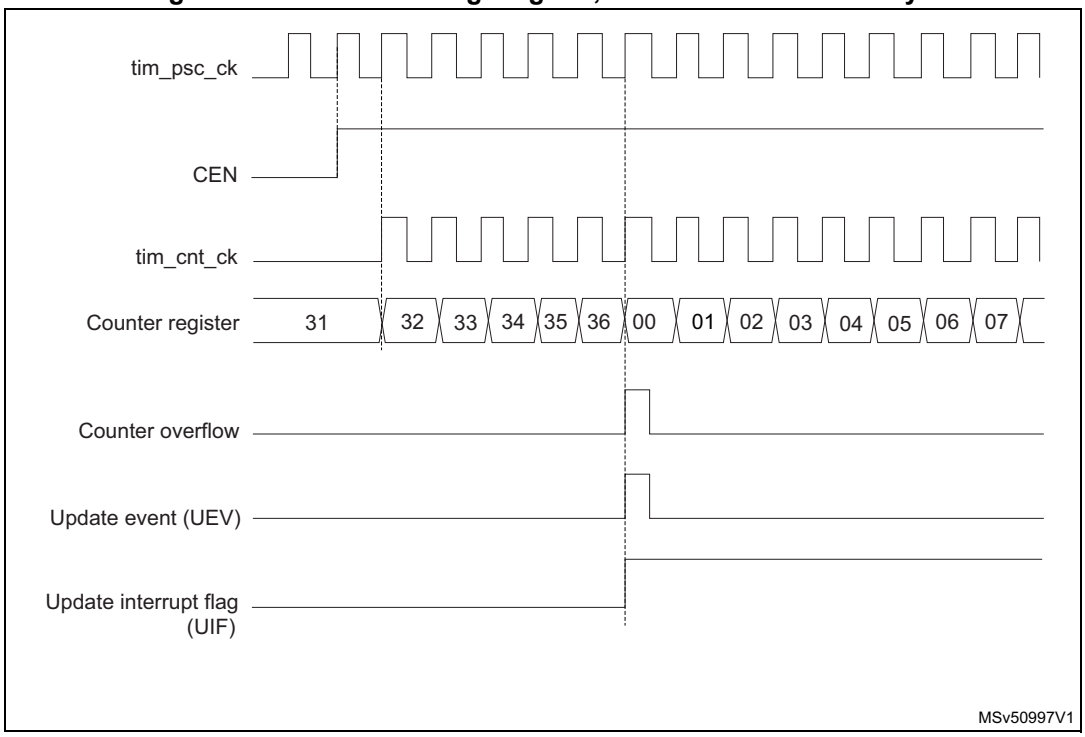


Figure 165. Counter timing diagram, internal clock divided by 2

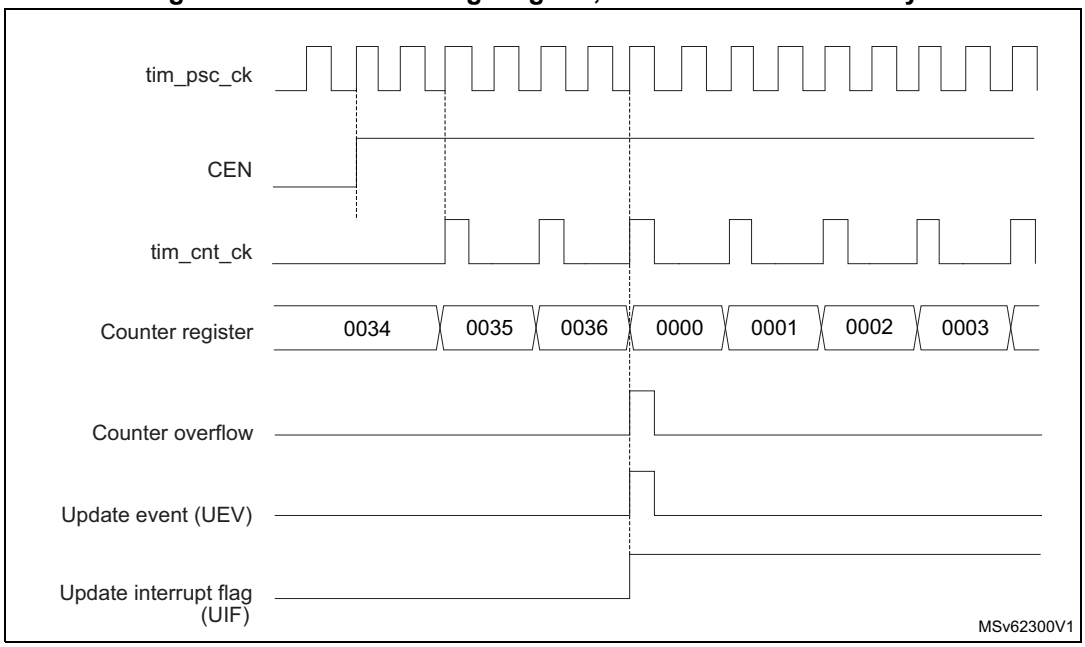


Figure 166. Counter timing diagram, internal clock divided by 4

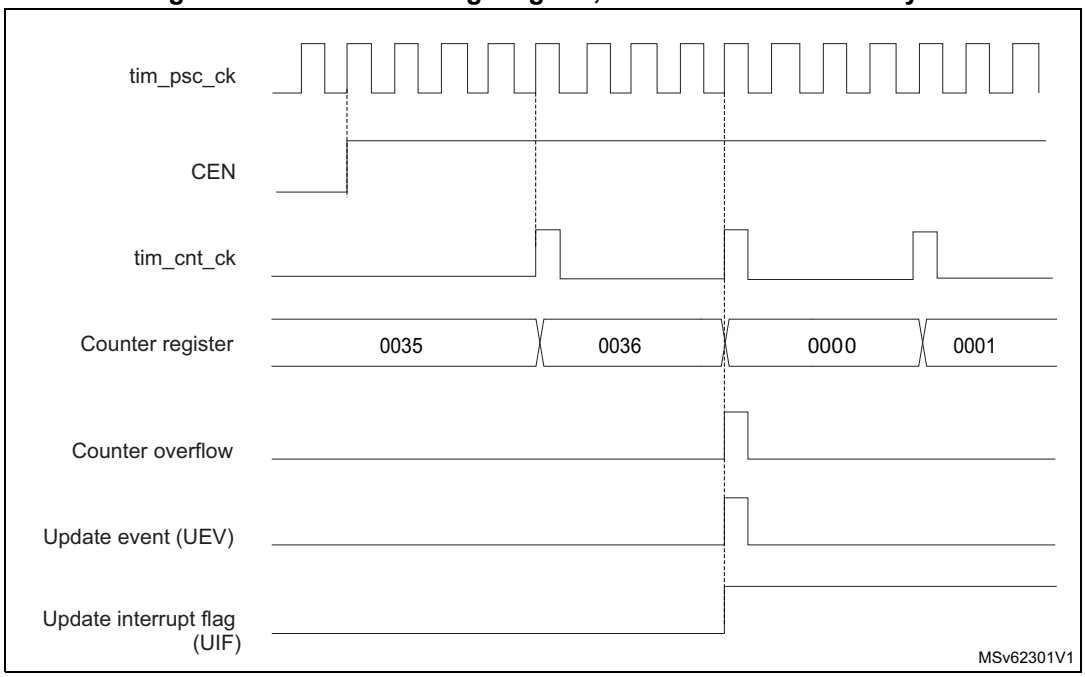


Figure 167. Counter timing diagram, internal clock divided by N

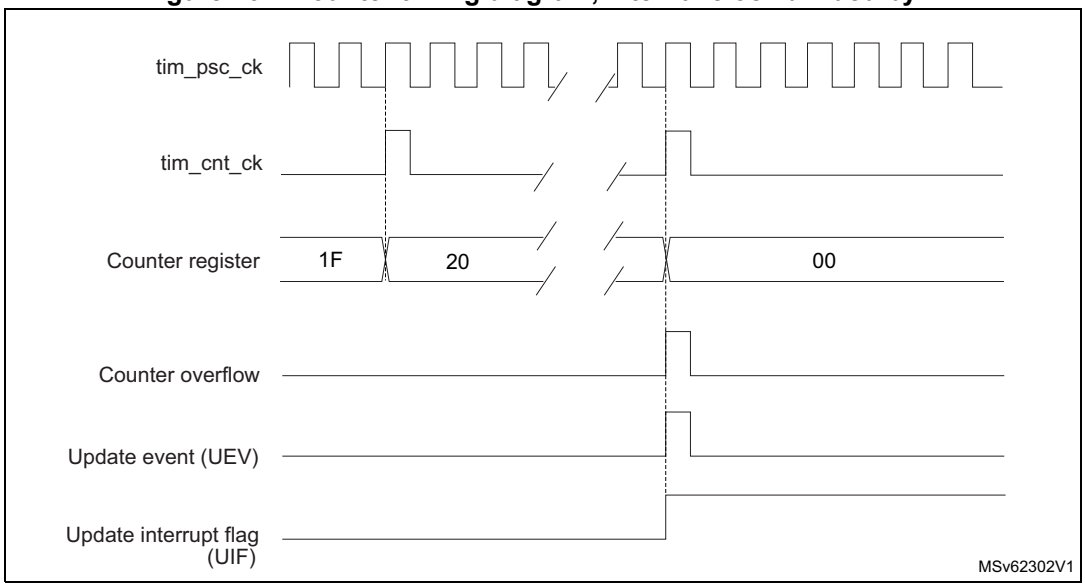


Figure 168. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

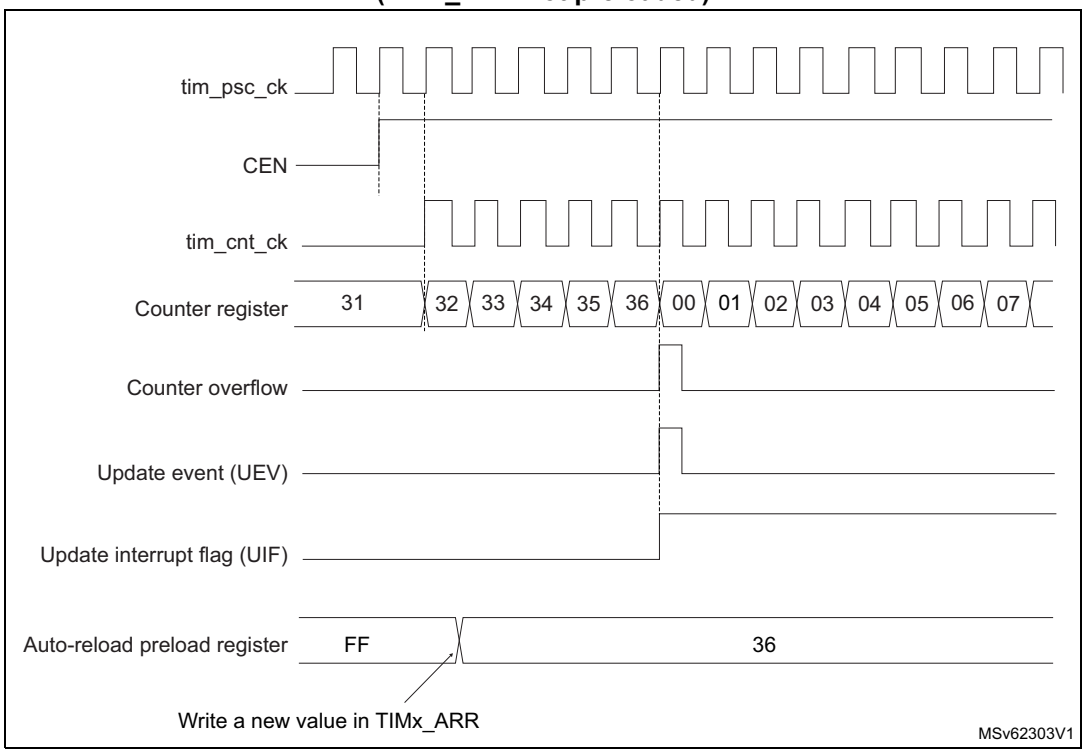
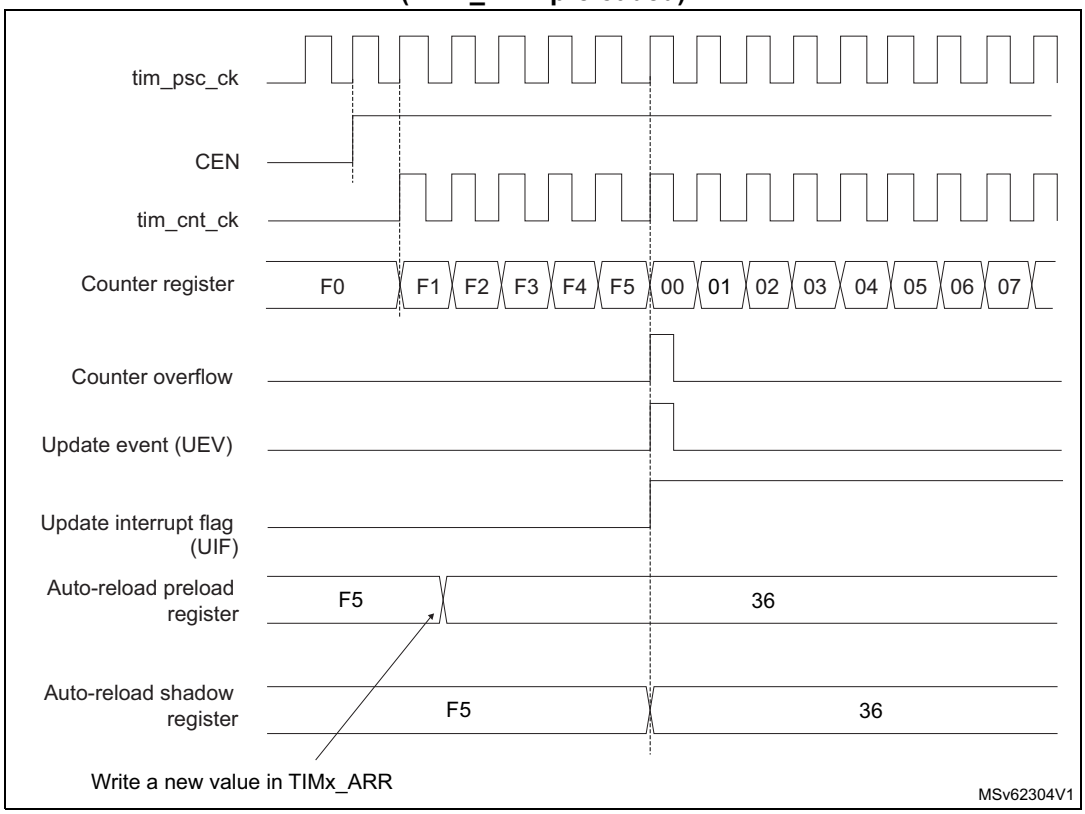


Figure 169. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)



Downcounting mode

In downcounting 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.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter underflow.

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 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 doesn't 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 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 170. Counter timing diagram, internal clock divided by 1

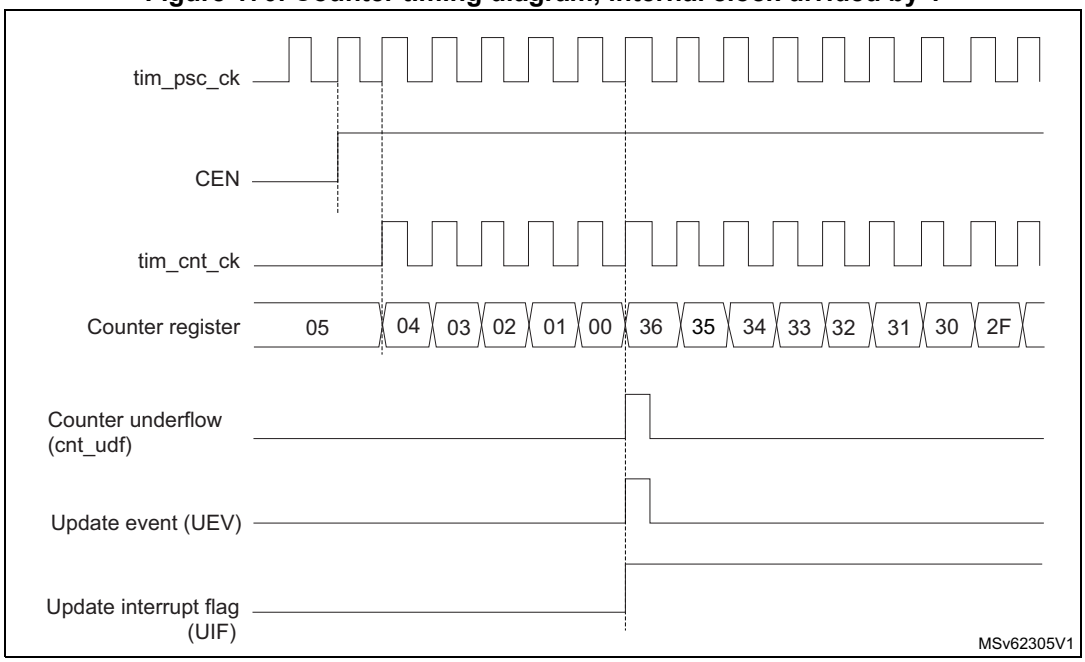


Figure 171. Counter timing diagram, internal clock divided by 2

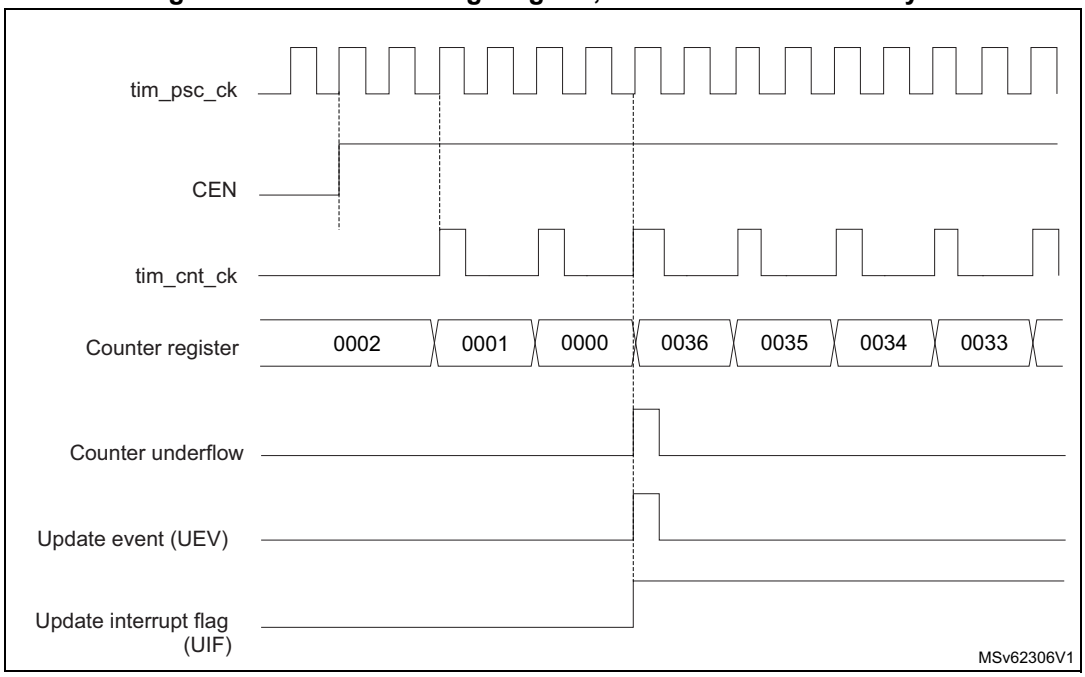


Figure 172. Counter timing diagram, internal clock divided by 4

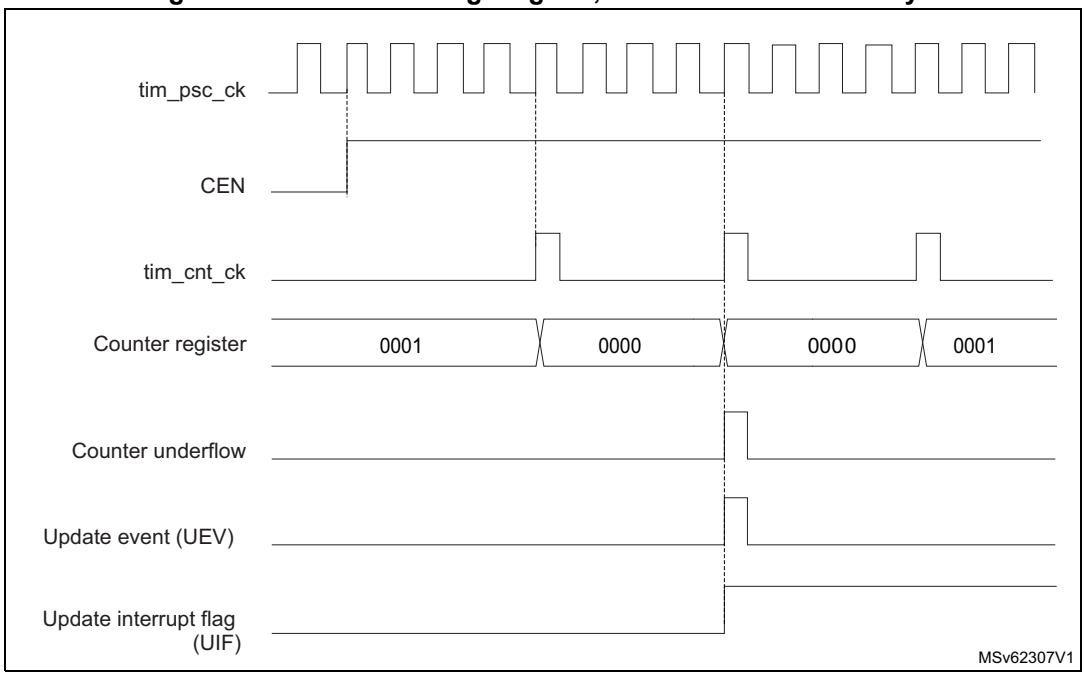


Figure 173. Counter timing diagram, internal clock divided by N

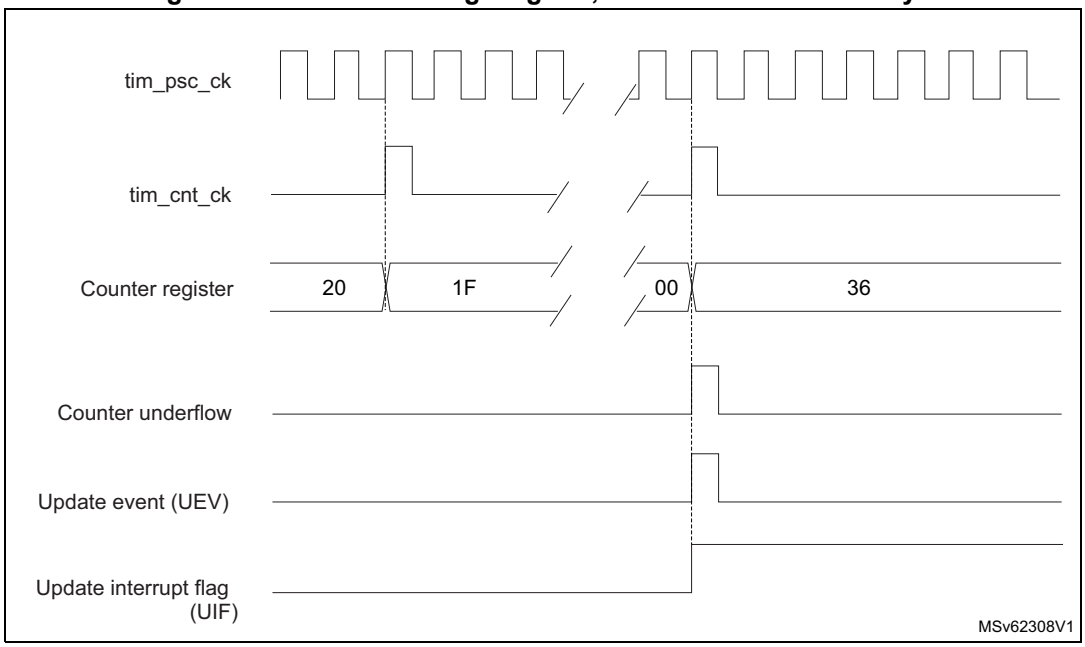
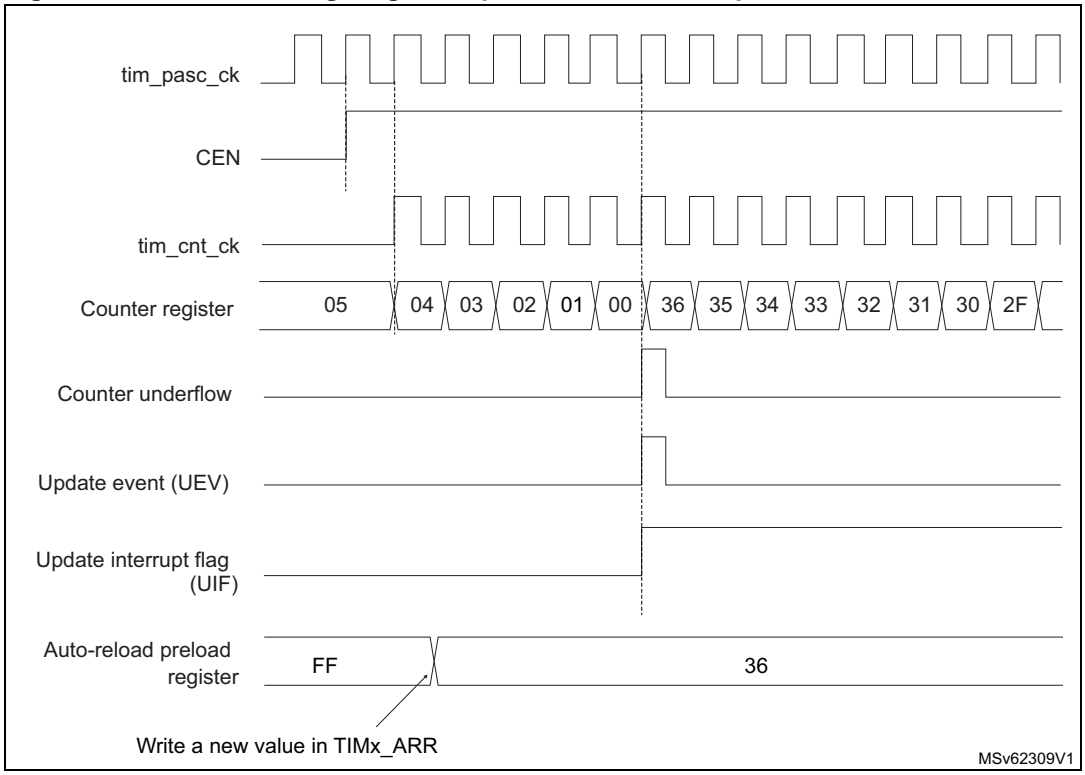


Figure 174. Counter timing diagram, update event when repetition counter is not used



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 DIR direction bit in the 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 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 UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current autoreload value.

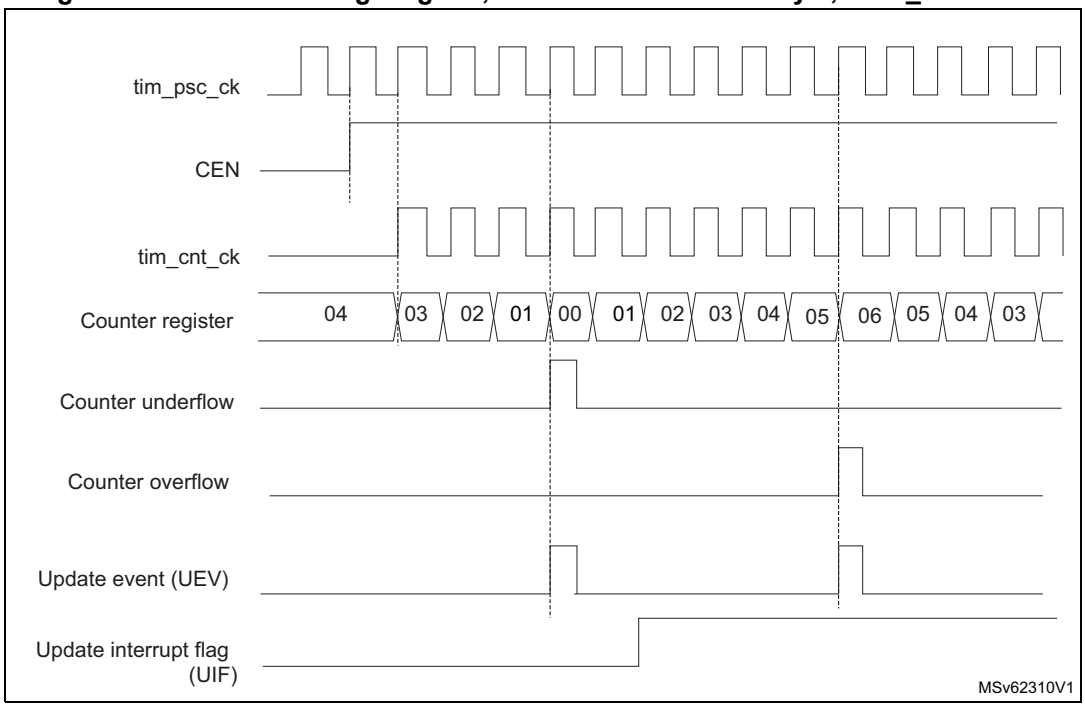
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on 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 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 175. 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 26.6: TIM1 registers*).

Figure 176. Counter timing diagram, internal clock divided by 2

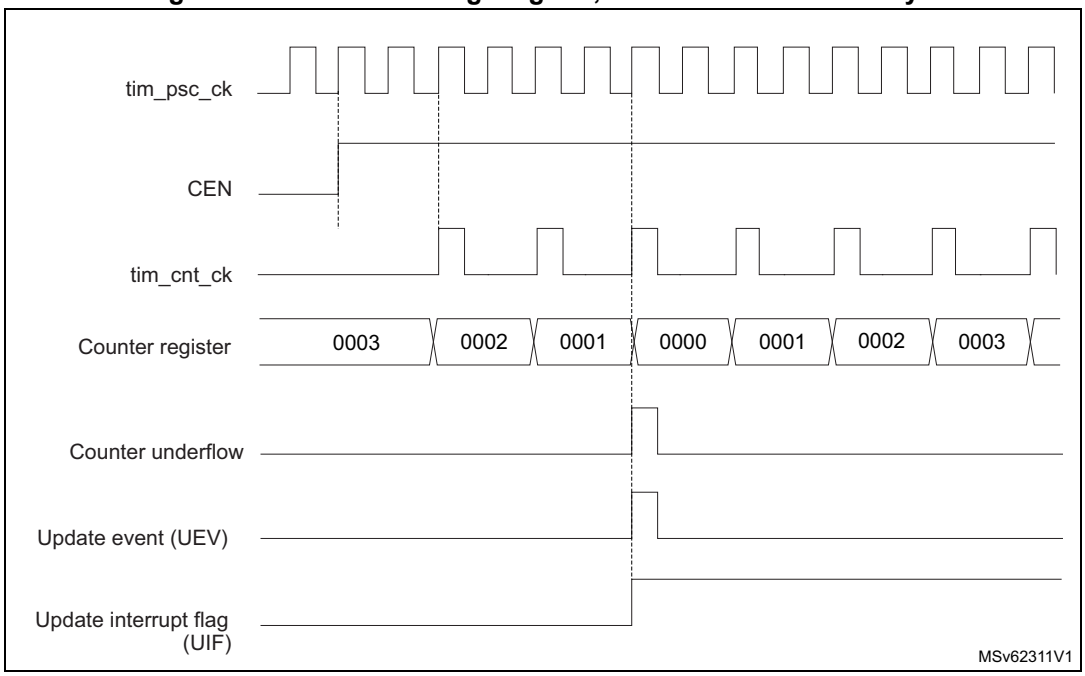


Figure 177. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36

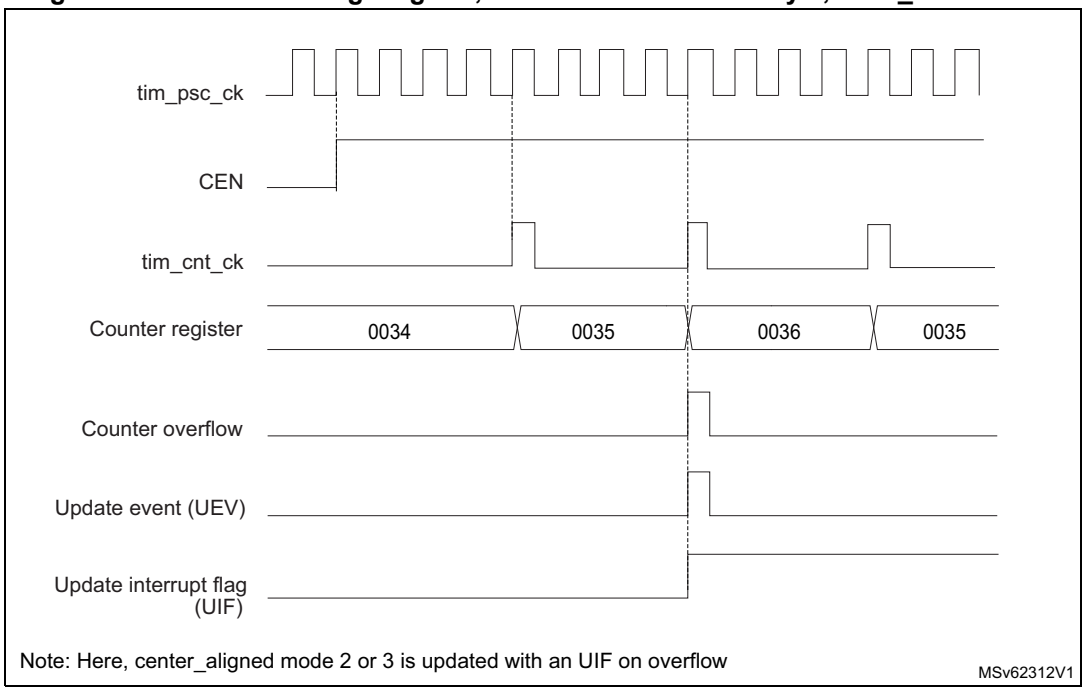


Figure 178. Counter timing diagram, internal clock divided by N

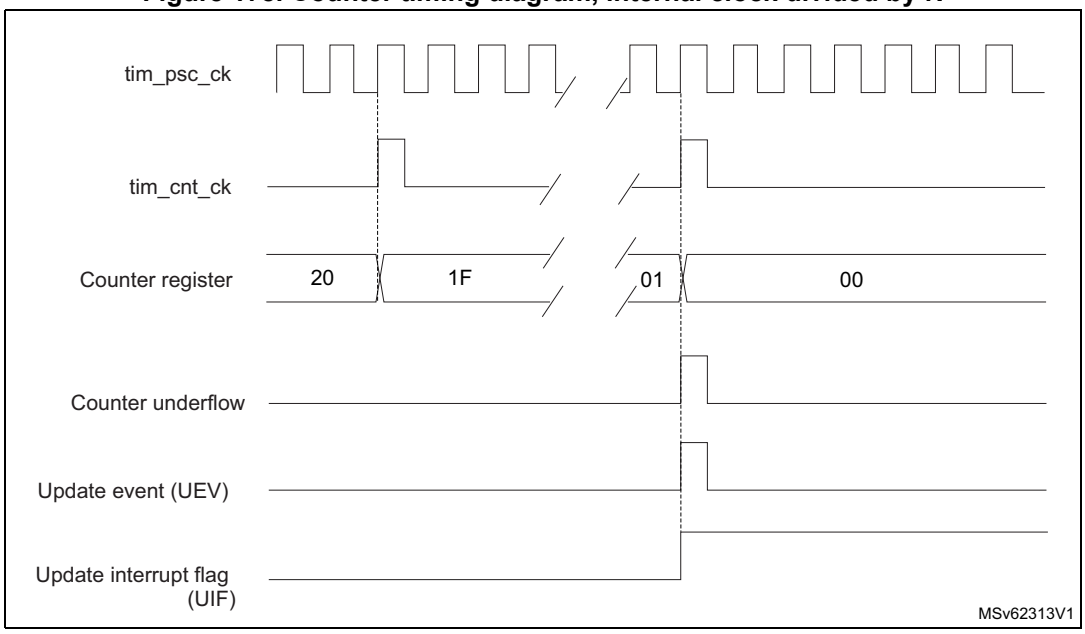


Figure 179. Counter timing diagram, update event with ARPE = 1 (counter underflow)

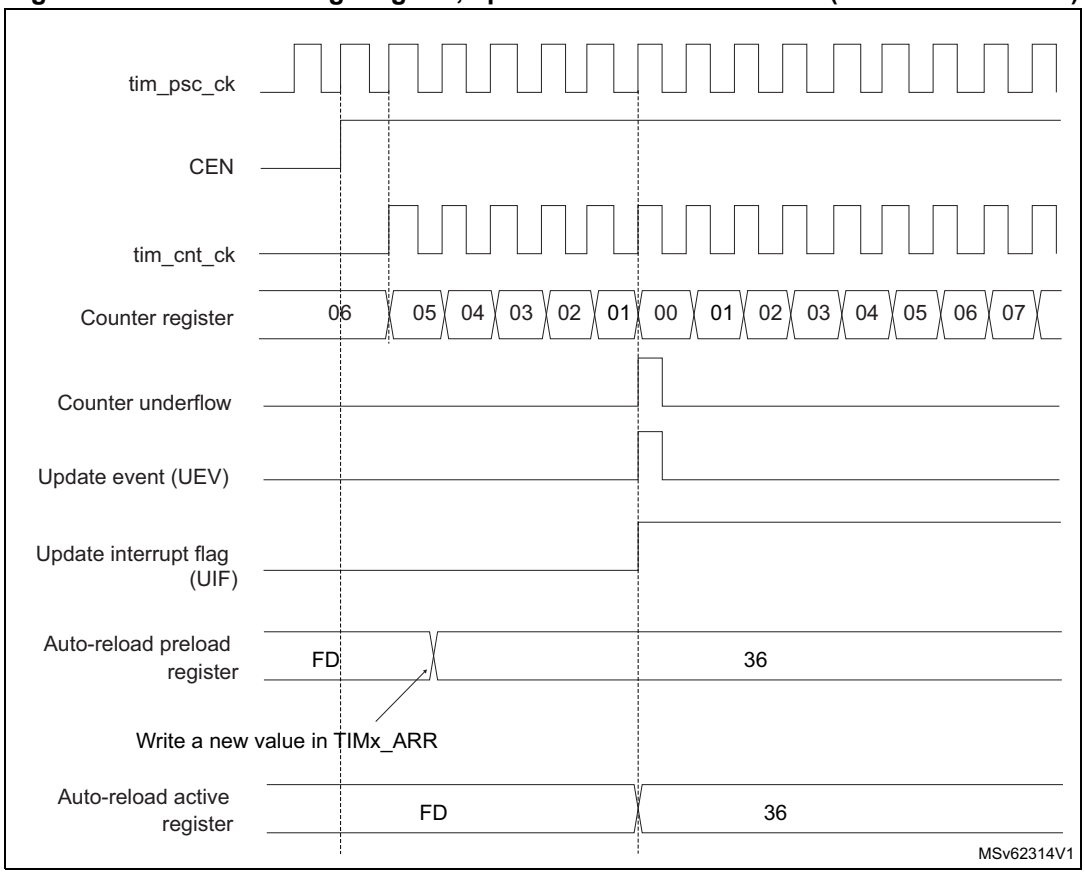
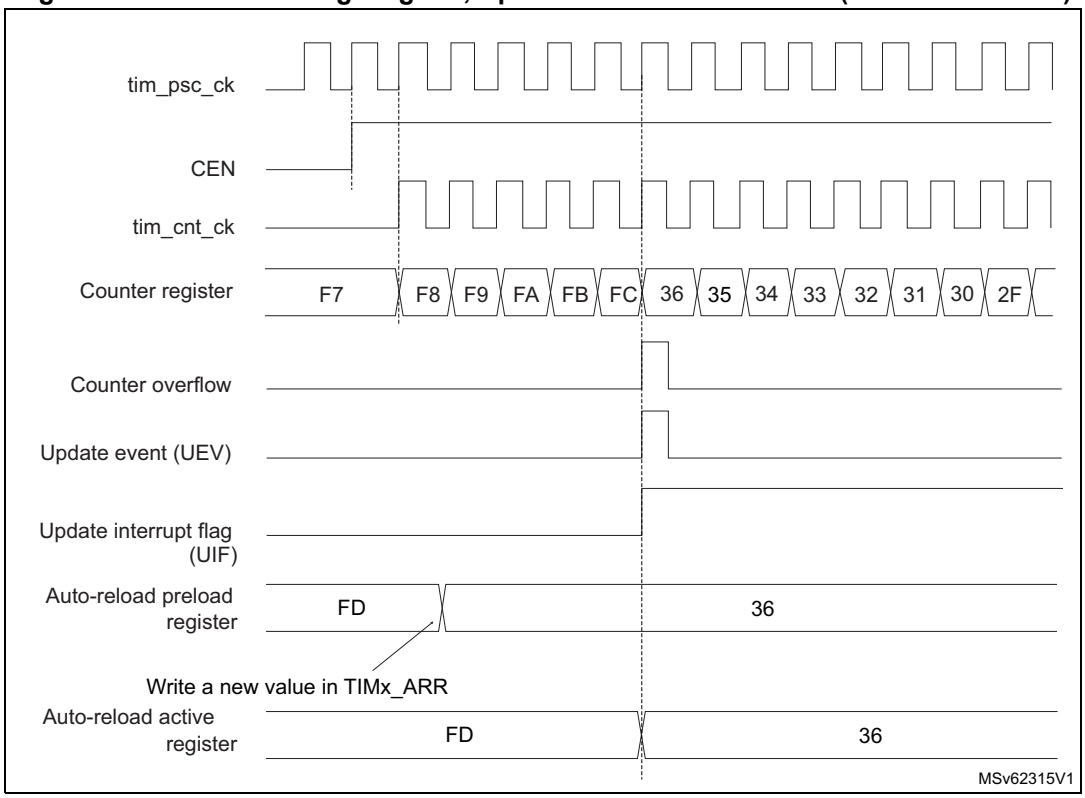


Figure 180. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



26.3.5 Repetition counter

Section 26.3.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. 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+1 counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented:

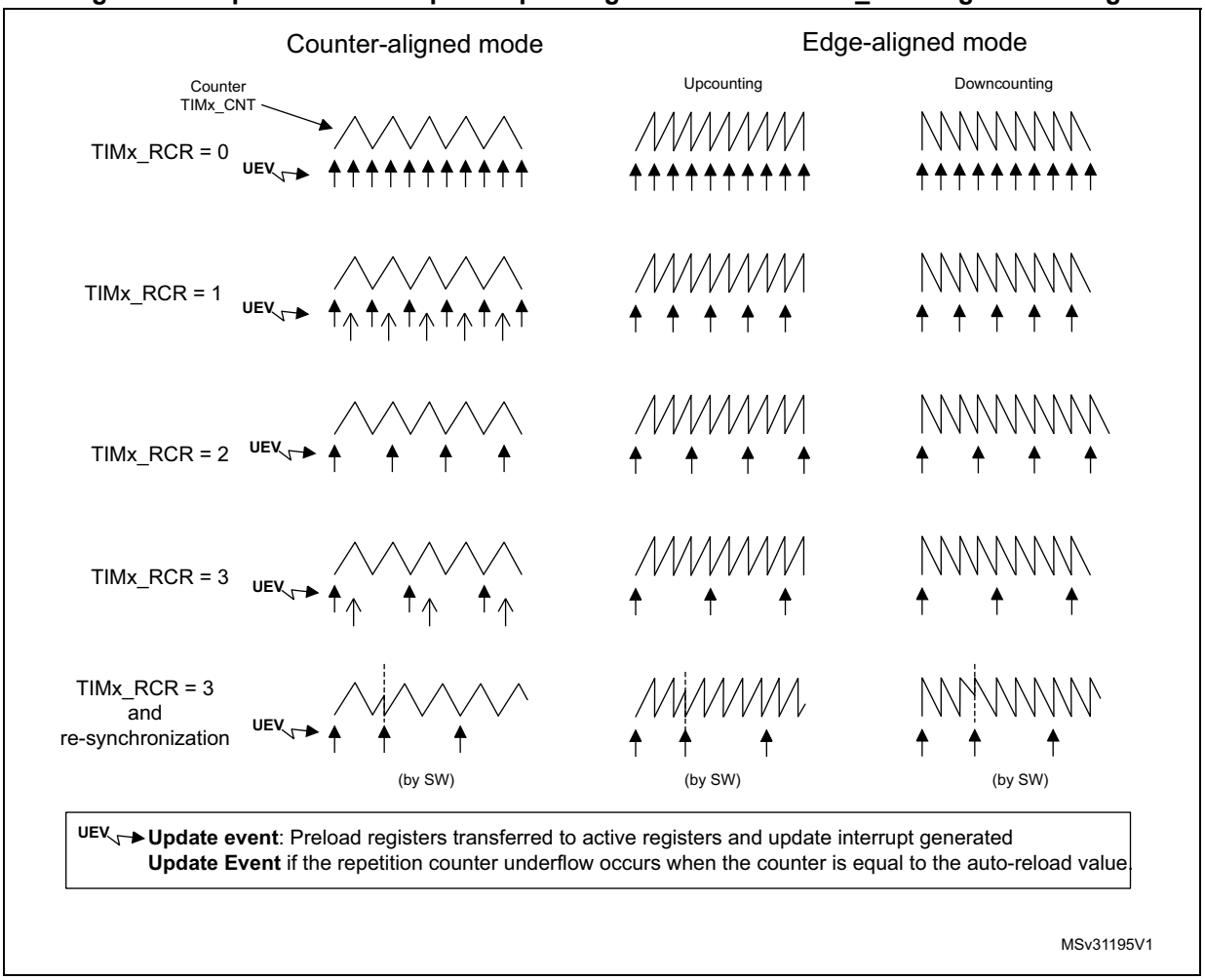
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an autoreload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 181*). 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.

In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

Figure 181. Update rate examples depending on mode and TIMx_RCR register settings



26.3.6 External trigger input

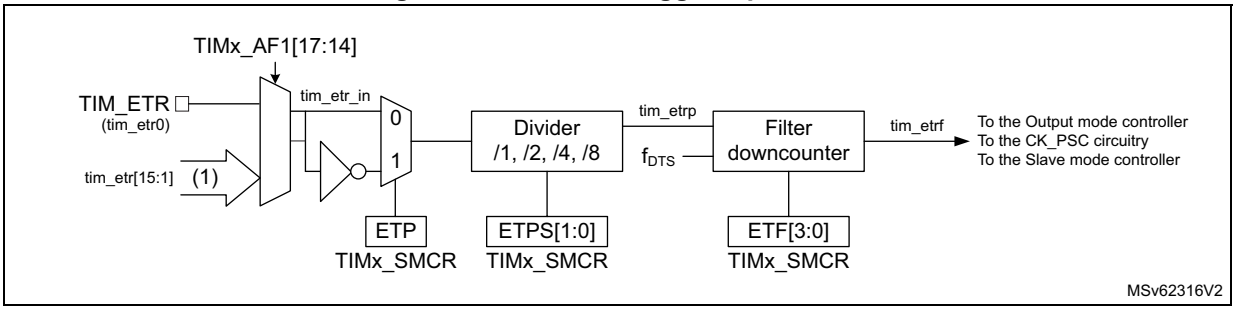
The timer features an external trigger input tim_etr_in. It can be used as:

- external clock (external clock mode 2, see *Section 26.3.7*)
- trigger for the slave mode (see *Section 26.3.30*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 26.3.9*)

Figure 182 below describes the tim_etr_in input conditioning. The input polarity is defined with the ETP bit in TIMxSMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bitfield. The resulting signal (tim_etrf) is available for three purposes: as an external clock, to condition

the output (typically to reset a PWM output for a current limitation), and as a trigger for the Slave mode controller.

Figure 182. External trigger input block



The tim_etr_in input comes from multiple sources: input pins (default configuration), or internal sources. The selection is done with the ETRSEL[3:0] bitfield in the TIMx_AF1 register.

Refer to *Section 26.3.2: TIM1 pins and internal signals* for the list of sources connected to the etr_in input in the product.

26.3.7 Clock selection

The counter clock can be provided by the following clock sources:

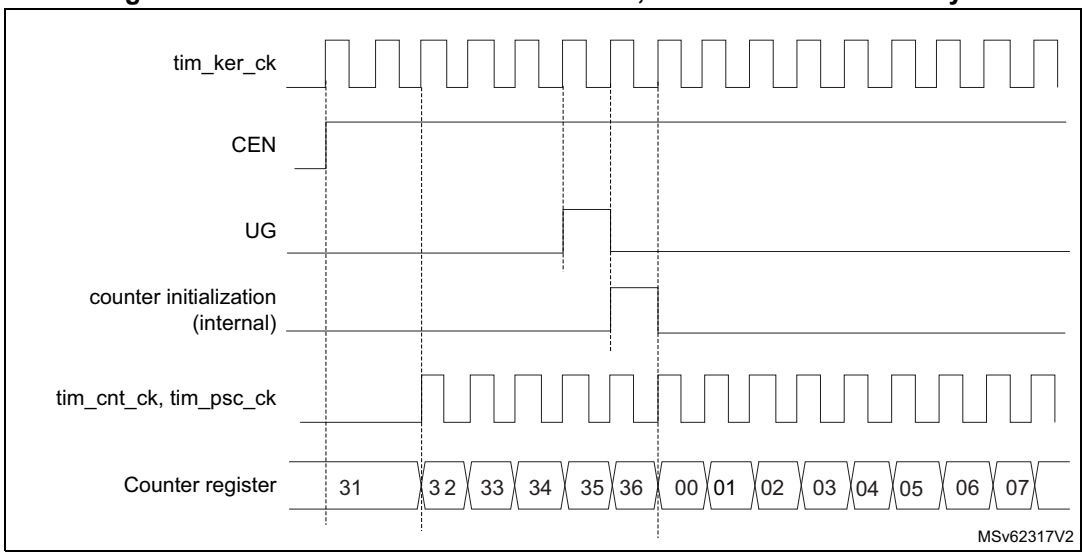
- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2)
- External clock mode2: external trigger input (tim_etr_in)
- Encoder mode

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 000), 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 183 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

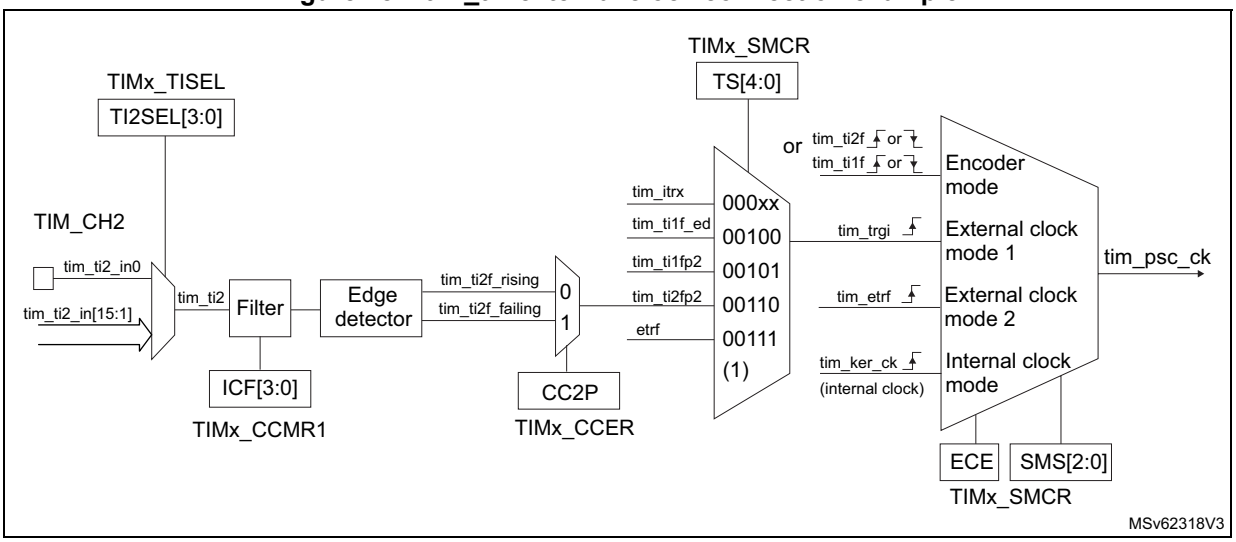
Figure 183. Control circuit in normal mode, internal clock divided by 1



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 184. tim_ti2 external clock connection example



1. Codes ranging from 01000 to 11111 are reserved.

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

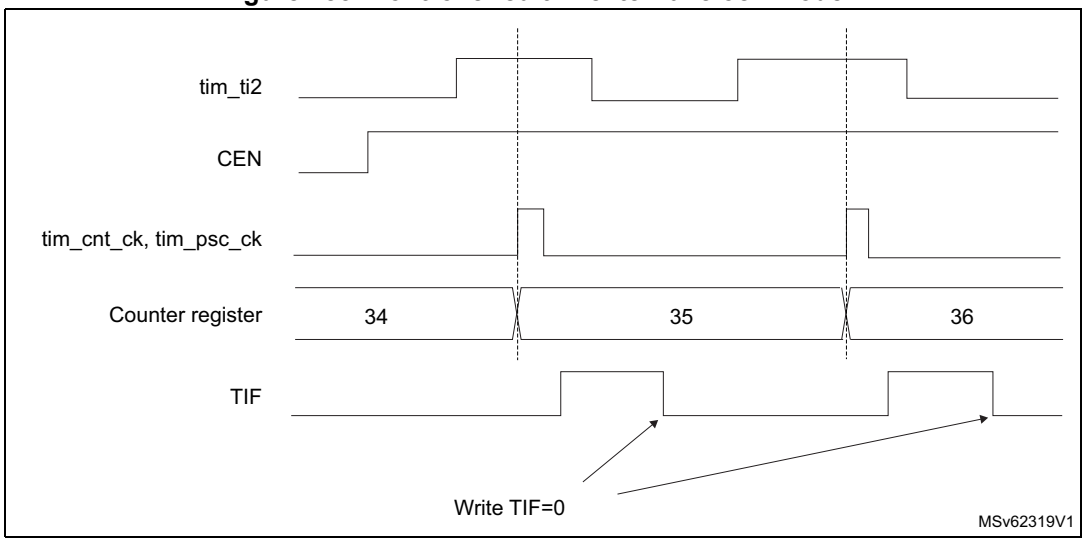
1. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F = 0000).
3. Select rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx_SMCR register.
5. Select tim_ti2 as the trigger input source by writing TS = 00110 in the TIMx_SMCR register.
6. 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 185. Control circuit in external clock mode 1



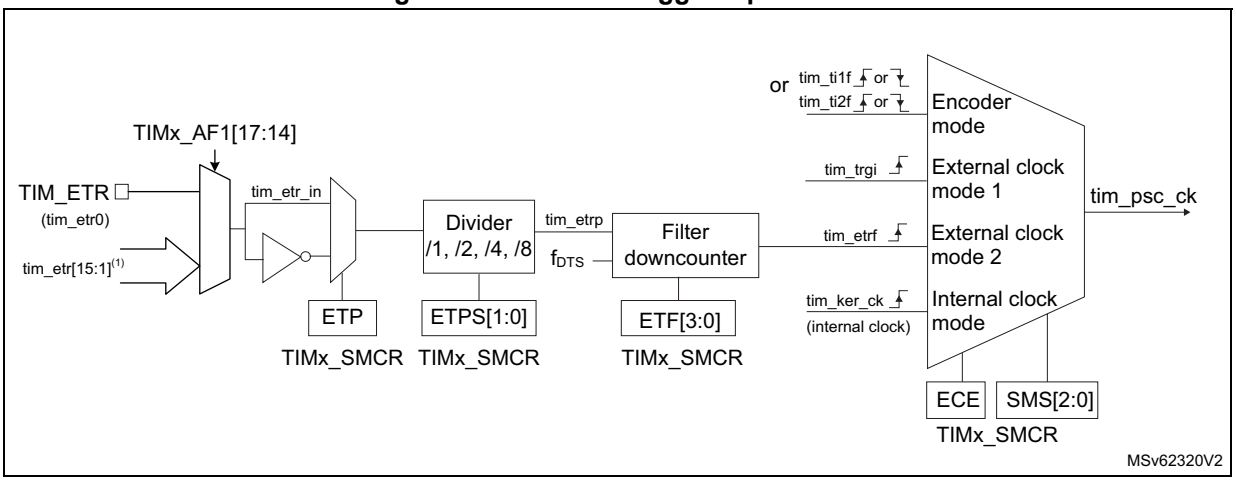
External clock source mode 2

This mode is selected by writing ECE = 1 in the TIMx_SMCR register.

The counter counts at each rising or falling edge on the external trigger input tim_etr_in.

The *Figure 186* gives an overview of the external trigger input block.

Figure 186. External trigger input block



1. Refer to *Section 26.3.2: TIM1 pins and internal signals*.

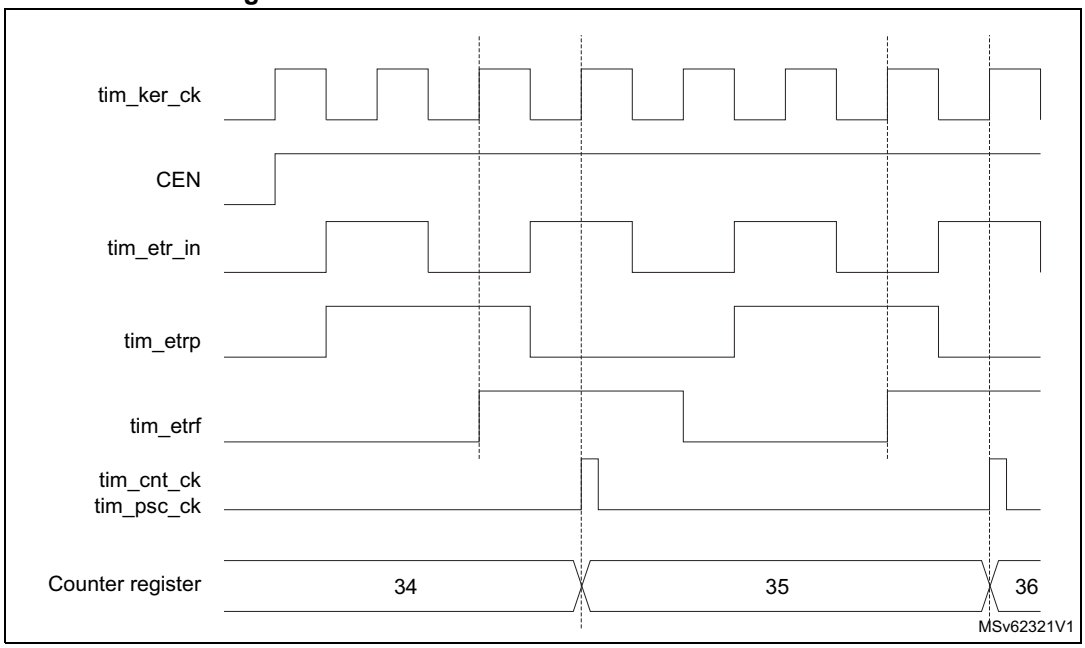
For example, to configure the upcounter to count each 2 rising edges on tim_etr_in, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0] = 0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0] = 01 in the TIMx_SMCR register
3. Select rising edge detection on the tim_etr_in input by writing ETP = 0 in the TIMx_SMCR register
4. Enable external clock mode 2 by writing ECE = 1 in the TIMx_SMCR register.
5. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter counts once each 2 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 which 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 187. Control circuit in external clock mode 2



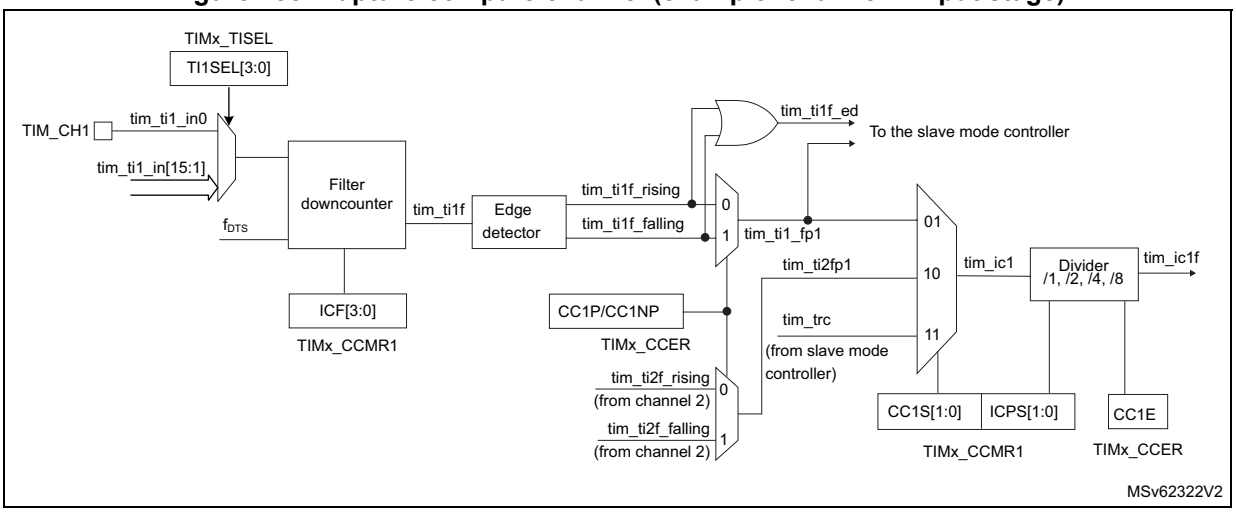
26.3.8 Capture/compare channels

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, except for channels 5 and 6) and an output stage (with comparator and output control).

Figure 188 to *Figure 191* 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 188. 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 189. Capture/compare channel 1 main circuit

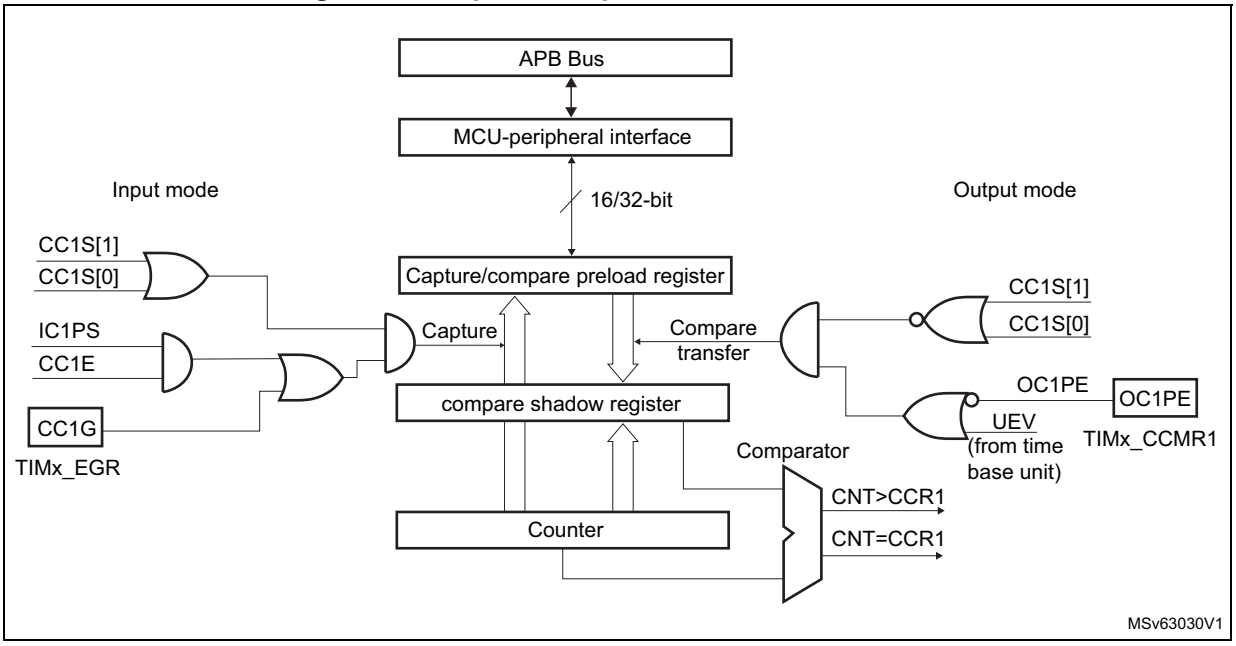
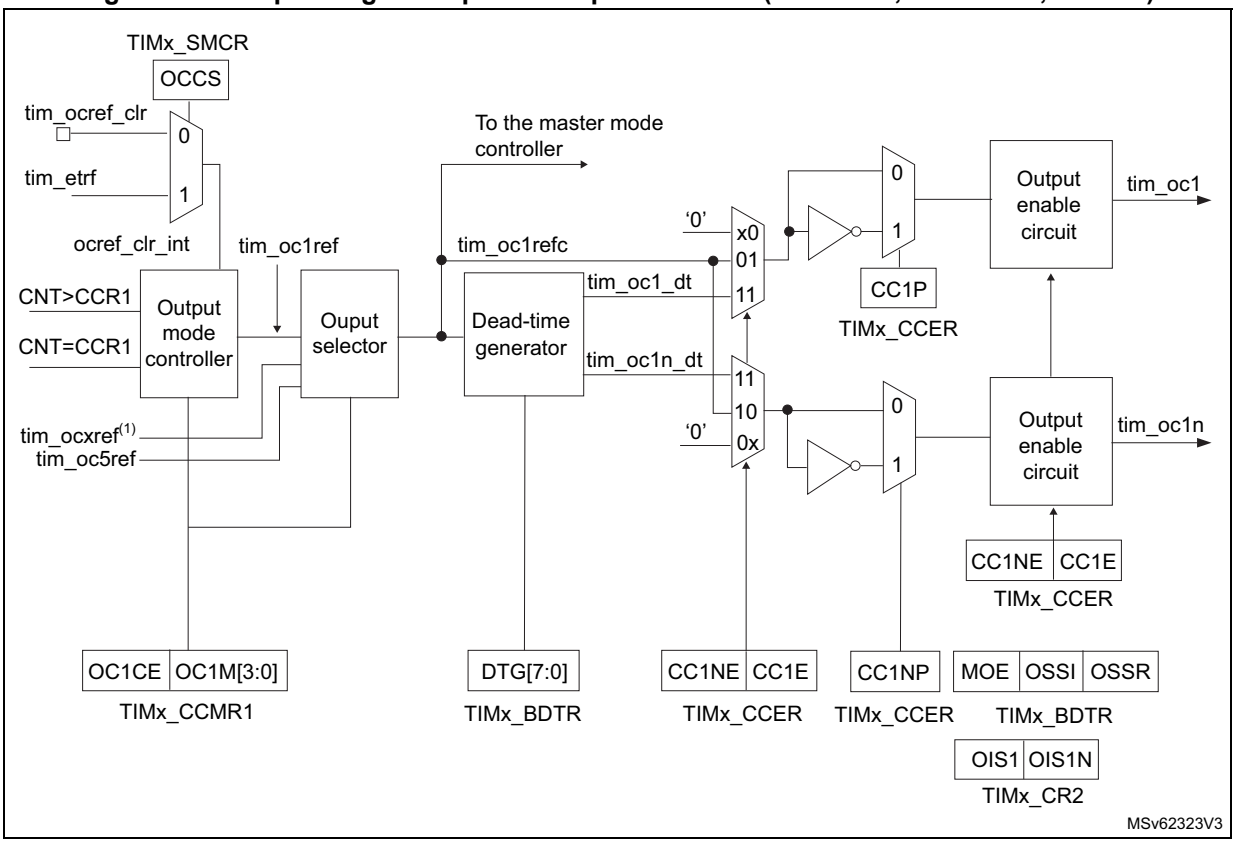
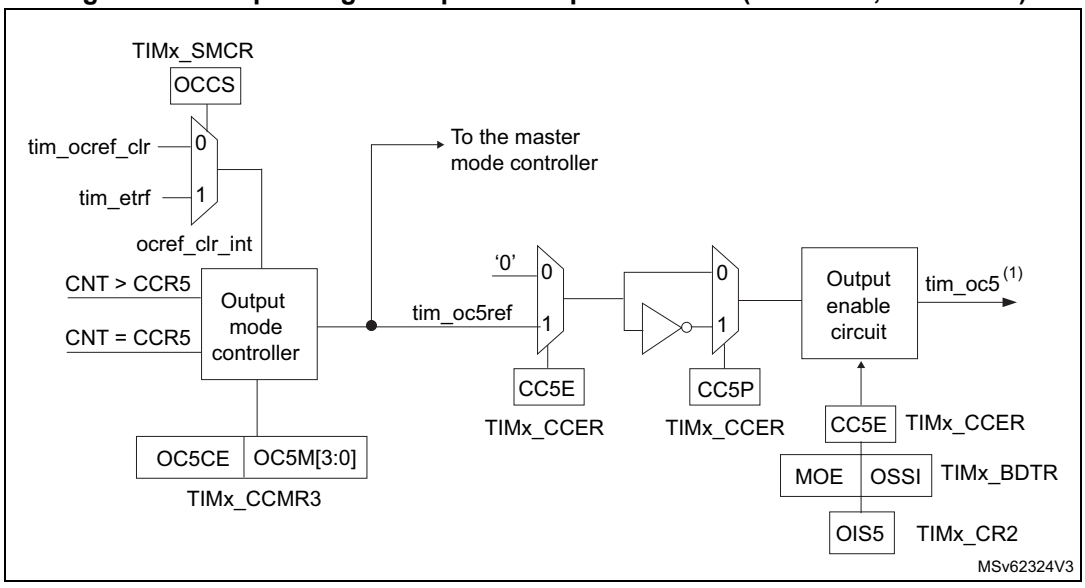


Figure 190. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4)



1. tim_ocxref, where x is the rank of the complementary channel

Figure 191. Output stage of capture/compare channel (channel 5, idem ch. 6)



1. Not available externally.

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.

26.3.9 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:

- 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.
- 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 must 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.
- Select the edge of the active transition on the tim_ti1 channel by writing CC1P and CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).
- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- 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.

26.3.10 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 TIM_CCR2 register holds the pulsewidth (interval between two consecutive rising and falling edges.

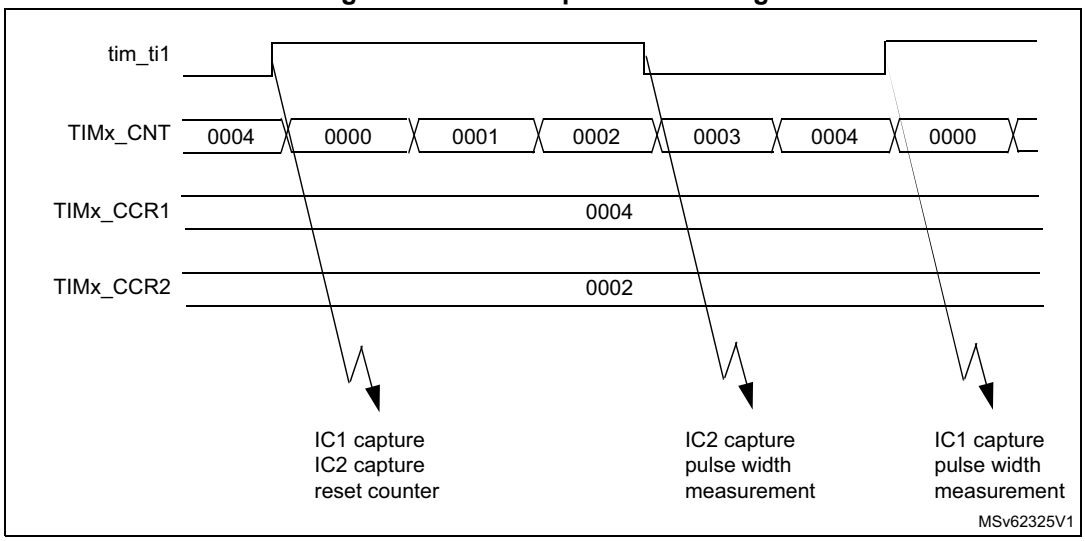
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tixfp1 input.
- These two ICx signals are active on edges with opposite polarity.
- One of the two tim_tixfp signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulsewidth of a PWM signal applied on tim_ti1 can be measured using the following procedure:

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to 0 (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP = 10 (active on falling edge).
- Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to 1 in the TIMx_CCER register.

Figure 192. PWM input mode timing



26.3.11 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, user just needs to write 0101 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 0100 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.

26.3.12 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output, while channel 5 and 6 are only available inside the microcontroller (for instance, for compound waveform generation or for ADC triggering).

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 = 0000), be

set active (OCxM = 0001), be set inactive (OCxM = 0010) or can toggle (OCxM = 0011) 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.

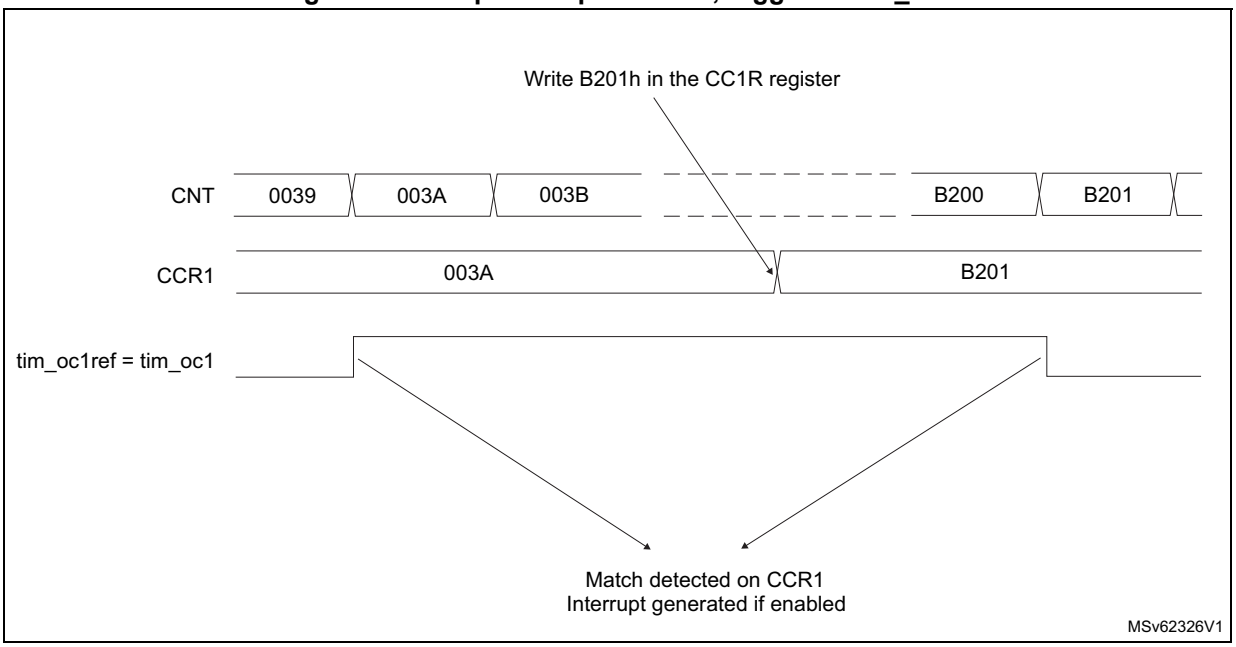
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

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 = 0011 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 193*.

Figure 193. Output compare mode, toggle on tim_oc1



26.3.13 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 0110 (PWM mode 1) or 0111 (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 timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

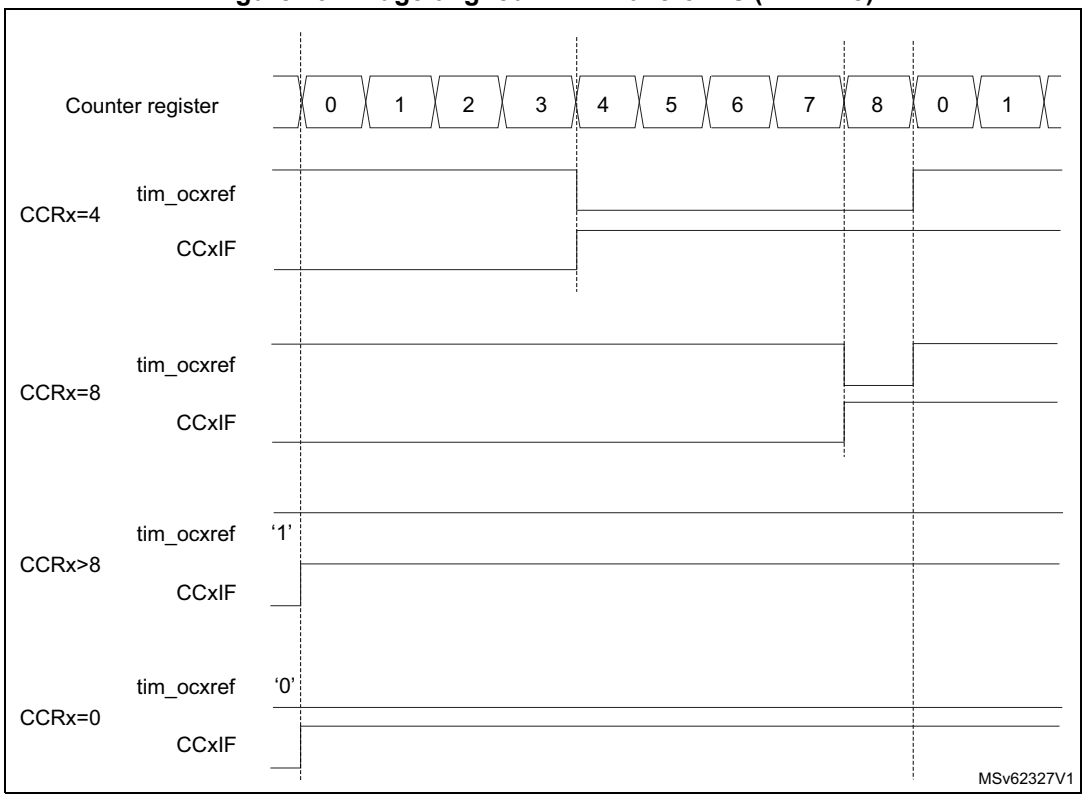
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Upcounting mode*.

  In the following example, the mode is 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 zero then tim_ocxref is held at 0. *Figure 194* shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 194. Edge-aligned PWM waveforms (ARR = 8)



- Downcounting configuration

  Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to the *Downcounting 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 1. 0% PWM is not possible in this mode.

PWM center-aligned mode

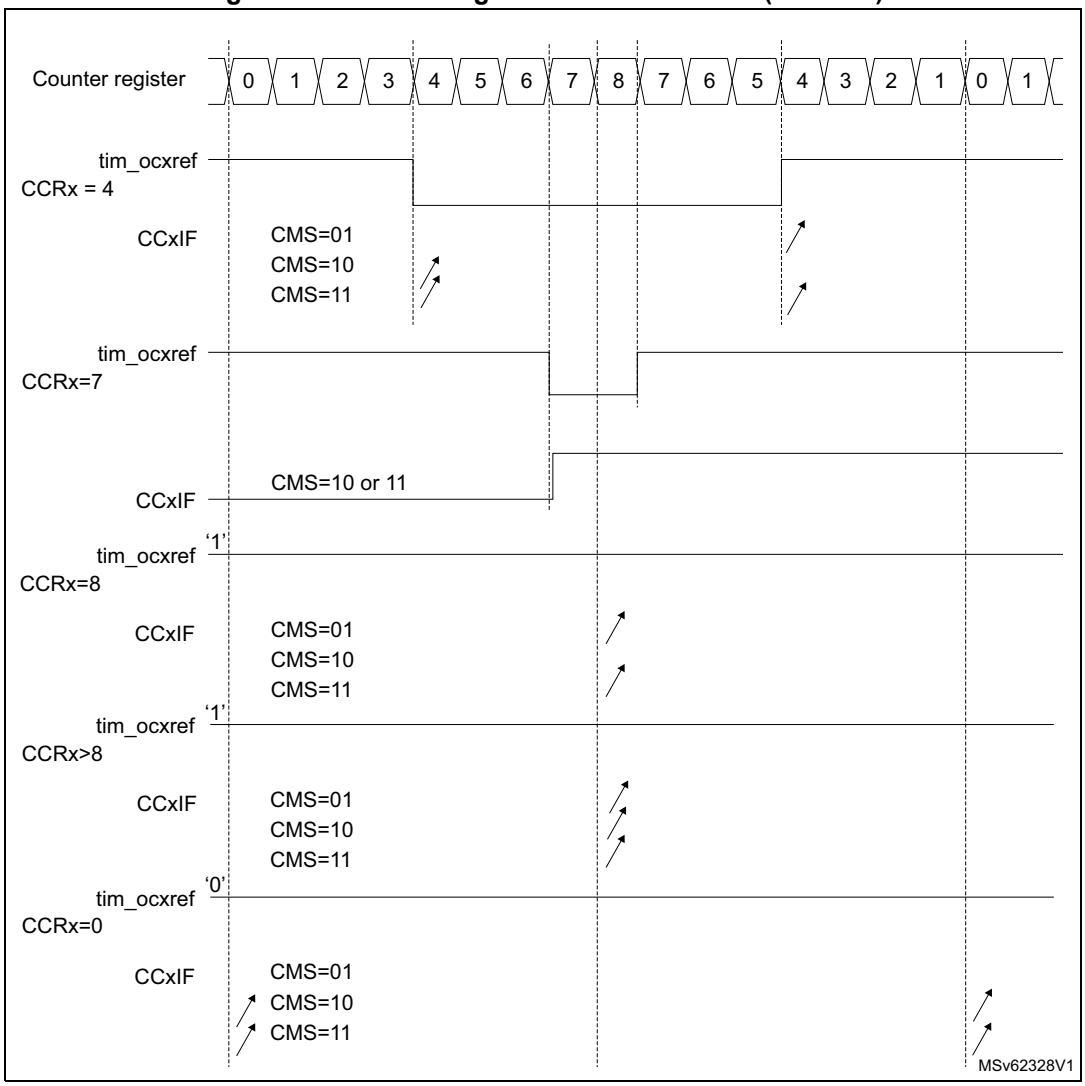
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending 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 195 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 195. 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.

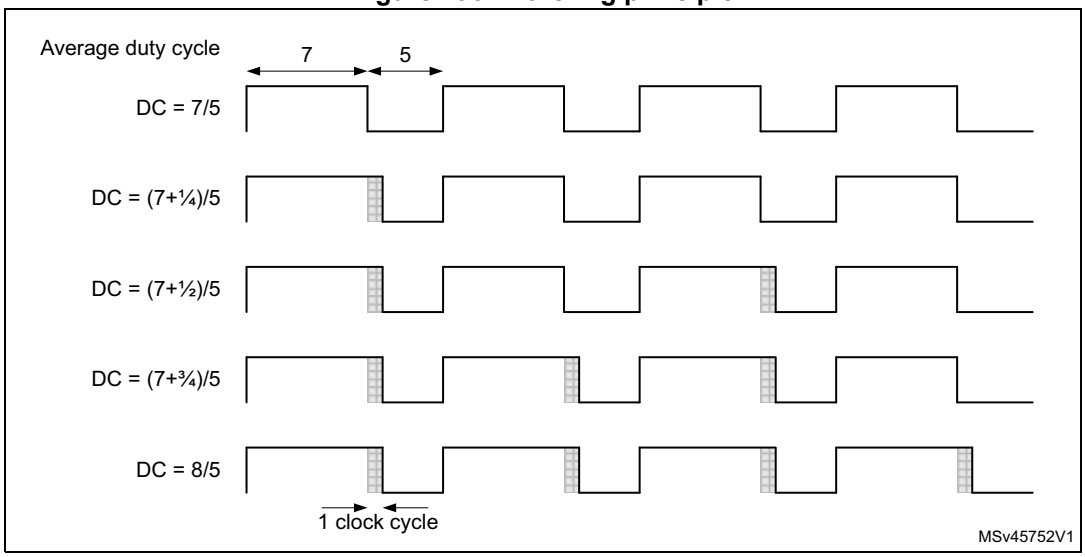
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIMx_CNT > TIMx_ARR). For example, if the counter was counting 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 196* presents the dithering principle applied to four consecutive PWM cycles.

Figure 196. Dithering principle

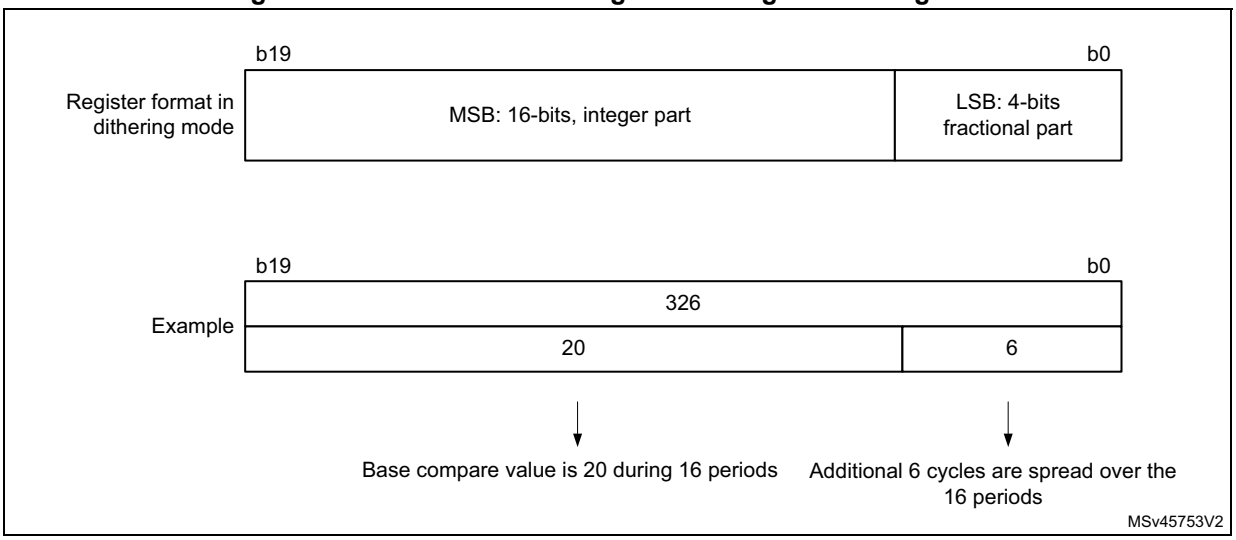


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 197* for example):

- 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 DITHEN bit must be reset.
3. The CCIF flags must be cleared.
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 197. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

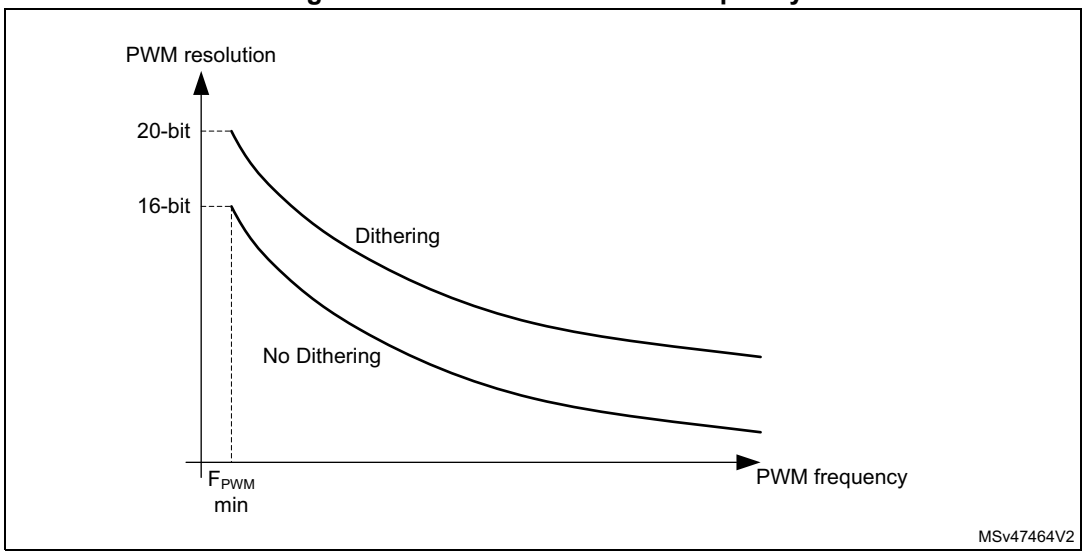
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

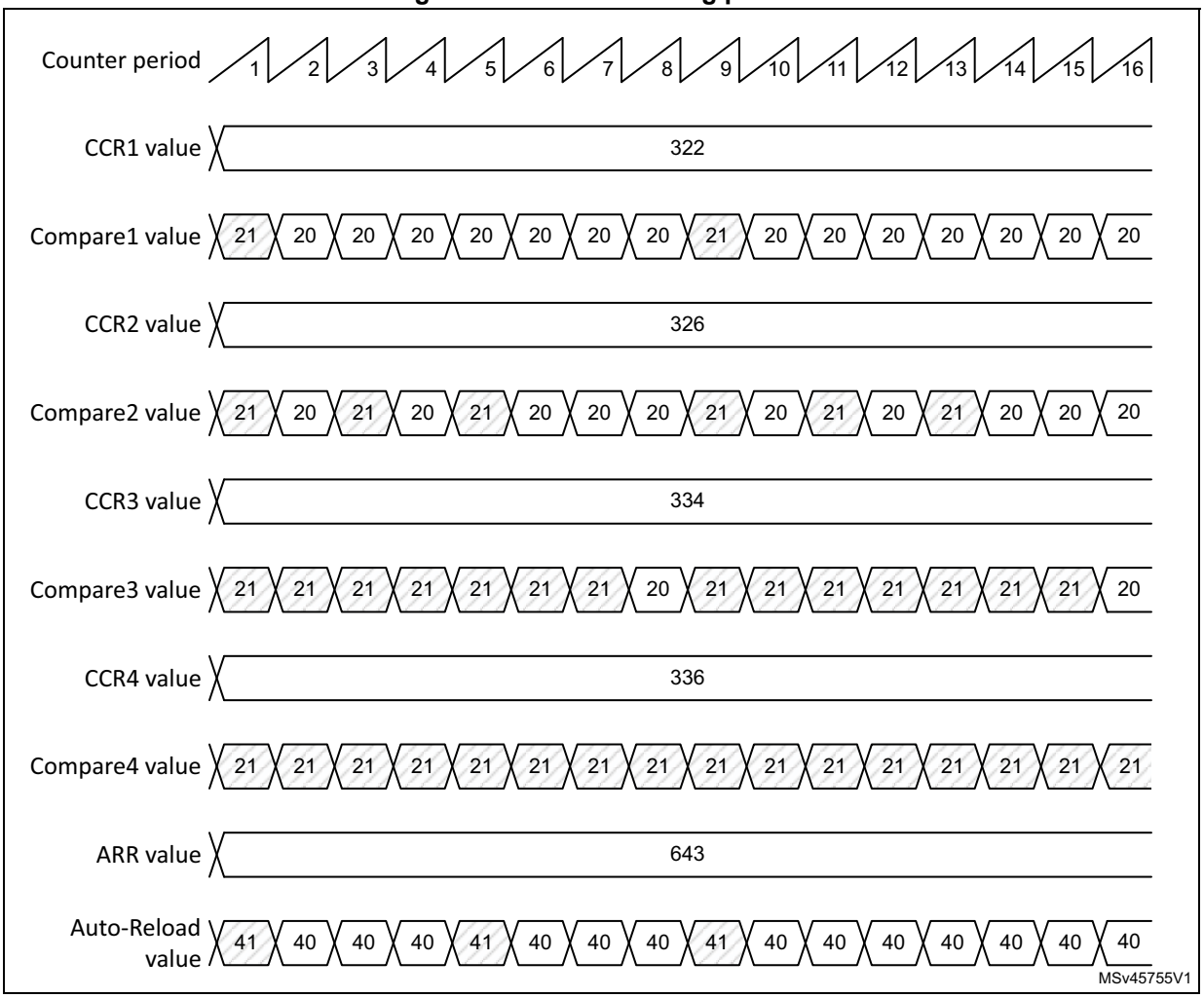
As shown on *Figure 198*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 198. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 199*.

Figure 199. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 176*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 176. 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	-	-	-

Table 176. CCR and ARR register change dithering pattern (continued)

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
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 200*.

Figure 200. Dithering effect on duty cycle in center-aligned PWM mode

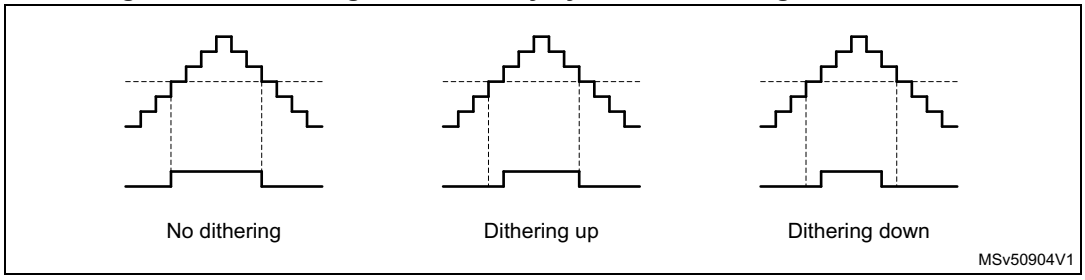


Table 177 shows how the dithering pattern is added in center-aligned PWM mode.

Table 177. 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	-	-	-

Table 177. CCR register change dithering pattern in center-aligned PWM mode (continued)

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
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	-

26.3.14 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 register. 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

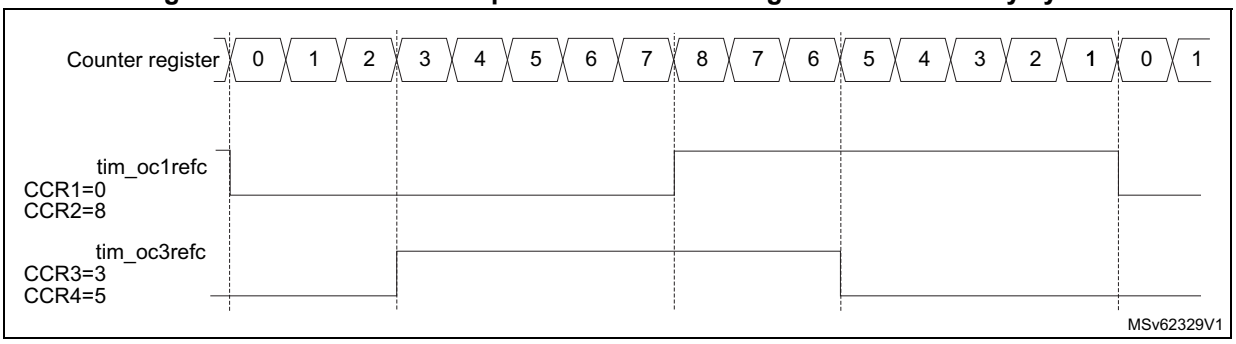
Asymmetric PWM mode can be selected independently on two channel (one tim_ocx output per pair of CCR registers) by writing 1110 (Asymmetric PWM mode 1) or 1111 (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

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 complementary 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 1.

Figure 201 represents an example of signals that can be generated using asymmetric PWM mode (channels 1 to 4 are configured in asymmetric PWM mode 2). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.

Figure 201. Generation of 2 phase-shifted PWM signals with 50% duty cycle



26.3.15 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.

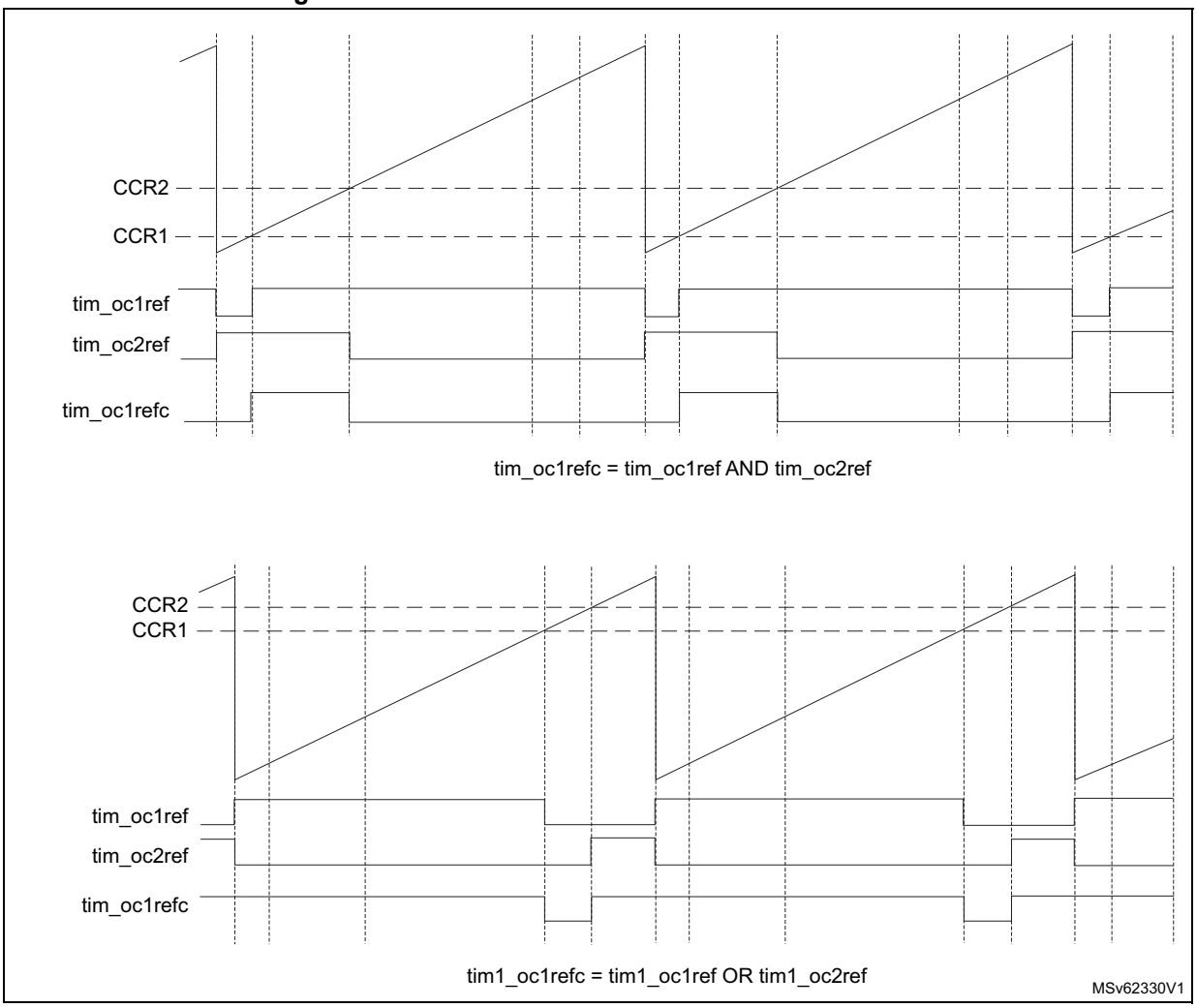
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

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 202 represents 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 2.
- Channel 2 is configured in PWM mode 1.
- Channel 3 is configured in Combined PWM mode 2.
- Channel 4 is configured in PWM mode 1.

Figure 202. Combined PWM mode on channel 1 and 3



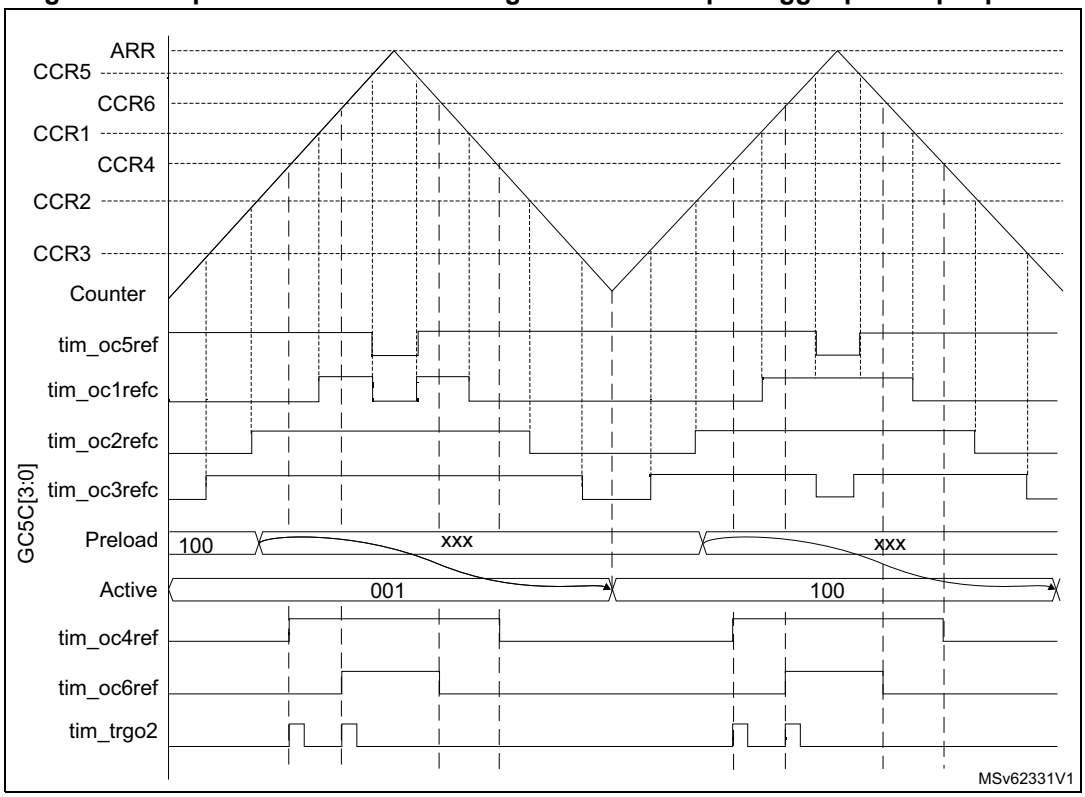
26.3.16 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The tim_oc5ref signal is used to define the resulting combined signal. The 3-bits GC5C[3:1] in the TIMx_CCR5 allow selection on which reference signal the tim_oc5ref is combined. The resulting signals, tim_ocxrefc, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, tim_oc1refc is controlled by TIMx_CCR1 and TIMx_CCR5.
- If GC5C2 is set, tim_oc2refc is controlled by TIMx_CCR2 and TIMx_CCR5.
- If GC5C3 is set, tim_oc3refc is controlled by TIMx_CCR3 and TIMx_CCR5.

Combined 3-phase PWM mode can be selected independently on channels 1 to 3 by setting at least one of the 3-bits GC5C[3:1].

Figure 203. 3-phase combined PWM signals with multiple trigger pulses per period



The tim_trgo2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to *Section 26.3.31: ADC triggers* for more details.

26.3.17 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1) can output two complementary signals and manage the switching-off and the switching-on instants 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, or 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 185: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature* 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

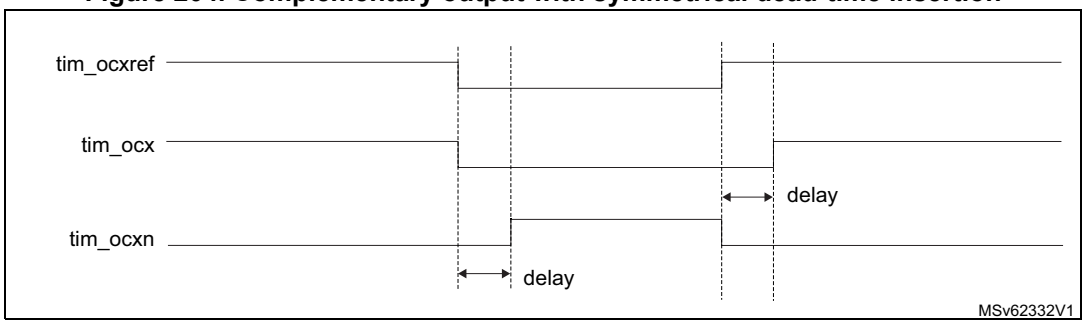
reference waveform tim_ocxref, it generates two outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- 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 considering CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1 in these examples.

Figure 204. 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 205*.

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 205. Asymmetrical deadtime

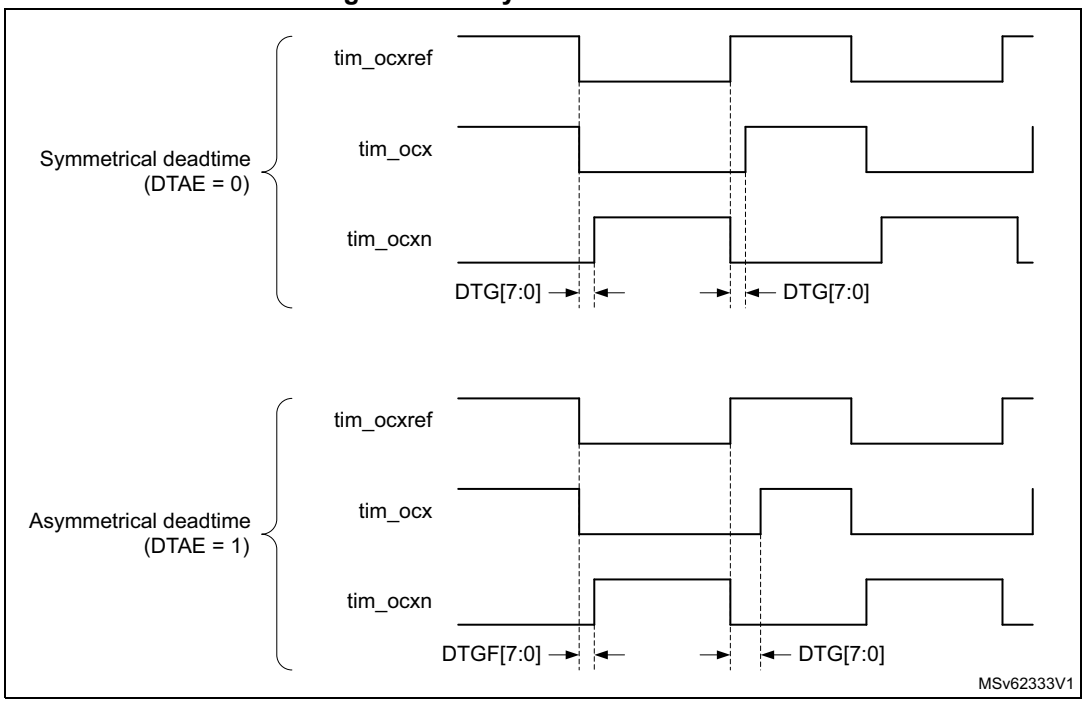


Figure 206. Dead-time waveforms with delay greater than the negative pulse

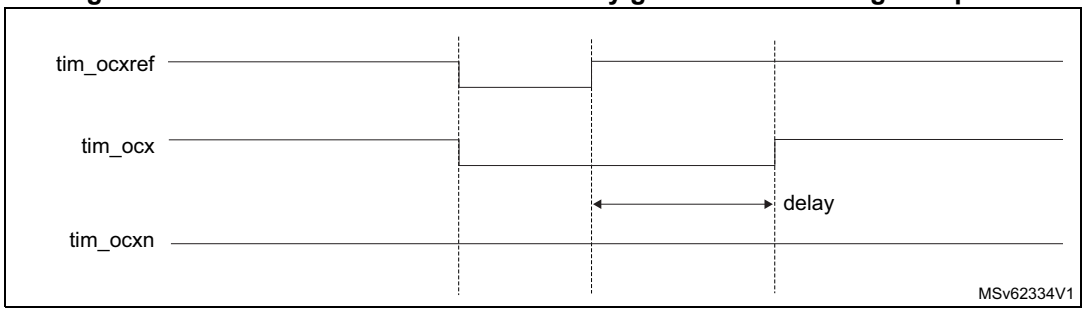
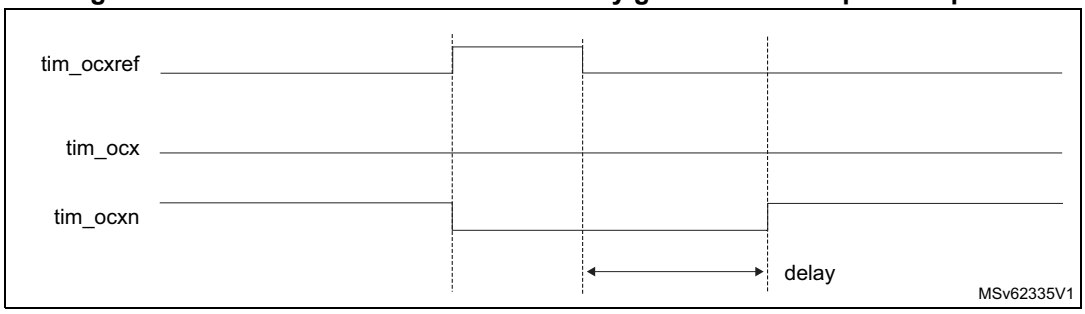


Figure 207. Dead-time waveforms with delay greater than the positive pulse



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 26.6.20: TIM1 break and dead-time register (TIM1_BDTR)* 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.

26.3.18 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The two break inputs are 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. A number of internal MCU events can also be selected to trigger an output shut-down.

The break features two channels. A break channel which gathers both system-level fault (clock failure, ECC/parity 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. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.

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 185: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break functions can be enabled by setting the BKE and BK2E bits in the TIMx_BDTR register. The break input polarities can be selected by configuring the BKP and BK2P bits in the same register. BKEx and BKPx can be modified at the same time. When the BKEx and BKPx 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 sources for break (tim_brk) channel are:

- External sources connected to one of the TIMx_BKIN pin (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 26.3.2: TIM1 pins and internal signals* for product specific implementation)
  - coming from a system break request (refer to *Section 26.3.2: TIM1 pins and internal signals* for product specific implementation)

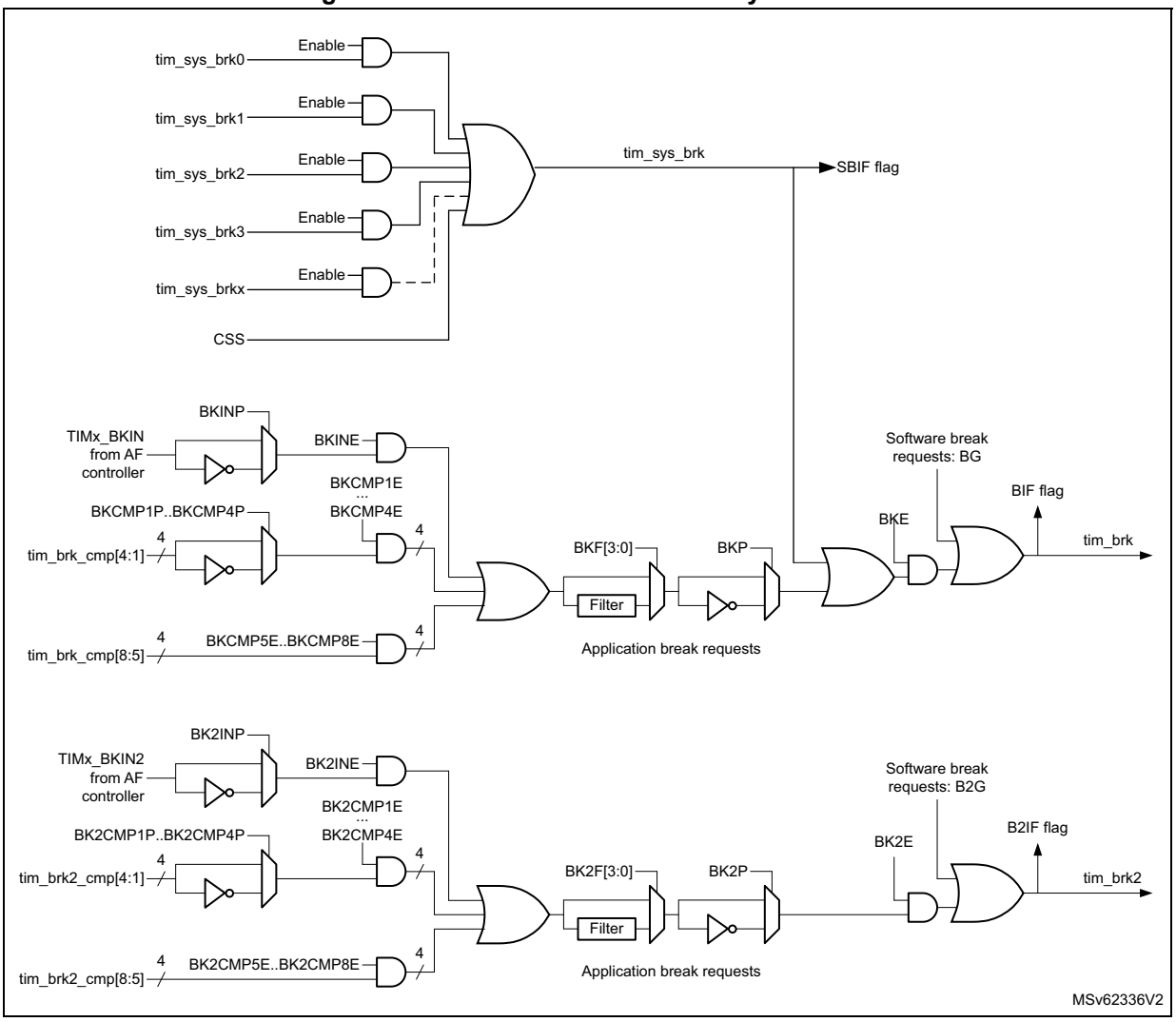
The sources for break2 (tim_brk2) are:

- External sources connected to one of the TIMx_BKIN2 pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources coming from a tim_brk2_cmpx input (refer to *Section 26.3.2: TIM1 pins and internal signals* for product specific implementation)

Break events can also be generated by software using BG and B2G bits in the TIMx_EGR register.

All sources are ORed before entering the timer tim_brk or tim_brk2 inputs, as per *Figure 208* below.

Figure 208. Break and Break2 circuitry overview



Note: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When one of the breaks occurs (selected level on one of the break inputs):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state, or even releasing the control to the GPIO controller (selected by the OSSI bit). This feature is enabled 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 controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in 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 slightly longer than usual (around 2 tim_ker_ck clock cycles).

  - If OSSI = 0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flag (SBIF, BIF, and B2IF bits in the TIMx_SR register) is set. An interrupt is 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). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to 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.

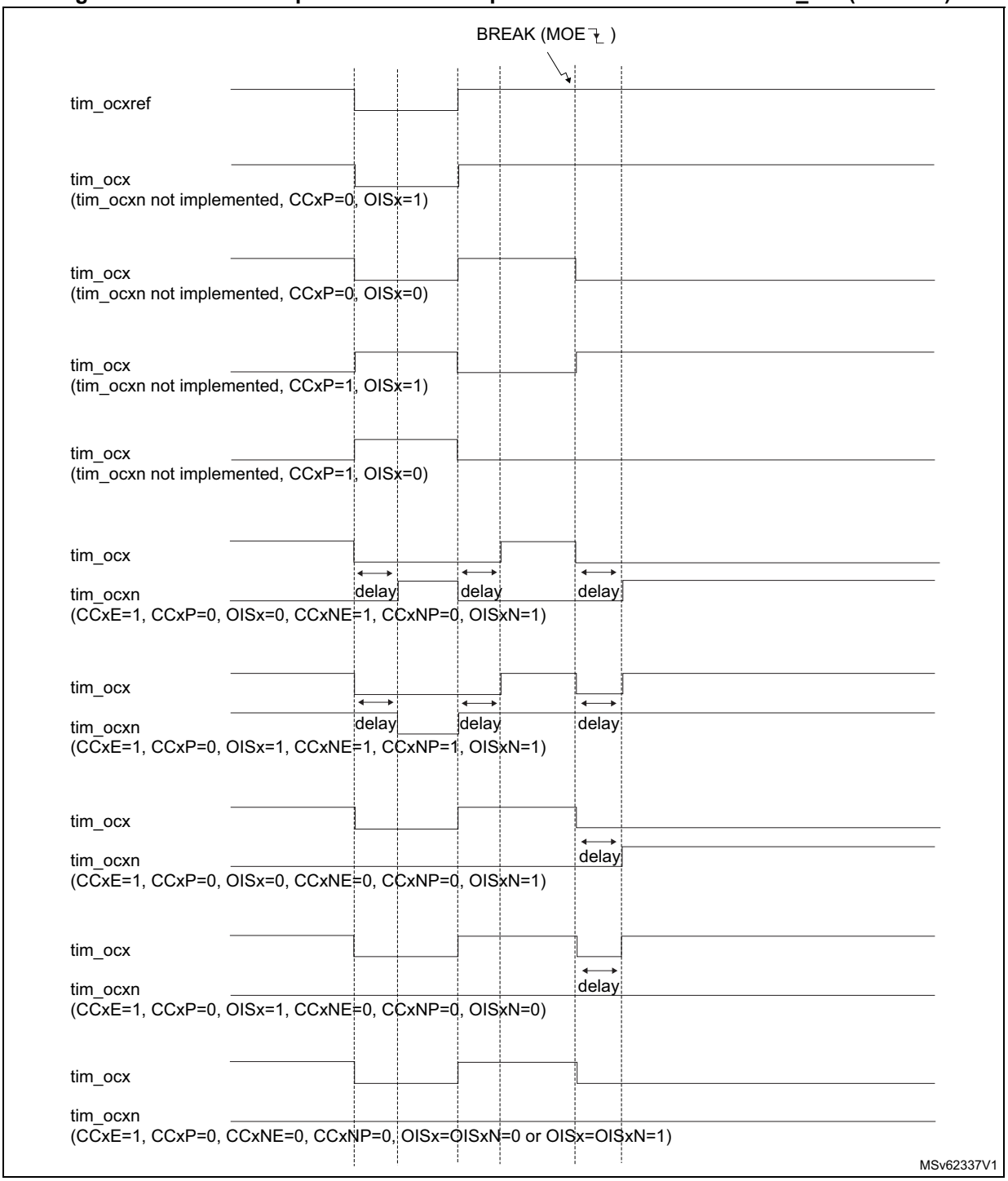
Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

The break inputs are active 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 and B2IF cannot be cleared.

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 application can choose from three levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to *Section 26.6.20: TIM1 break and dead-time register (TIM1_BDTR)*. The LOCK bits can be written only once after an MCU reset.

Figure 209 shows an example of behavior of the outputs in response to a break.

Figure 209. Various output behavior in response to a break event on tim_brk (OSSI = 1)



The two break inputs have different behaviors on timer outputs:

- The tim_brk input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- tim_brk2 can only disable (inactive state) the PWM outputs.

The tim_brk has a higher priority than tim_brk2 input, as described in *Table 178*.

Note: tim_brk2 must only be used with OSSR = OSSI = 1.

Table 178. Behavior of timer outputs versus tim_brk/tim_brk2 inputs

tim_brk	tim_brk2	Timer outputs state	Typical use case	
			tim_ocxn output (low side switches)	tim_ocx output (high side switches)
Active	X	– Inactive then forced output state (after a deadtime) – Outputs disabled if OSSI = 0 (control taken over by GPIO logic)	ON after deadtime insertion	OFF
Inactive	Active	Inactive	OFF	OFF

Figure 210 gives an example of tim_ocx and tim_ocxn output behavior in case of active signals on tim_brk and tim_brk2 inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIMx_CCER register).

Figure 210. PWM output state following tim_brk and tim_brk2 assertion (OSSI = 1)

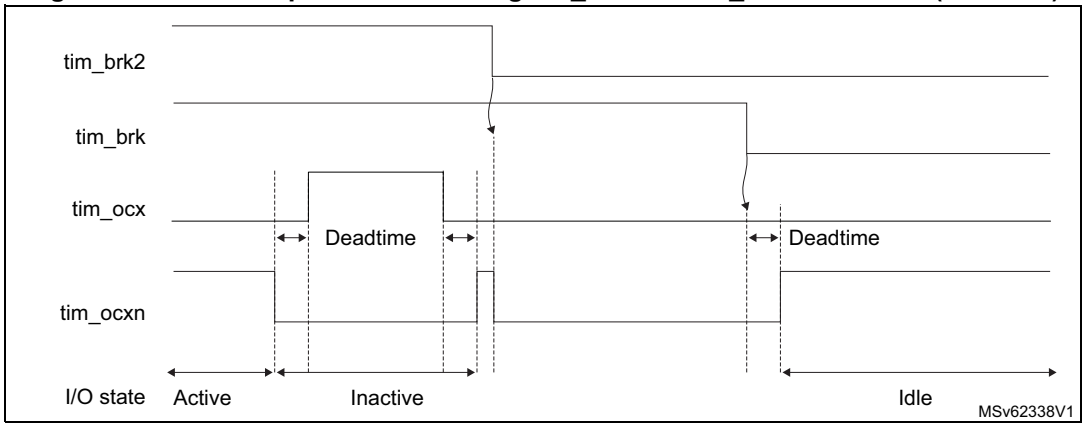
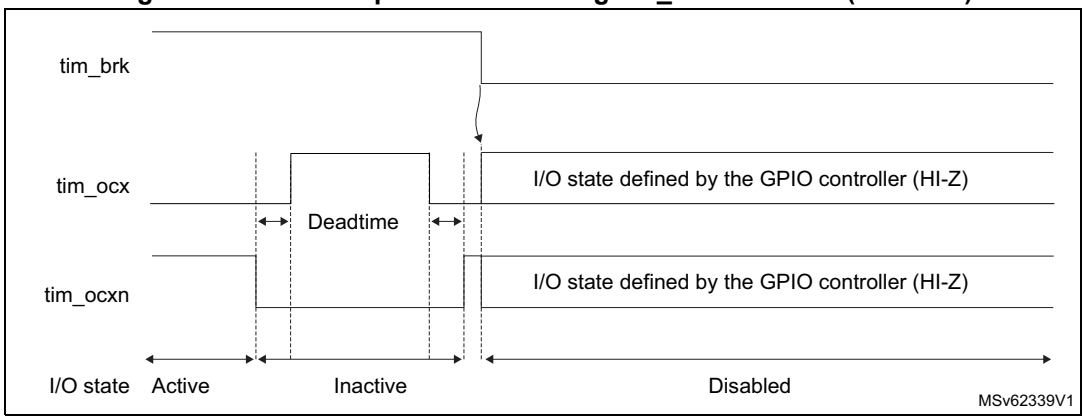


Figure 211. PWM output state following tim_brk assertion (OSSI = 0)



26.3.19 Bidirectional break inputs

The TIM1 features bidirectional break I/Os, as represented on *Figure 212*.

This provides support for:

- 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 and tim_brk2 inputs are configured in bidirectional mode using the BKBID and BK2BID bits in the TIMxBDTR register. The BKBID programming bits can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode is available for both the tim_brk and tim_brk2 inputs, and require the I/O to be configured in open-drain mode with active low polarity (using BKINP, BKP, BK2INP and BK2P 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 events (BG and B2G) also cause the break I/O to be forced to 0 to indicate to the external components that the timer is entered in break state. However, this is valid only if the break is enabled (BKE or B2KE = 1). When a software break event is generated with BKE or B2KE = 0), the outputs are put in safe state and the break flag is set, but there is no effect on the TIMx_BKIN and TIMx_BKIN2 I/Os.

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 (BK2DSRM) 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 (BK2DSRM) 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 (BK2DSRM) bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 179*).

Table 179. Break protection disarming conditions

MOE	BKBID (BK2BID)	BKDSRM (BK2DSRM)	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and rearming break circuitry

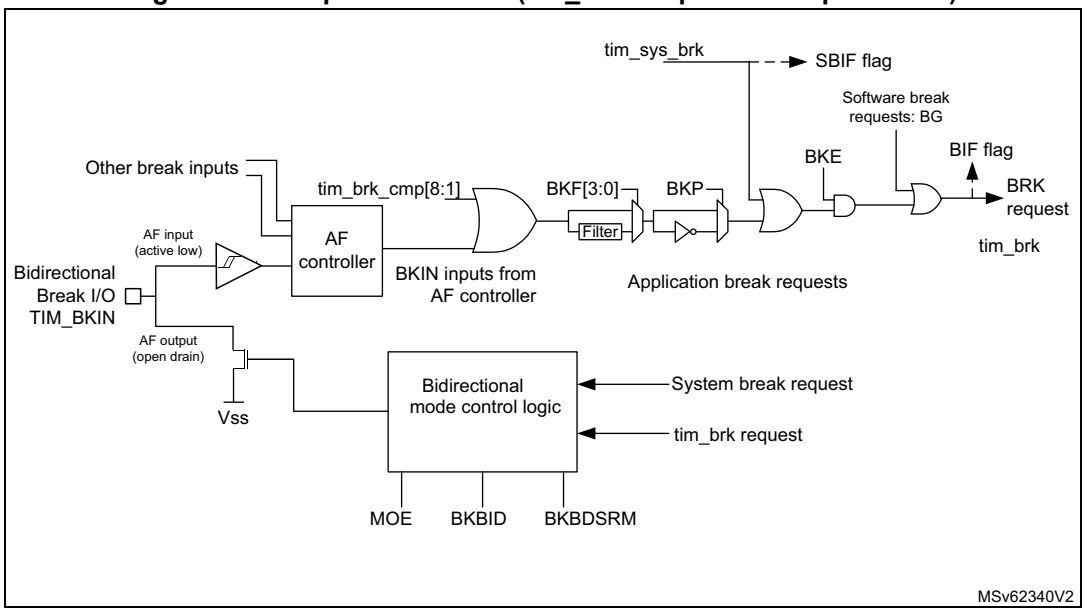
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break (break2) event:

- The BKDSRM (BK2DSRM) 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 (BK2DSRM) 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 212. Output redirection (tim_brk2 request not represented)



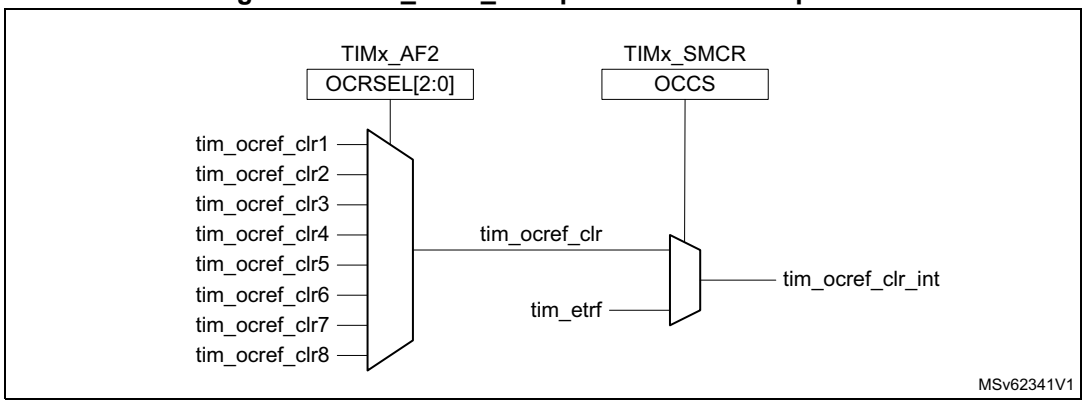
26.3.20 Clearing the tim_ocxref signal on an external event

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. tim_ocref_clr_int input can be selected between the tim_ocref_clr input and tim_etrf (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 inputs, using the OCRSEL[2:0] bitfield in the TIMx_AF2 register, as shown on the *Figure 213* below. Refer to *Section 26.3.2: TIM1 pins and internal signals* for a list of sources available in the product.

Figure 213. tim_ocref_clr input selection multiplexer

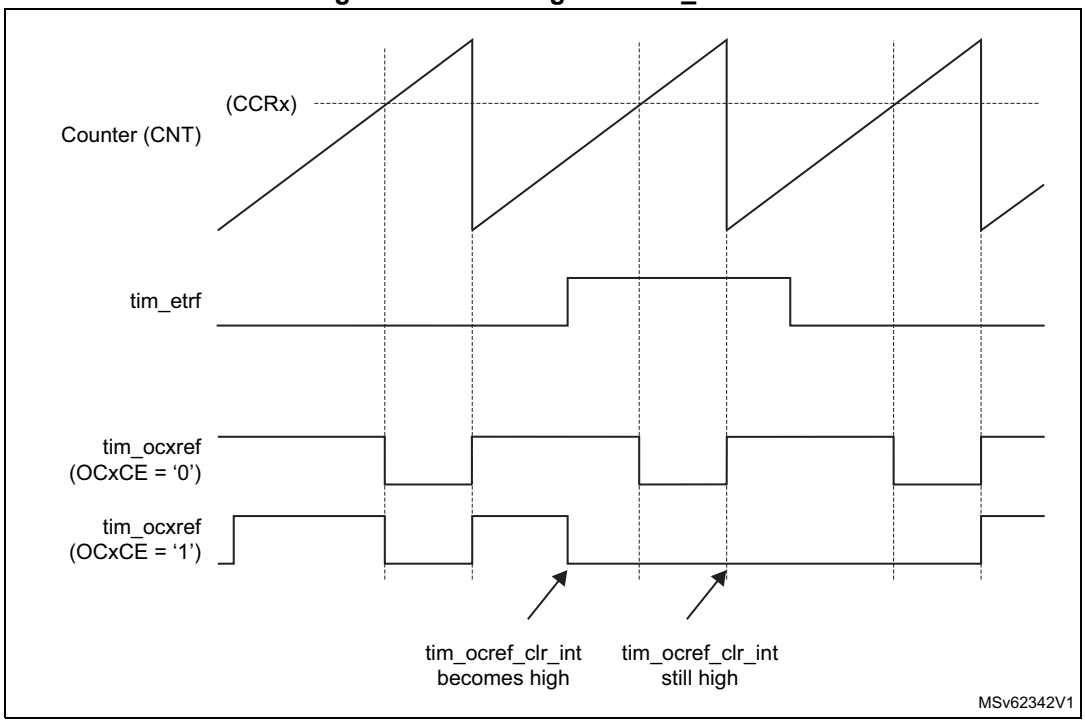


When tim_etrf is chosen, tim_etr_in must be configured as follows:

1. The external trigger prescaler must be kept off: bits ETPS[1:0] of the TIMx_SMCR register set to 00.
2. The external clock mode 2 must be disabled: bit ECE of the TIMx_SMCR register set to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to application needs (as per polarity of the source connected to the trigger and eventual need to remove noise using the filter).

Figure 214 shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

Figure 214. Clearing TIMx tim_ocxref



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), then tim_ocxref is enabled again at the next counter overflow.

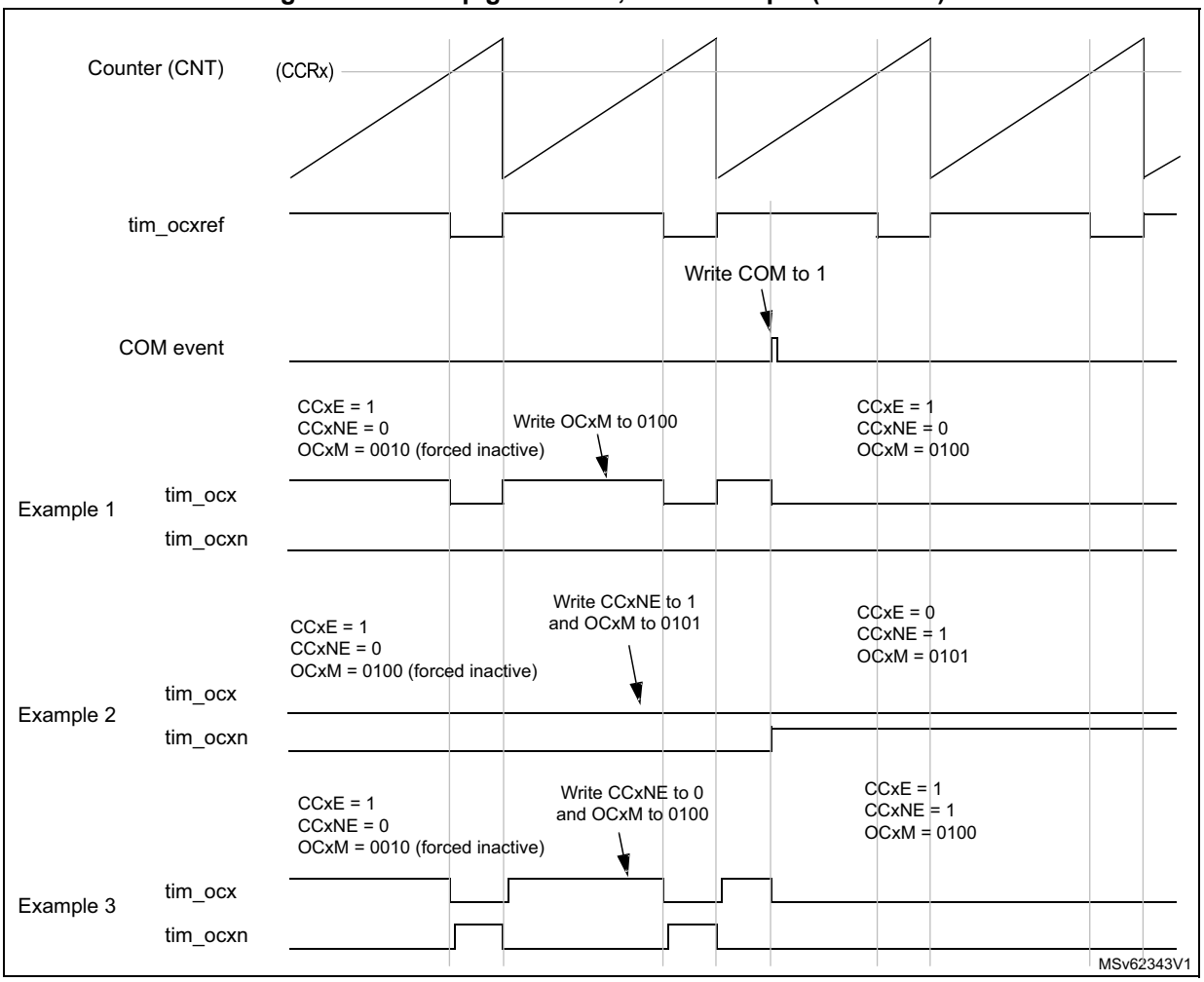
26.3.21 6-step PWM generation

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).

Figure 215 describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in three different examples of programmed configurations.

Figure 215. 6-step generation, COM example (OSSR = 1)



26.3.22 One-pulse mode

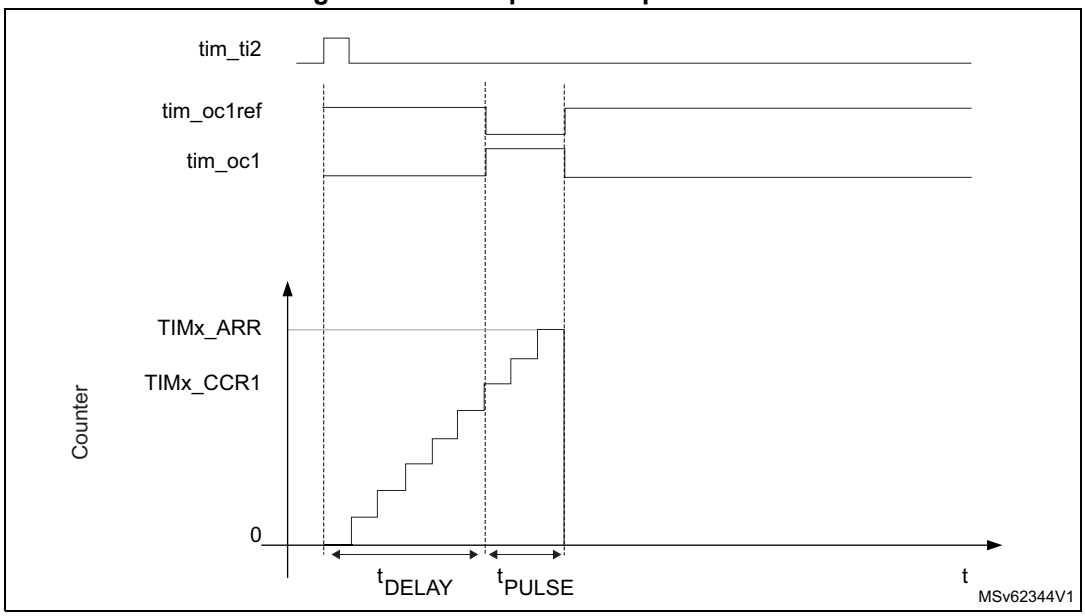
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

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:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 216. Example of one pulse mode.



In the following example, 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:

- Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIMx_CCMR1 register.
- tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
- Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
- 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 auto-reload value. This is achieved by enabling PWM mode 2 (OC1M = 111 in TIMx_CCMR1). 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 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). 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 that can be achieved.

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.

26.3.23 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 nonretriggerable one-pulse mode described in *Section 26.3.22*:

- 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.

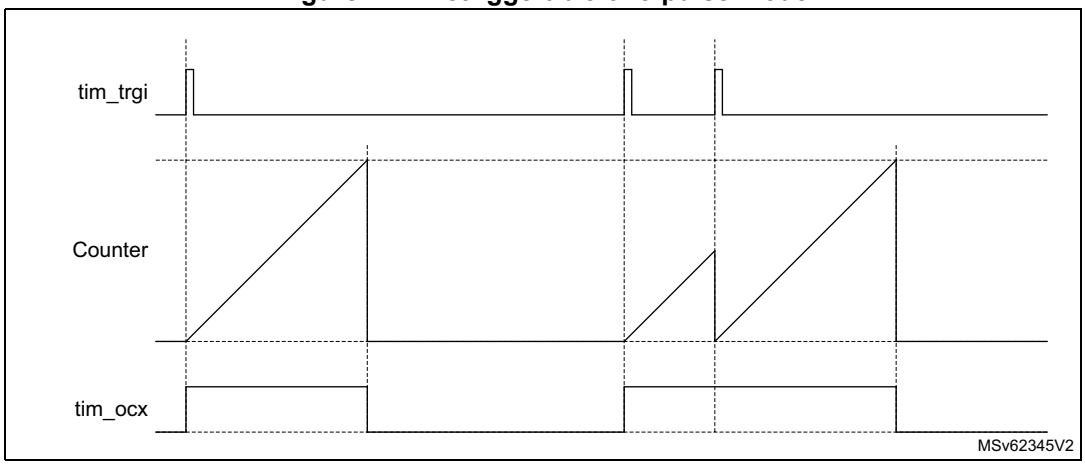
The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for retriggerable OPM mode 1 or 2.

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: The OCxM[3:0] and SMS[3:0] bitfields are split into two parts for compatibility reasons, the most significant bit are 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 217. Retriggerable one-pulse mode



26.3.24 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulsewidth 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 218*.

Figure 218. Pulse generator circuitry

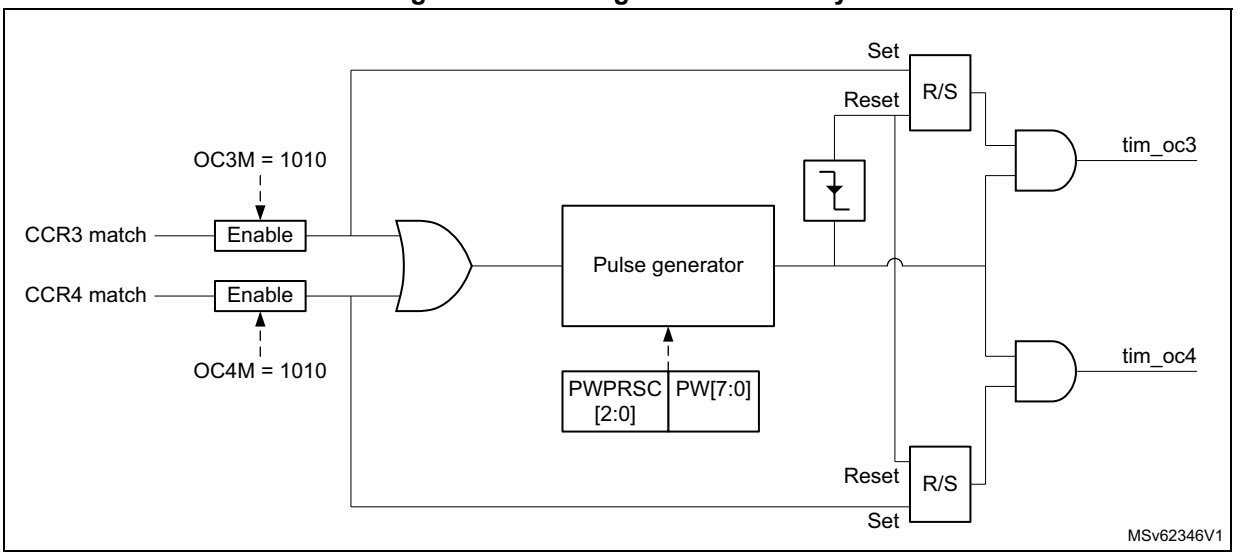
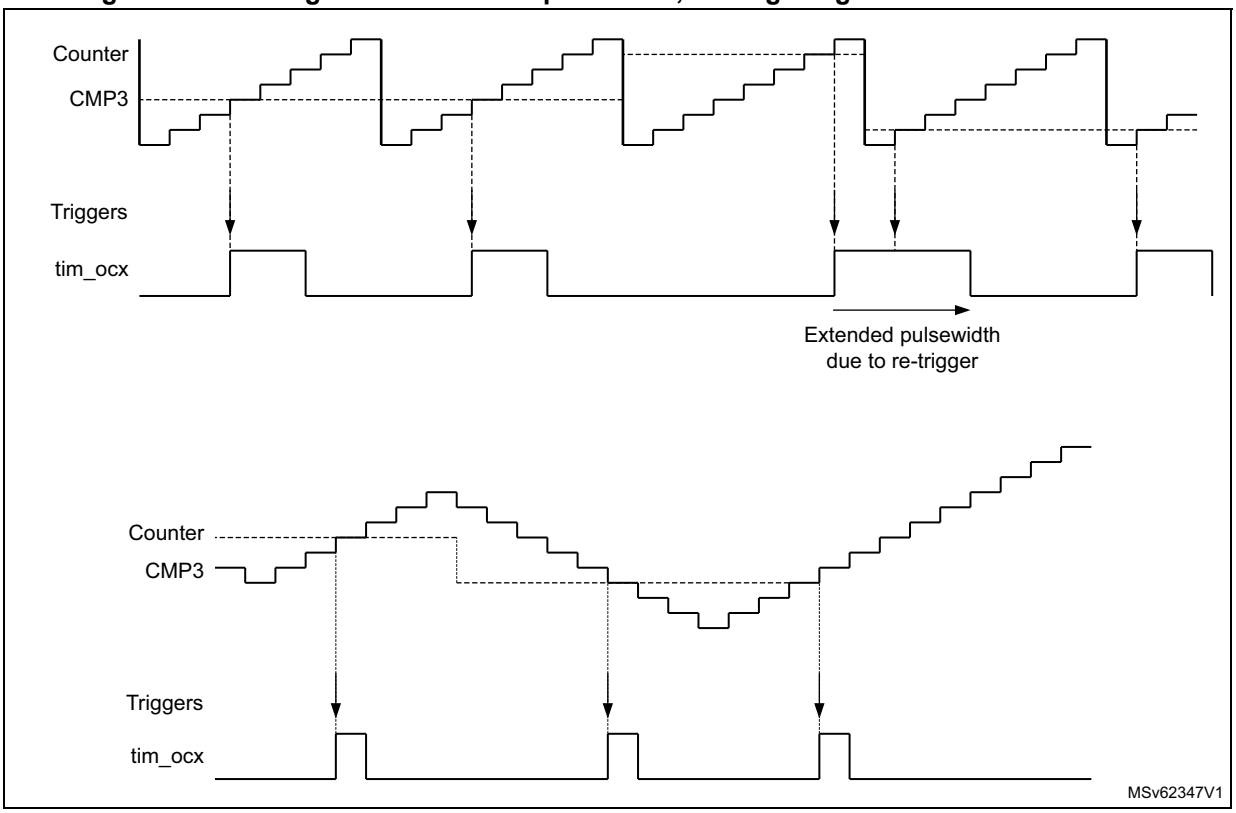


Figure 219 shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 219. 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 pulsewidth is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

$t_{PW} = PW[7:0] \times t_{PWG}$

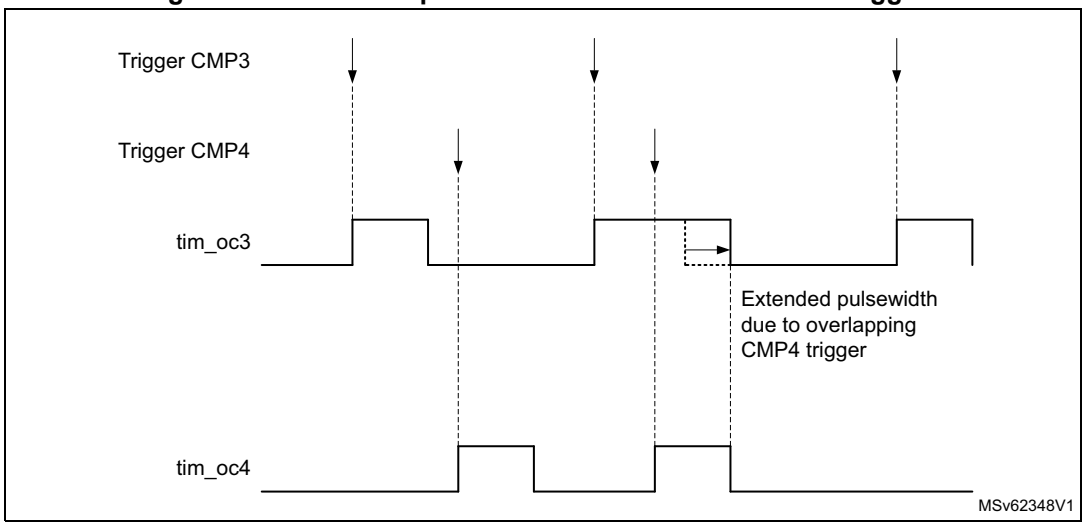
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim\_ker\_ck}$

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 220).

Figure 220. Extended pulsewidth in case of concurrent triggers



26.3.25 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. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs tim_ti1 and tim_ti2 are used to interface to an quadrature encoder. Refer to *Table 180*. 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, repetition counter, 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 correspond 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 180. 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	No count
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 indicate the mechanical zero position, may be connected to the external trigger input and trigger a counter reset.

Figure 221 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 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 = 0 and CC1NP = 0 (TIMx_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P = 0 and CC2NP = 0 (TIMx_CCER register, tim_ti1fp2 noninverted, tim_ti1fp2= tim_ti2).
- SMS = 0011 (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIMx_CR1 register, Counter enabled).

Figure 221. Example of counter operation in encoder interface mode.

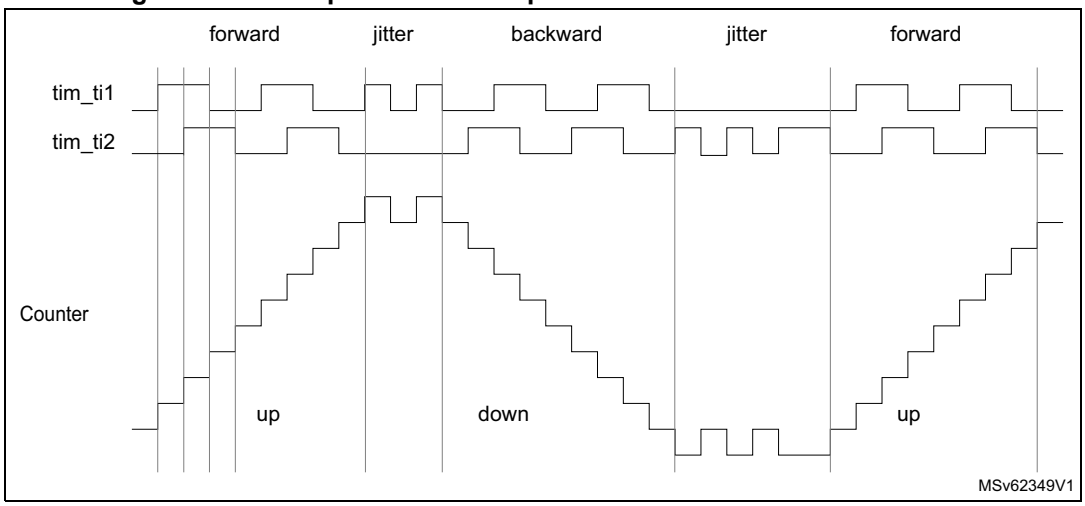


Figure 222 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 222. Example of encoder interface mode with tim_ti1fp1 polarity inverted.

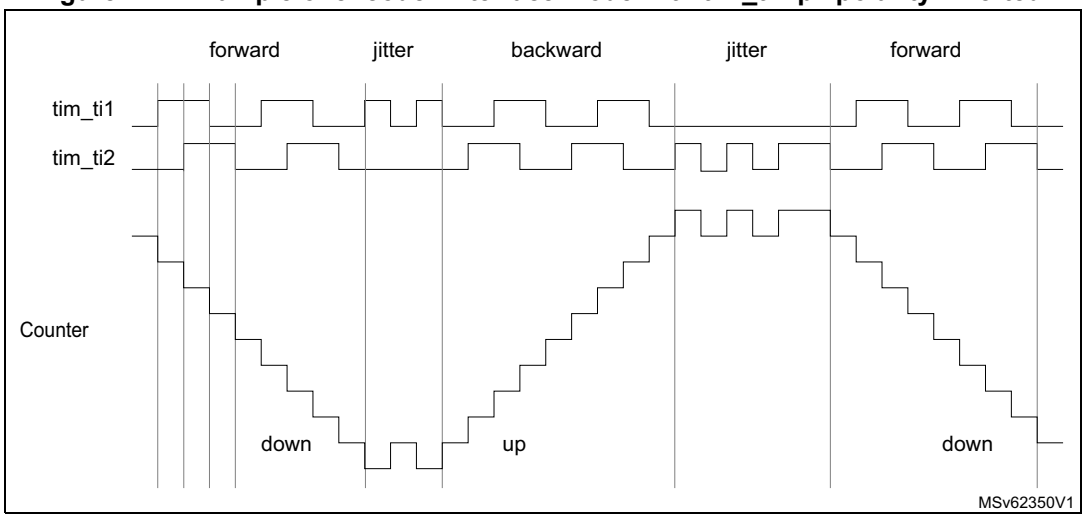
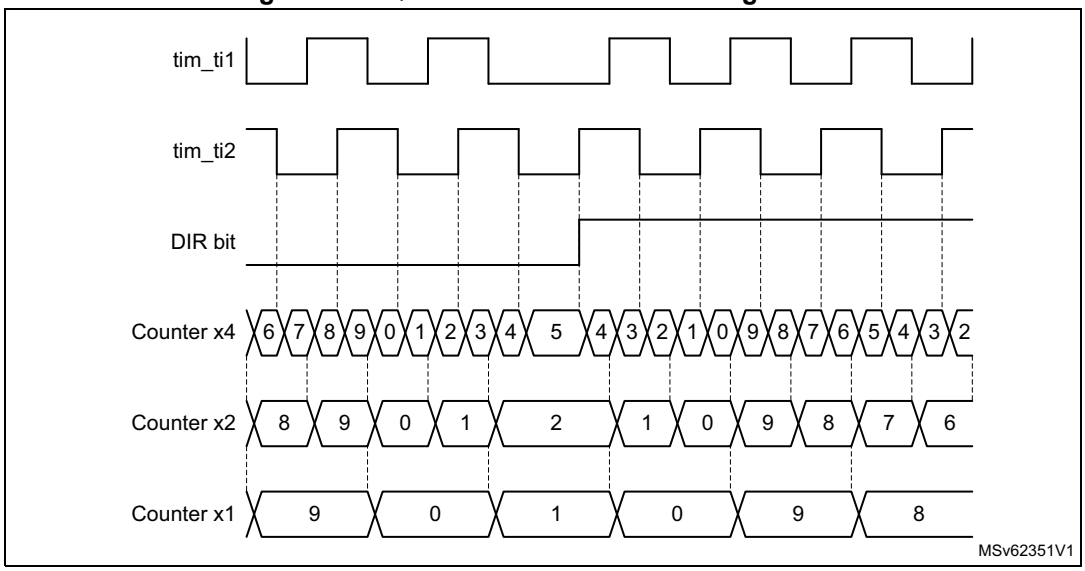


Figure 223 shows the timer counter value during a speed reversal, for various counting modes.

Figure 223. 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 (TIMxCNT[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.

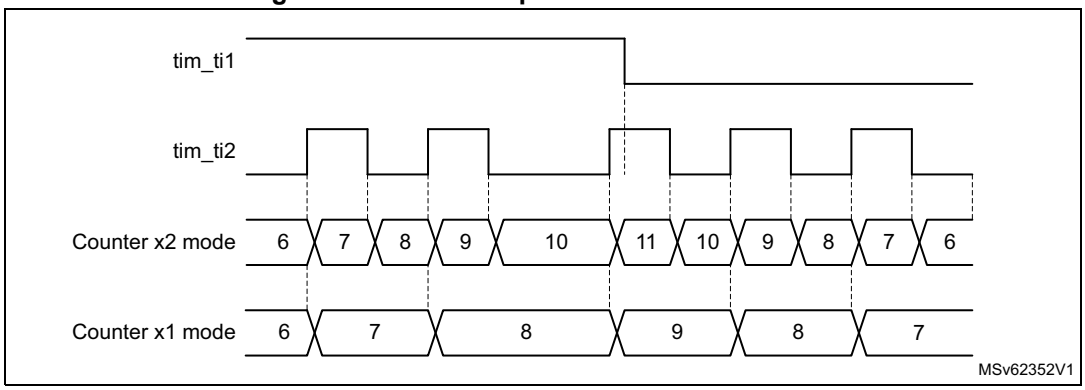
In the clock plus direction mode shown on *Figure 224*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

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 224. Direction plus clock encoder mode



Directional clock encoder mode

In the directional clock mode on *Figure 225*, 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 line. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 225*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 226*).
- 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 225*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 226*).

Figure 225. Directional clock encoder mode (CC1P = CC2P = 0)

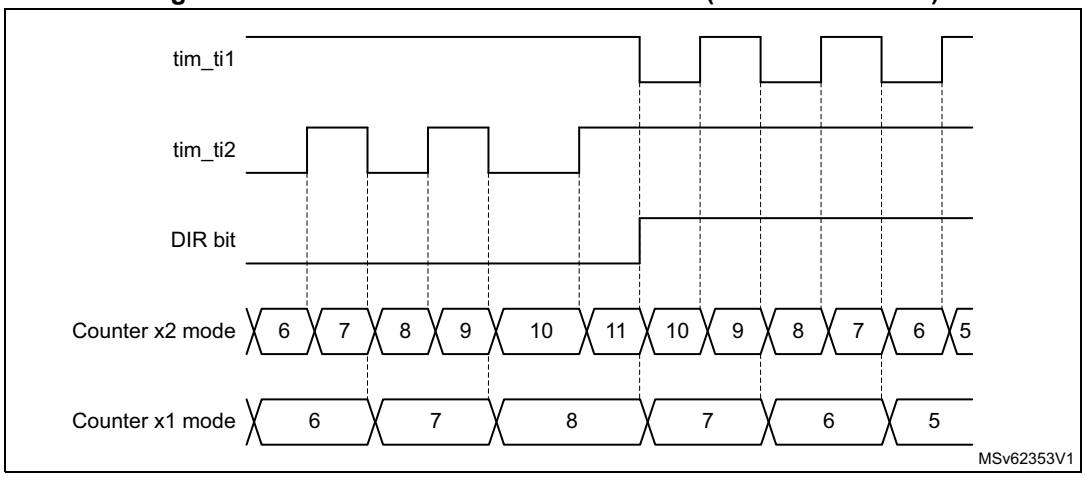


Figure 226. Directional clock encoder mode (CC1P = CC2P = 1)

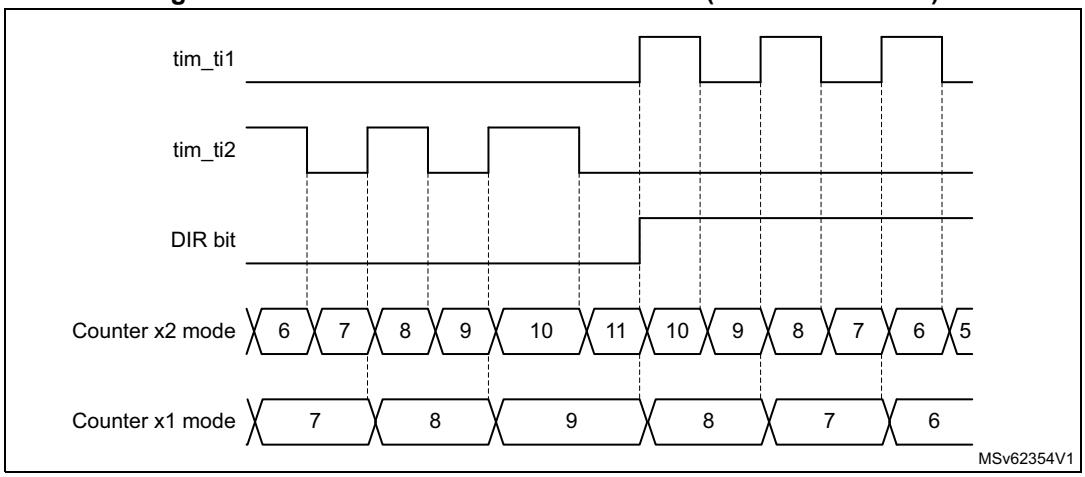


Table 181 here-below details how the directional clock mode operates, for any input transition.

Table 181. 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

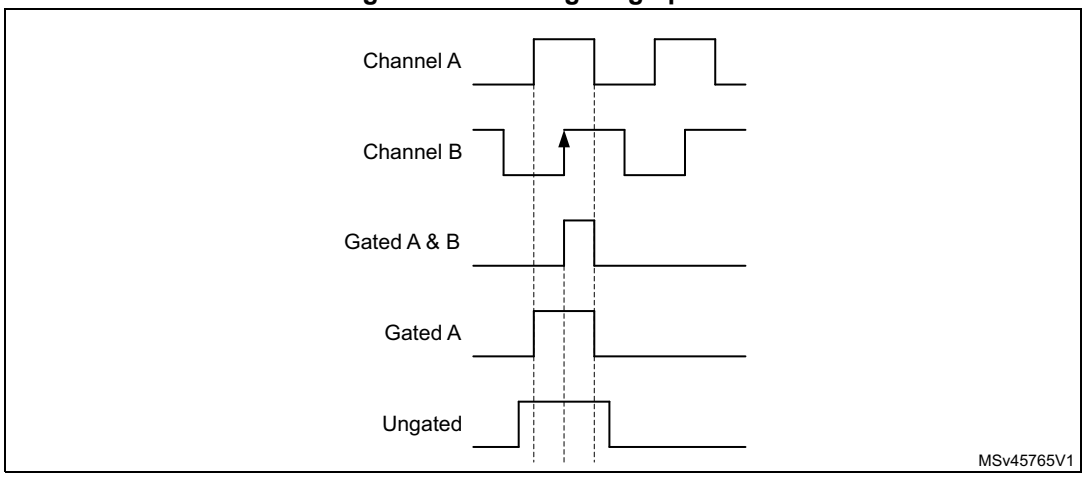
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

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 227*:

- gated with A and B: the pulsewidth is 1/4 of one channel period, aligned with both A and B edges
- gated with A (or gated with B): the pulsewidth is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B)
- ungated: the pulsewidth is up to one channel period, without any alignment to the edges

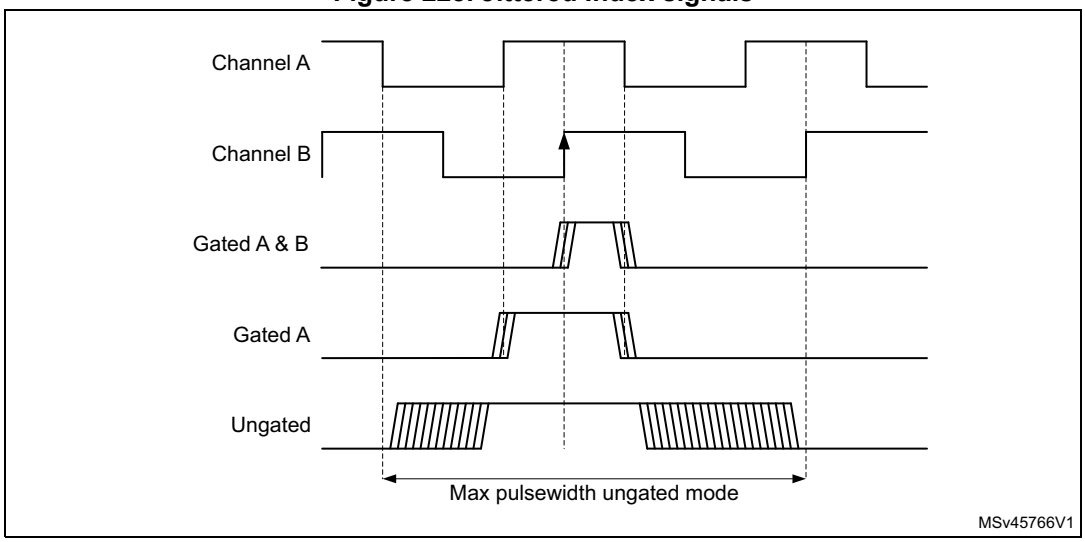
Figure 227. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as show on *Figure 228*.

In ungated mode, the signal must be strictly below two encoder periods. If the pulsewidth is greater or equal to two encoder period, the counter is reset multiple times.

Figure 228. 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 229* shows at which position is the index generated, for a simplistic example (an encoder providing four edges par mechanical rotation).

Figure 229. Index generation for IPOS[1:0] = 11

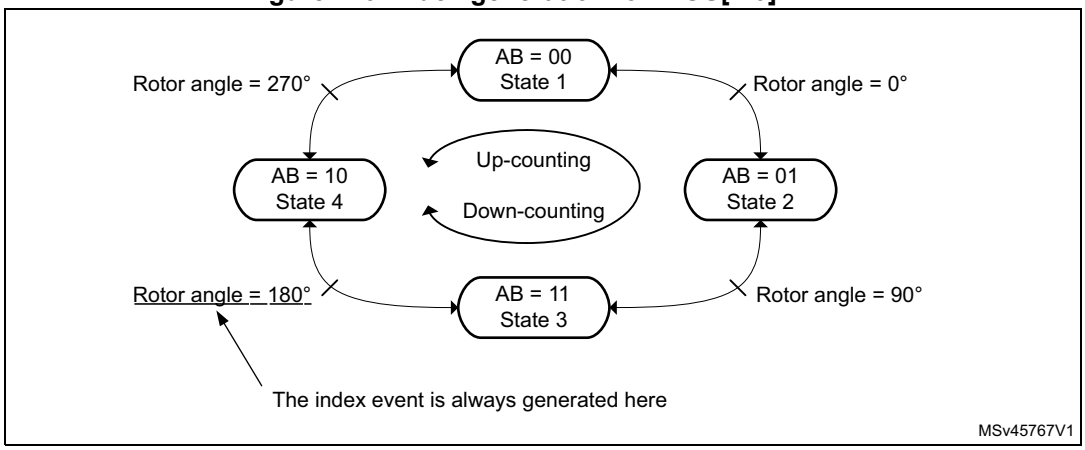


Figure 230 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- 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 230. Counter reading with index gated on channel A (IPOS[1:0] = 11)

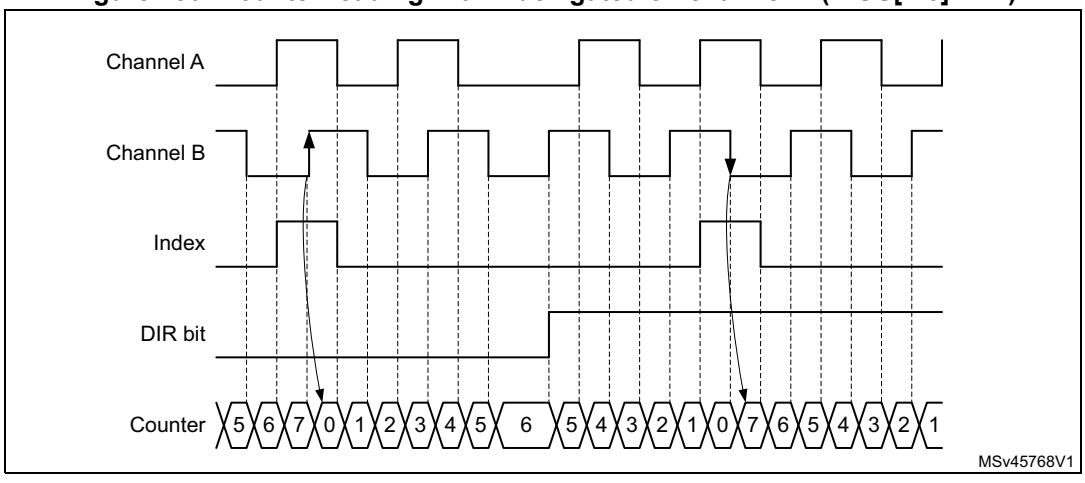


Figure 231. presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 231. Counter reading with index ungated (IPOS[1:0] = 00)

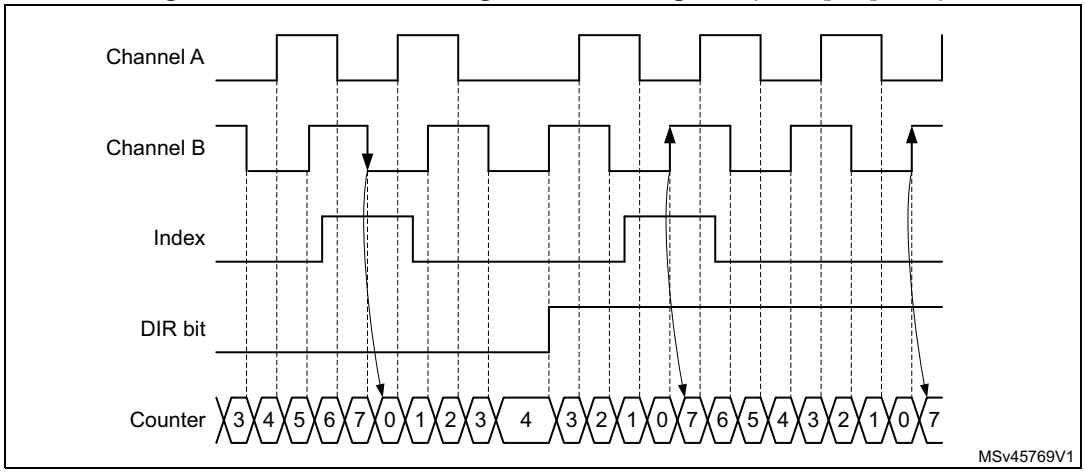


Figure 232. shows how the 'gated on A & B' mode is handled, for various pulse alignment scenario. The arrows are indicating on which transition is the index event generated.

Figure 232. Counter reading with index gated on channel A and B

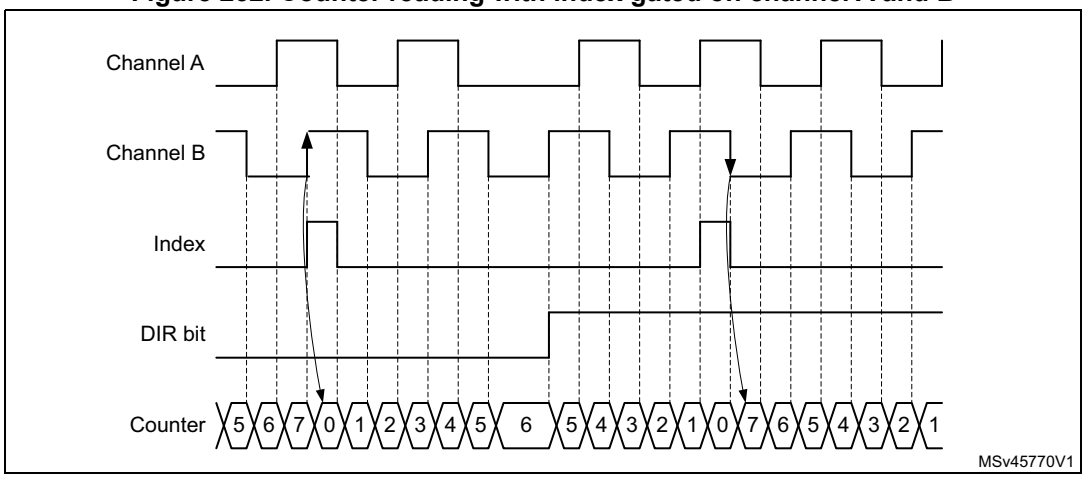


Figure 233 and *Figure 234* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 233. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

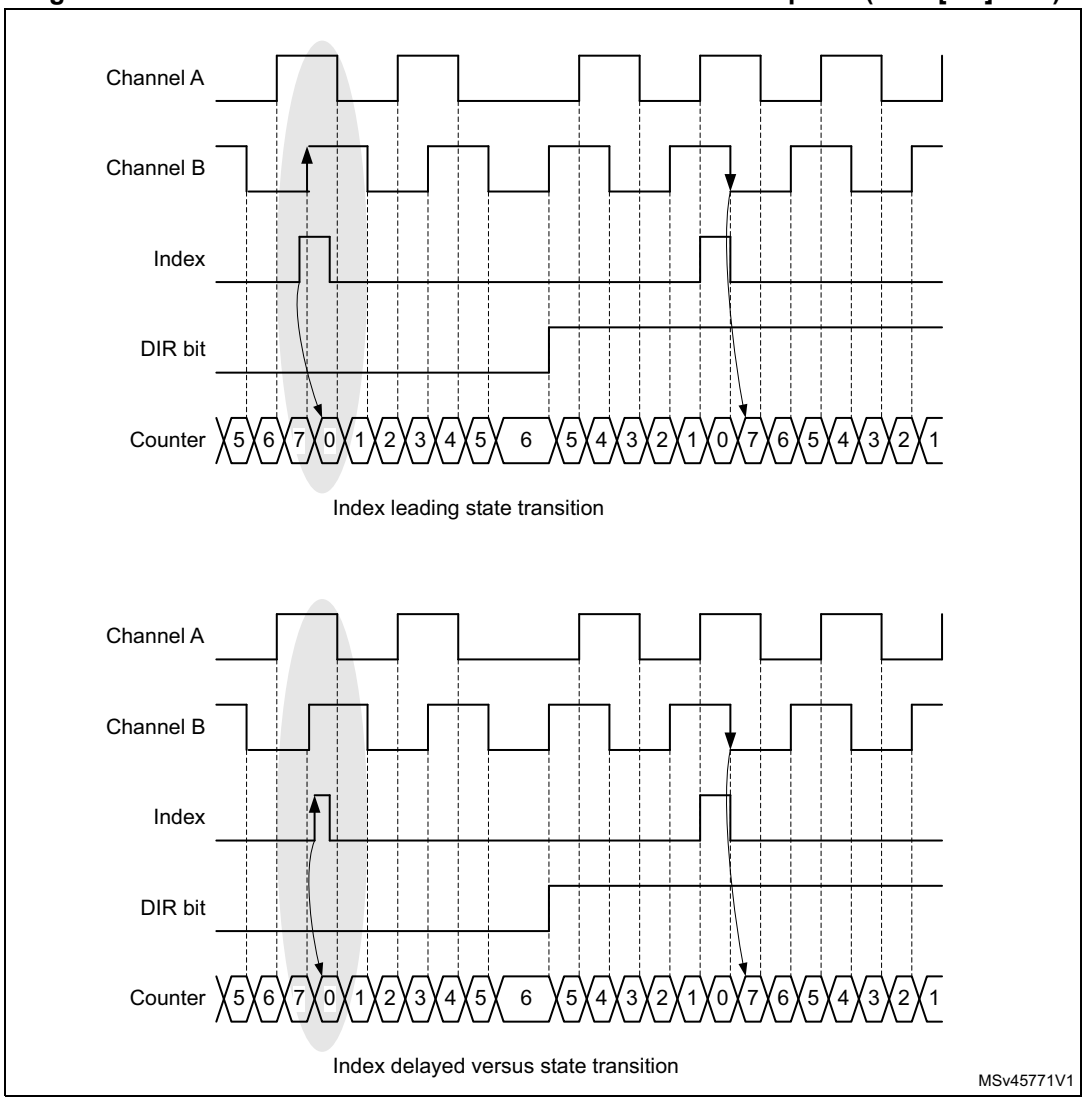


Figure 234. Counter reset Narrow index pulse (closer view, ARR = 0x07)

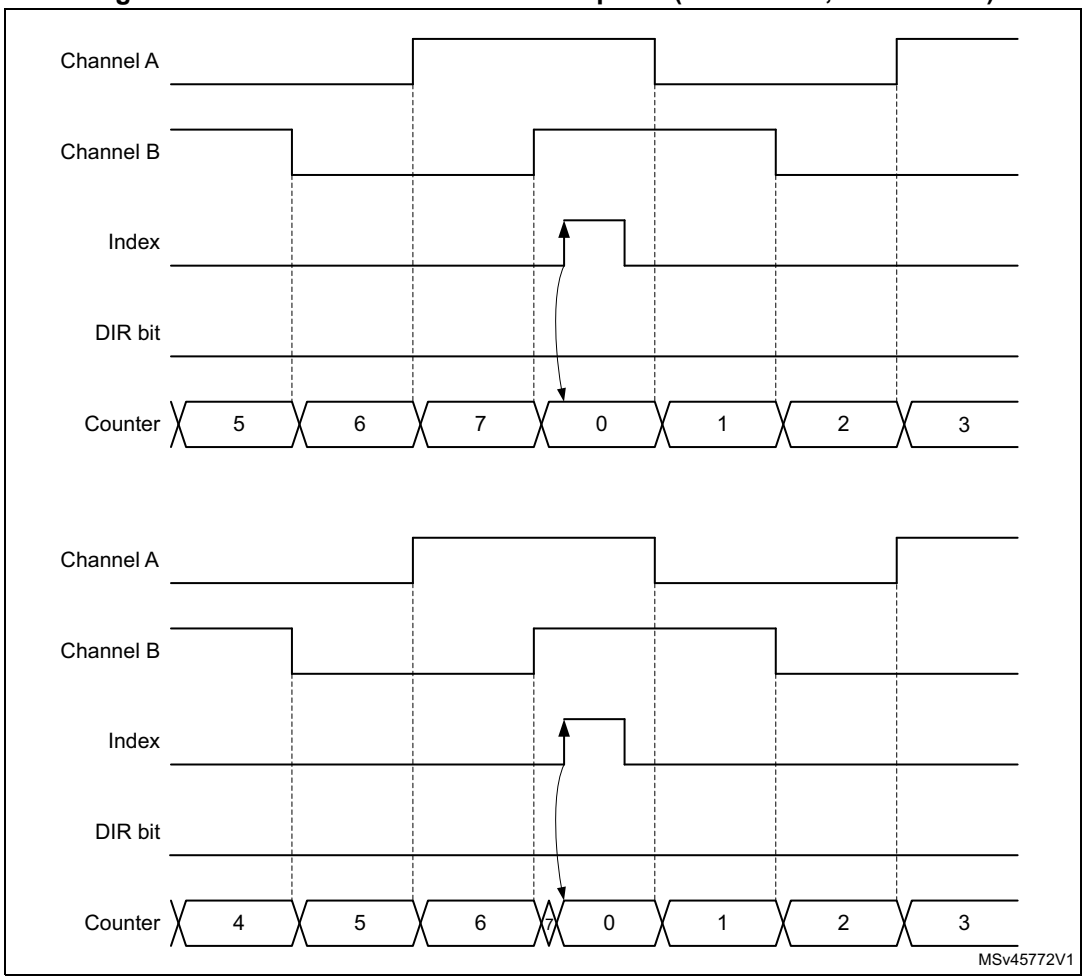
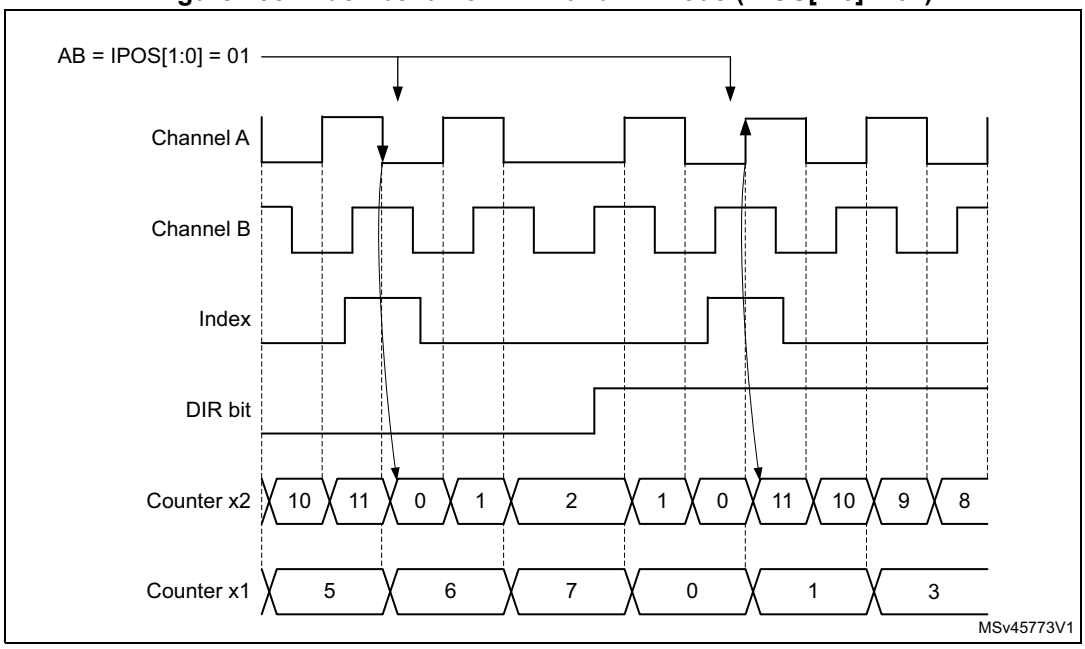


Figure 235 shows how the index is managed in x1 and x2 modes.

Figure 235. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)

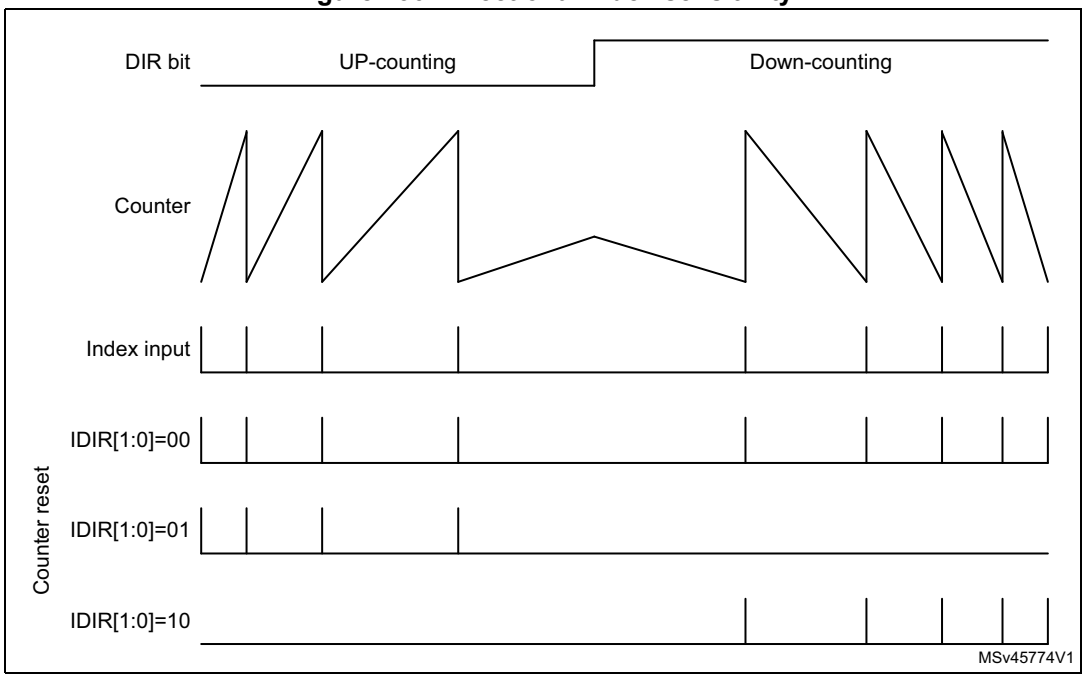


Directional index sensitivity

The IDIR[1:0] bitfield in the TIMx_ECR register allows the index to be active only in a selected counting direction.

Figure 236 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

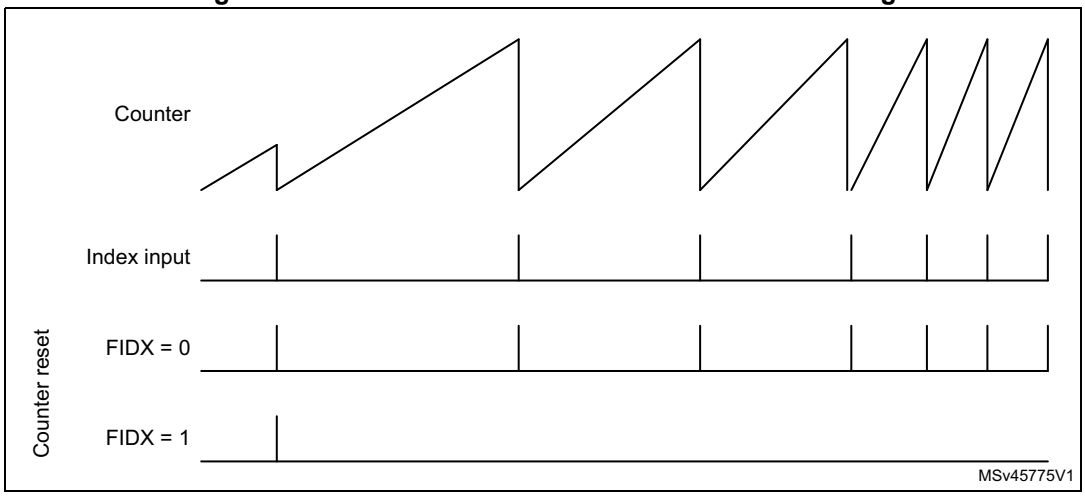
Figure 236. Directional index sensitivity



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 237*. 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 237. Counter reset as function of FIDX bit setting



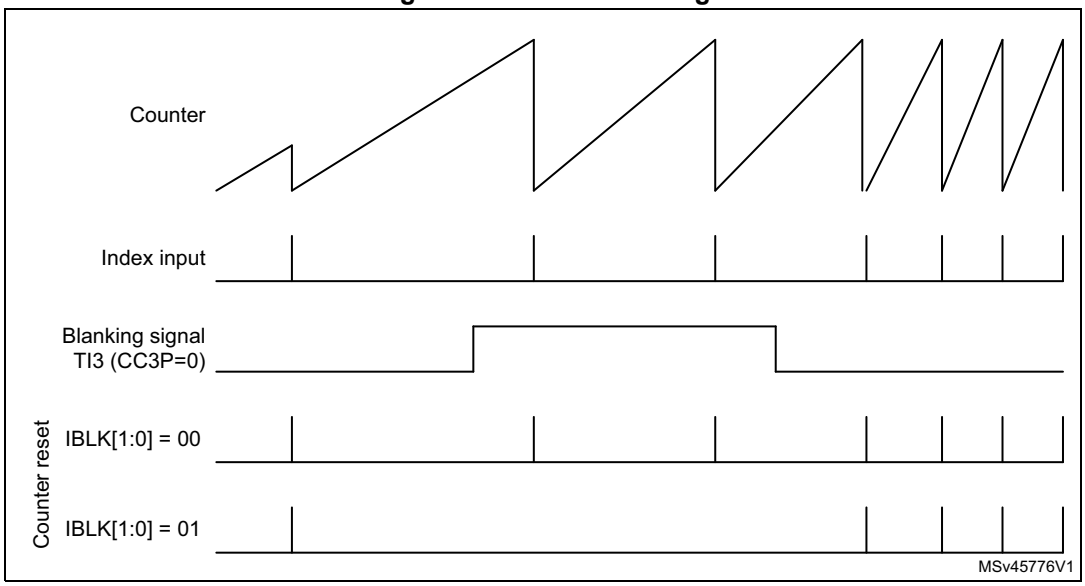
Index blanking

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 the *Figure 238* below.

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 238. Index blanking



Index management in nonquadrature mode

Figure 239 and *Figure 240* 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 239. Index behavior in clock + direction mode, IPOS[0] = 1

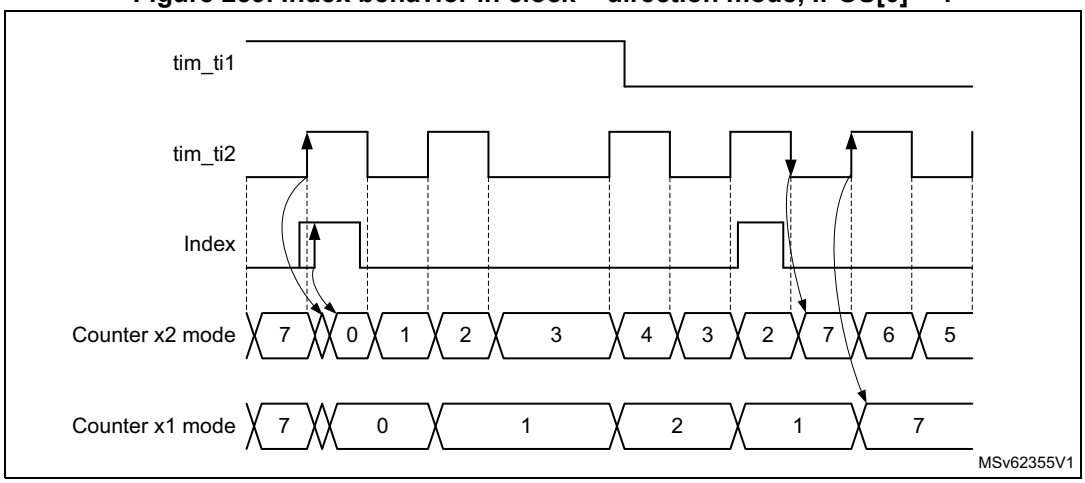
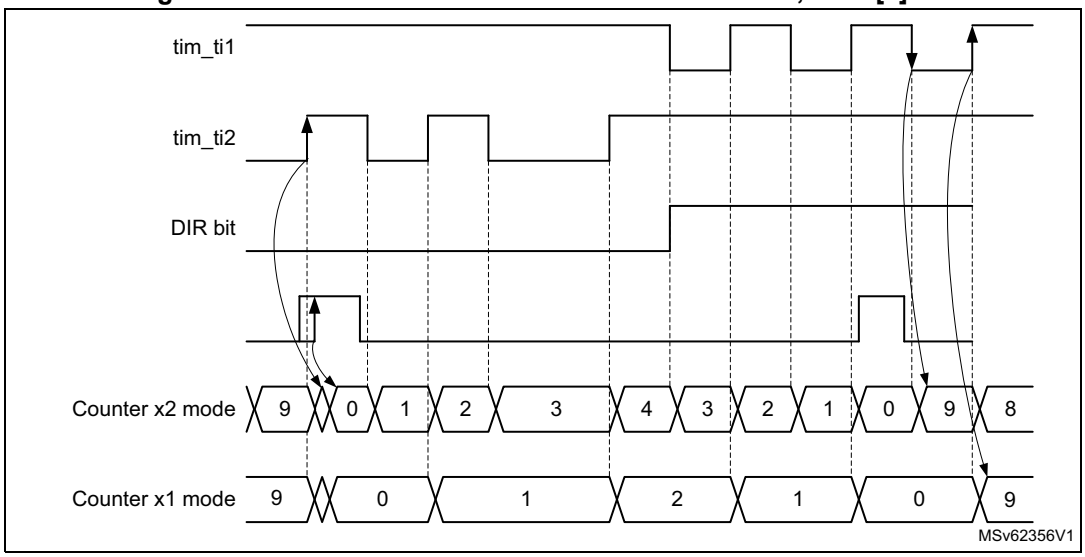


Figure 240. Index behavior in directional clock mode, IPOS[0] = 1

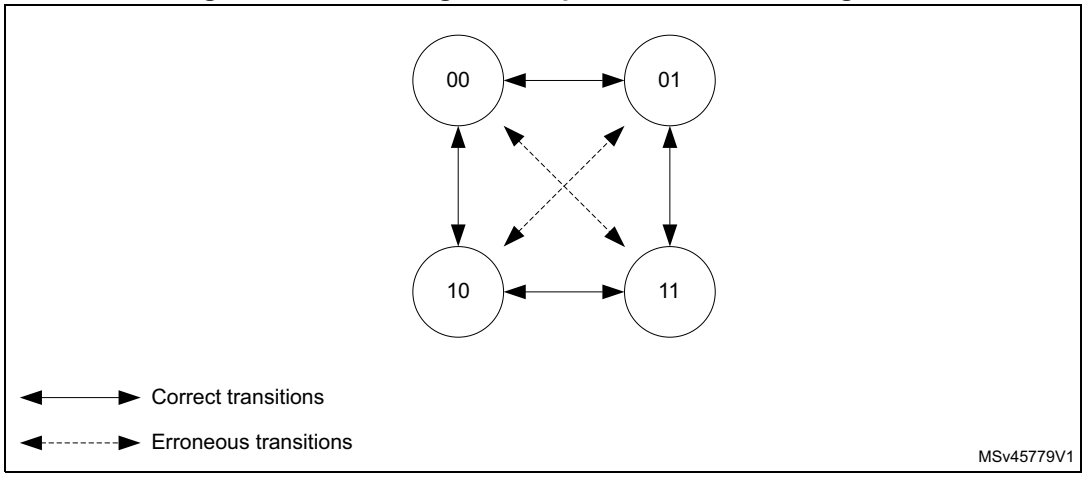


Encoder error management

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 241.*. 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 241. State diagram for quadrature encoded signals



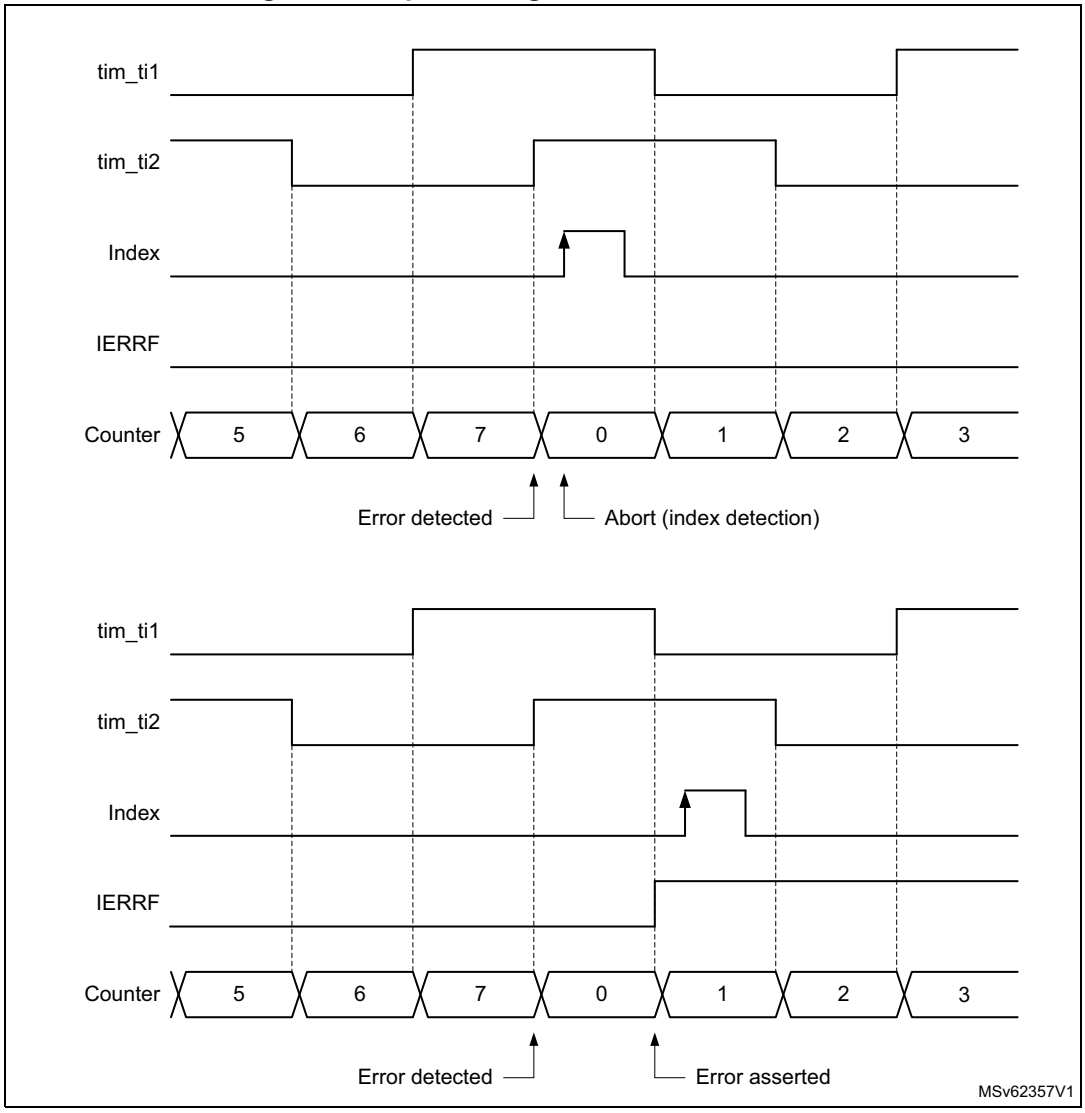
For encoder having an index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIMx_ARR to 0 or decremented from 0 to TIMxARR 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.

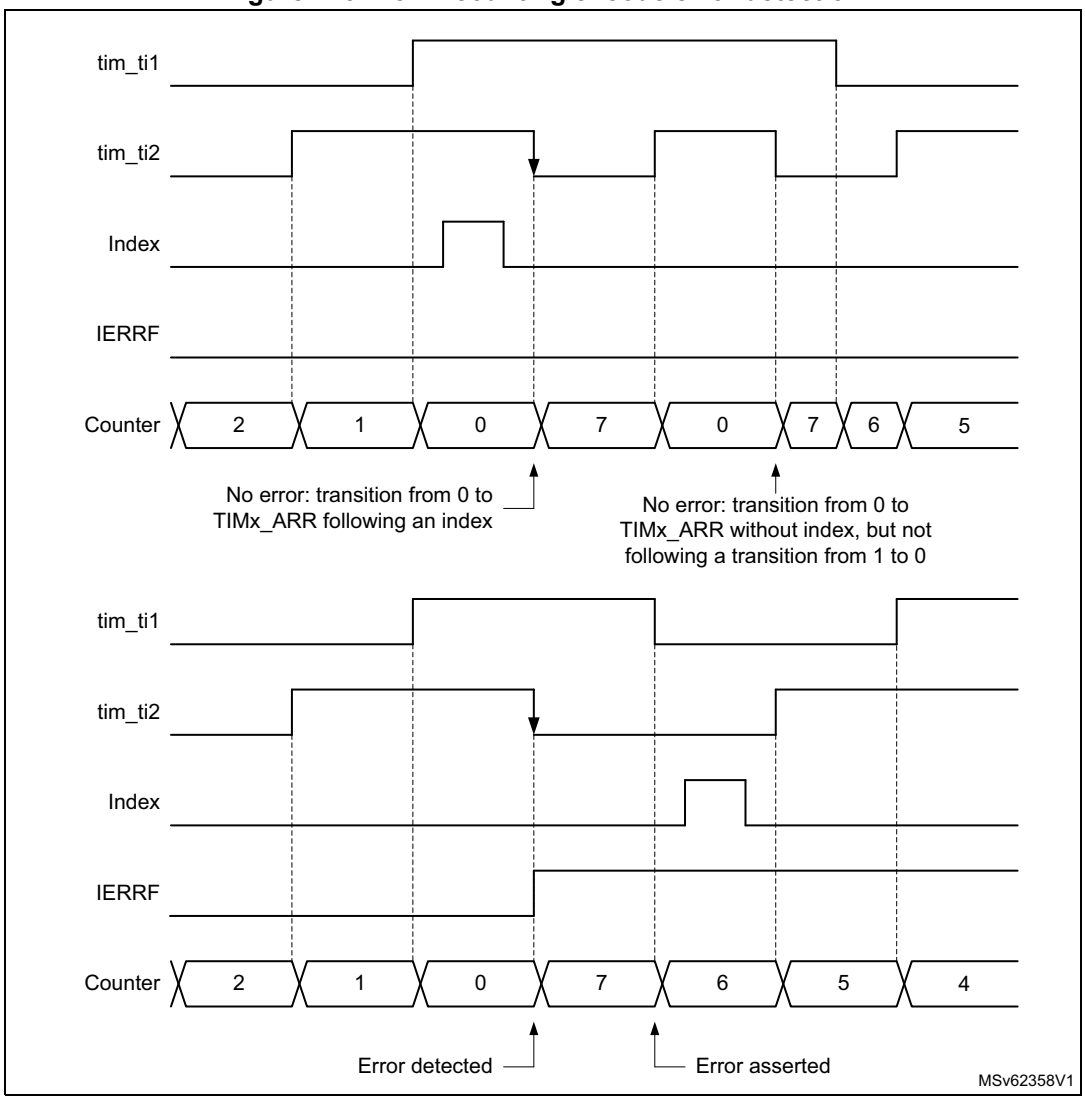
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is cope with narrow index pulses in gated A and B mode, as shown on *Figure 242*.

Figure 242. 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 243*, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

Figure 243. 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.

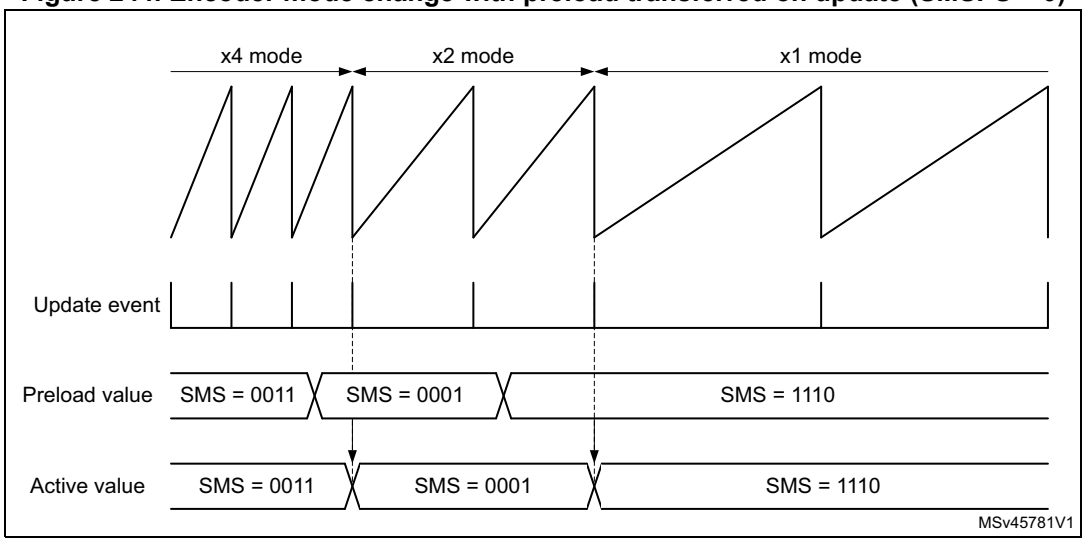
Slave mode selection preload for run-time encoder mode update

It may 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 244*.

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 upcounting, and underflows when downcounting.
- SMSPS = 1: the transfer is triggered by the index event.

Figure 244. 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.

26.3.26 Direction bit output

Its is possible to output a direction signal out of the timer, on the tim_oc3n 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.

26.3.27 UIF bit remapping

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 (TIMxCNT[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. 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).

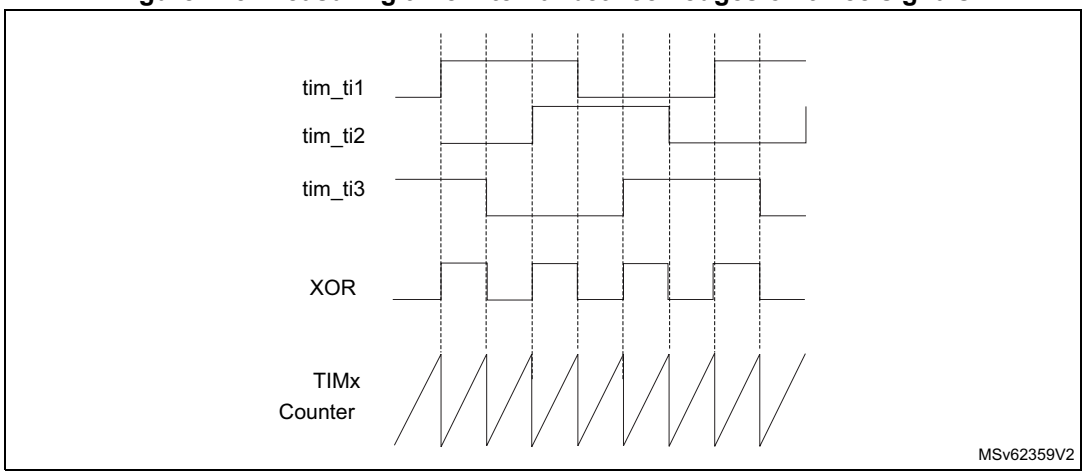
There is no latency between the UIF and UIFCPY flags assertion.

26.3.28 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. It is convenient to measure the interval between edges on two input signals, as per *Figure 245*.

Figure 245. Measuring time interval between edges on three signals



26.3.29 Interfacing with Hall sensors

This is done using the advanced-control timers to generate PWM signals to drive the motor and another timer TIMx referred to as "interfacing timer" in *Figure 246*. The "interfacing timer" captures the three timer input pins (tim_ti1, tim_ti2 and tim_ti3) connected through a XOR to the tim_ti1 input channel (selected by setting the TI1S bit in the TIMx_CR2 register).

The slave mode controller is configured in reset mode; the slave input is tim_ti1f_ed. Thus, each time one of the three inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the "interfacing timer", capture/compare channel 1 is configured in capture mode, capture signal is tim_trc (See *Figure 188*). The captured value, which corresponds to the time elapsed between two changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (by triggering a COM event). The advanced-control timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer through the tim_trgo output.

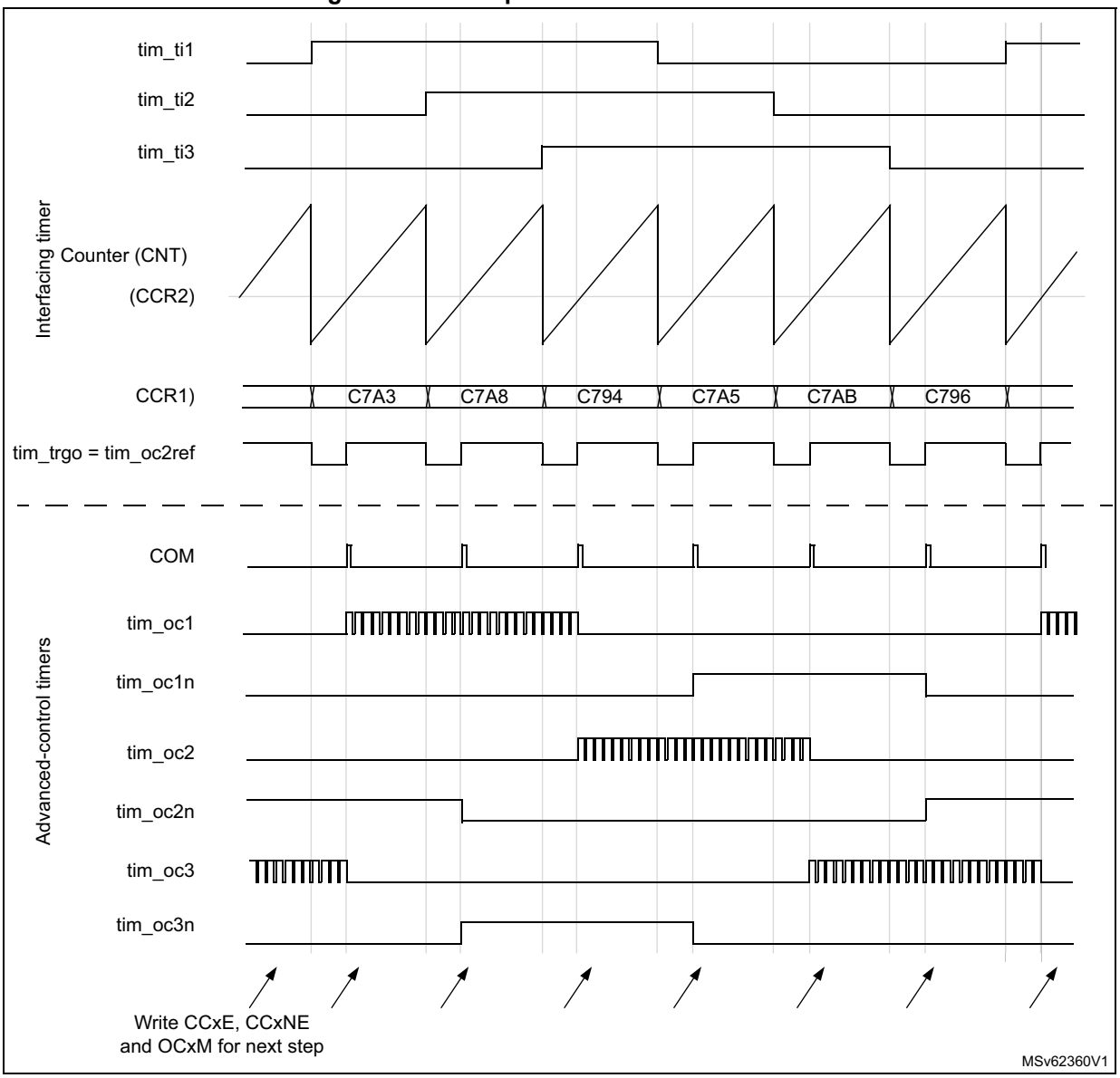
In this example the user wants to change the PWM configuration of the advanced-control timer after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure three timer inputs ORed to the tim_ti1 input channel by writing the TI1S bit in the TIMx_CR2 register to 1.
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the tim_ti1 change. Set the prescaler to get a maximum counter period longer than the time between two changes on the sensors.
- Program the channel 1 in capture mode (tim_trc selected): write the CC1S bits in the TIMx_CCMR1 register to 01. The digital filter can also be programmed if needed.
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to 111 and the CC2S bits to 00 in the TIMx_CCMR1 register.
- Select tim_oc2ref as trigger output on tim_trgo: write the MMS bits in the TIMx_CR2 register to 101.

In the advanced-control timer, the right tim_itrx input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC = 1 in the TIMx_CR2 register) and the COM event is controlled by the trigger input (CCUS = 1 in the TIMx_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of tim_oc2ref).

Figure 246 describes this example.

Figure 246. Example of Hall sensor interface



26.3.30 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 27.4.23: Timer synchronization* for details. They can be synchronized 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.

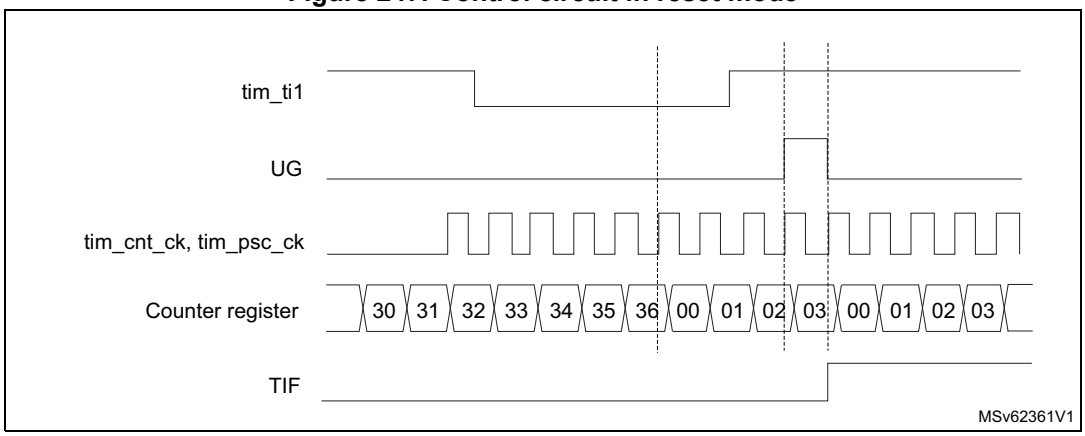
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

- 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).
- 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.
- 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 247. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

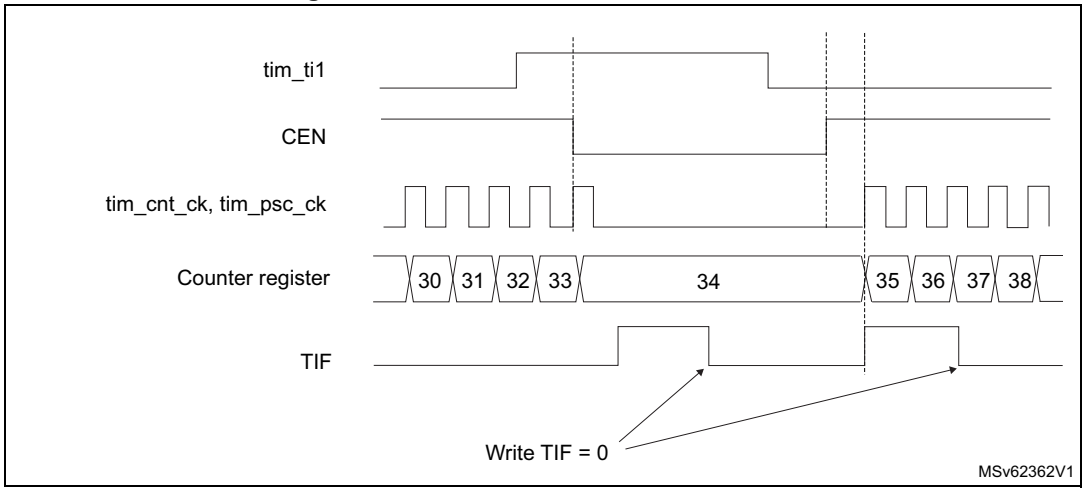
In the following example, the upcounter counts only when tim_ti1 input is low:

- 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).
- 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.
- 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 248. Control circuit in Gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

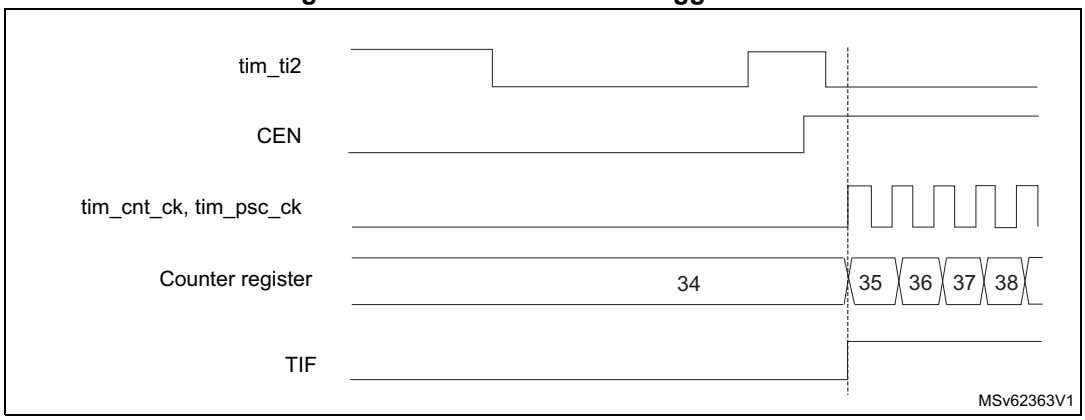
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

- 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. The CC2S bits are configured to select 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).
- 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 249. Control circuit in trigger mode



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 (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.

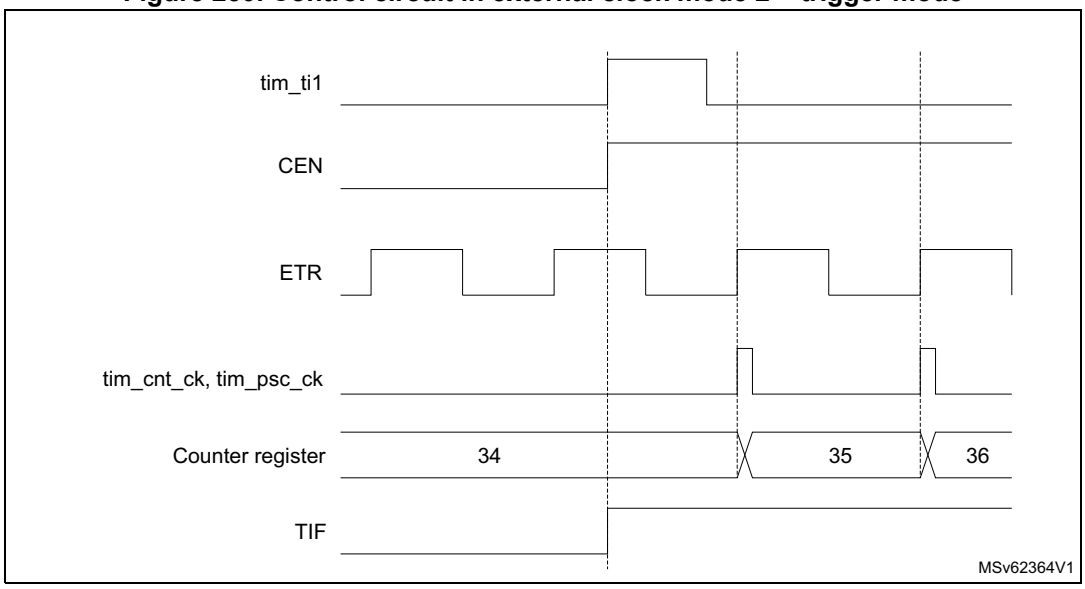
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

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 250. Control circuit in external clock mode 2 + trigger mode



Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo or the tim_trgo2 signals 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.

26.3.31 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

The triggers are issued on the tim_trgo2 internal line which is redirected to the ADC. There is a total of 16 possible events, which can be selected using the MMS2[3:0] bits in the TIMx_CR2 register.

An example of an application for 3-phase motor drives is given in *Figure 203*.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals 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.

The clock of the 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 timer.

26.3.32 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 register define the DMA 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

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 26.6.29: TIM1 DMA control register (TIM1_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 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.

26.3.33 TIM1 DMA requests

The TIM1 can generate a DMA request, as shown in the table below.

Table 182. 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_com_dma	Commutation (COM)	COMDE
tim_trgi_dma	Trigger	TDE

26.3.34 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stop, depending on DBG_TIMx_STOP configuration bit in DBG module.

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), typically to force a Hi-Z.

For more details, refer to section Debug support (DBG).

26.4 TIM1 low-power modes

Table 183. Effect of low-power modes on TIM1

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

26.5 TIM1 interrupts

The TIM1 can generate multiple interrupts, as shown in *Table 184*.

Table 184. 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_COM	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	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_BRK	Break	BIF	BIE	write 0 in BIF	Yes	No
	Break2	B2IF		write 0 in B2IF	Yes	No
	System Break	SBIF		write 0 in SBIF	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

26.6 TIM1 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).

26.6.1 TIM1 control register 1 (TIM1_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 the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_etr_in, 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, do not program this value

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.

26.6.2 TIM1 control register 2 (TIM1_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.	MMS[3]	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5
						rw		rw	rw	rw	rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
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 24 Reserved, must be kept at reset value.

Bits 23:20 **MMS2[3:0]**: Master mode selection 2

These bits allow the information to be sent to ADC for synchronization (tim_trgo2) to be selected. The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on tim_trgo2 is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo2). 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 the 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_trgo2, 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_trgo2). For instance, a master timer can then be used as a prescaler for a slave timer.

0011: **Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (tim_trgo2).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo2)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo2)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo2)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo2)

1000: **Compare** - tim_oc5refc signal is used as trigger output (tim_trgo2)

1001: **Compare** - tim_oc6refc signal is used as trigger output (tim_trgo2)

1010: **Compare Pulse** - tim_oc4refc rising or falling edges generate pulses on tim_trgo2

1011: **Compare pulse** - tim_oc6refc rising or falling edges generate pulses on tim_trgo2

1100: **Compare pulse** - tim_oc4refc or tim_oc6refc rising edges generate pulses on tim_trgo2

1101: **Compare pulse** - tim_oc4refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

1110: **Compare pulse** - tim_oc5refc or tim_oc6refc rising edges generate pulses on tim_trgo2

1111: **Compare pulse** - tim_oc5refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

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 19 Reserved, must be kept at reset value.

Bit 18 **OIS6**: Output idle state 6 (tim_oc6 output)
Refer to OIS1 bit

Bit 17 Reserved, must be kept at reset value.

Bit 16 **OIS5**: Output idle state 5 (tim_oc5 output)
Refer to OIS1 bit

Bit 15 **OIS4N**: Output idle state 4 (tim_oc4n output)
Refer to OIS1N bit

Bit 14 **OIS4**: Output idle state 4 (tim_oc4 output)
Refer to OIS1 bit

Bit 13 **OIS3N**: Output idle state 3 (tim_oc3n output)
Refer to OIS1N bit

Bit 12 **OIS3**: Output idle state 3 (tim_oc3n output)
Refer to OIS1 bit

Bit 11 **OIS2N**: Output idle state 2 (tim_oc2n output)
Refer to OIS1N bit

Bit 10 **OIS2**: Output idle state 2 (tim_oc2 output)
Refer to OIS1 bit

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_BDTR 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_BDTR register).

Bit 7 **TI1S**: tim_ti1 selection
0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input
1: 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, 6:4 **MMS[3:0]**: Master mode selection

These bits 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 enable. 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.

Other codes 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

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 an rising edge occurs on tim_trgi

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 a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).

Note: This bit acts only on channels that have a complementary output.

26.6.3 TIM1 slave mode control register (TIM1_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 21:20 **TS[4:3]**: Trigger selection - bit 4:3

Refer to TS[2:0] description - bits 6:4

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

Bit 15 **ETP**: External trigger polarity

This bit selects whether tim_etr_in or tim_$\overline{\text{etr_in}}$ 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 TIMxCLK 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_etr_in frequency divided by 2
10: tim_etr_in frequency divided by 4
11: tim_etr_in 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 6:4 **TS[2:0]:** Trigger selection

This bitfield is combined with TS[4:3] bits.

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 *Table 171: Internal trigger connection* for more details on tim_itrx meaning for each Timer.

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

Bits 16, 2:0 **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: Quadrature encoder mode 1, x2 mode- Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

0010: Quadrature encoder mode 2, x2 mode - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

0011: Quadrature encoder mode 3, x4 mode - 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 TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals 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.

26.6.4 TIM1 DMA/interrupt enable register (TIM1_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	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	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 Change 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 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM DMA request enabled

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 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

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

26.6.5 TIM1 status register (TIM1_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.	CC6IF	CC5IF
								rc_w0	rc_w0	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.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	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	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 19:18 Reserved, must be kept at reset value.

Bit 17 **CC6IF**: Compare 6 interrupt flag
Refer to CC1IF description
Note: Channel 6 can only be configured as output.

Bit 16 **CC5IF**: Compare 5 interrupt flag
Refer to CC1IF description
Note: Channel 5 can only be configured as output.

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

Bit 13 **SBIF**: System break interrupt flag
This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.
This flag must be reset to re-start PWM operation.
0: No break event occurred.
1: An active level has been detected on the system break input. An interrupt is generated if BIE = 1 in the TIMx_DIER register.

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

Bit 8 **B2IF**: Break 2 interrupt flag

This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.

0: No break event occurred.

1: An active level has been detected on the break 2 input. An interrupt is generated if BIE = 1 in the TIMx_DIER register.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break 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. An interrupt is generated if BIE = 1 in the TIMx_DIER register.

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 **COMIF**: COM interrupt flag

This flag is set by hardware on COM event (when capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.

0: No COM event occurred.

1: COM interrupt pending.

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 downcounting 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 or underflow 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.
–When CNT is reinitialized by a trigger event (refer to *Section 26.6.3: TIM1 slave mode control register (TIM1_SMCR)*), if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

26.6.6 TIM1 event generation register (TIM1_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.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
							w	w	w	w	w	w	w	w	w

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **B2G**: Break 2 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 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

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 **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 **COMG**: Capture/compare control update generation
This bit can be set by software, it is automatically cleared by hardware
0: No action
1: CCxE, CCxNE and OCxM bits update (providing CCPC bit is set)
Note: This bit acts only on channels having a complementary output.

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: 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). The counter is cleared if the center-aligned mode is selected or if DIR = 0 (upcounting), else it takes the autoreload value (TIMx_ARR) if DIR = 1 (downcounting).

26.6.7 TIM1 capture/compare mode register 1 (TIM1_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).

26.6.8 TIM1 capture/compare mode register 1 [alternate] (TIM1_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
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	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

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 signal

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf 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 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 upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (tim_oc1ref = 0) as long as TIMx_CNT>TIMx_CCR1 else active (tim_oc1ref = 1).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 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: 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: 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 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.

Note: On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

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, 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 3 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).

26.6.9 TIM1 capture/compare mode register 2 (TIM1_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 3 in input capture mode and channel 4 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).

26.6.10 TIM1 capture/compare mode register 2 [alternate] (TIM1_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 3 in input capture mode and channel 4 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
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE	OC3M[2:0]			OC3 PE	OC3 FE	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 description

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 upcounting, channel 3 is active as long as TIMx_CNT<TIMx_CCR3 else inactive. In downcounting, channel 3 is inactive (tim_oc3ref = 0) as long as TIMx_CNT>TIMx_CCR3 else active (tim_oc3ref = 1).

0111: PWM mode 2 - In upcounting, channel 3 is inactive as long as TIMx_CNT<TIMx_CCR3 else active. In downcounting, 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 TIMxECR.

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).

26.6.11 TIM1 capture/compare enable register (TIM1_CCER)

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.	CC6P	CC6E	Res.	Res.	CC5P	CC5E
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CC6P**: Capture/compare 6 output polarity
Refer to CC1P description

Bit 20 **CC6E**: Capture/compare 6 output enable
Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC5P**: Capture/compare 5 output polarity
Refer to CC1P description

Bit 16 **CC5E**: Capture/compare 5 output enable
Refer to CC1E description

Bit 15 **CC4NP**: Capture/compare 4 complementary output polarity
Refer to CC1NP description

Bit 14 **CC4NE**: Capture/compare 4 complementary output enable
Refer to CC1NE description

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 complementary output polarity
Refer to CC1NP description

Bit 10 **CC3NE**: Capture/compare 3 complementary output enable
Refer to CC1NE description

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 complementary output polarity
Refer to CC1NP description

Bit 6 **CC2NE**: Capture/compare 2 complementary output enable
Refer to CC1NE description

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 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 and tim_ti2fp1.
Refer to CC1P description.

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 (channel configured as output).

Note: On channels having 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.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.

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:the 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).

Note: On channels having 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 185* for details.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1E active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Table 185. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature

Control bits					Output states[1]	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	tim_ocx output state	tim_ocxn output state
1	X	X	0	0	Output disabled (not driven by the timer: Hi-Z) tim_ocx = 0, tim_ocxn = 0	
		0	0	1	Output disabled (not driven by the timer: Hi-Z) tim_ocx = 0	tim_ocxref + Polarity tim_ocxn = tim_ocxref xor CCxNP
		0	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref xor CCxP	Output Disabled (not driven by the timer: Hi-Z) tim_ocxn = 0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_ocx = CCxP	tim_ocxref + Polarity tim_ocxn = tim_ocxref x or CCxNP
		1	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref xor CCxP	Off-State (output enabled with inactive state) tim_ocxn = CCxNP
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_ocx = CCxP, tim_ocxn = CCxNP (if tim_brk or tim_brk2 is triggered). Then (this is valid only if tim_brk is triggered), if the clock is present: tim_ocx = OISx and tim_ocxn = OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and tim_ocxn both in active state (may cause a short circuit when driving switches in half-bridge configuration). *Note: tim_brk2 can only be used if OSSI = OSSR = 1.*	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and the GPIO registers.

26.6.12 TIM1 counter (TIM1_CNT)

Address offset: 0x024

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 the TIMxCR1 is reset, bit 31 is reserved and read at 0.

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.

26.6.13 TIM1 prescaler (TIM1_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 ($f_{tim\_cnt\_ck}$) 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”).

26.6.14 TIM1 autoreload register (TIM1_ARR)

Address offset: 0x02C

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]**: Autoreload value

ARR is the value to be loaded in the actual autoreload register.

Refer to the *Section 26.3.3: 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[19:4]. The ARR[3:0] bitfield contains the dithered part.

26.6.15 TIM1 repetition counter register (TIM1_RCR)

Address offset: 0x030

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **REP[15:0]**: 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 half PWM period in center-aligned mode.

26.6.16 TIM1 capture/compare register 1 (TIM1_CCR1)

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.	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: CR1 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 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.

26.6.17 TIM1 capture/compare register 2 (TIM1_CCR2)

Address offset: 0x038

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.	CCR2[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: 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_CCMR1 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 in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. 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 in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

26.6.18 TIM1 capture/compare register 3 (TIM1_CCR3)

Address offset: 0x03C

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.	CCR3[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR3[19:0]**: Capture/compare 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_CCMR2 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 in CCR3[15:0]. The CCR3[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR3[19:4]. 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 in CCR3[15:0]. The CCR3[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR3[19:4]. The CCR3[3:0] bits are reset.

26.6.19 TIM1 capture/compare register 4 (TIM1_CCR4)

Address offset: 0x040

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.	CCR4[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR4[19:0]**: Capture/compare 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_CCMR2 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 signalled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[19:4]. 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 in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[19:4]. The CCR4[3:0] bits are reset.

26.6.20 TIM1 break and dead-time register (TIM1_BDTR)

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.	BK2BID	BKBID	BK2 DSRM	BK DSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]			
		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
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 bits BKBID/BK2BID/BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **BK2BID**: Break2 bidirectional
Refer to BKBID description

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).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 **BK2DSRM**: Break2 disarm
Refer to BKDSRM description

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.

Bit 25 **BK2P**: Break 2 polarity
0: Break input tim_brk2 is active low
1: Break input tim_brk2 is active high
Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 24 **BK2E**: Break 2 enable
This bit enables the complete break 2 protection, see *Figure 208: Break and Break2 circuitry overview*).
0: Break2 function disabled
1: Break2 function enabled
Note: The BRKIN2 must only be used with OSSR = OSSI = 1.
Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 23:20 **BK2F[3:0]**: Break 2 filter

This bitfield defines the frequency used to sample tim_brk2 input and the length of the digital filter applied to tim_brk2. 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, tim_brk2 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

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

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 consecutive 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

Note: 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 one of the break inputs is active (tim_brk or tim_brk2). 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: In response to a break 2 event. OC and OCN outputs are disabled
In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register).
See OC/OCN enable description for more details (*Section 26.6.11: TIM1 capture/compare enable register (TIM1_CCER)*).

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 none of the break inputs tim_brk and tim_brk2 is 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).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

This bit enables the complete break protection (including all sources connected to tim_sys_brk and BKIN sources, as per *Figure 208: Break and Break2 circuitry overview*).

0: Break function disabled

1: Break function enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: 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 having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 26.6.11: TIM1 capture/compare enable register (TIM1_CCER)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state).

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE = 1 or CCxNE = 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 due to a break event or by a software write, on channels configured as outputs.
See OC/OCN enable description for more details (*Section 26.6.11: TIM1 capture/compare enable register (TIM1_CCER)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).

1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

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/BK2BID/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).

26.6.21 TIM1 capture/compare register 5 (TIM1_CCR5)

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[19:16]			
rw	rw	rw										rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR5[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GC5C3**: Group channel 5 and channel 3
Distortion on channel 3 output:
0: No effect of tim_oc5ref on tim_oc3refc
1: tim_oc3refc is the logical AND of tim_oc3ref and tim_oc5ref
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR2).
Note: it is also possible to apply this distortion on combined PWM signals.

Bit 30 **GC5C2**: Group channel 5 and channel 2
Distortion on channel 2 output:
0: No effect of tim_oc5ref on tim_oc2refc
1: tim_oc2refc is the logical AND of tim_oc2ref and tim_oc5ref
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).
Note: it is also possible to apply this distortion on combined PWM signals.

Bit 29 **GC5C1**: Group channel 5 and channel 1
Distortion on channel 1 output:
0: No effect of oc5ref on oc1refc
1: oc1refc is the logical AND of oc1ref and oc5ref
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).
Note: it is also possible to apply this distortion on combined PWM signals.

Bits 28:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR5[19:0]**: Capture/compare 5 value
CCR5 is the value to be loaded in the actual capture/compare 5 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC5PE). Else the preload value is copied in the active capture/compare 5 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_oc5 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR5[15:0]. The CCR5[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR5[19:4]. The CCR5[3:0] bitfield contains the dithered part.

26.6.22 TIM1 capture/compare register 6 (TIM1_CCR6)

Address offset: 0x04C

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.	CCR6[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR6[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 **CCR6[19:0]**: Capture/compare 6 value

CCR6 is the value to be loaded in the actual capture/compare 6 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC6PE). Else the preload value is copied in the active capture/compare 6 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_oc6 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR6[15:0]. The CCR6[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR6[19:4]. The CCR6[3:0] bitfield contains the dithered part.

26.6.23 TIM1 capture/compare mode register 3 (TIM1_CCMR3)

Address offset: 0x050

Reset value: 0x0000 0000

Refer to the above CCMR1 register description. Channels 5 and 6 can only be configured in output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC6 CE	OC6M[2:0]			OC6 PE	OC6FE	Res.	Res.	OC5 CE	OC5M[2:0]			OC5PE	OC5FE	Res.	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		

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

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC6CE**: Output compare 6 clear enable

Bits 24, 14:12 **OC6M[3:0]**: Output compare 6 mode

Bit 11 **OC6PE**: Output compare 6 preload enable

Bit 10 **OC6FE**: Output compare 6 fast enable

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **OC5CE:** Output compare 5 clear enable

Bits 16, 6:4 **OC5M[3:0]**: Output compare 5 mode

Bit 3 **OC5PE**: Output compare 5 preload enable

Bit 2 **OC5FE**: Output compare 5 fast enable

Bits 1:0 Reserved, must be kept at reset value.

26.6.24 TIM1 timer deadtime register 2 (TIM1_DTR2)

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).

26.6.25 TIM1 timer encoder control register (TIM1_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 or tim_ti4 input
00: Index always active
01: Index disabled when 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

26.6.26 TIM1 timer input selection register (TIM1_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 26.3.2: TIM1 pins and internal signals* for interconnects list.

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_CH2
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 26.3.2: TIM1 pins and internal signals* for interconnects list.

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 26.3.2: TIM1 pins and internal signals* for interconnects list.

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 26.3.2: TIM1 pins and internal signals* for interconnects list.

26.6.27 TIM1 alternate function option register 1 (TIM1_AF1)

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.	ETRSEL[3:2]	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		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	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 26.3.2: TIM1 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).

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 polarity is not inverted (active low if BKP = 0, active high if BKP = 1)
1: tim_brk_cmp4 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

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 polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp3 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

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 polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp2 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

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 polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp1 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

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 polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: TIMx_BKIN input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

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).

Note: Refer to Section 26.3.2: TIM1 pins and internal signals for product specific implementation.

26.6.28 TIM1 alternate function register 2 (TIM1_AF2)

Address offset: 0x064

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.	OCRSEL[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK2CMP4P	BK2CMP3P	BK2CMP2P	BK2CMP1P	BK2INP	BK2CMP8E	BK2CMP7E	BK2CMP6E	BK2CMP5E	BK2CMP4E	BK2CMP3E	BK2CMP2E	BK2CMP1E	BK2INE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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 26.3.2: TIM1 pins and internal signals* for product specific information.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

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

Bit 13 **BK2CMP4P**: tim_brk2_cmp4 input polarity

This bit selects the tim_brk2_cmp4 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp4 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp4 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 **BK2CMP3P**: tim_brk2_cmp3 input polarity

This bit selects the tim_brk2_cmp3 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp3 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp3 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BK2CMP2P**: tim_brk2_cmp2 input polarity

This bit selects the tim_brk2_cmp2 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BK2CMP1P**: tim_brk2_cmp1 input polarity

This bit selects the tim_brk2_cmp1 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp1 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp1 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BK2INP**: TIMx_BKIN2 input polarity

This bit selects the TIMx_BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.

0: TIMx_BKIN2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: TIMx_BKIN2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BK2CMP8E**: tim_brk2_cmp8 enable

This bit enables the tim_brk2_cmp8 for the timer's tim_brk2 input. tim_brk2_cmp8 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp8 input disabled

1: tim_brk2_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 **BK2CMP7E**: tim_brk2_cmp7 enable

This bit enables the tim_brk2_cmp7 for the timer's tim_brk2 input. tim_brk2_cmp7 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp7 input disabled

1: tim_brk2_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 **BK2CMP6E**: tim_brk2_cmp6 enable

This bit enables the tim_brk2_cmp6 for the timer's tim_brk2 input. tim_brk2_cmp6 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp6 input disabled

1: tim_brk2_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 **BK2CMP5E**: tim_brk2_cmp5 enable

This bit enables the tim_brk2_cmp5 for the timer's tim_brk2 input. tim_brk2_cmp5 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp5 input disabled

1: tim_brk2_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 **BK2CMP4E**: tim_brk2_cmp4 enable

This bit enables the tim_brk2_cmp4 for the timer's tim_brk2 input. tim_brk2_cmp4 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp4 input disabled

1: tim_brk2_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 **BK2CMP3E**: tim_brk2_cmp3 enable

This bit enables the tim_brk2_cmp3 for the timer's tim_brk2 input. tim_brk2_cmp3 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp3 input disabled

1: tim_brk2_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 **BK2CMP2E**: tim_brk2_cmp2 enable

This bit enables the tim_brk2_cmp2 for the timer's tim_brk2 input. tim_brk2_cmp2 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp2 input disabled

1: tim_brk2_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 **BK2CMP1E**: tim_brk2_cmp1 enable

This bit enables the tim_brk2_cmp1 for the timer's tim_brk2 input. tim_brk2_cmp1 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp1 input disabled

1: tim_brk2_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 **BK2INE**: TIMx_BKIN2 input enable

This bit enables the TIMx_BKIN2 alternate function input for the timer's tim_brk2 input. TIMx_BKIN2 input is 'ORed' with the other tim_brk2 sources.

0: TIMx_BKIN2 input disabled

1: TIMx_BKIN2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Section 26.3.2: TIM1 pins and internal signals for product specific implementation.

26.6.29 TIM1 DMA control register (TIM1_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
...

26.6.30 TIM1 DMA address for full transfer (TIM1_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).

26.6.31 TIM1 register map

TIM1 registers are mapped as 16-bit addressable registers as described in the table below:

Table 186. TIM1 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	**TIMx_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	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	MMS[3]	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5	OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]			CCDS	CCUS	Res.	CCPC
	Reset value							0		0	0	0	0		0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x008	**TIMx_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	ETP	ECE	ETP S [1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	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	0
0x00C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value									0	0	0	0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	CC6IF	CC5IF	Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value									0	0	0	0			0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	**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.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																								0	0	0	0	0	0	0	0	0

Table 186. TIM1 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
0x018	**TIMx_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]		CC2 S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_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	CC2 S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	**TIMx_CCMR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4 S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3 S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4 S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	**TIMx_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E	CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value											0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	**TIMx_CNT**	UIFCPY	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
0x028	**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
0x02C	**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
0x030	**TIMx_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	**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
0x038	**TIMx_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x03C	**TIMx_CCR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x040	**TIMx_CCR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x044	**TIMx_BDTR**	Res.	Res.	BK2BID	BKBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]				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	0	0	0	0	0	0	0	0

Table 186. TIM1 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
0x048	**TIMx_CCR5**	GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[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
0x04C	**TIMx_CCR6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	**TIMx_CCMR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]	OC6CE	OC6M [2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M [2:0]			OC5PE	OC5FE	Res.	Res.
	Reset value								0								0	0	0	0	0	0	0			0	0	0	0	0	0		
0x054	**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
0x058	**TIMx_ECR**	Res.	Res.	Res.	Res.	Res.	PWPR SC[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
0x05C	**TIMx_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	**TIMx_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res ETRSEL [3:0]				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	0	0	0	0	1
0x064	**TIMx_AF2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL [2:0]			Res.	Res.	BK2CMP4P	BK2CMP3P	BK2CMP2P	BK2CMP1P	BK2INP	BK2CMP8E	BK2CMP7E	BK2CMP6E	BK2CMP5E	BK2CMP4E	BK2CMP3E	BK2CMP2E	BK2CMP1E	BK2INE
	Reset value														0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x068.. 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 2.2* for the register boundary addresses.

27 General-purpose timers (TIM2/TIM3)

27.1 TIM2/TIM3 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 27.4.23: Timer synchronization*.

27.2 TIM2/TIM3 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.

27.3 TIM2/TIM3 implementation

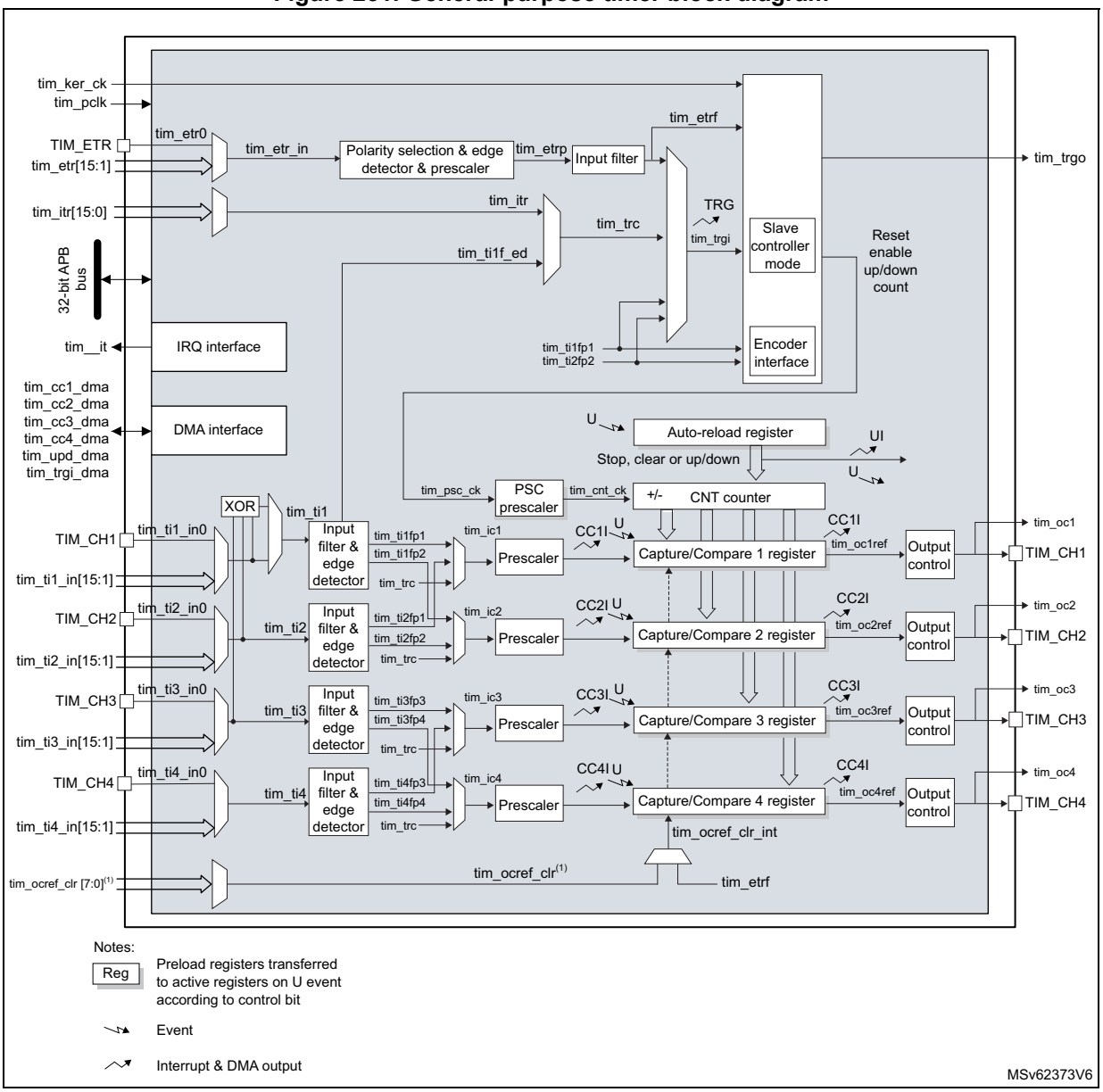
Table 187. STM32H503xx general purpose timers

Timer instance	TIM2	TIM3
Resolution	32-bit	16-bit
OCREF clear selection	Yes	Yes
Sources	tim_etrf Reserved	tim_etrf Reserved

27.4 TIM2/TIM3 functional description

27.4.1 Block diagram

Figure 251. General-purpose timer block diagram



1. This feature is not available on all timers, refer to *Section 27.3: TIM2/TIM3 implementation*.

27.4.2 TIM2/TIM3 pins and internal signals

Table 188 and *Table 189* in this section summarize the TIM inputs and outputs.

Table 188. 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 189. 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 189. 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 190, *Table 191*, *Table 192* and *Table 193* are listing the sources connected to the tim_ti[4:1] input multiplexers.

Table 190. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources	
	TIM2	TIM3
tim_ti1_in0	TIM2_CH1	TIM3_CH1
tim_ti1_in1	RCC_LSI	COMP1_OUT
tim_ti1_in2	RCC_LSE	RCC_MCO1
tim_ti1_in3	RTC_WKUP_IT	AFI_TIM2_TI1
tim_ti1_in4	AFI_TIM3_TI1	RCC_HSE_1MHz
tim_ti1_in[15:5]	Reserved	

Table 191. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources	
	TIM2	TIM3
tim_ti2_in0	TIM2_CH2	TIM3_CH2
tim_ti2_in1	RCC_HSI/1024	RCC_CSI/128
tim_ti2_in2	RCC_CSI/128	RCC_MCO2
tim_ti2_in3	RCC_MCO2	RCC_HSI/1024
tim_ti2_in4	RCC_MCO1	Reserved
tim_ti2_in[15:5]	Reserved	

Table 192. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources	
	TIM2	TIM3
tim_ti3_in0	TIM2_CH3	TIM3_CH3
tim_ti3_in[15:1]	Reserved	

Table 193. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources	
	TIM2	TIM3
tim_ti4_in0	TIM2_CH4	TIM3_CH4
tim_ti4_in1	COMP1_OUT	Reserved
tim_ti4_in[15:2]	Reserved	

Table 194 lists the internal sources connected to the tim_etr input multiplexer.

Table 194. TIMx internal trigger connection

TIMx	TIM2	TIM3
tim_itr0	tim1_trgo	tim1_trgo
tim_itr1	Reserved	tim2_trgo
tim_itr2	tim3_trgo	Reserved
tim_itr[11:3]	Reserved	
tim_itr12	USBSOF	Reserved
tim_itr[15:13]	Reserved	

Table 195 lists the internal sources connected to the tim_etr input multiplexer.

Table 195. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signals assignment	
	TIM2	TIM3
tim_etr0	TIM2_ETR	TIM3_ETR
tim_etr1	COMP1_OUT	COMP1_OUT
tim_etr2	Reserved	

Table 195. Interconnect to the tim_etr input multiplexer (continued)

Timer external trigger input signal	Timer external trigger signals assignment	
	TIM2	TIM3
tim_etr3	RCC_LSE	Reserved
tim_etr4	Reserved	
tim_etr5		
tim_etr6		
tim_etr7		
tim_etr8		TIM2_ETR
tim_etr9	TIM3_ETR	Reserved
tim_etr[15:10]	Reserved	

27.4.3 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 252 and *Figure 253* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 252. Counter timing diagram with prescaler division change from 1 to 2

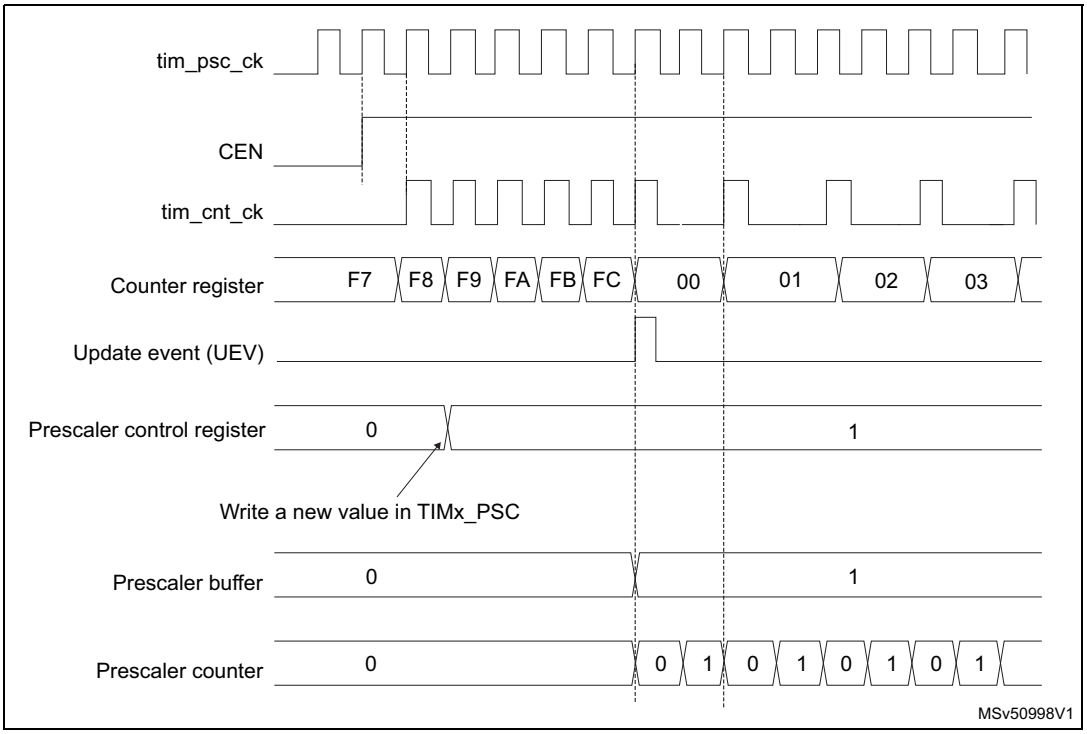
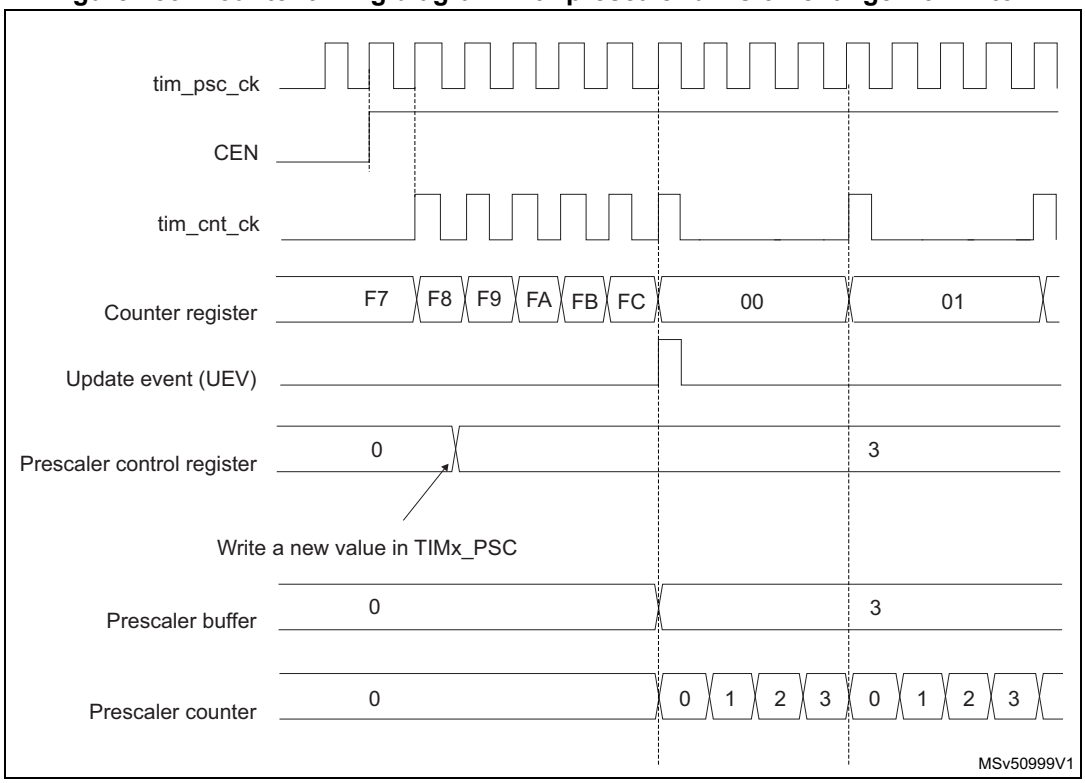


Figure 253. Counter timing diagram with prescaler division change from 1 to 4



27.4.4 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 254. Counter timing diagram, internal clock divided by 1

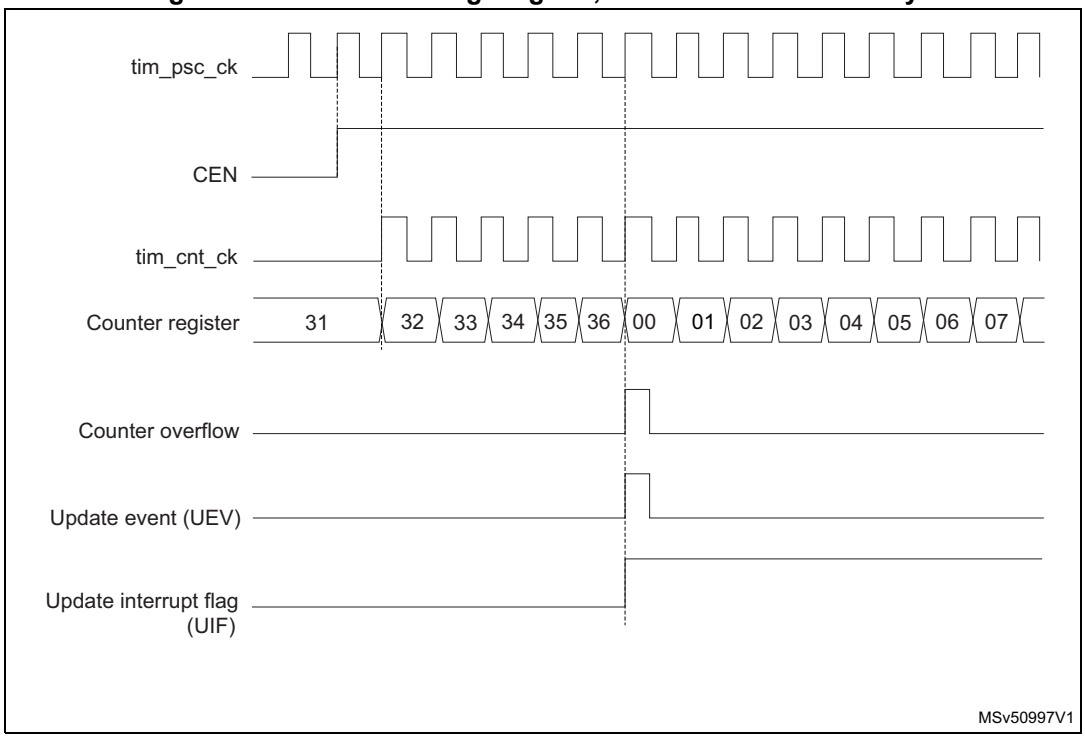


Figure 255. Counter timing diagram, internal clock divided by 2

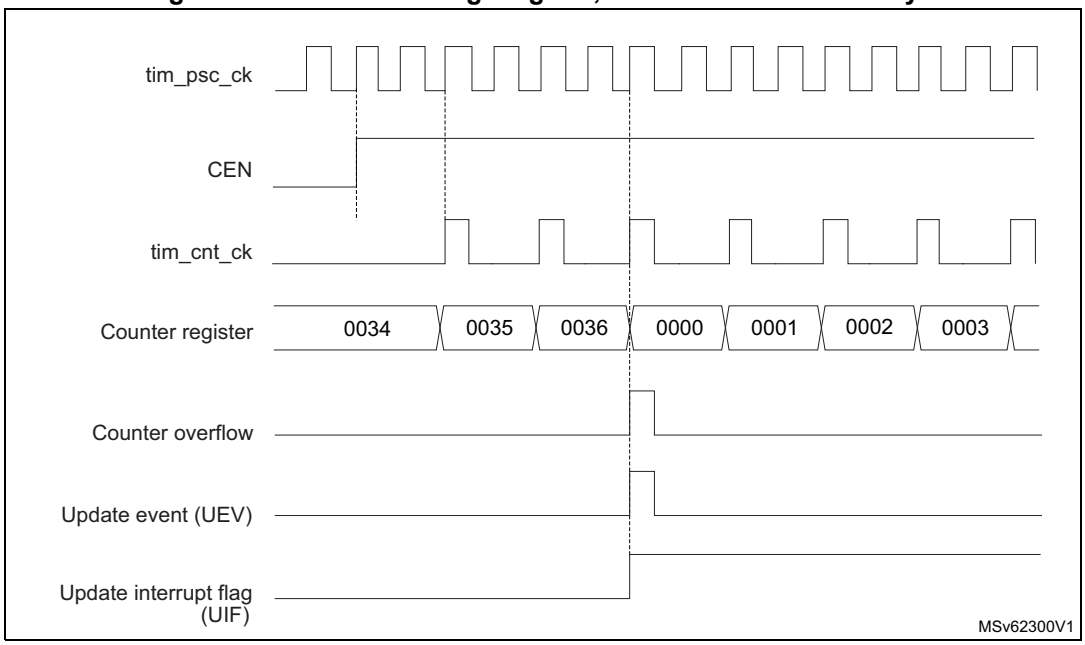


Figure 256. Counter timing diagram, internal clock divided by 4

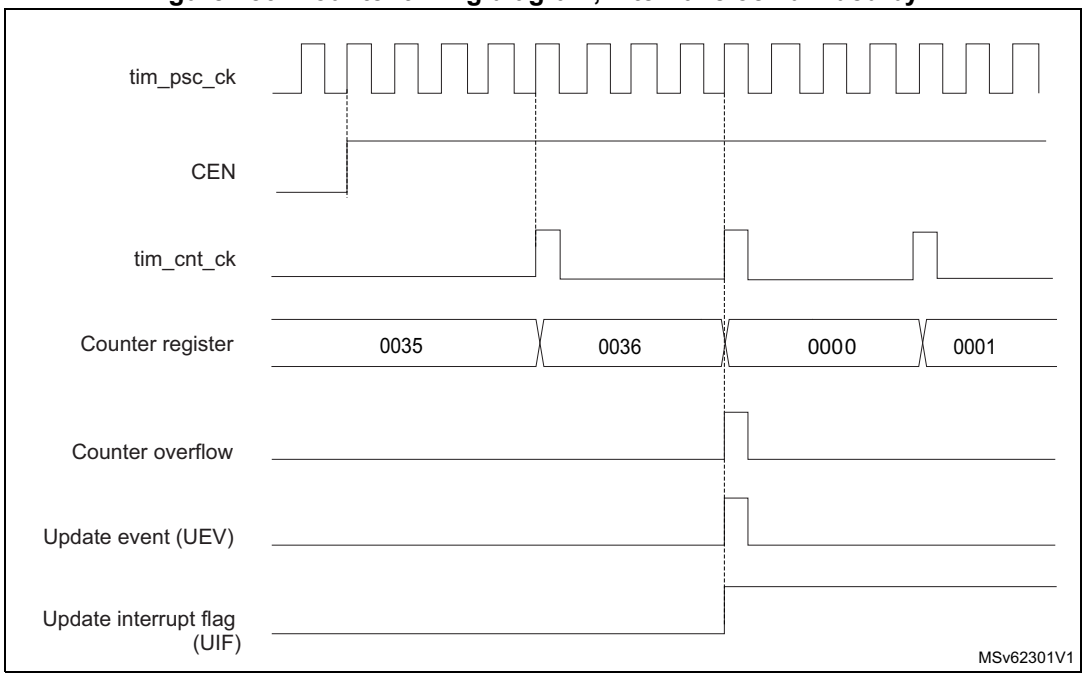


Figure 257. Counter timing diagram, internal clock divided by N

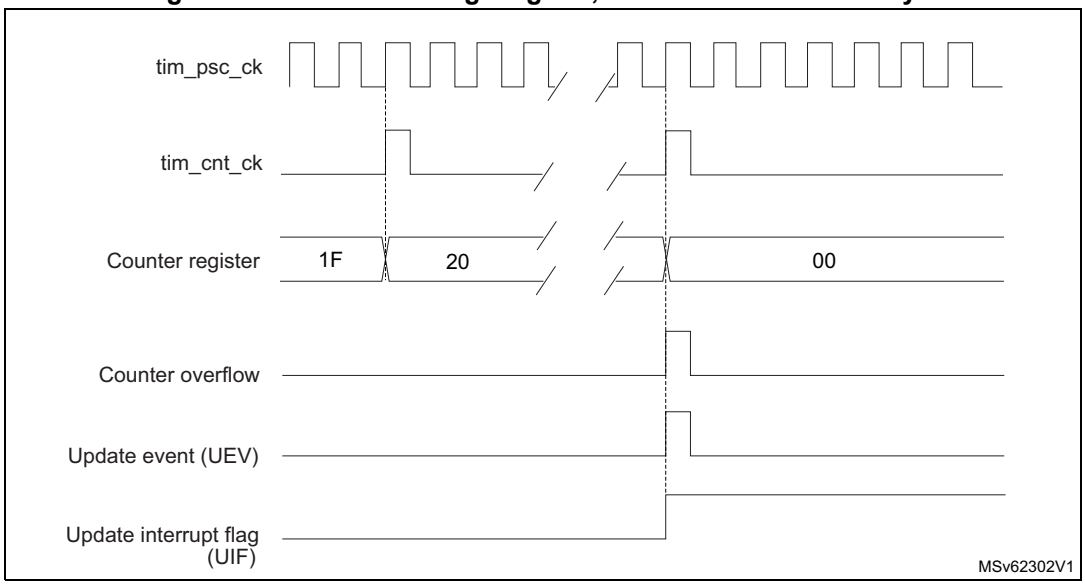


Figure 258. Counter timing diagram, Update event when ARPE = 0 (TIMx_ARR not preloaded)

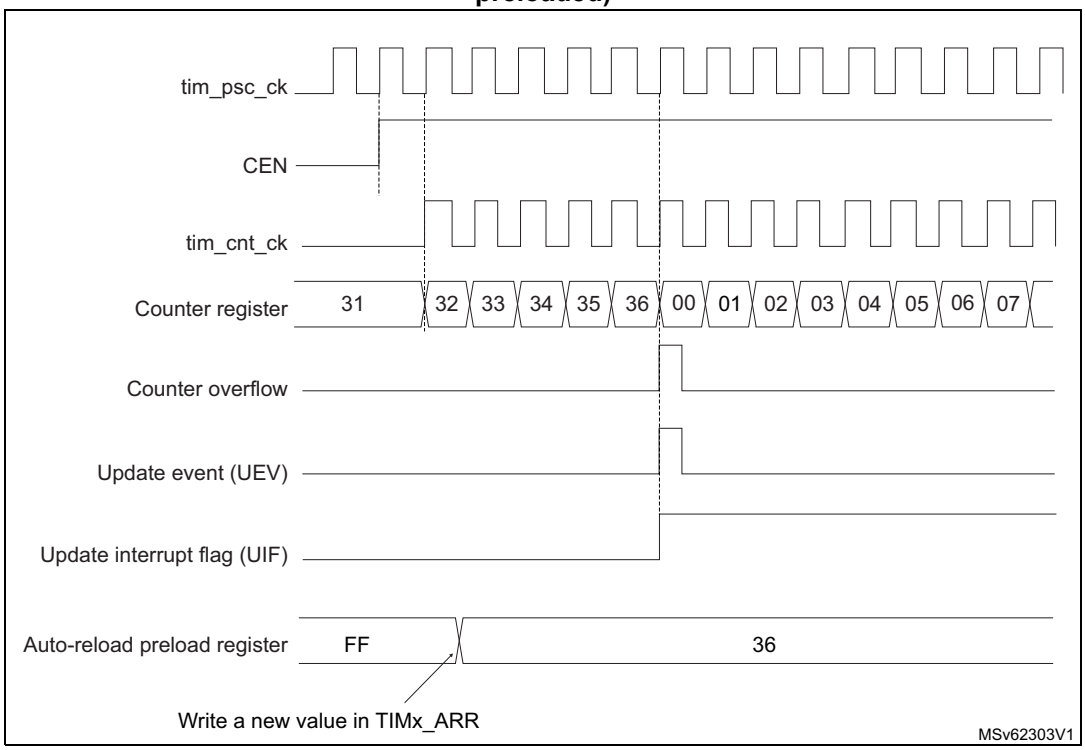
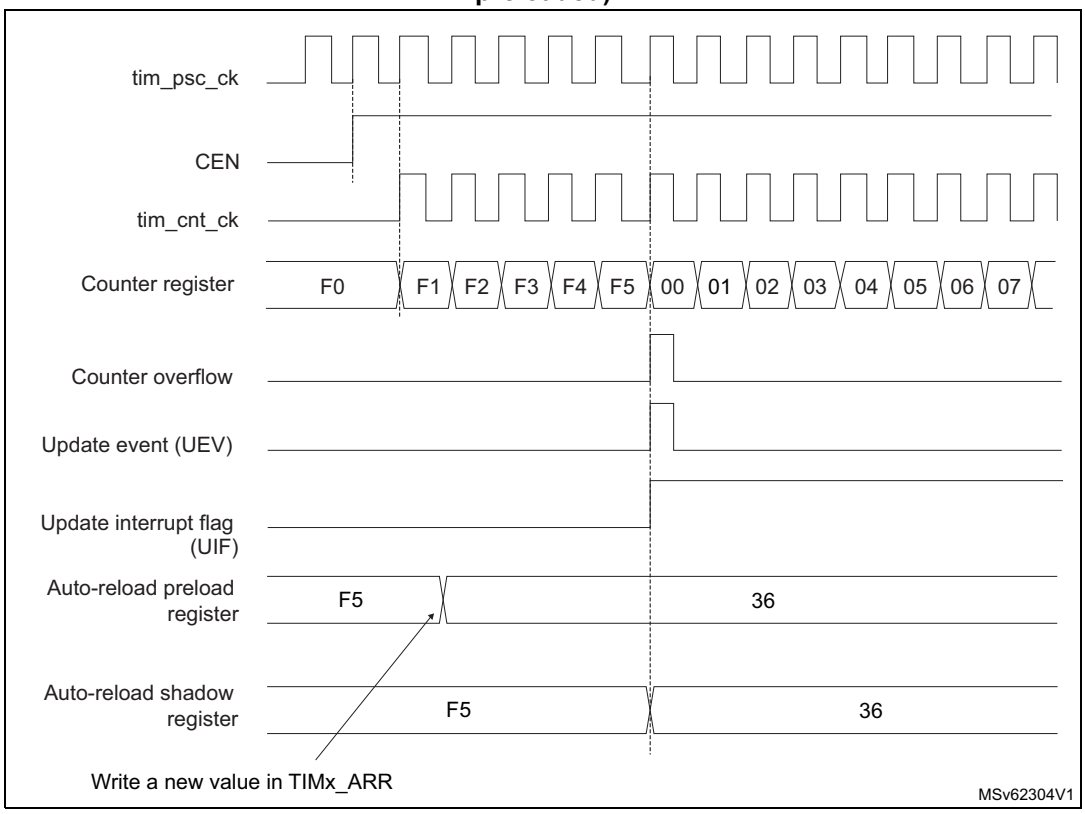


Figure 259. Counter timing diagram, Update event when ARPE = 1 (TIMx_ARR preloaded)



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 260. Counter timing diagram, internal clock divided by 1

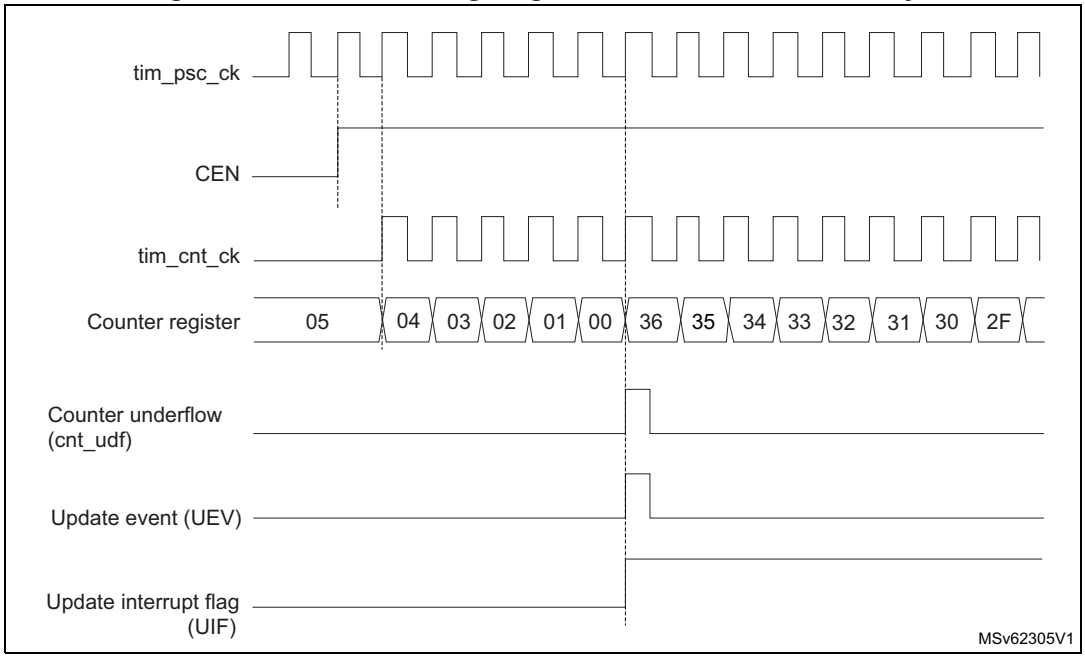


Figure 261. Counter timing diagram, internal clock divided by 2

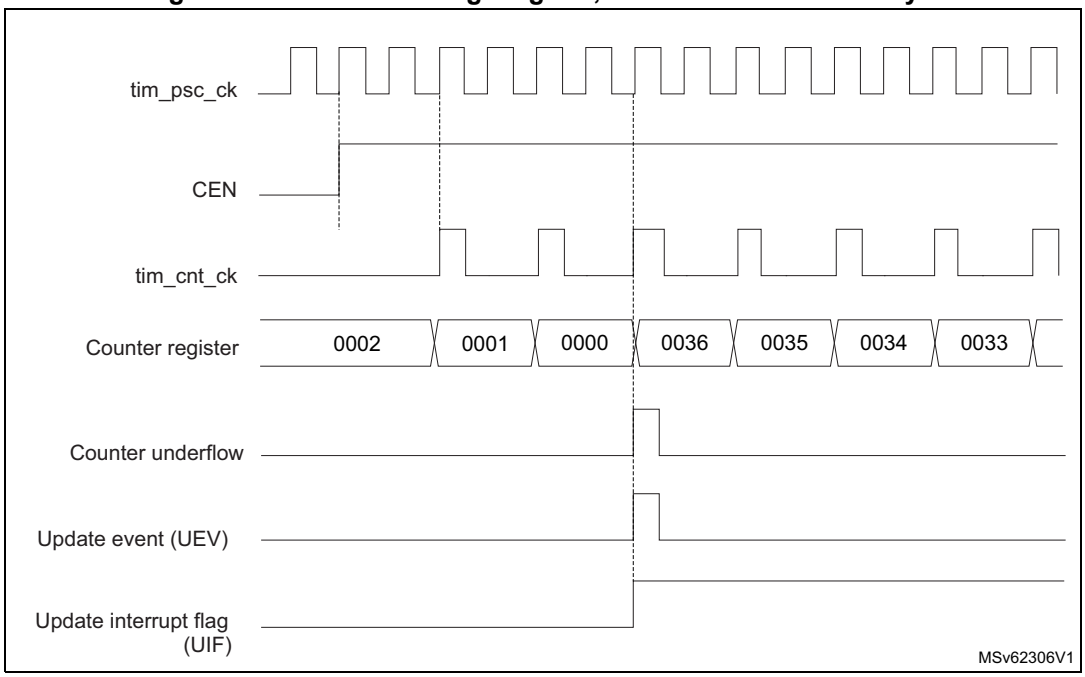


Figure 262. Counter timing diagram, internal clock divided by 4

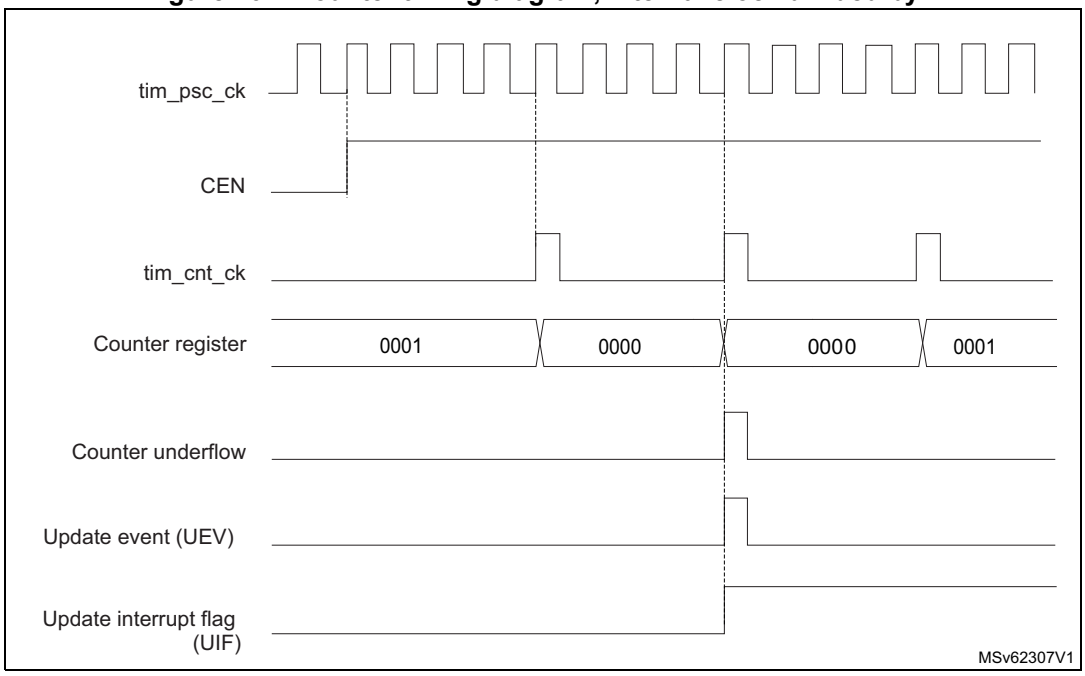


Figure 263. Counter timing diagram, internal clock divided by N

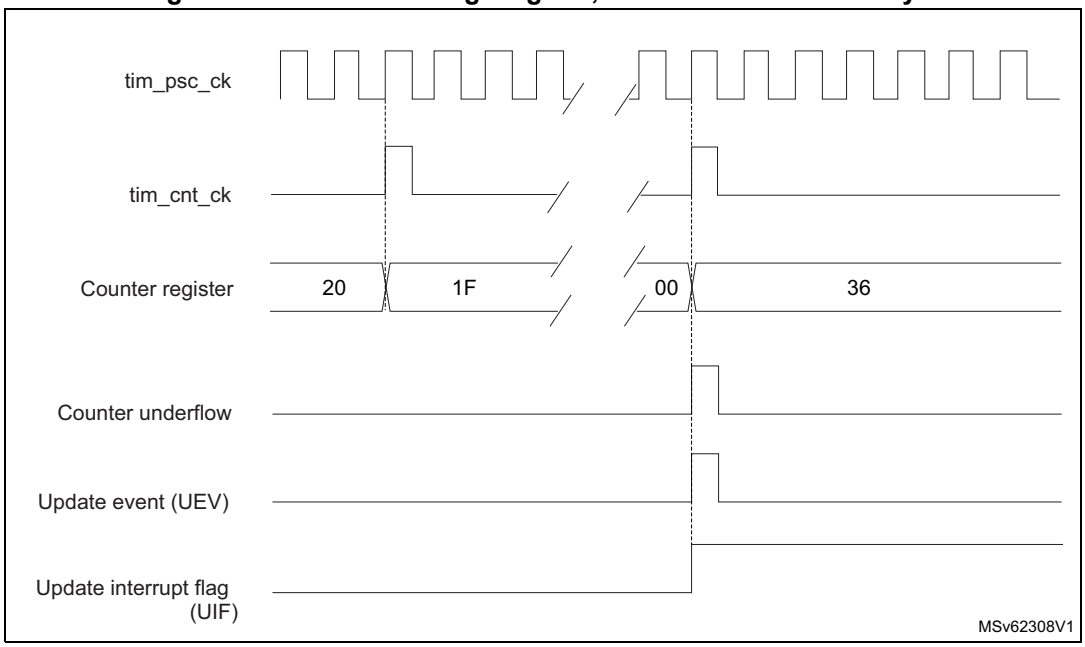
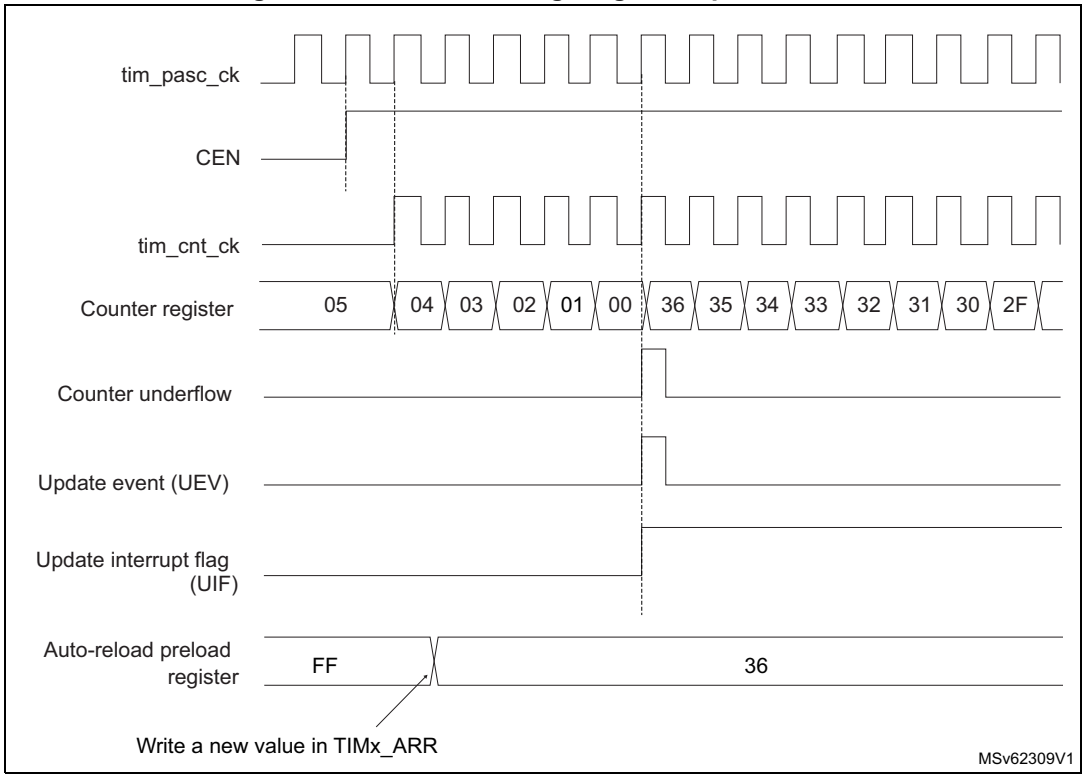


Figure 264. Counter timing diagram, Update event



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.

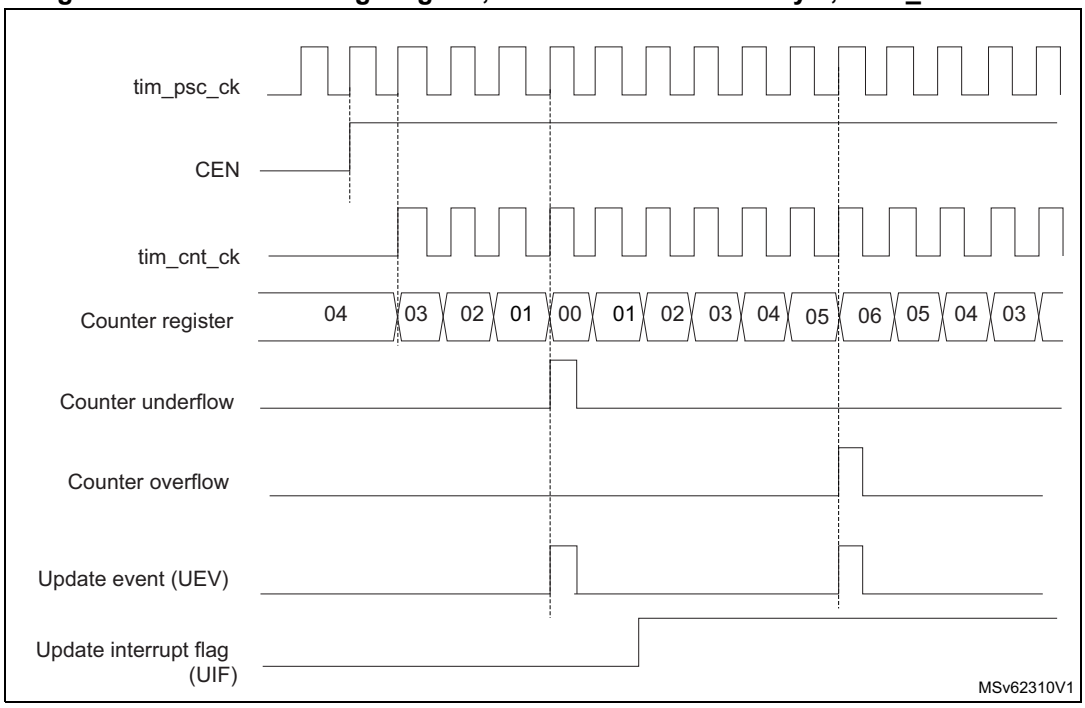
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on 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 265. 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 27.5.1: TIMx control register 1 (TIMx_CR1)(x = 2, 3)*).

Figure 266. Counter timing diagram, internal clock divided by 2

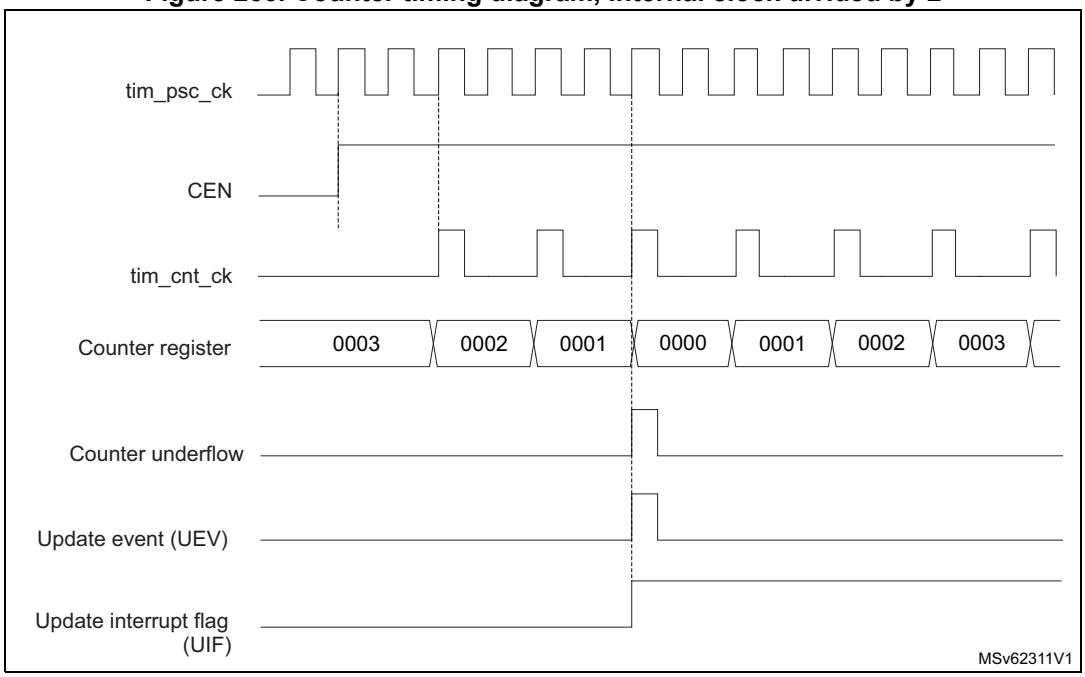
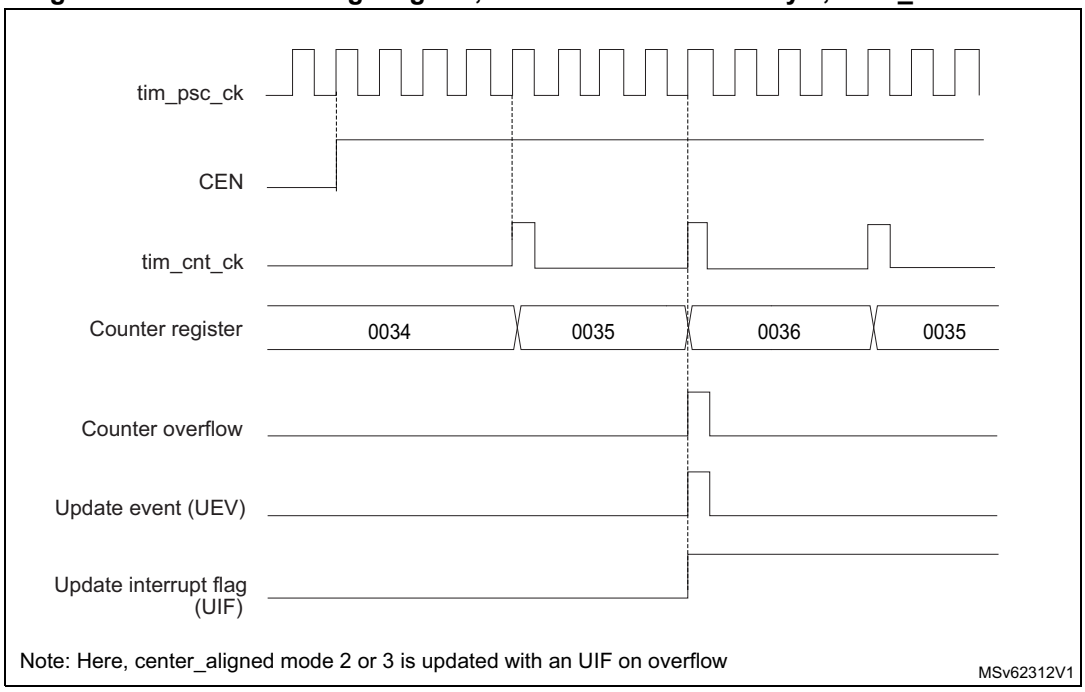


Figure 267. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36



1. Center-aligned mode 2 or 3 is used with a UIF on overflow.

Figure 268. Counter timing diagram, internal clock divided by N

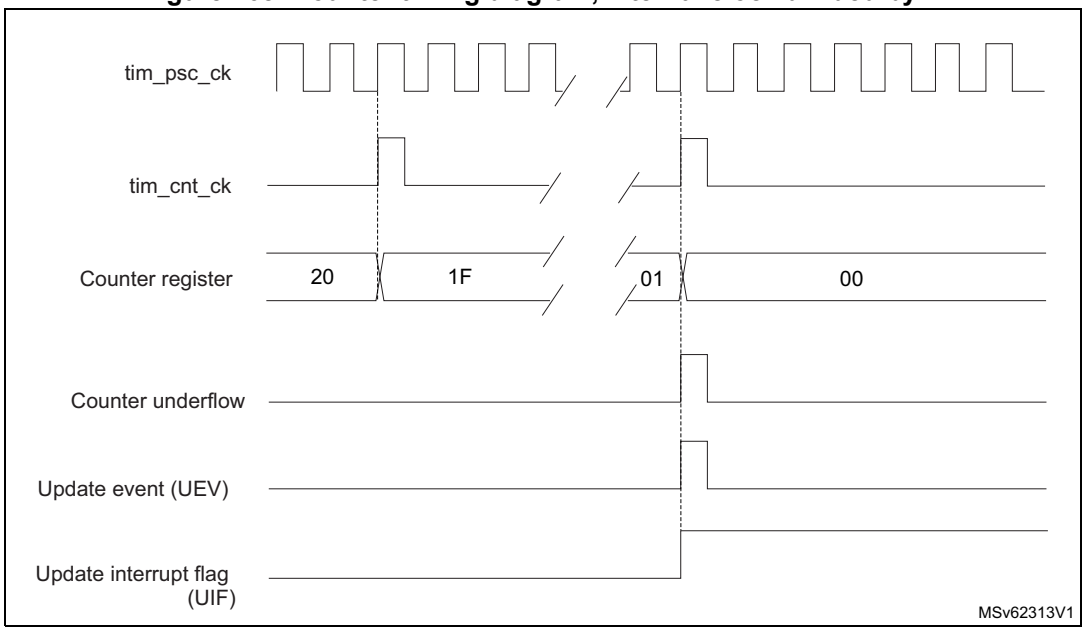


Figure 269. Counter timing diagram, Update event with ARPE = 1 (counter underflow)

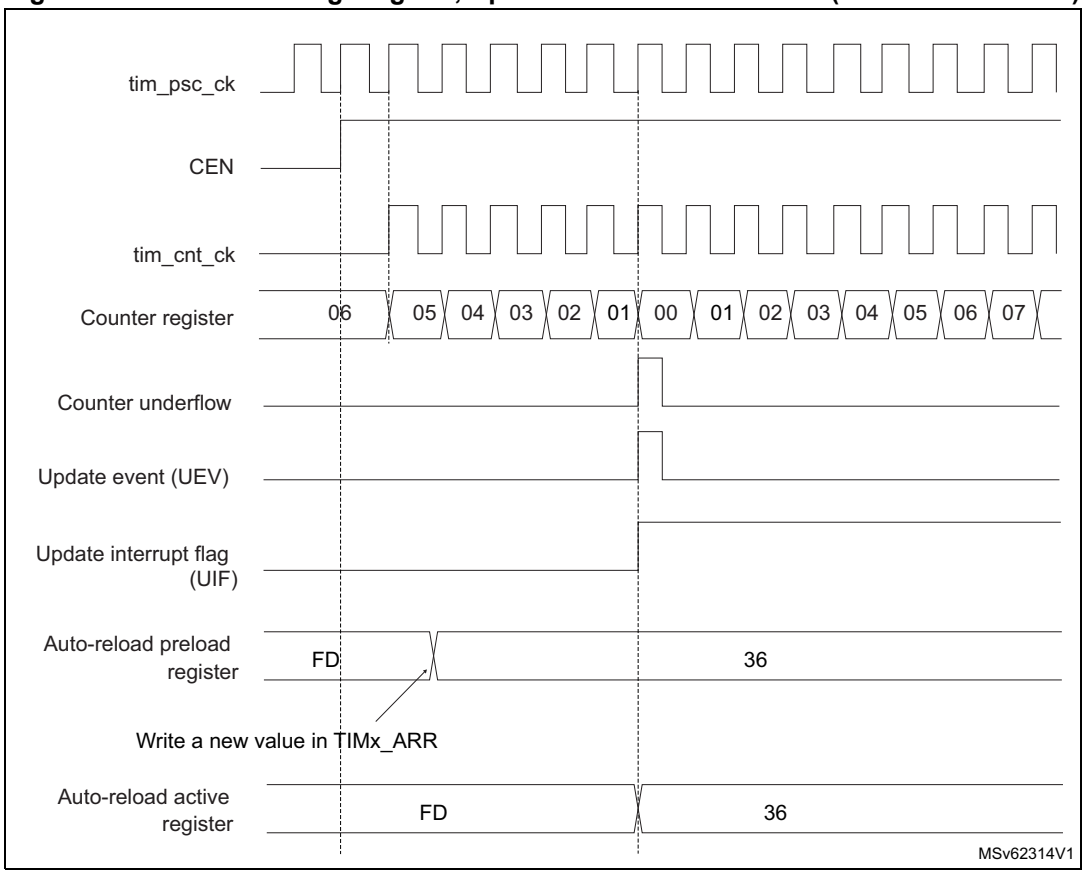
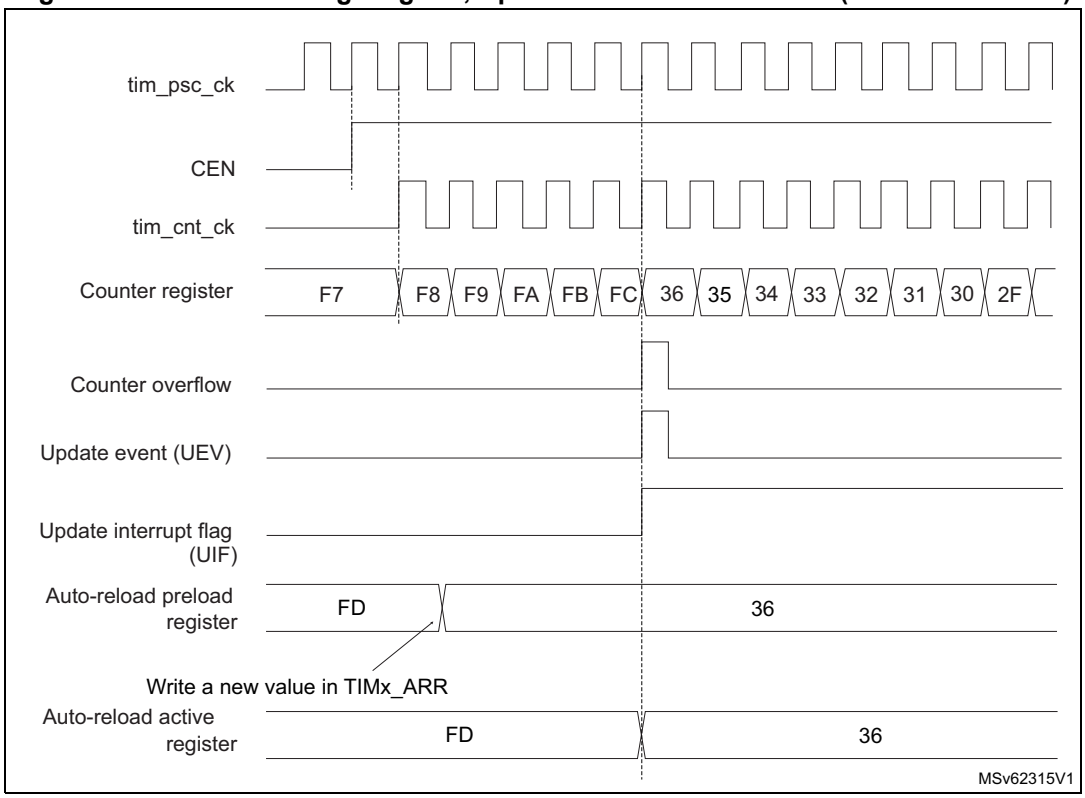


Figure 270. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



27.4.5 Clock selection

The counter clock can be provided by the following clock sources:

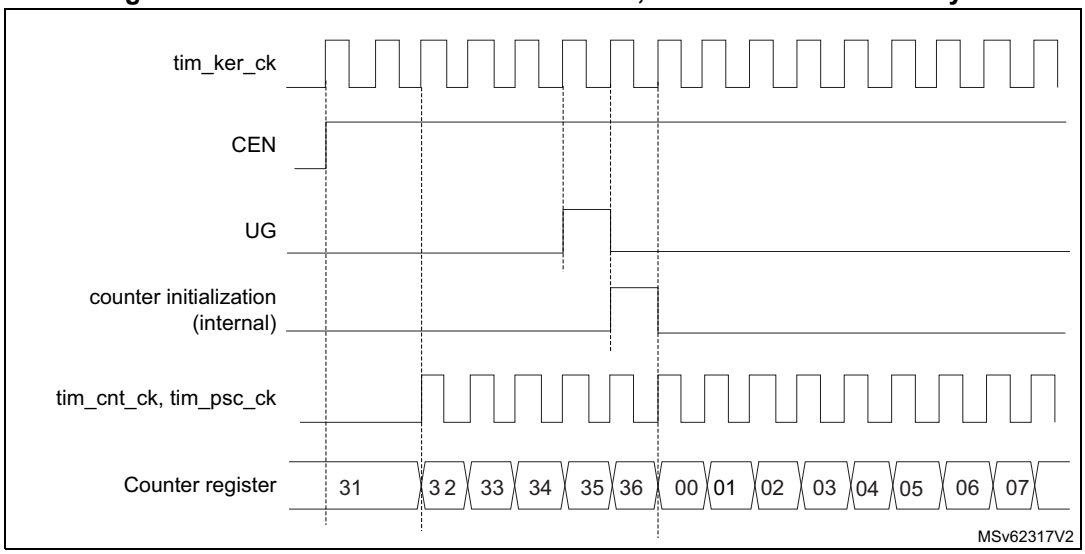
- Internal clock (tim_ker_ck).
- External clock mode1: external input pin (tim_ti1 or tim_ti2).
- External clock mode2: external trigger input (tim_etr_in).
- Internal trigger inputs (tim_itr): using one timer as prescaler for another timer, for example, timer 1 can be configured to act as a prescaler 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 271 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

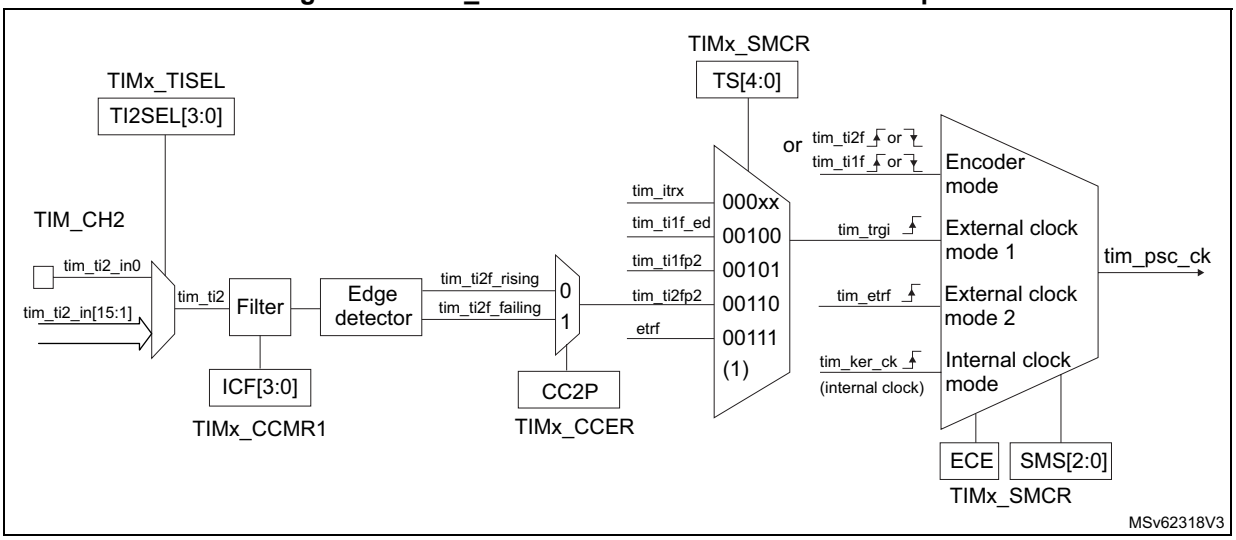
Figure 271. Control circuit in normal mode, internal clock divided by 1



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 272. 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).

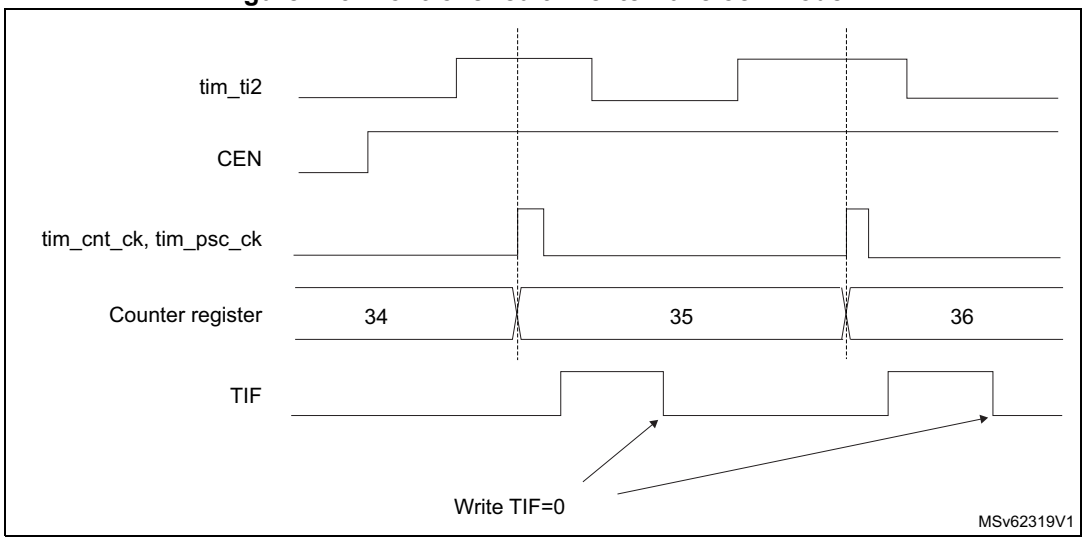
Note: The capture prescaler is not used for triggering, so it does not need to be configured.

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 273. Control circuit in external clock mode 1



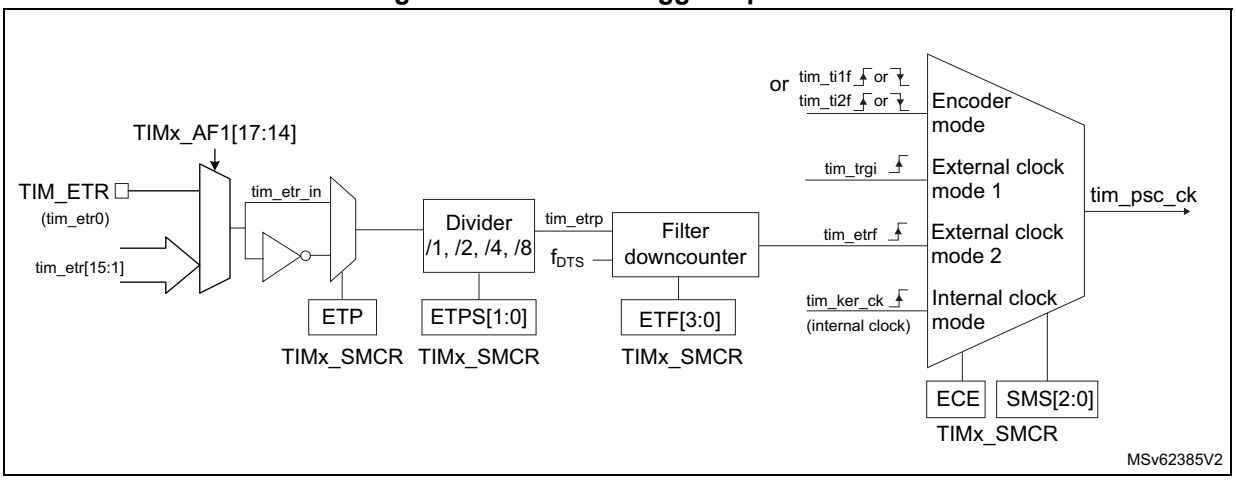
External clock source mode 2

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 274 gives an overview of the external trigger input block.

Figure 274. External trigger input block



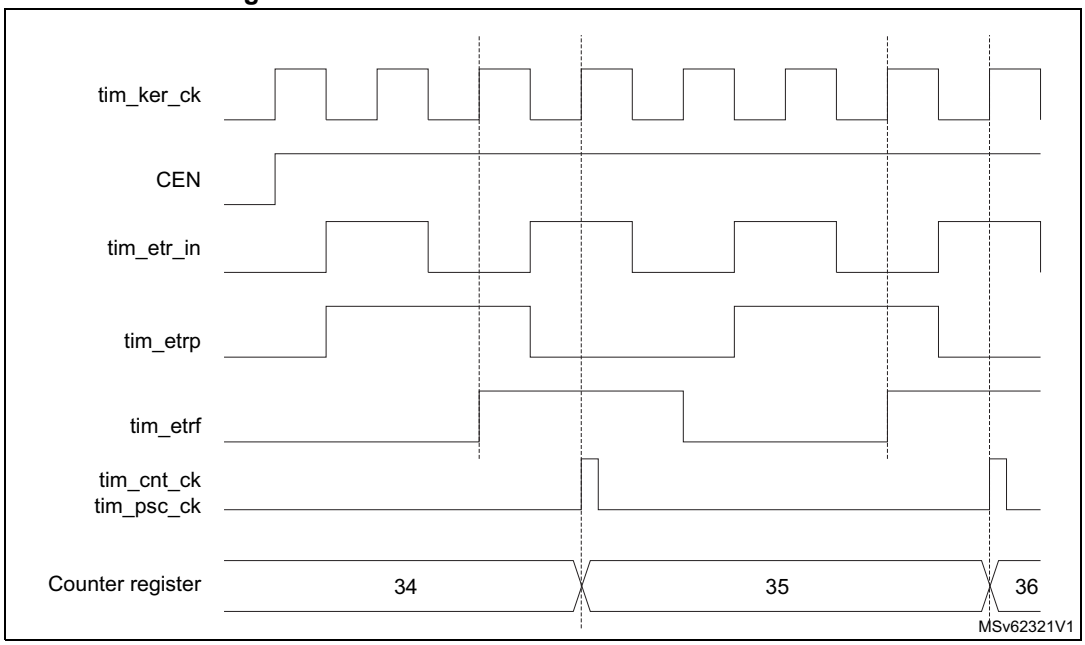
For example, to configure the upcounter to count each two rising edges on tim_etr_in, use the following procedure:

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 TIMxCLK frequency. When the ETRP signal is faster, the user must apply a division of the external signal by a proper ETPS prescaler setting.

Figure 275. Control circuit in external clock mode 2



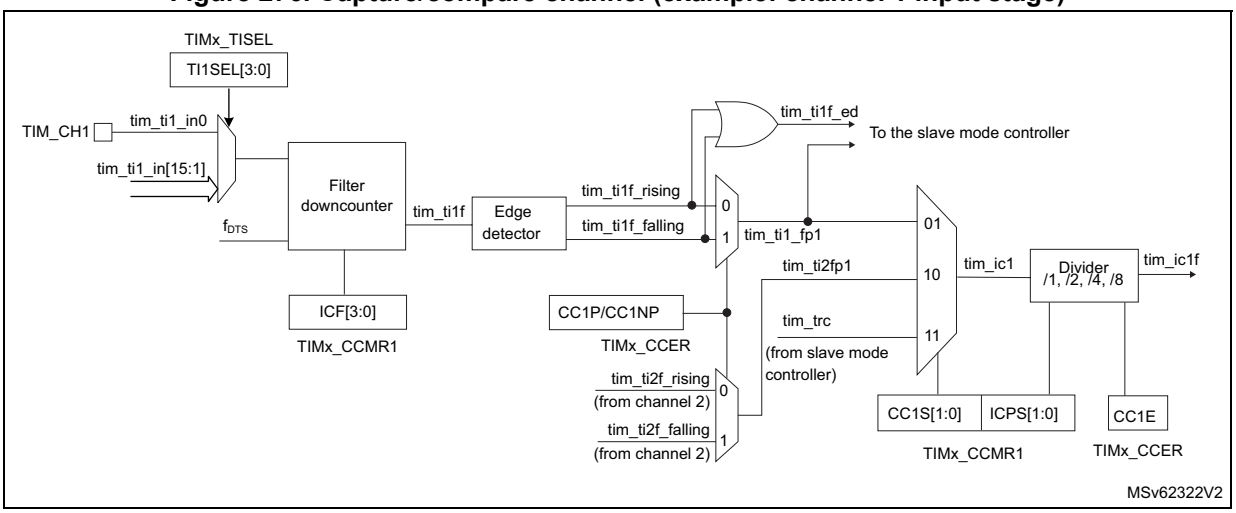
27.4.6 Capture/compare channels

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 276. 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 277. Capture/compare channel 1 main circuit

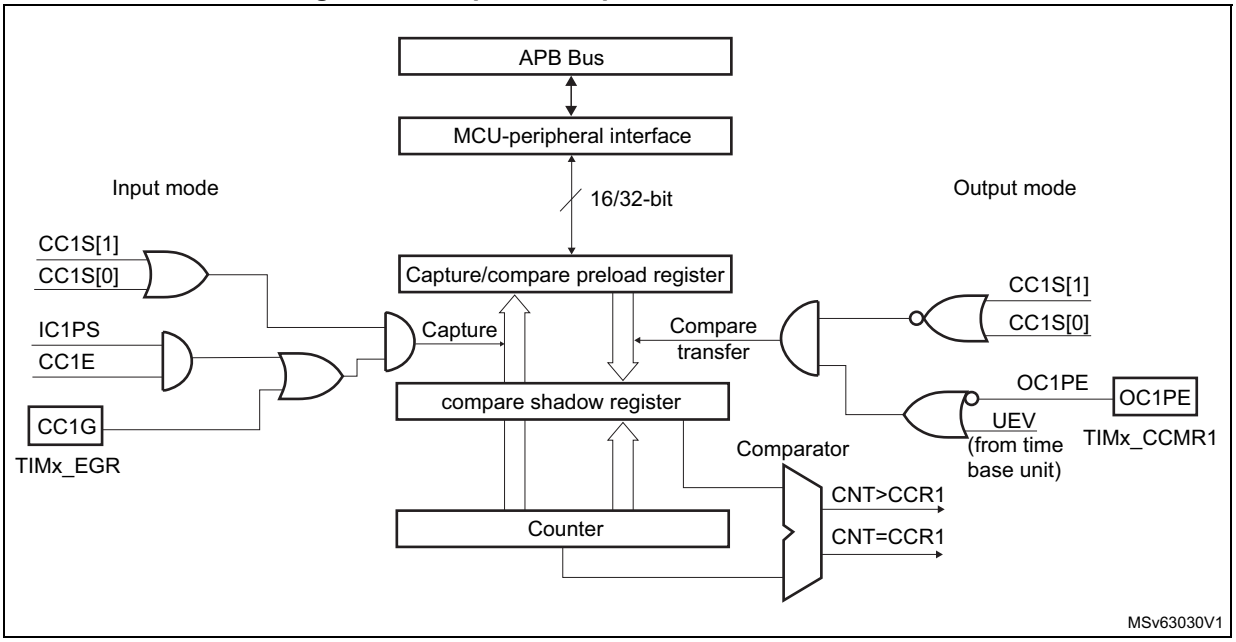
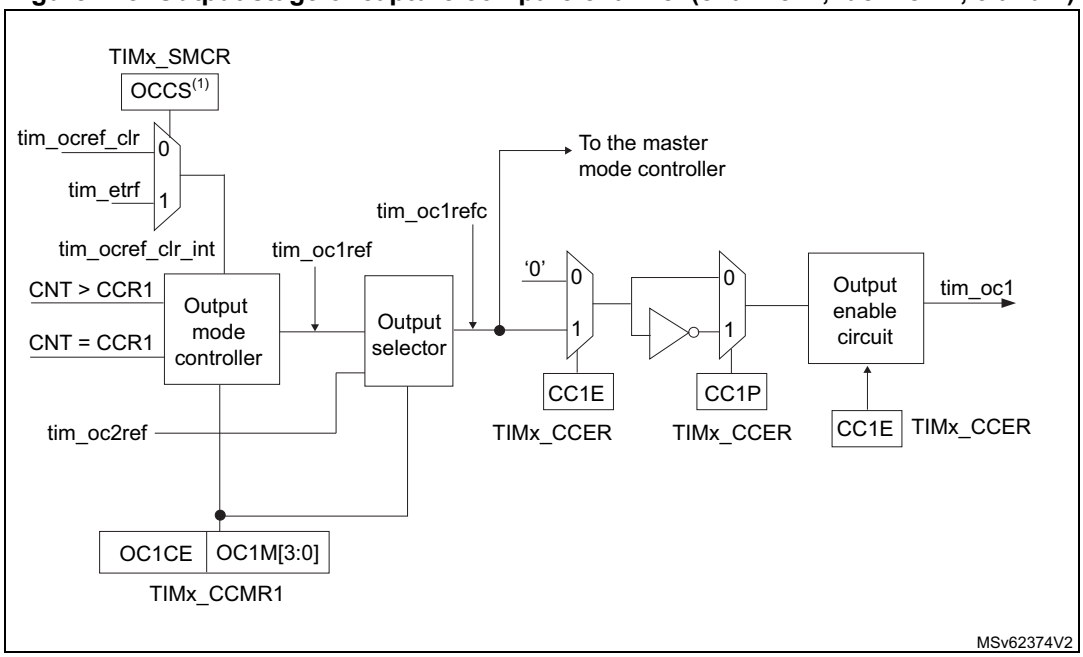


Figure 278. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4)



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.

27.4.7 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.

27.4.8 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 TIM_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges).

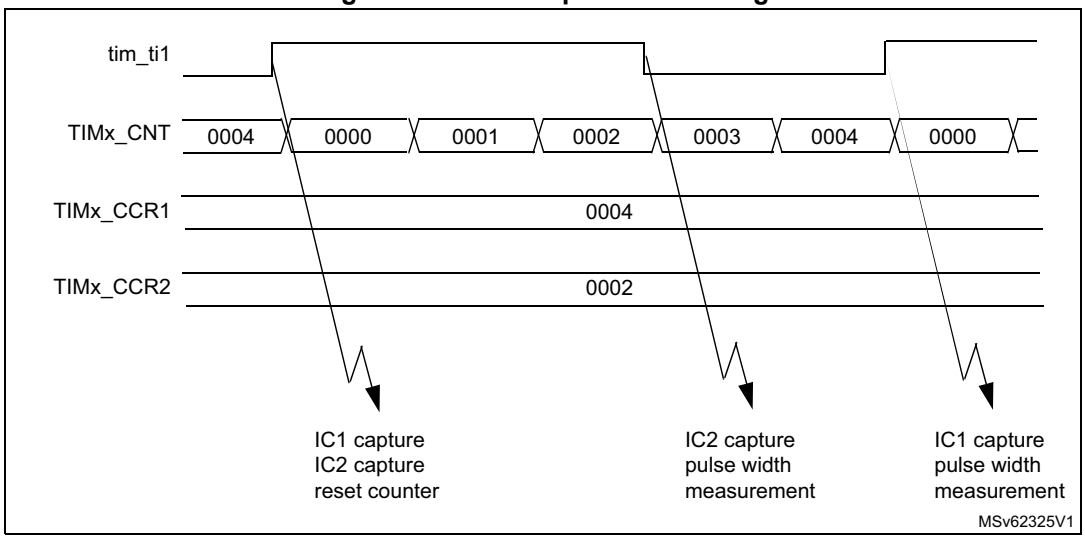
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- 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 279. 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.

27.4.9 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.

27.4.10 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.

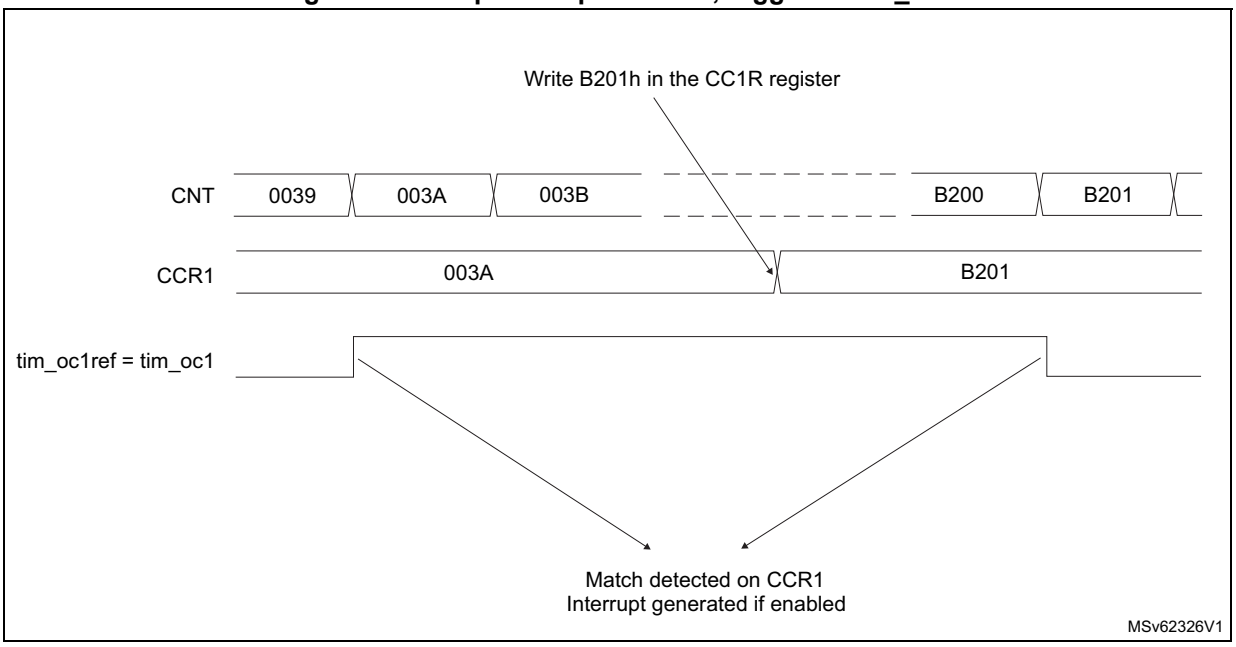
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

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 280*.

Figure 280. Output compare mode, toggle on tim_oc1



27.4.11 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.

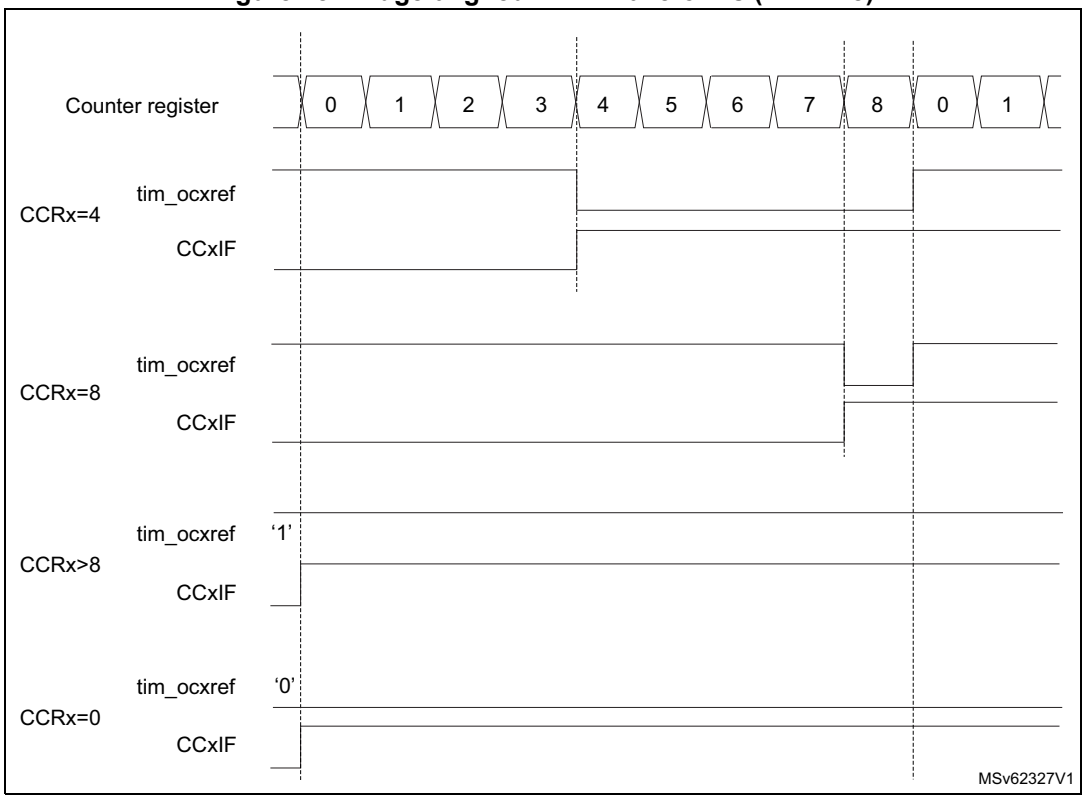
The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

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 281* shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 281. 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

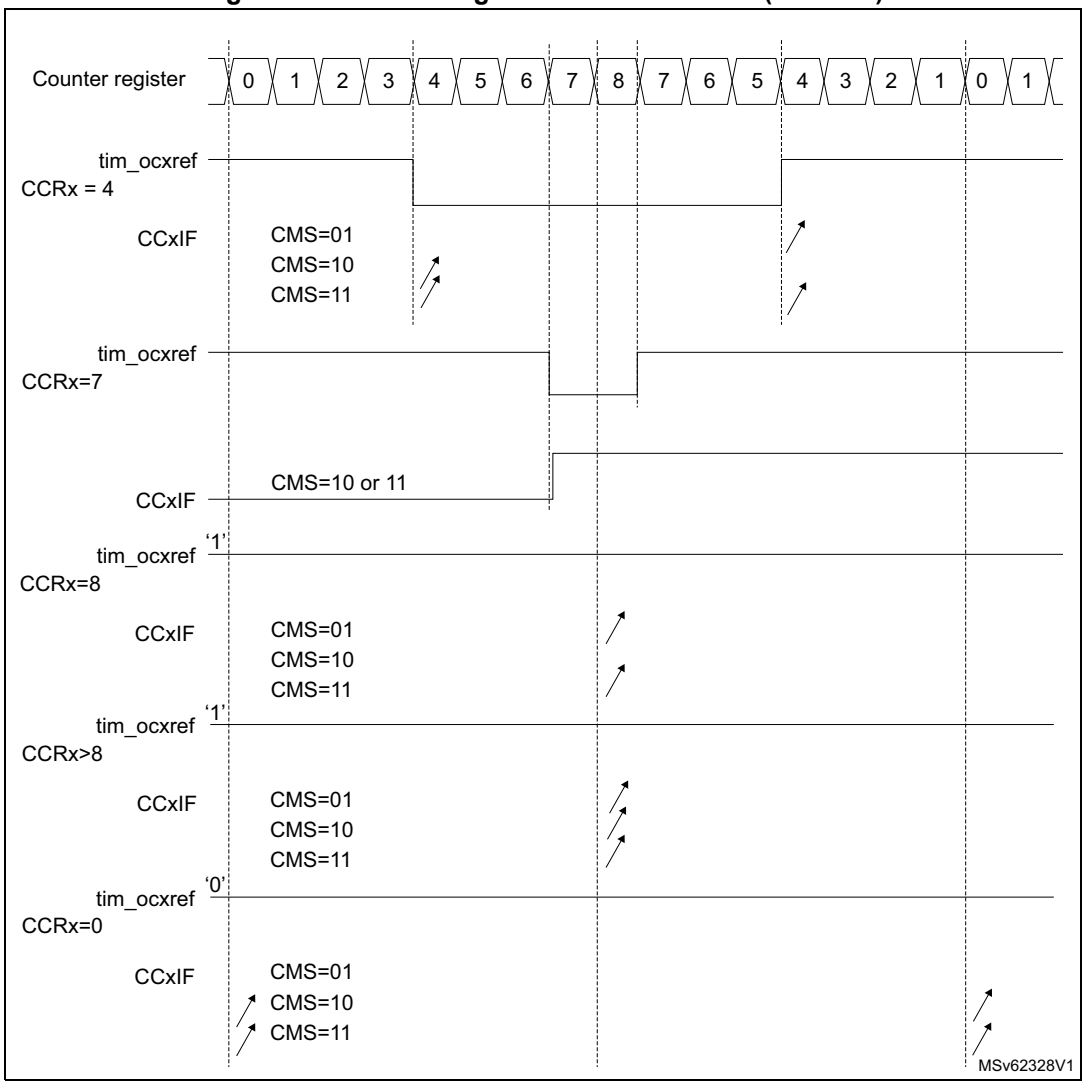
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending 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 282 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 282. 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.

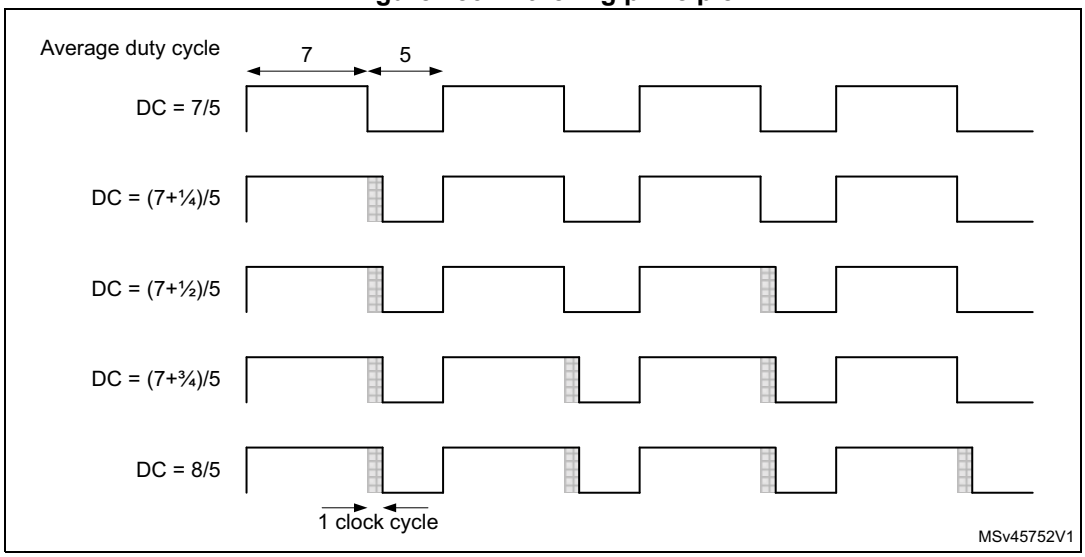
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, 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 283* presents the dithering principle applied to four consecutive PWM cycles.

Figure 283. Dithering principle



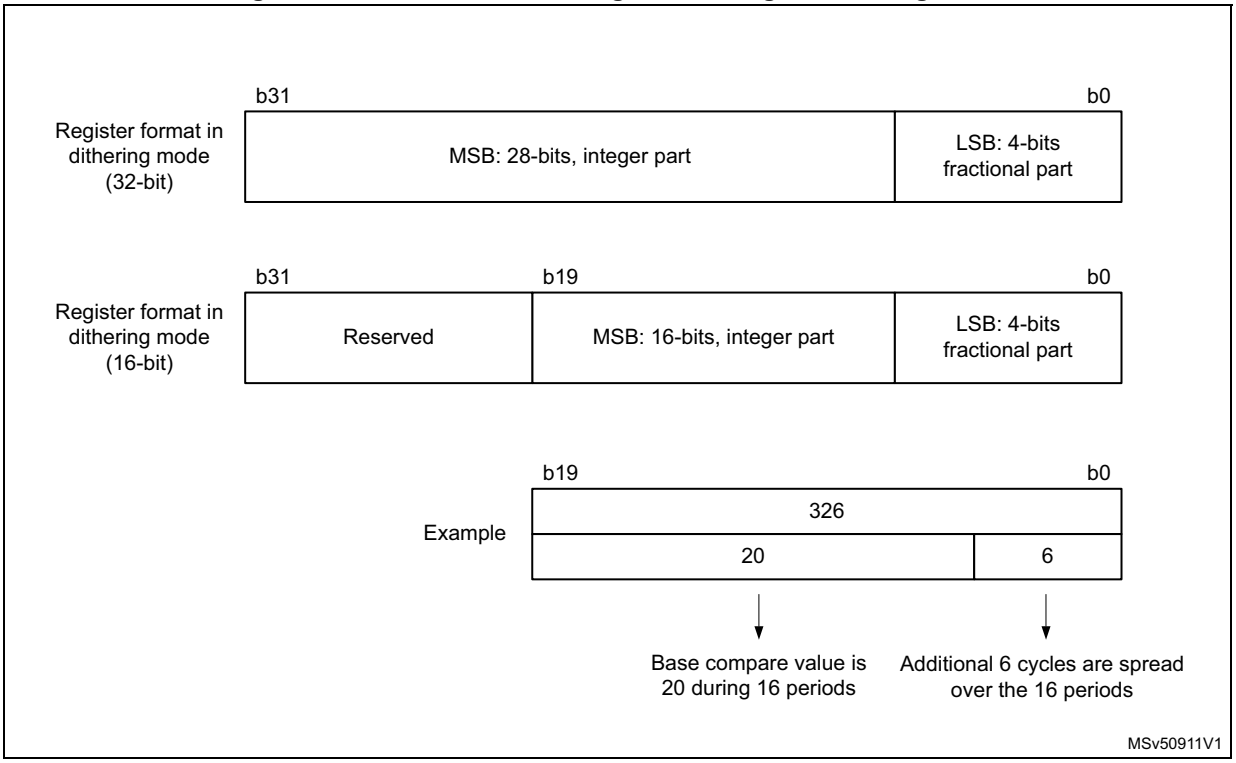
When the dithering mode is enabled, the register coding is changed as following (see *Figure 284* for example):

- 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 284. 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 TIMxCCRy 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 TIMxCCRy 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 285* and *Figure 286*, the dithering mode is used to increase the PWM resolution.

Figure 285. PWM resolution vs frequency (16-bit mode)

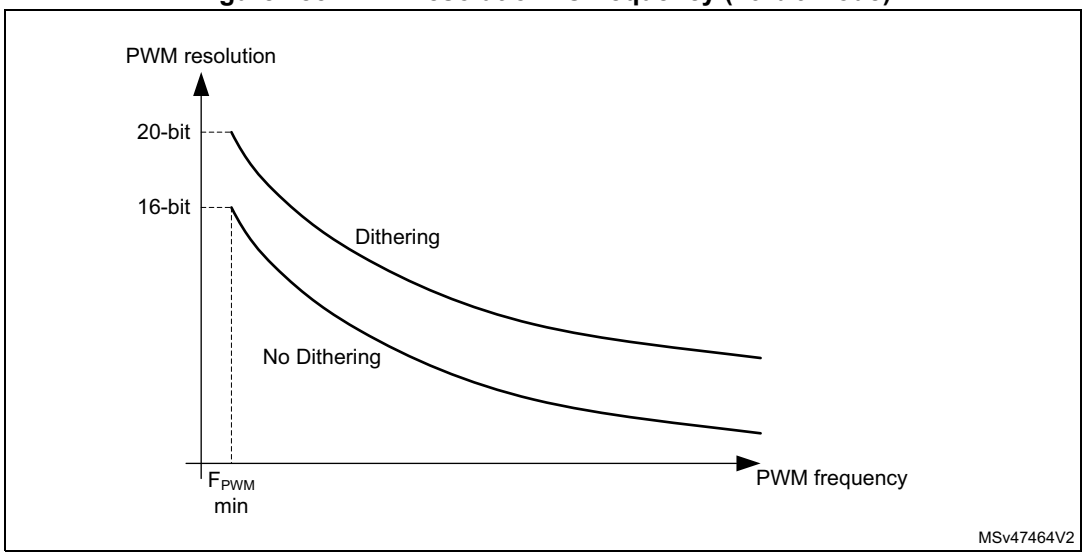
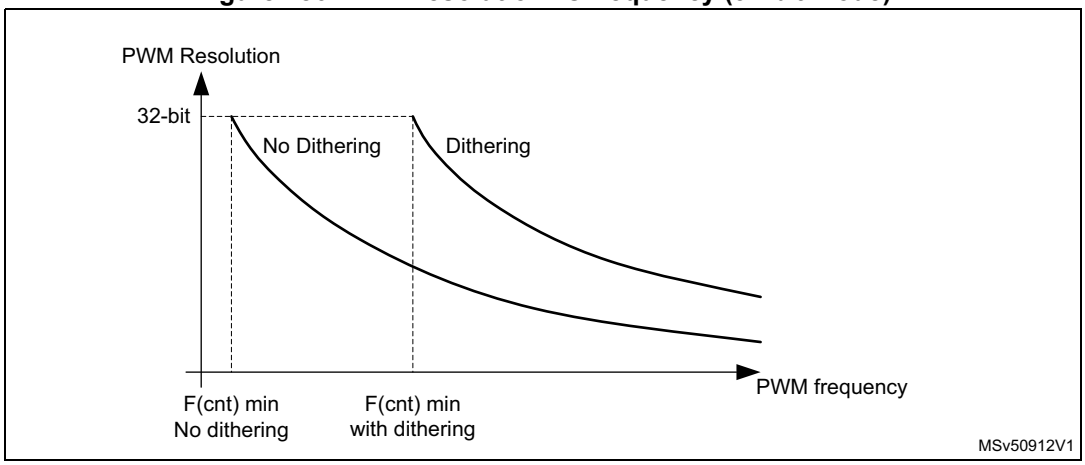
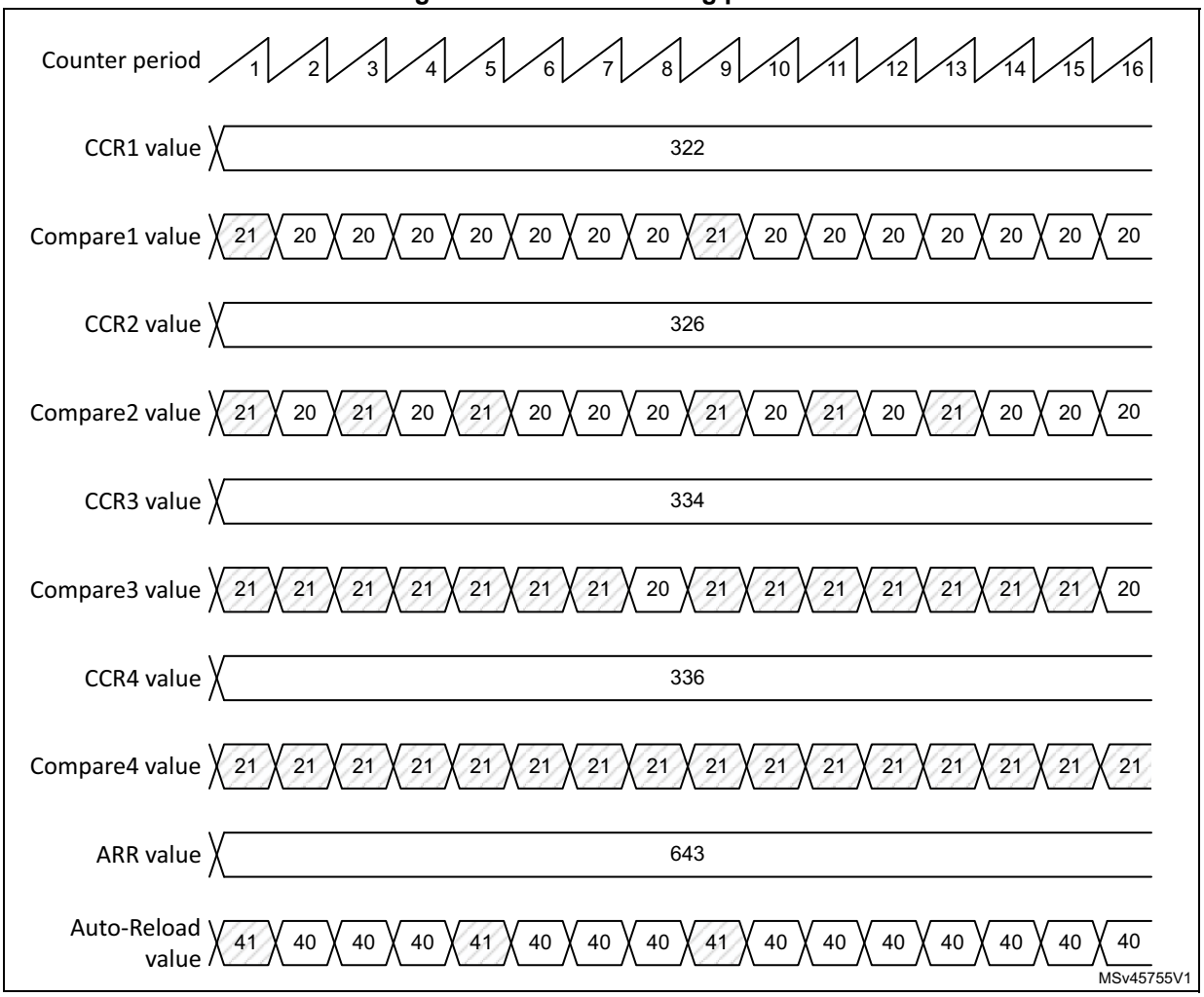


Figure 286. PWM resolution vs frequency (32-bit mode)



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 287*.

Figure 287. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 196*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 196. 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 288*.

Figure 288. Dithering effect on duty cycle in center-aligned PWM mode

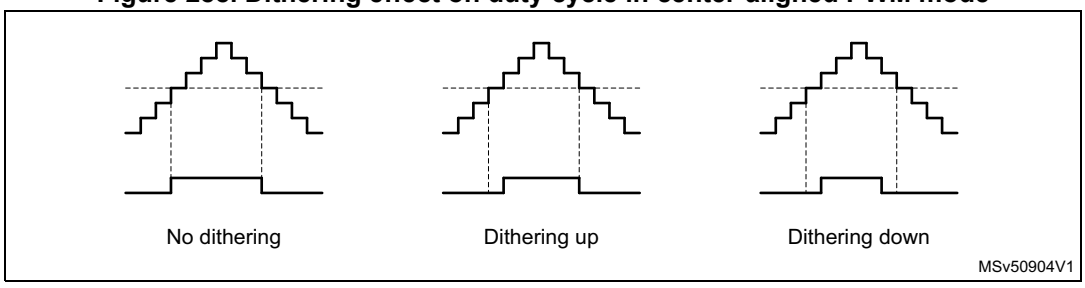


Table 197 shows how the dithering pattern is added in center-aligned PWM mode.

Table 197. 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	-

27.4.12 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.

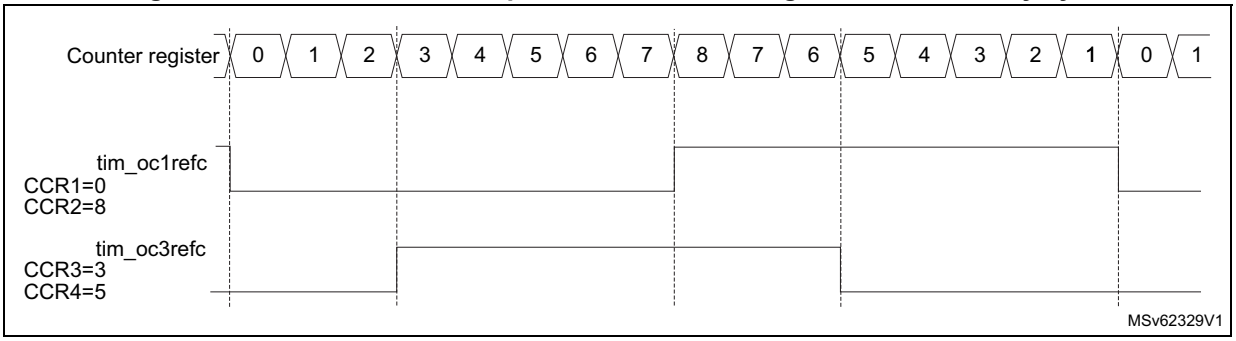
Asymmetric PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1110 (Asymmetric PWM mode 1) or 1111 (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

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 289 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 289. Generation of two phase-shifted PWM signals with 50% duty cycle



27.4.13 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.

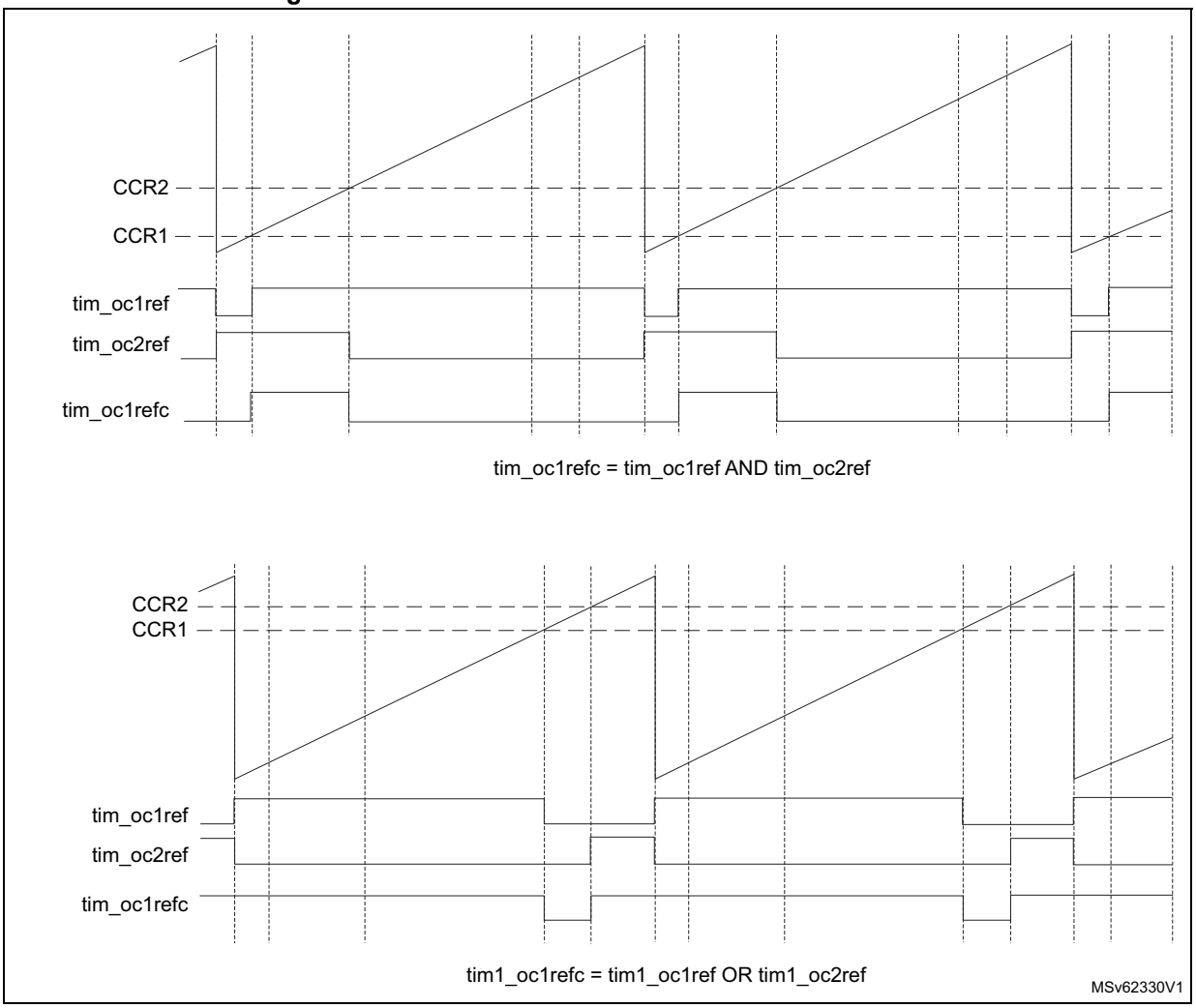
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

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 290 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 2.
- Channel 2 is configured in PWM mode 1.
- Channel 3 is configured in Combined PWM mode 2.
- Channel 4 is configured in PWM mode 1.

Figure 290. Combined PWM mode on channels 1 and 3



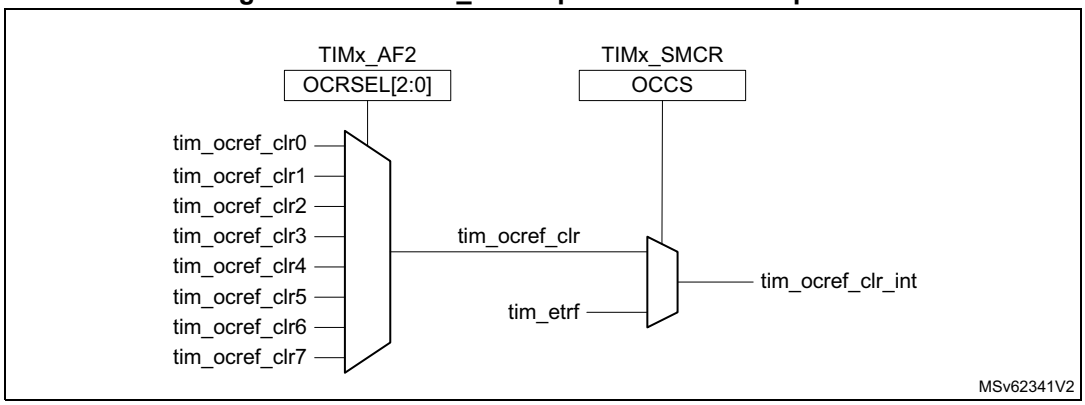
27.4.14 Clearing the tim_ocxref signal on an external event

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 27.3: TIM2/TIM3 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 291*.

Figure 291. OCREF_CLR input selection multiplexer



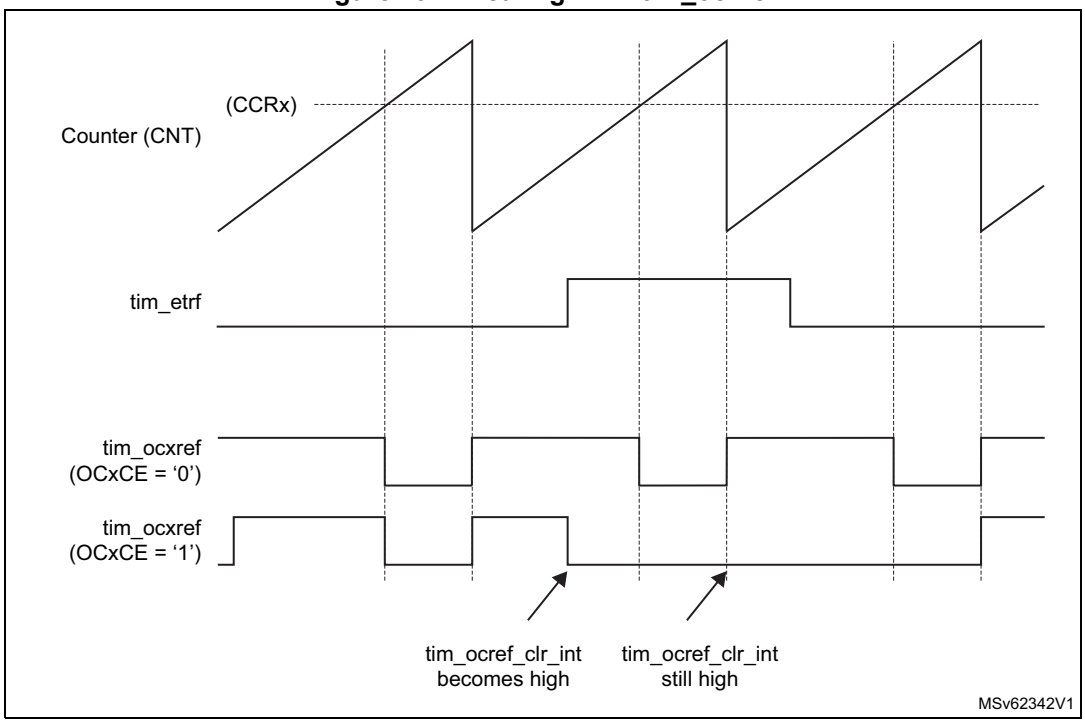
If the OCREF clear selection feature is not implemented, the tim_ocref_clr_int input is directly connected to the tim_etrf input.

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 TIM1_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 292 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 292. 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.

27.4.15 One-pulse mode

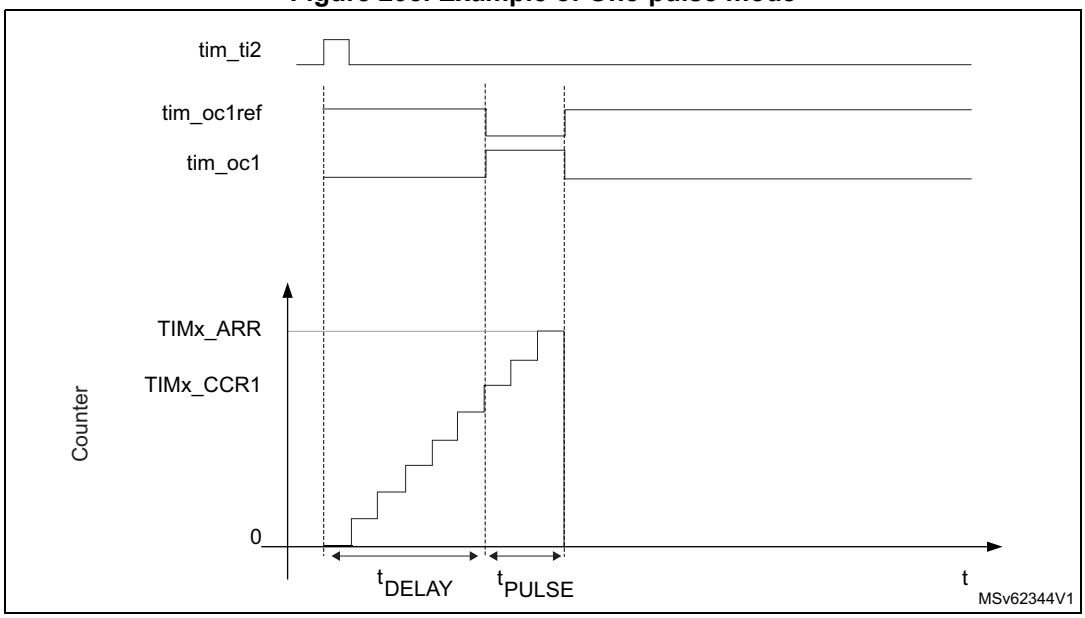
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

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 293. 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.

27.4.16 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 27.4.15*:

- 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.

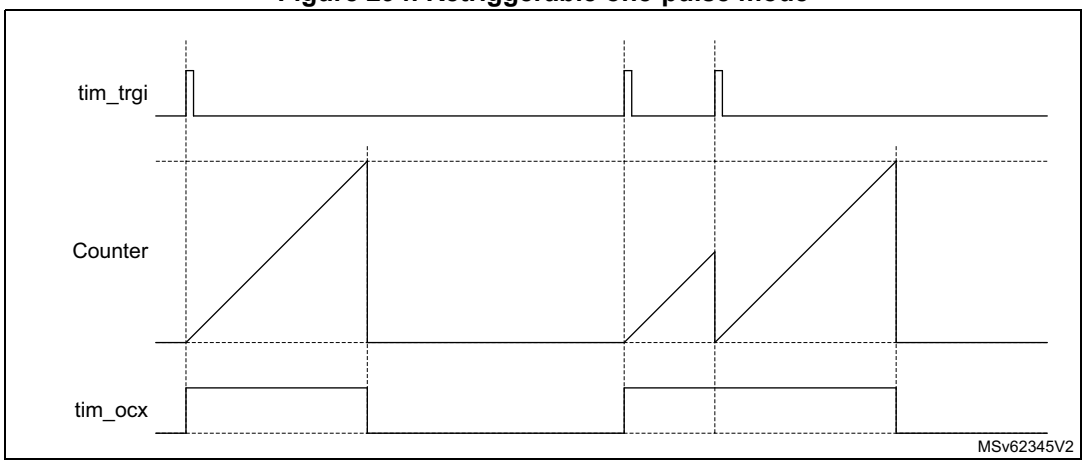
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode CCRx must be above or equal to ARR.

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 294. Retriggerable one-pulse mode



27.4.17 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 295*.

Figure 295. Pulse generator circuitry

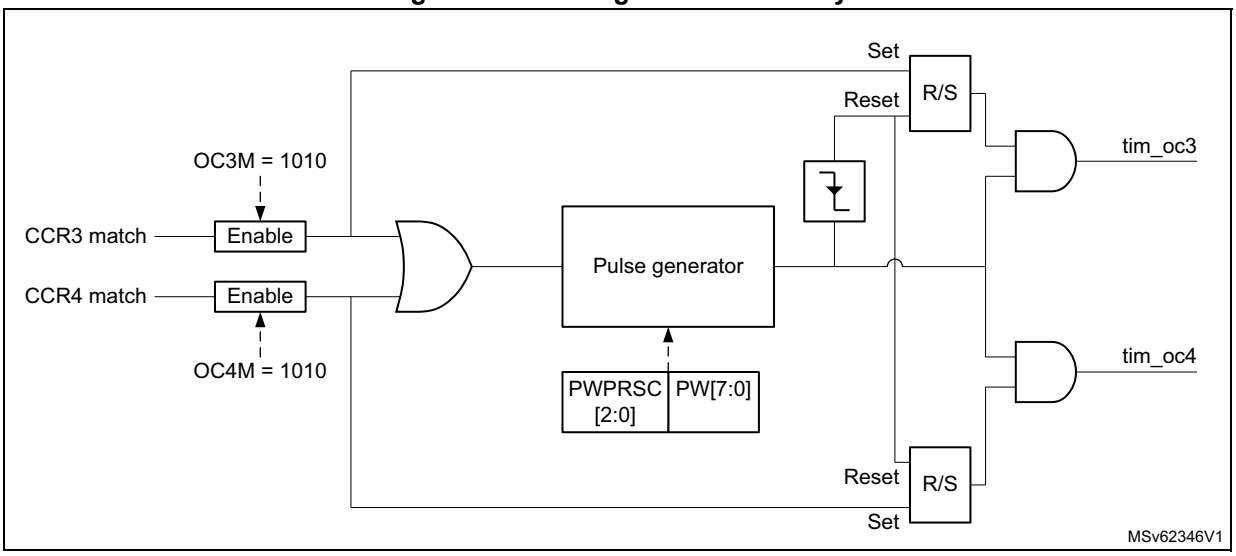
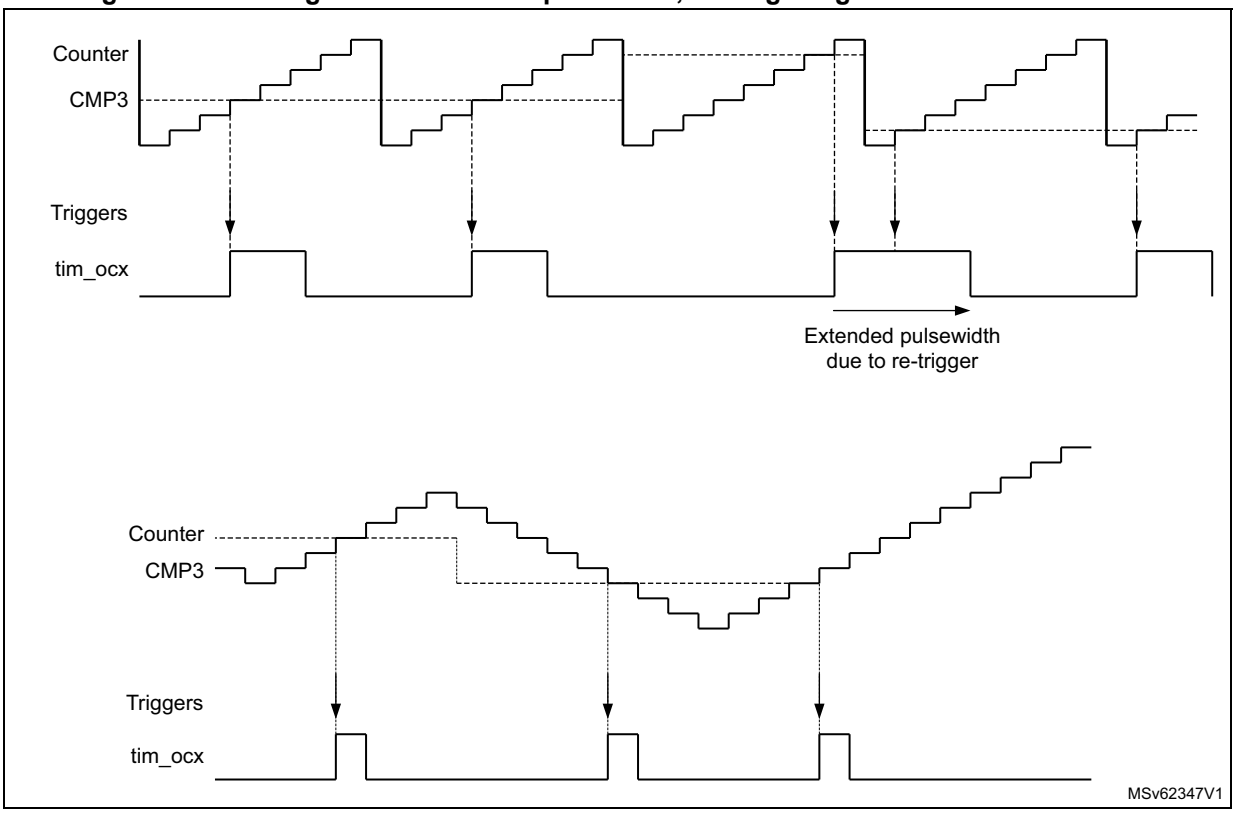


Figure 296 shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 296. 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:

$t_{PW} = PW[7:0] \times t_{PWG}$

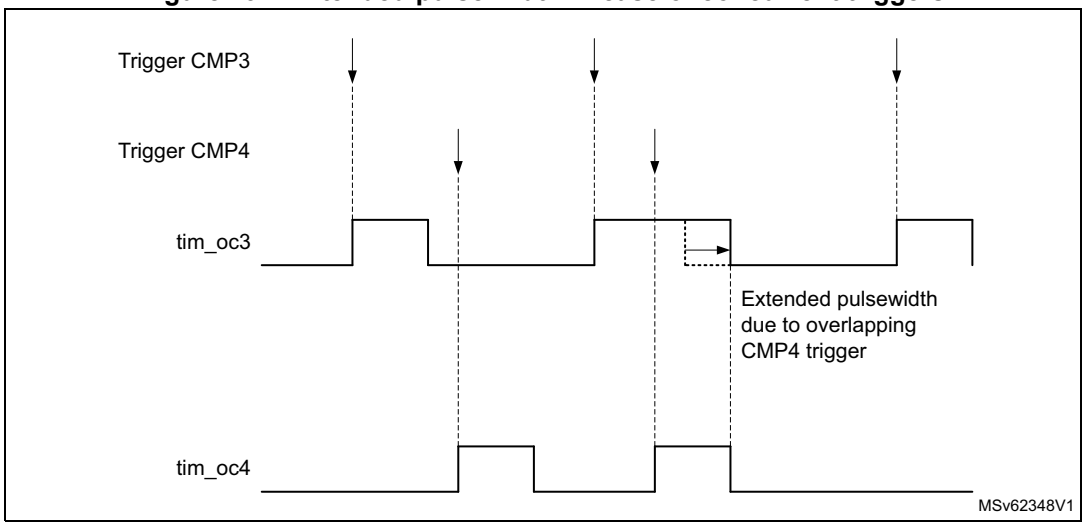
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim\_ker\_ck}$.

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 297).

Figure 297. Extended pulse width in case of concurrent triggers



27.4.18 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 198*. 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 198. 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 298 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 298. Example of counter operation in encoder interface mode

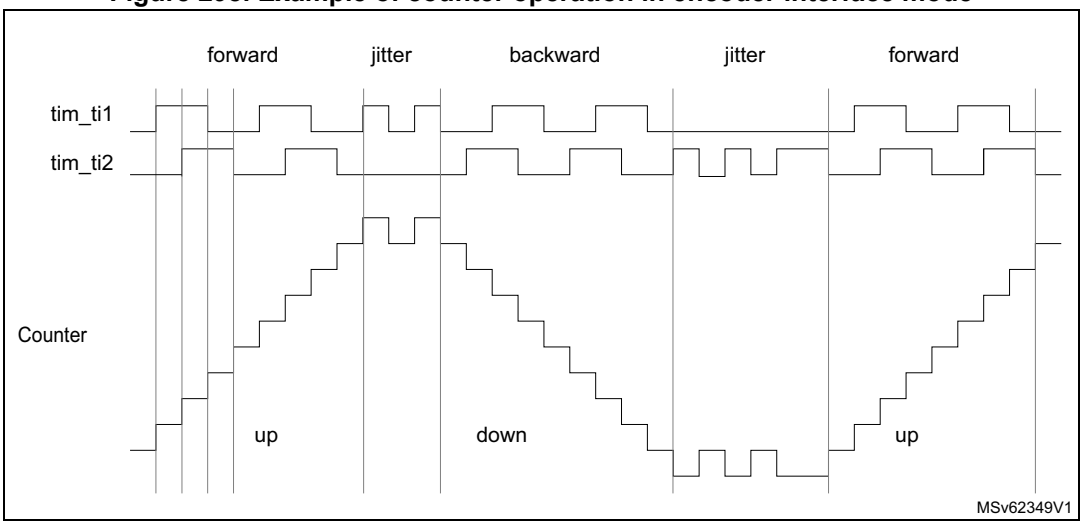


Figure 299 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 299. Example of encoder interface mode with tim_ti1fp1 polarity inverted

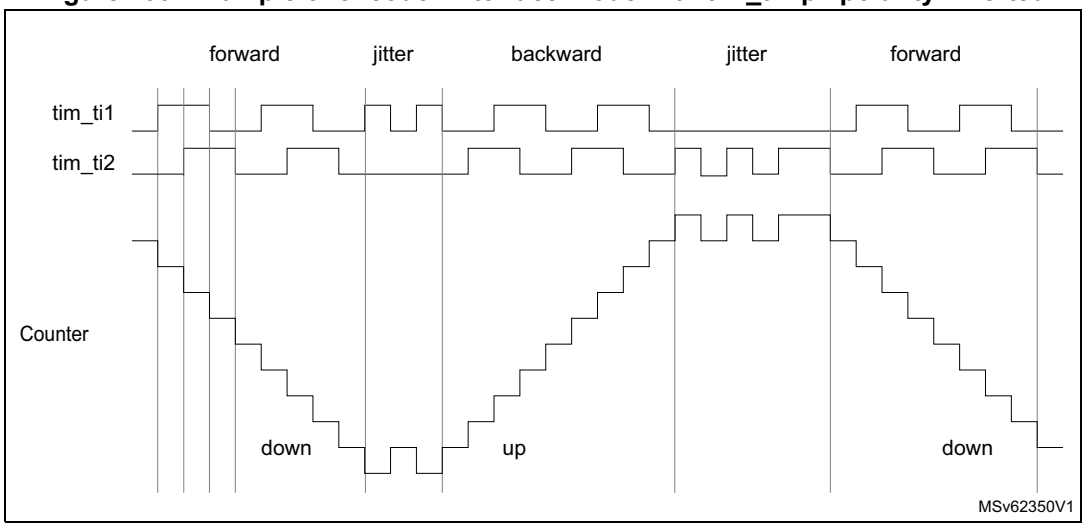
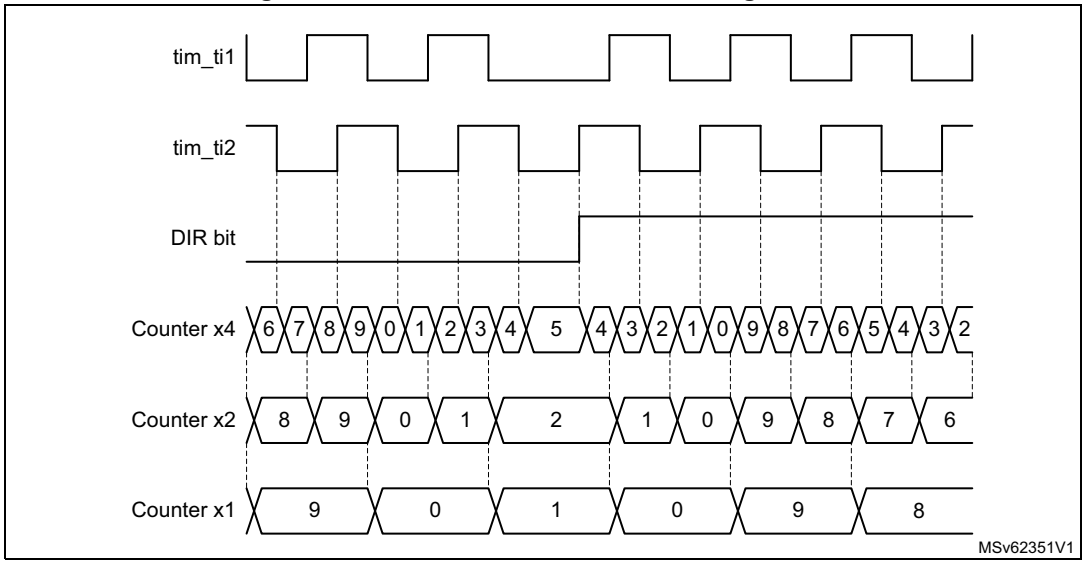


Figure 300 shows the timer counter value during a speed reversal, for various counting modes.

Figure 300. 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 (TIMxCNT[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.

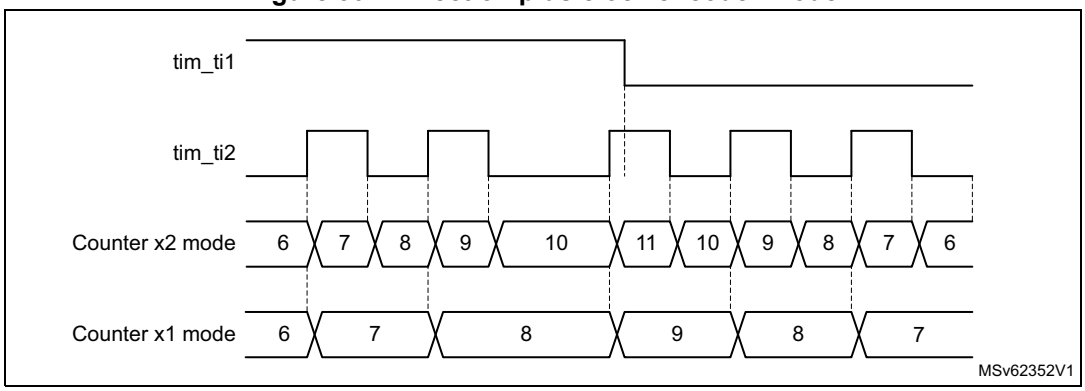
In the "clock plus direction" mode shown on *Figure 301*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

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 301. Direction plus clock encoder mode



Directional clock encoder mode

In the “directional clock” mode on *Figure 302*, 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 302*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 303*).
- 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 302*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 303*).

Figure 302. Directional clock encoder mode (CC1P = CC2P = 0)

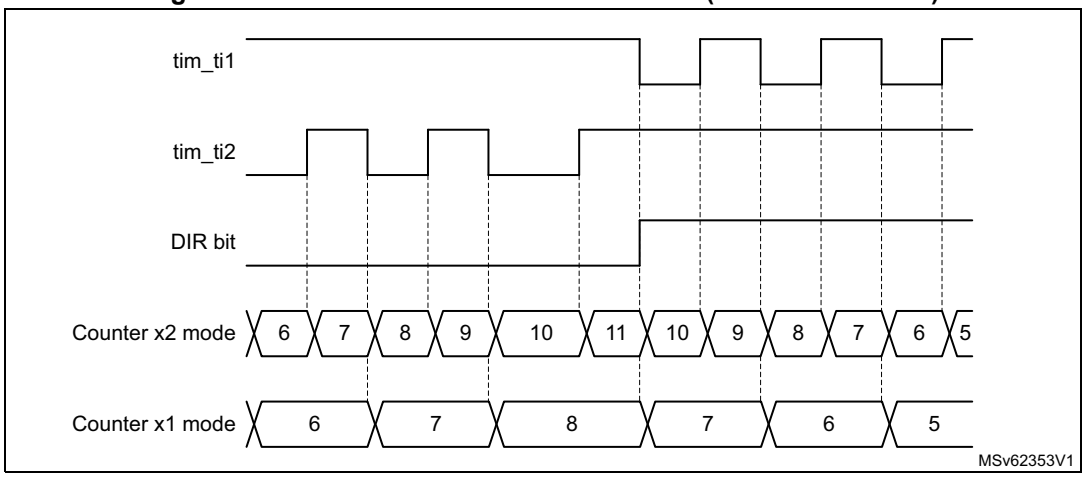


Figure 303. Directional clock encoder mode (CC1P = CC2P = 1)

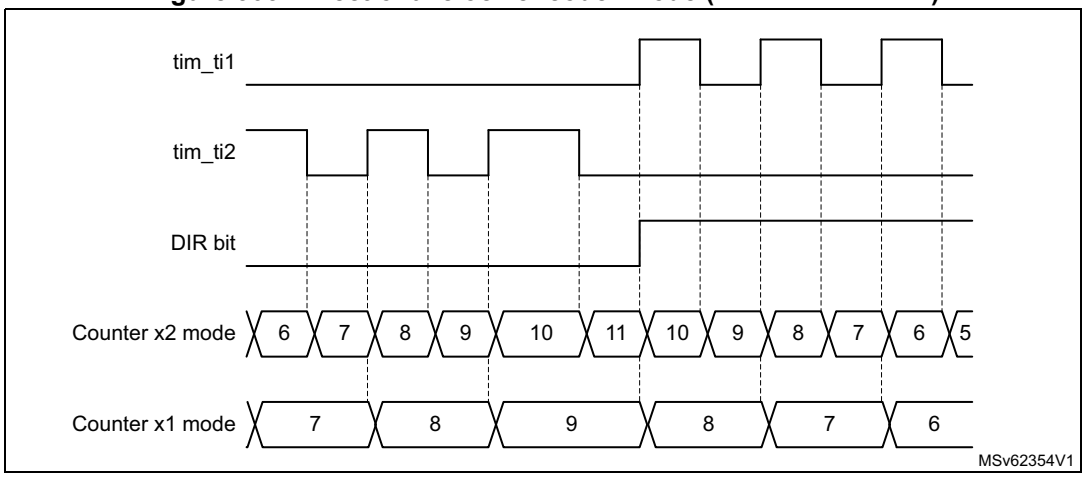


Table 199 details how the directional clock mode operates, for any input transition.

Table 199. 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

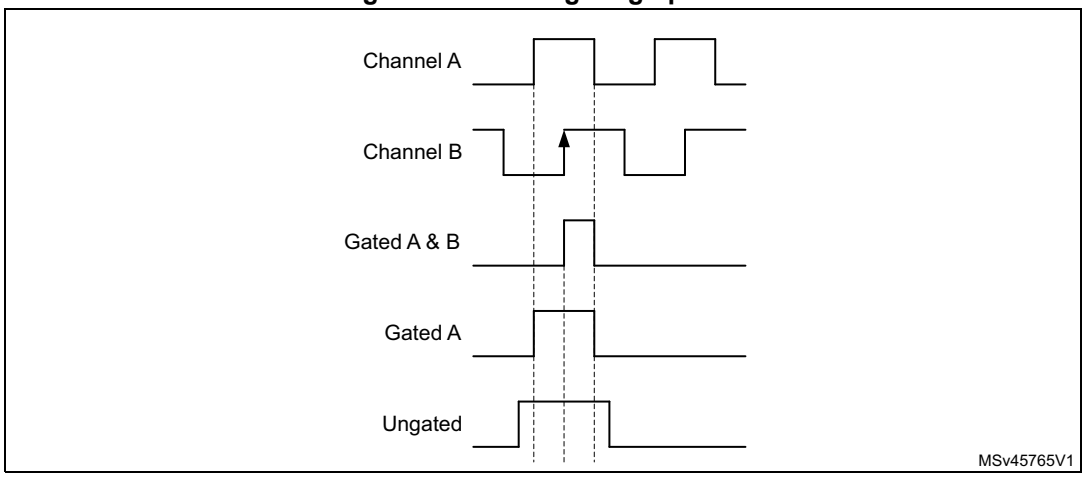
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

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 304*:

- 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.

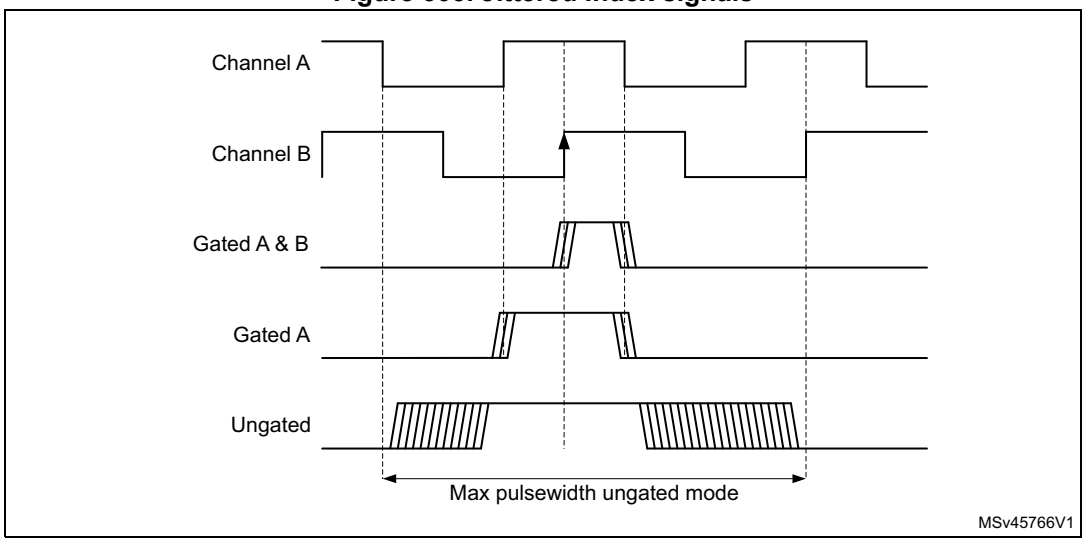
Figure 304. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as shown on *Figure 305*.

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 305. 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 306* shows at which position is the index generated, for a simplistic example (an encoder providing four edges par mechanical rotation).

Figure 306. Index generation for IPOS[1:0] = 11

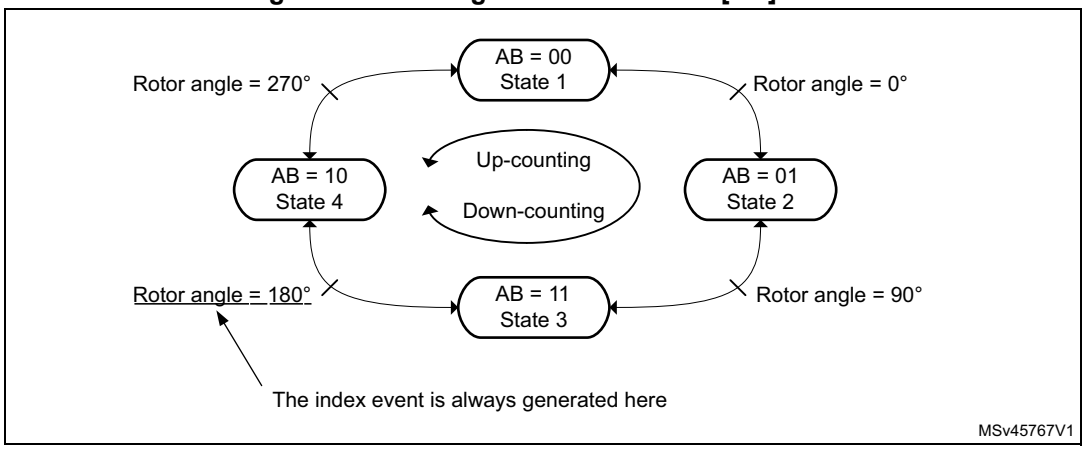


Figure 307 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- 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 307. Counter reading with index gated on channel A (IPOS[1:0] = 11)

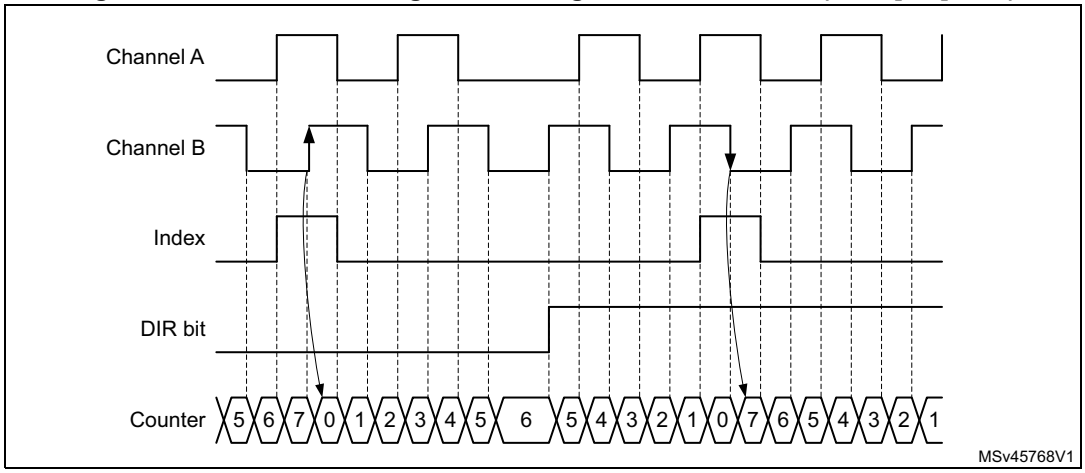


Figure 308 presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 308. Counter reading with index ungated (IPOS[1:0] = 00)

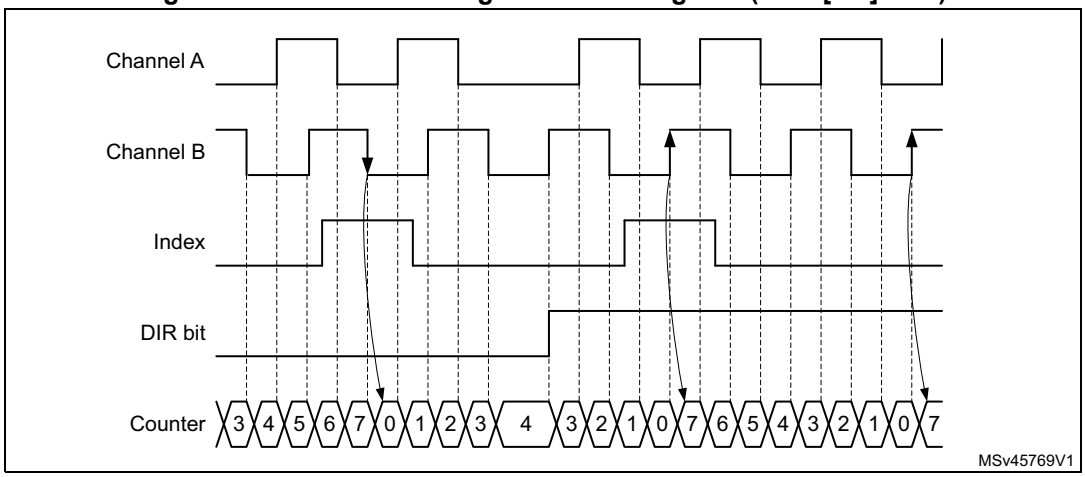


Figure 309 shows how the gated on A & B mode is handled, for various pulse alignment scenarios. The arrows are indicating on which transition is the index event generated.

Figure 309. Counter reading with index gated on channel A and B

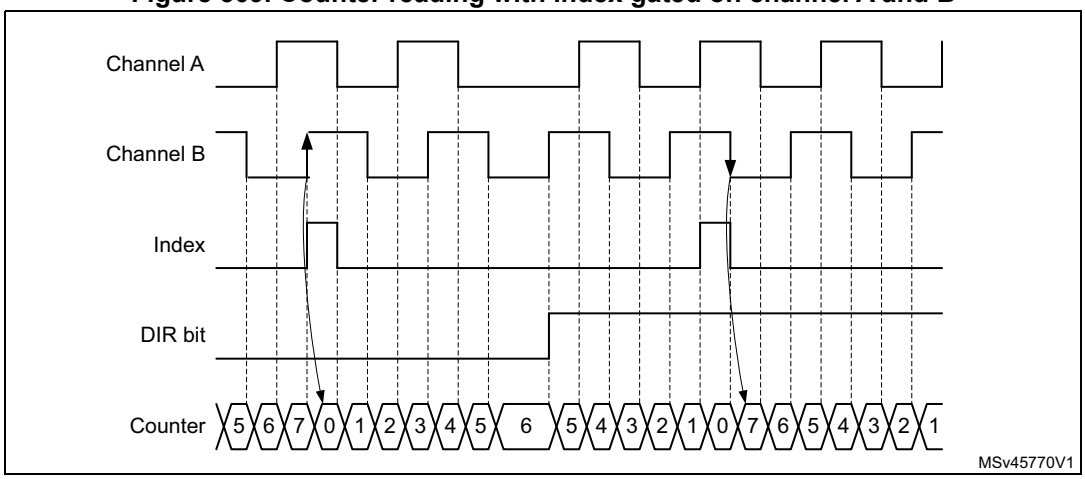


Figure 310 and *Figure 311* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 310. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

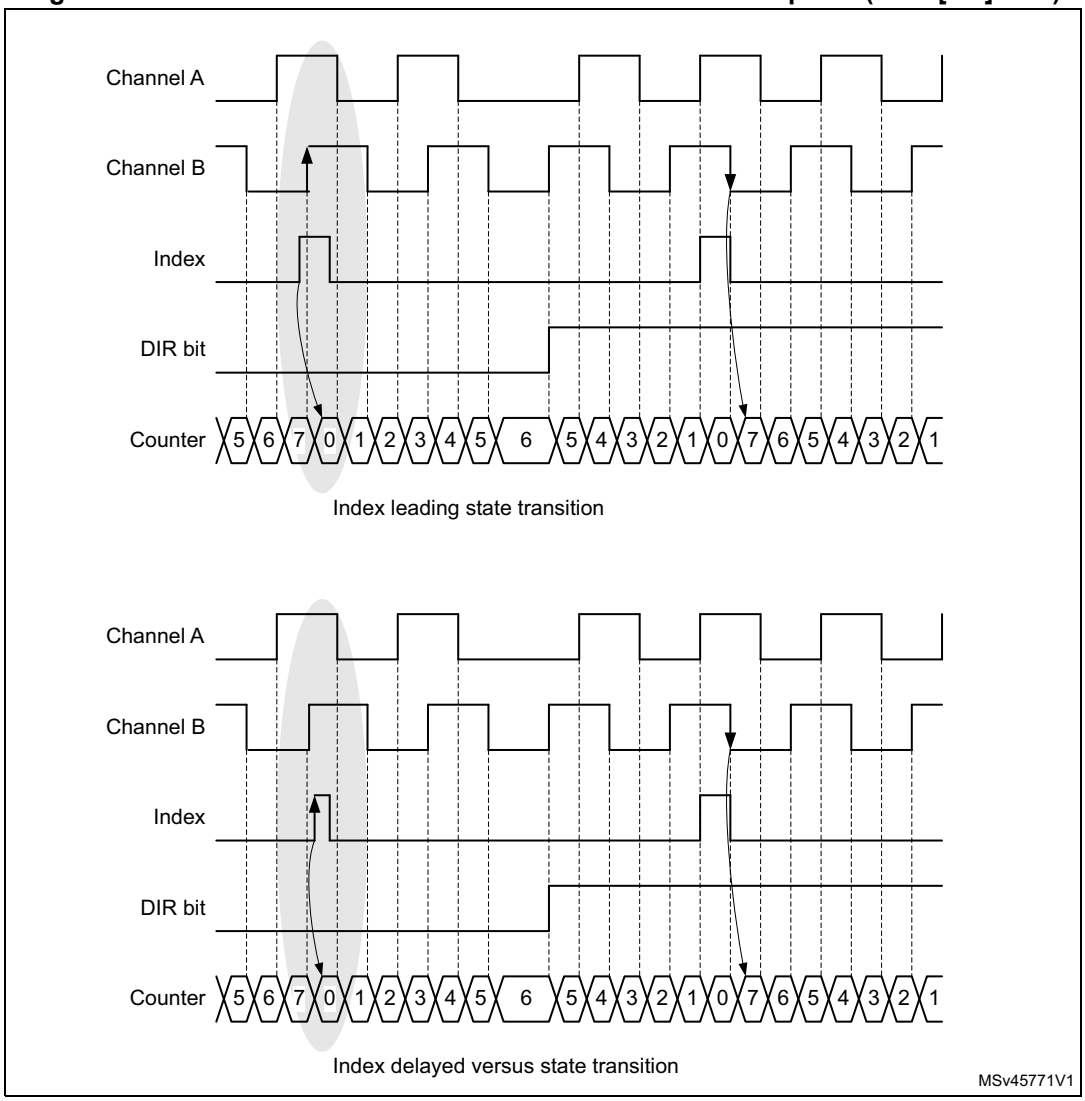


Figure 311. Counter reset Narrow index pulse (closer view, ARR = 0x07)

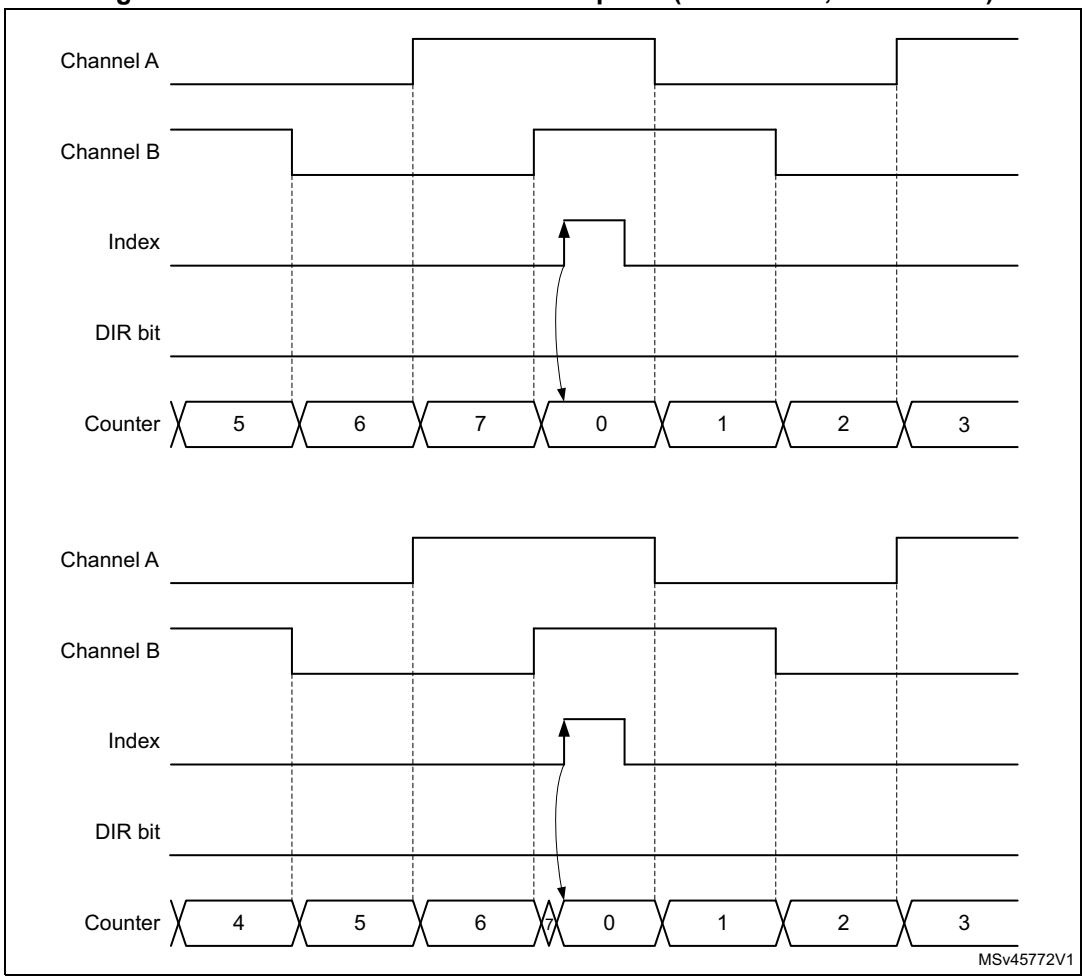
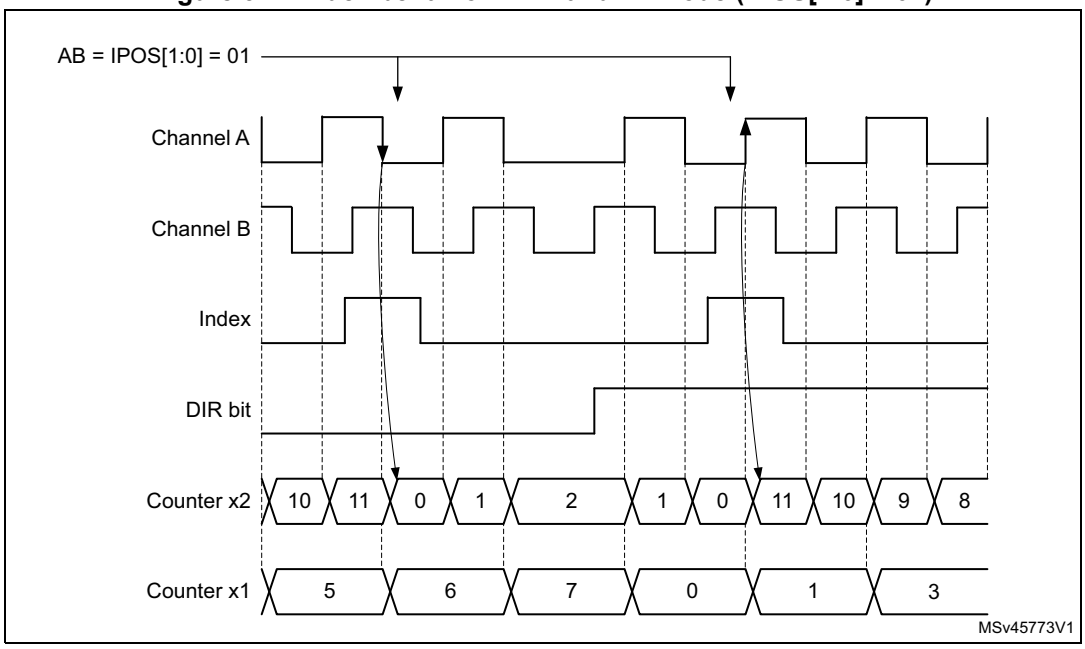


Figure 312 shows how the index is managed in x1 and x2 modes.

Figure 312. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)

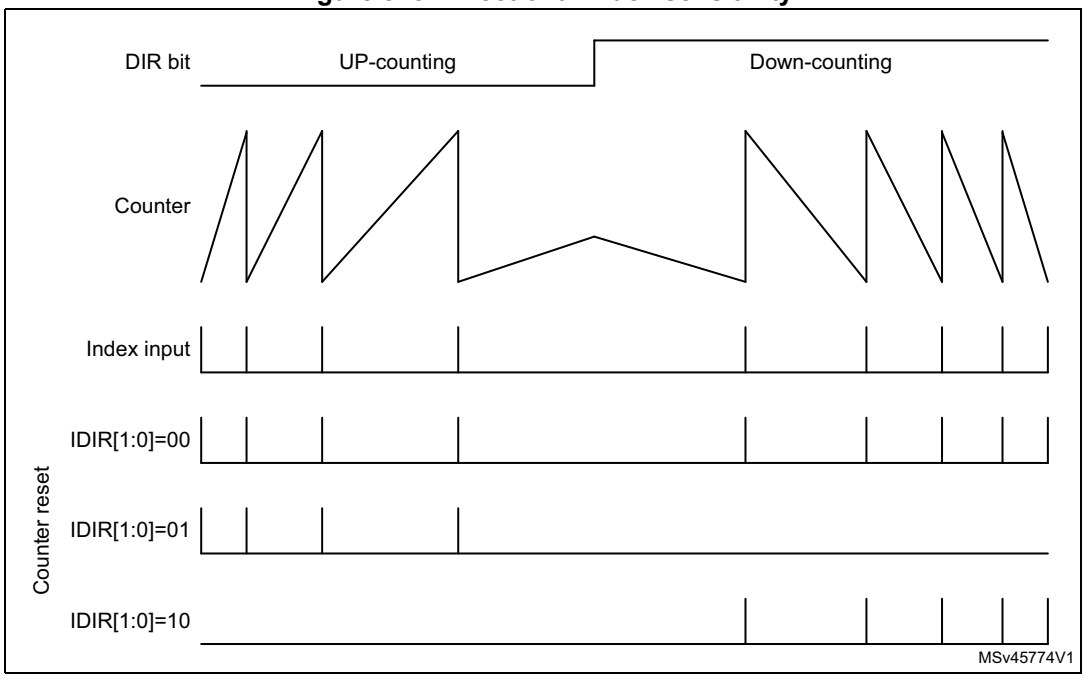


Directional index sensitivity

The IDIR[1:0] bitfield in the TIMx_ECR register allows the index to be active only in a selected counting direction.

Figure 313 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

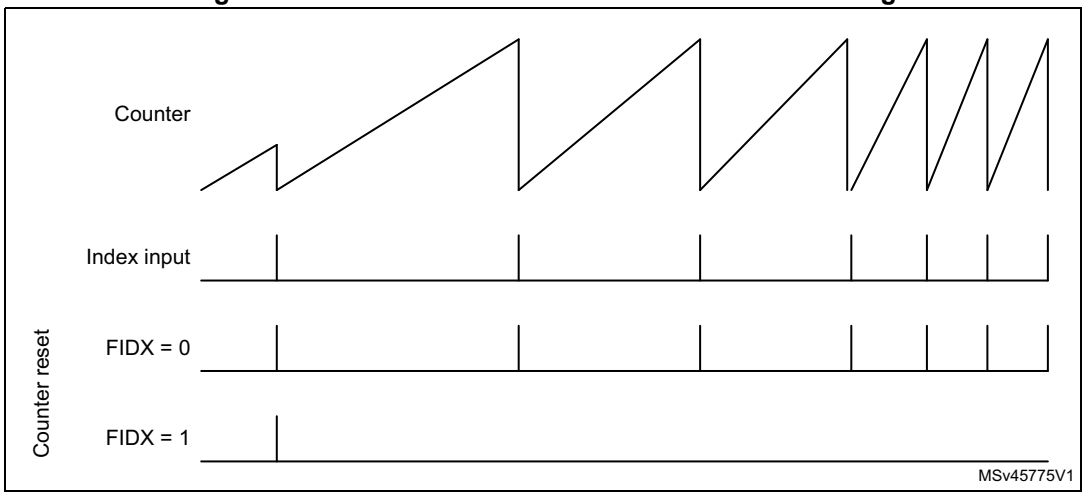
Figure 313. Directional index sensitivity



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 314*. 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 314. Counter reset as function of FIDX bit setting



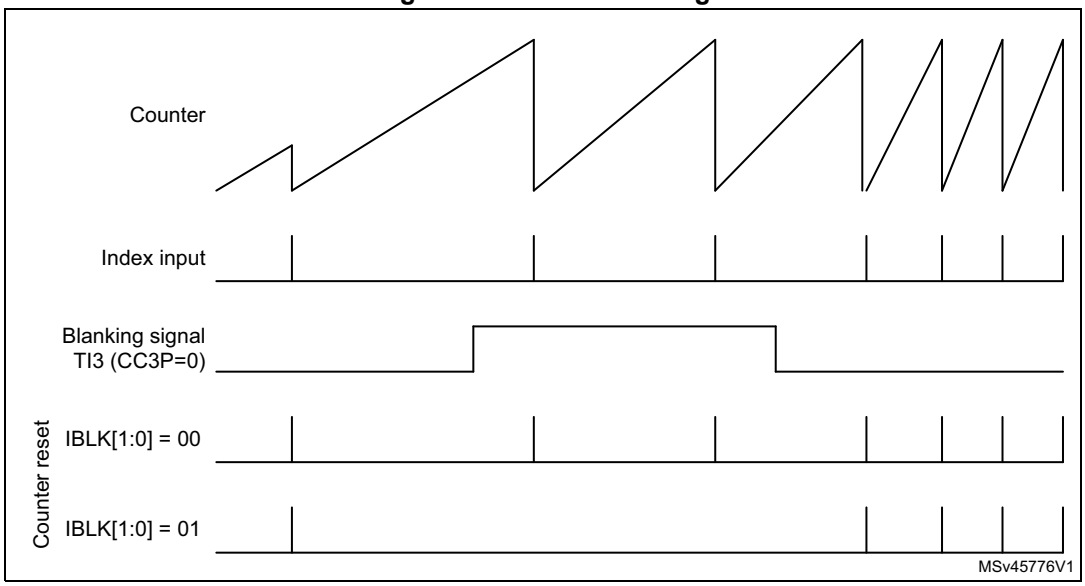
Index blanking

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 315*.

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 315. Index blanking



Index management in nonquadrature mode

Figure 316 and *Figure 317* 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 316. Index behavior in clock + direction mode, IPOS[0] = 1

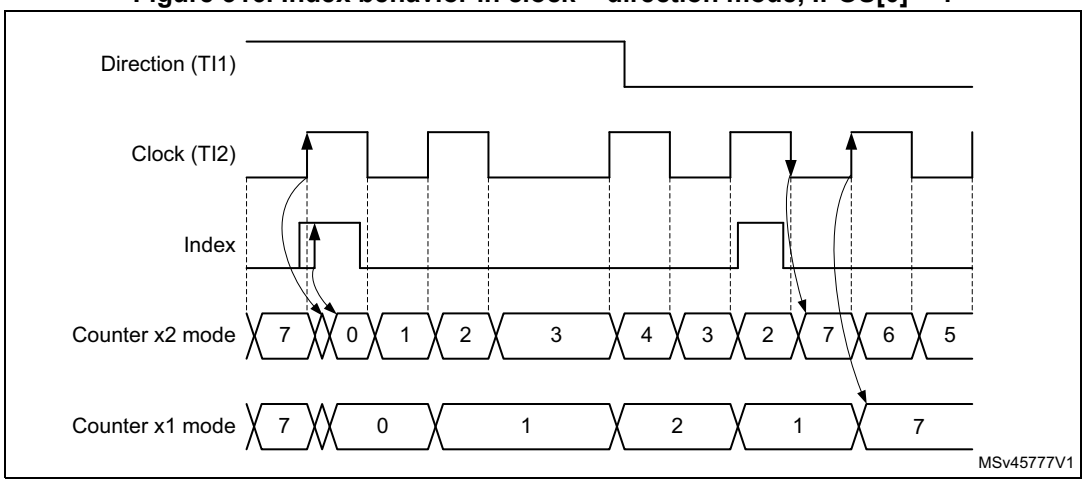
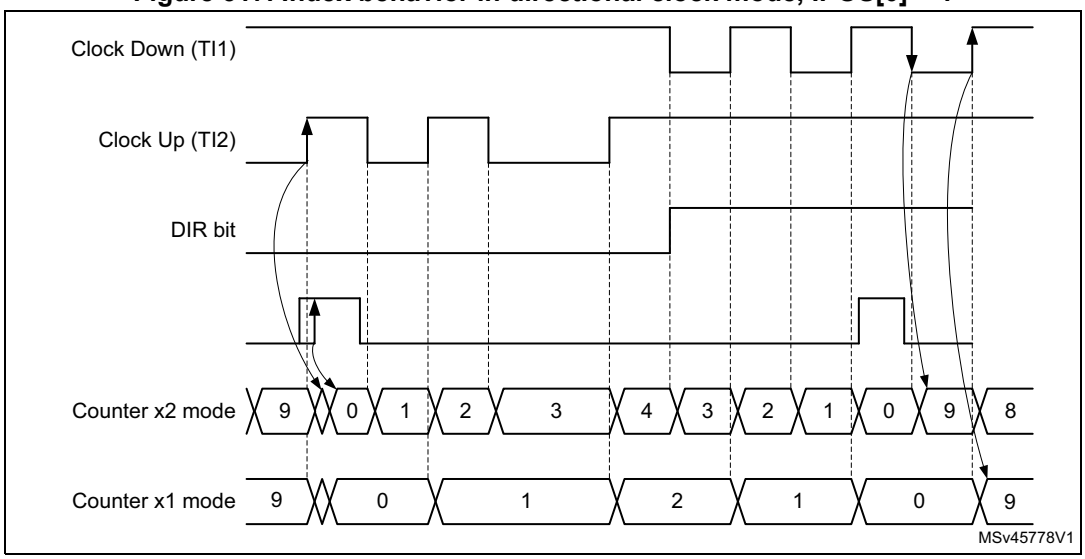


Figure 317. Index behavior in directional clock mode, IPOS[0] = 1

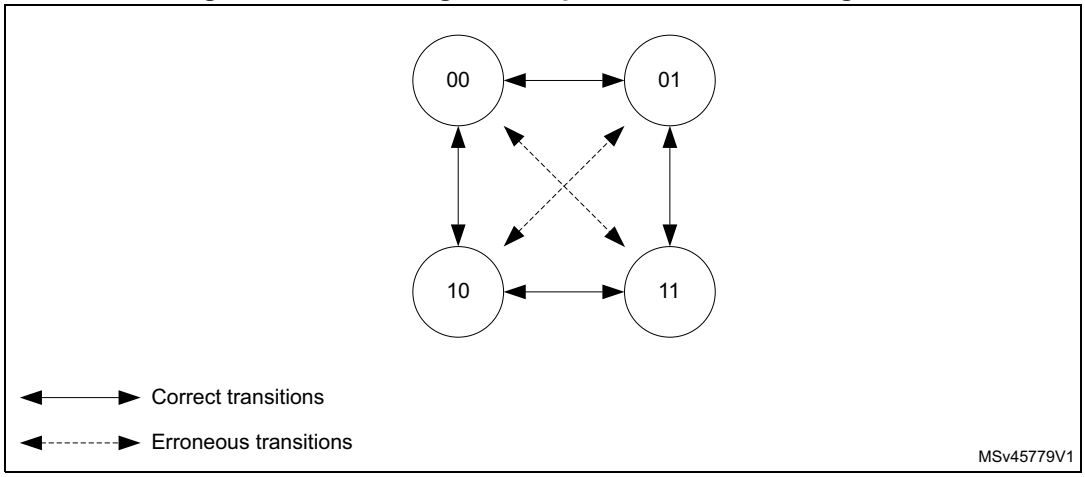


Encoder error management

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 318*. 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 318. State diagram for quadrature encoded signals



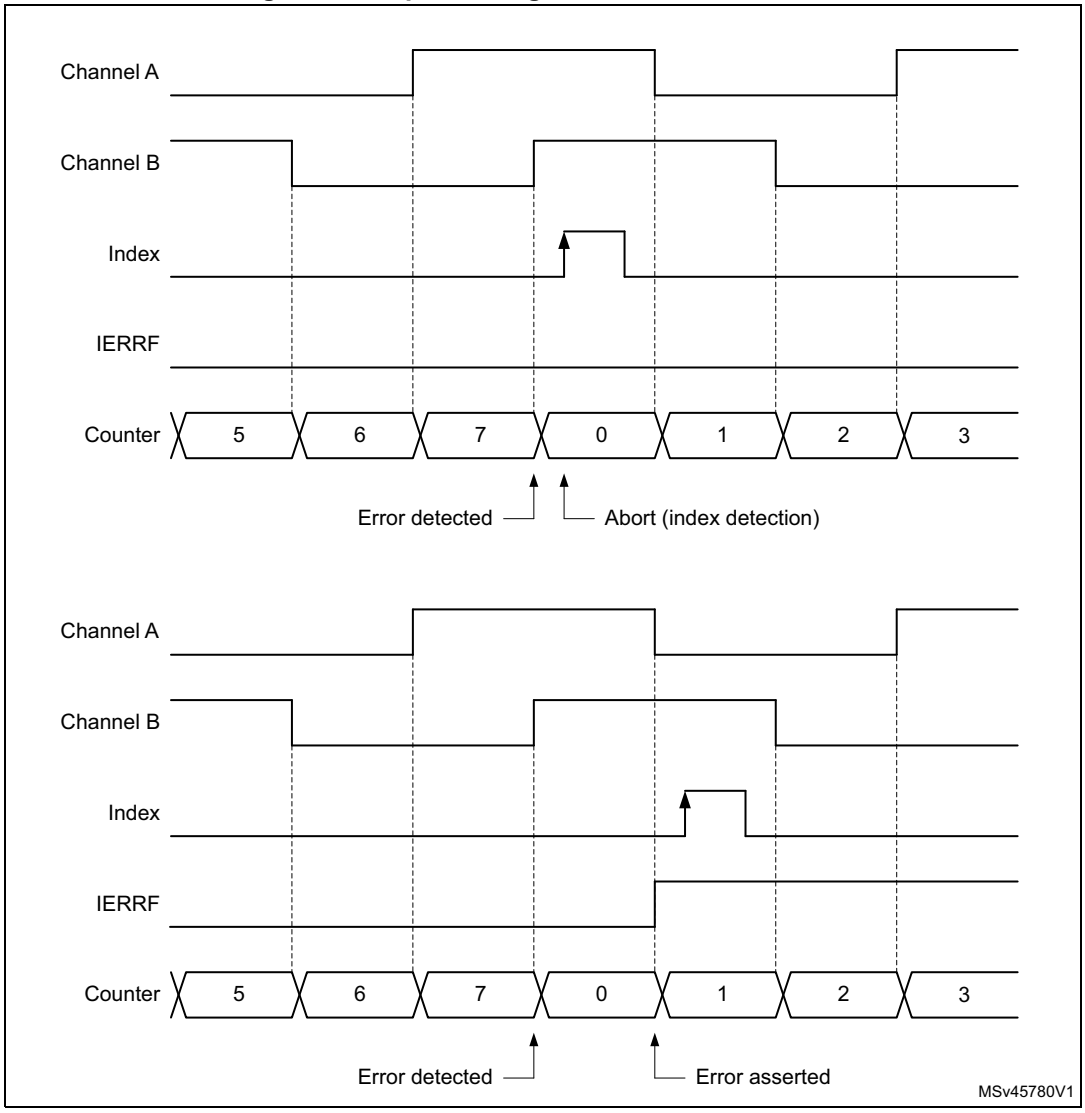
For encoder having an index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIMx_ARR to 0 or decremented from 0 to TIMxARR 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.

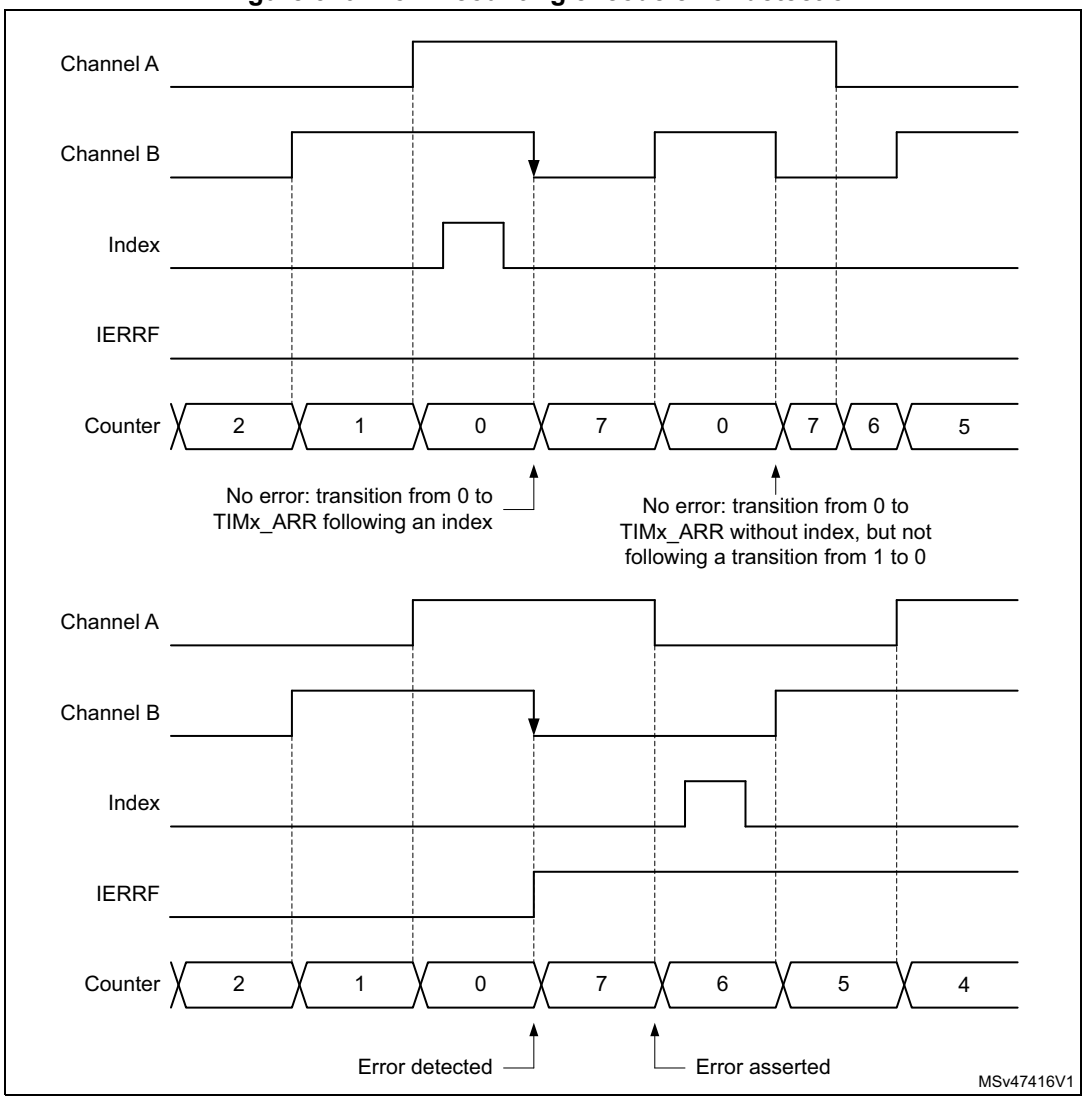
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 319*.

Figure 319. 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 320*, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

Figure 320. 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.

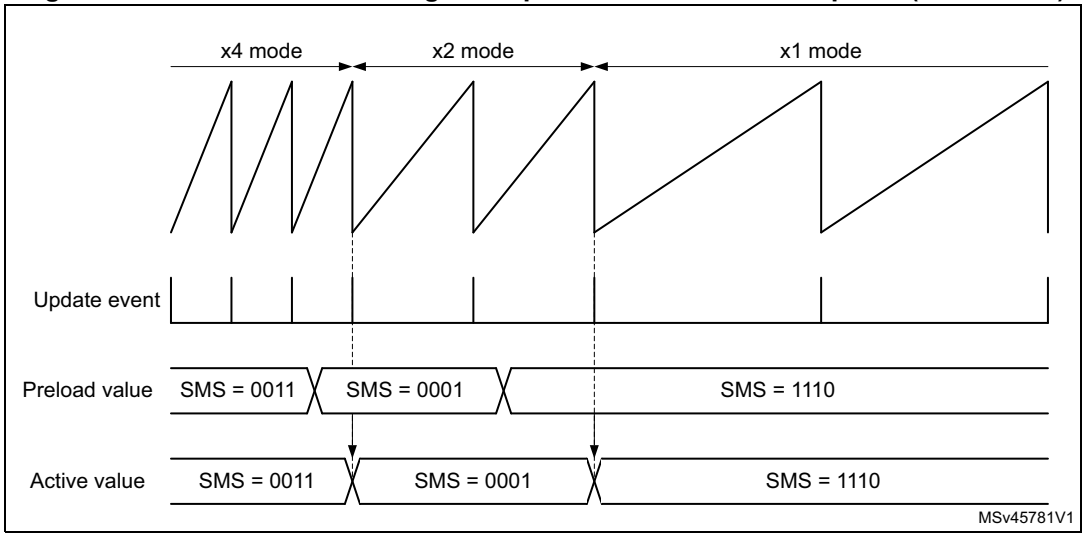
Slave mode selection preload for run-time encoder mode update

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 321*.

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 321. 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.

27.4.19 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.

27.4.20 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 (TIMxCNT[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).

27.4.21 Timer input XOR function

The TI1S bit in the TIM1xx_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.

An example of this feature used to interface Hall sensors is given in *Section 26.3.29: Interfacing with Hall sensors*.

27.4.22 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:

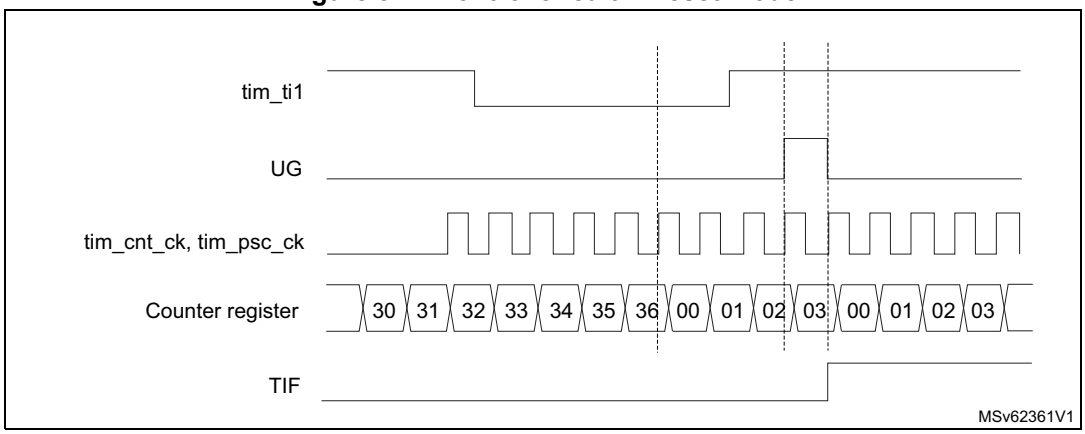
1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source 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 322. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

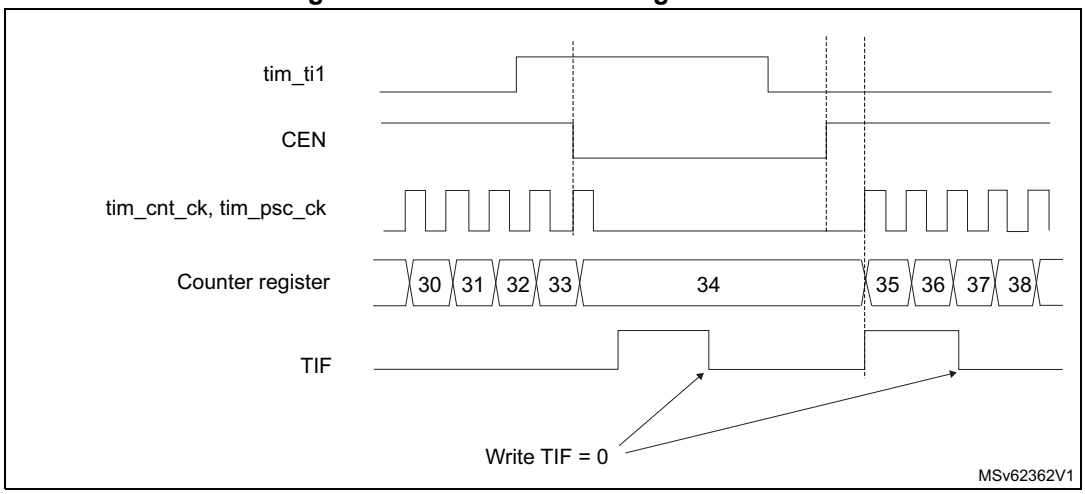
In the following example, the upcounter counts only when tim_ti1 input is low:

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 323. 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.

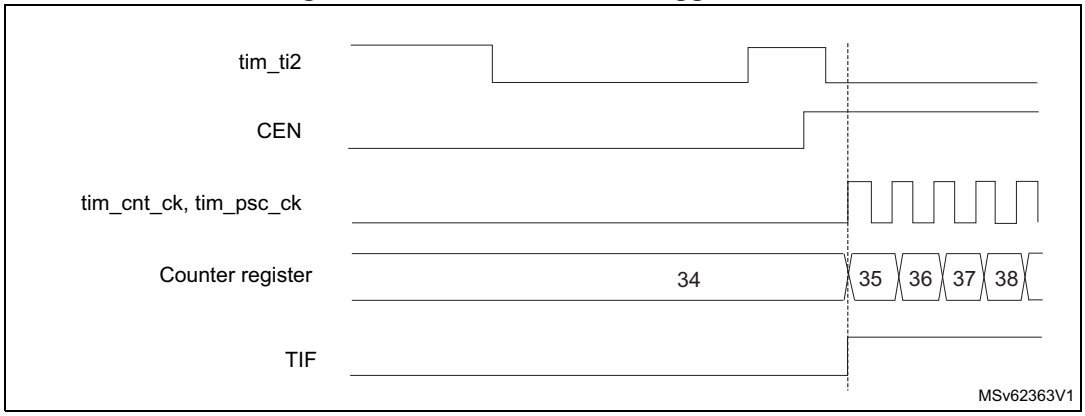
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

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 324. 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.

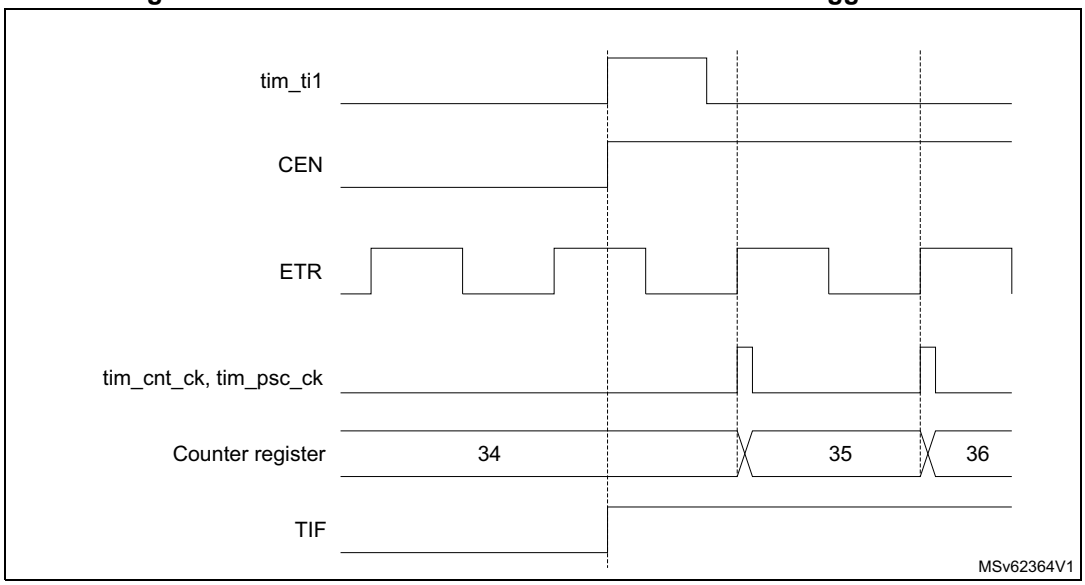
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

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 325. Control circuit in external clock mode 2 + trigger mode



27.4.23 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 326 and *Figure 327* show examples of master/slave timer connections.

Figure 326. Master/Slave timer example

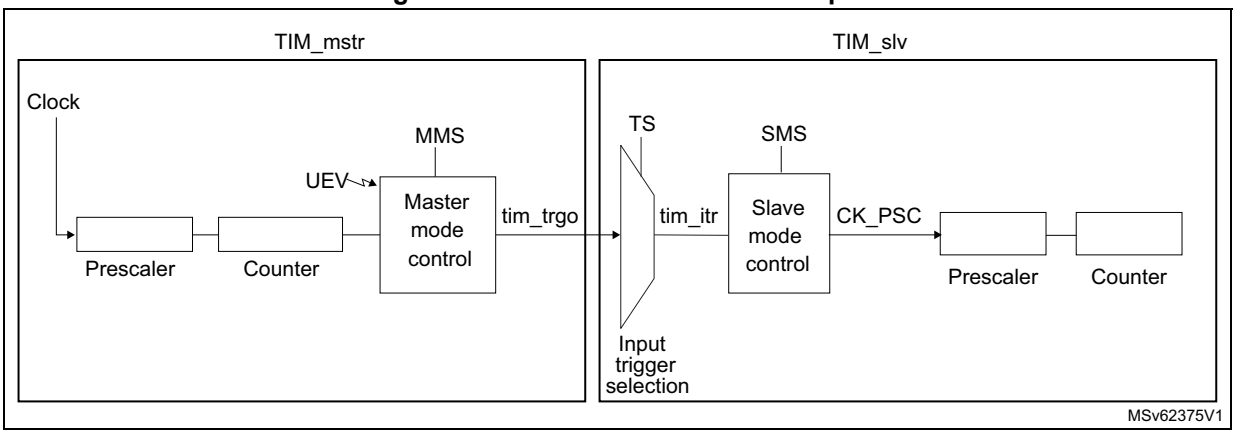
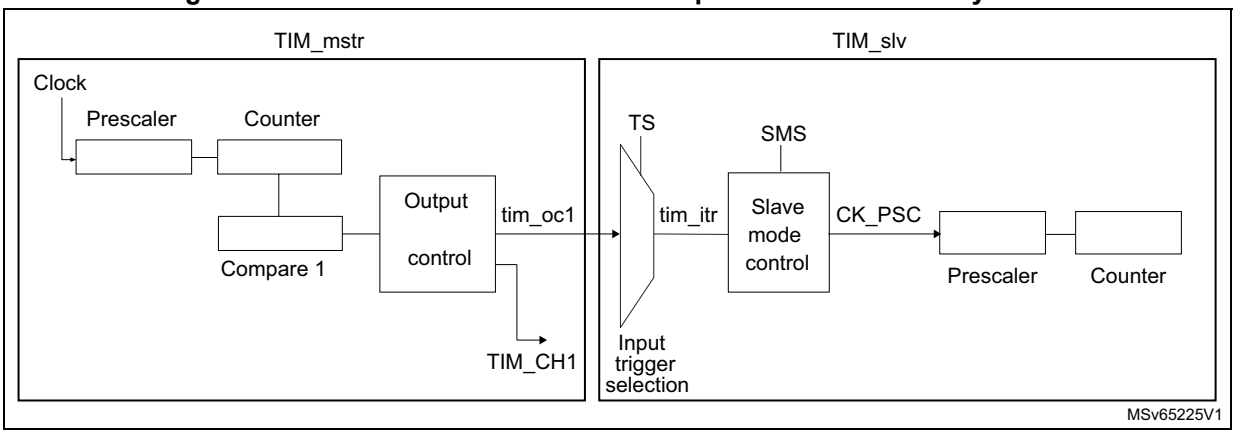


Figure 327. Master/slave connection example with 1 channel only timers



Note: The timers with one channel only (see Figure 327) 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 27.4.2: TIM2/TIM3 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 326*. 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.

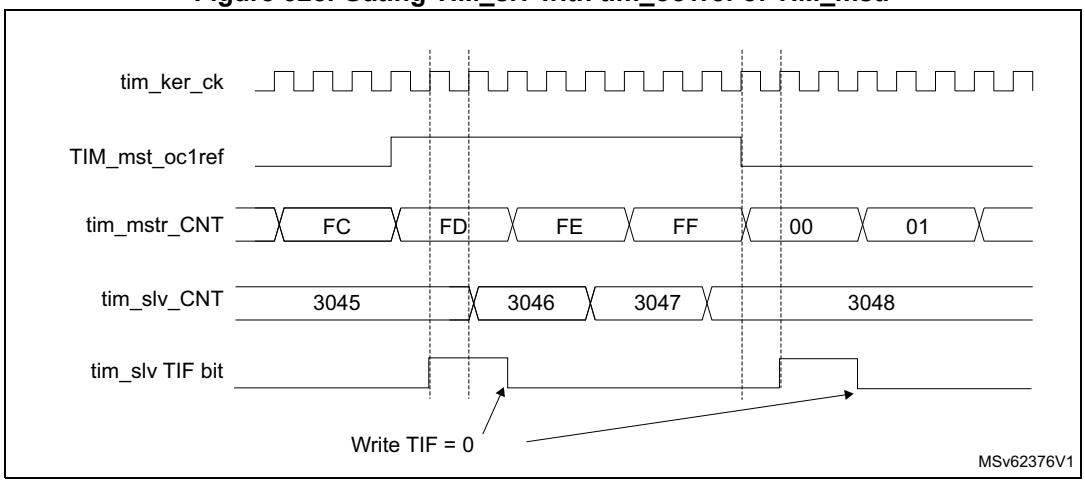
Using one timer to enable another timer

In this example, we control the enable of TIM_slv with the output compare 1 of TIM_mstr. Refer to *Figure 326* 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 328. Gating TIM_slv with tim_oc1ref of TIM_mstr

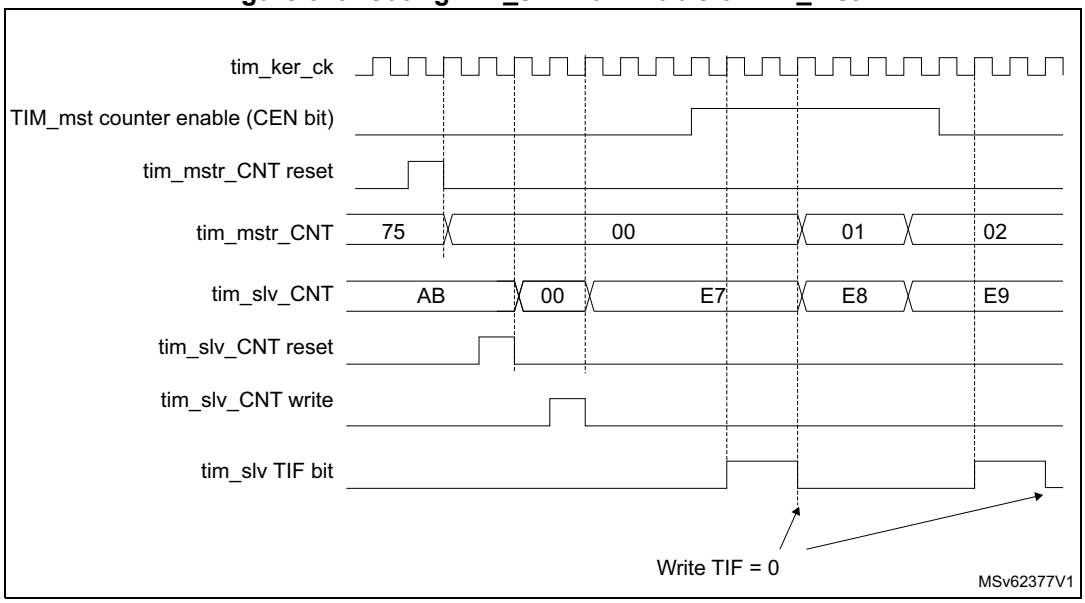


In the example in *Figure 328*, the TIM_slv counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM_mstr. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example (refer to *Figure 329*), 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 329. Gating TIM_slv with Enable of TIM_mstr

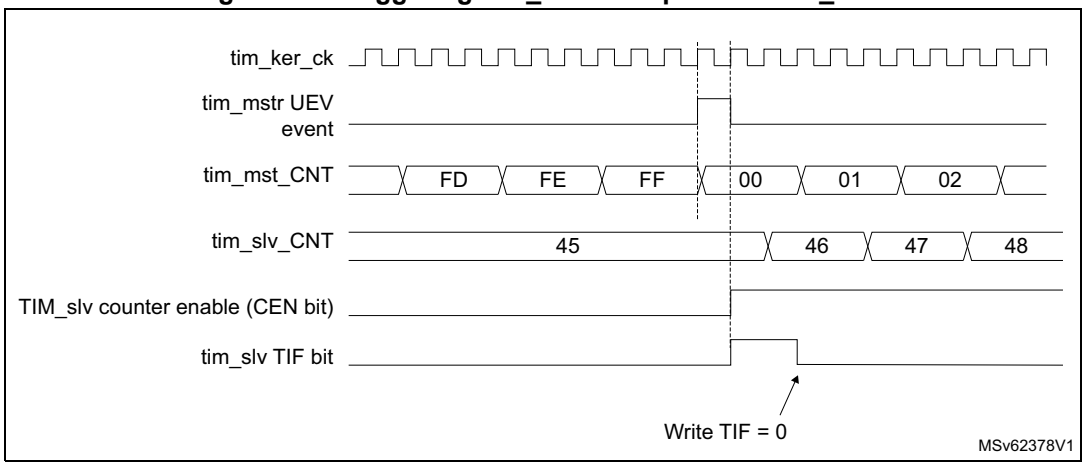


Using one timer to start another timer

In this example, we set the enable of TIM_slv with the update event of TIM_mstr. Refer to *Figure 326* 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 ($f_{tim\_cnt\_ck} = f_{tim\_ker\_ck}/3$).

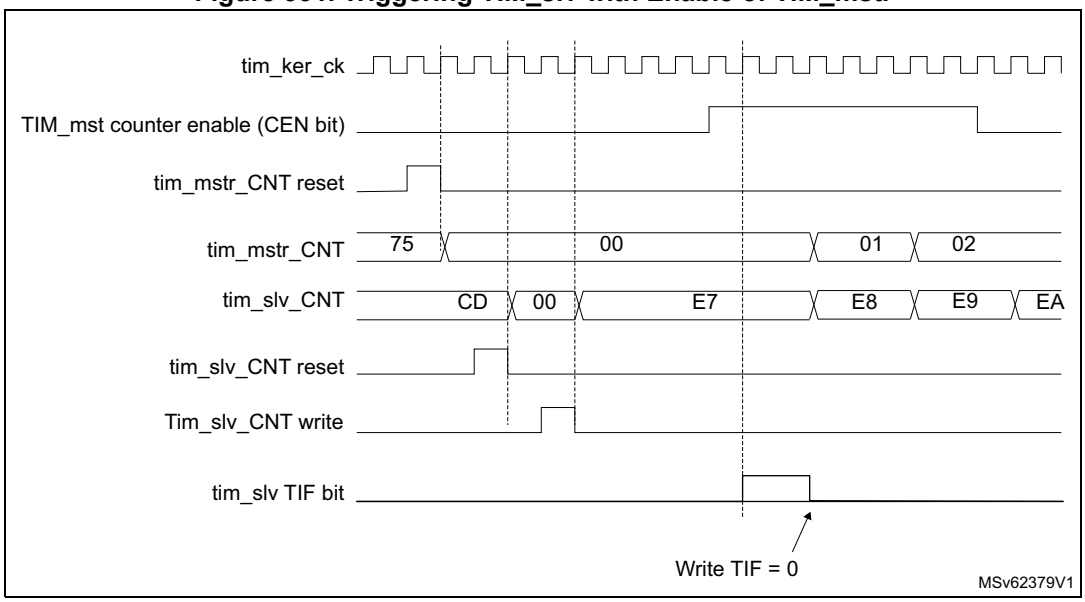
1. Configure TIM_mstr master mode to send its update event (UEV) as trigger output (MMS = 010 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr period (TIM_mstr_ARR registers).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv 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 330. Triggering TIM_slv with update of TIM_mstr



As in the previous example, both counters can be initialized before starting counting. *Figure 331* shows the behavior with the same configuration as in *Figure 330* but in trigger mode (SMS = 110 in the TIM_slv_SMCR register) instead of gated mode.

Figure 331. Triggering TIM_slv with Enable of TIM_mstr



Starting two timers synchronously in response to an external trigger

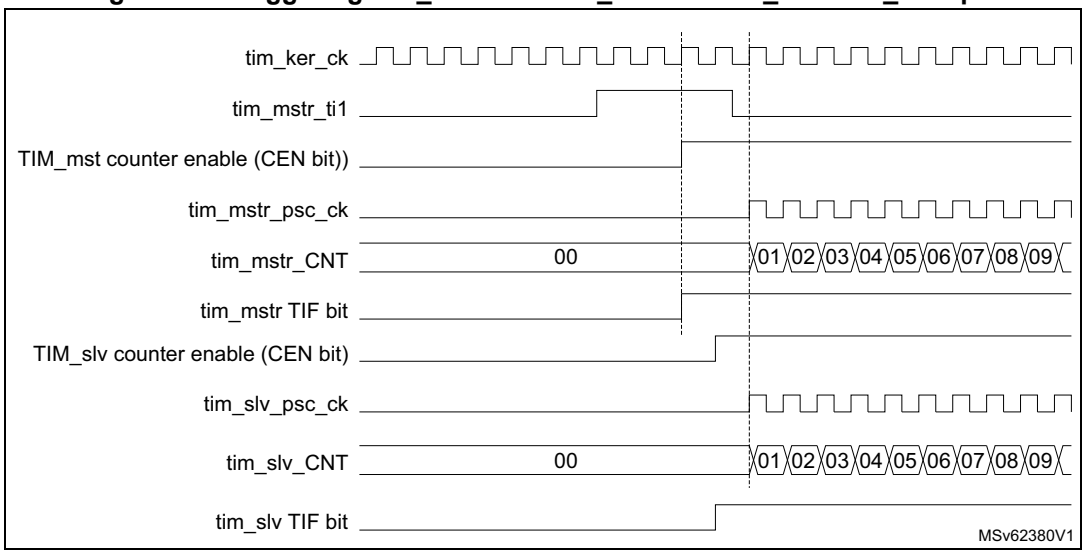
In this example, we set the enable of TIM_mstr when its tim_ti1 input rises, and the enable of TIM_slv with the enable of TIM_mstr. Refer to *Figure 326* for connections. To ensure the counters are aligned, TIM_mstr must be configured in Master/Slave mode (slave with respect to tim_ti1, master with respect 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 332. 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.

27.4.24 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.

27.4.25 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 27.5.29: TIMx DMA control register (TIMx_DCR)(x = 2, 3)* 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.

27.4.26 TIM2/TIM3 DMA requests

The TIM2/TIM3 can generate a DMA requests, as shown in *Table 200*.

Table 200. 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.

27.4.27 Debug mode

When the microcontroller enters debug mode (Cortex-M33 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).

27.4.28 TIM2/TIM3 low-power modes

Table 201. Effect of low-power modes on TIM2/TIM3

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

27.4.29 TIM2/TIM3 interrupts

The TIM2/TIM3 can generate multiple interrupts, as shown in *Table 202*.

Table 202. 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

27.5 TIM2/TIM3 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).

27.5.1 TIMx control register 1 (TIMx_CR1)(x = 2, 3)

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.

27.5.2 TIMx control register 2 (TIMx_CR2)(x = 2, 3)

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.	MMS[3]	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.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
								rw	rw	rw	rw	rw			

Bits 31:26 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. See also *Section 26.3.29: Interfacing with Hall sensors*.

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.

27.5.3 TIMx slave mode control register (TIMx_SMCR)(x = 2, 3)

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 27.4.2: TIM2/TIM3 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 27.3: TIM2/TIM3 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.

27.5.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2, 3)

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.

27.5.5 TIMx status register (TIMx_SR)(x = 2, 3)

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.

27.5.6 TIMx event generation register (TIMx_EGR)(x = 2, 3)

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).

27.5.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 2, 3)

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).

27.5.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2, 3)

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).

27.5.9 TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 2, 3)

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).

27.5.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2, 3)

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 TIMxECR.

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).

27.5.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2, 3)

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 203. 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.

27.5.12 TIM3 counter (TIM3_CNT)

Address offset: 0x024

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.
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**: Value depends on IUFREMAP in TIMx_CR1.
If UIFREMAP = 0
Reserved
If UIFREMAP = 1
UIFCPY: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register

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 holds the non-dithered part in CNT[15:0]. The fractional part is not available.

27.5.13 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.

27.5.14 TIMx prescaler (TIMx_PSC)(x = 2, 3)

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”).

27.5.15 TIM3 autoreload register (TIM3_ARR)

Address offset: 0x02C

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]**: Low autoreload value

ARR is the value to be loaded in the actual autoreload register.
Refer to the *Section 27.4.3: 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[19:4]. The ARR[3:0] bitfield contains the dithered part.

27.5.16 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 27.4.3: 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.

27.5.17 TIM3 capture/compare register 1 (TIM3_CCR1)

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.	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). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The CCR1[15:0] bits hold the capture value. The CCR1[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:0]. The CCR1[3:0] bits are reset.

27.5.18 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.

27.5.19 TIM3 capture/compare register 2 (TIM3_CCR2)

Address offset: 0x038

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.	CCR2[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 1 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 in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. 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 CCR2[15:0] bits hold the capture value. The CCR2[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:0]. The CCR2[3:0] bits are reset.

27.5.20 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.

27.5.21 TIM3 capture/compare register 3 (TIM3_CCR3)

Address offset: 0x03C

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.	CCR3[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR3[19:0]**: 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 in CCR3[15:0]. The CCR3[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR3[19:4]. 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 CCR3[15:0] bits hold the capture value. The CCR3[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR3[19:0]. The CCR3[3:0] bits are reset.

27.5.22 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.

27.5.23 TIM3 capture/compare register 4 (TIM3_CCR4)

Address offset: 0x040

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.	CCR4[19:16]			
												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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR4[19:0]**: 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 in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[19:4]. 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 CCR4[15:0] bits hold the capture value. The CCR4[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[19:0]. The CCR4[3:0] bits are reset.

27.5.24 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.

27.5.25 TIMx timer encoder control register (TIMx_ECR)(x = 2, 3)

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

27.5.26 TIMx timer input selection register (TIMx_TISEL)(x = 2, 3)

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 27.4.2: TIM2/TIM3 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 27.4.2: TIM2/TIM3 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 27.4.2: TIM2/TIM3 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 27.4.2: TIM2/TIM3 pins and internal signals* for product specific implementation.

27.5.27 TIMx alternate function register 1 (TIMx_AF1)(x = 2, 3)

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 27.4.2: TIM2/TIM3 pins and internal signals* for product specific implementation.

Bits 13:0 Reserved, must be kept at reset value.

27.5.28 TIMx alternate function register 2 (TIMx_AF2)(x = 2, 3)

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 27.4.2: TIM2/TIM3 pins and internal signals* for product specific implementation.

Bits 15:0 Reserved, must be kept at reset value.

27.5.29 TIMx DMA control register (TIMx_DCR)(x = 2, 3)

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,
...

27.5.30 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2, 3)

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).

27.5.31 TIMx register map

TIMx registers are mapped as described in the table below.

Table 204. TIM2/TIM3 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	**TIMx_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	**TIMx_CR2**	Res.	Res.	Res.	Res.	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			
0x008	**TIMx_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	**TIMx_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	**TIMx_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	**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.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x018	**TIMx_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
	TIMx_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	**TIMx_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
	TIMx_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 204. TIM2/TIM3 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	**TIMx_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	**TIMx_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	**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
0x02C	**TIMx_ARR** (x = 3)	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
0x02C	**TIMx_ARR** (x = 2)	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																																
0x034	**TIMx_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	**TIMx_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	**TIMx_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	**TIMx_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	**TIMx_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
0x05C	**TIMx_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.		TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x060	**TIMx_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.
	Reset value															0	0	0	0														
0x064	**TIMx_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.																															

Table 204. TIM2/TIM3 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
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 2.2* for the register boundary addresses.

28 Basic timers (TIM6/TIM7)

28.1 TIM6/TIM7 introduction

The basic timers TIM6/TIM7 consist in a 16-bit autoreload counter driven by a programmable prescaler.

They can be used as generic timers for time-base generation.

The basic timer can also be used for triggering the digital-to-analog converter. This is done with the trigger output of the timer.

The timers are completely independent, and do not share any resources.

28.2 TIM6/TIM7 main features

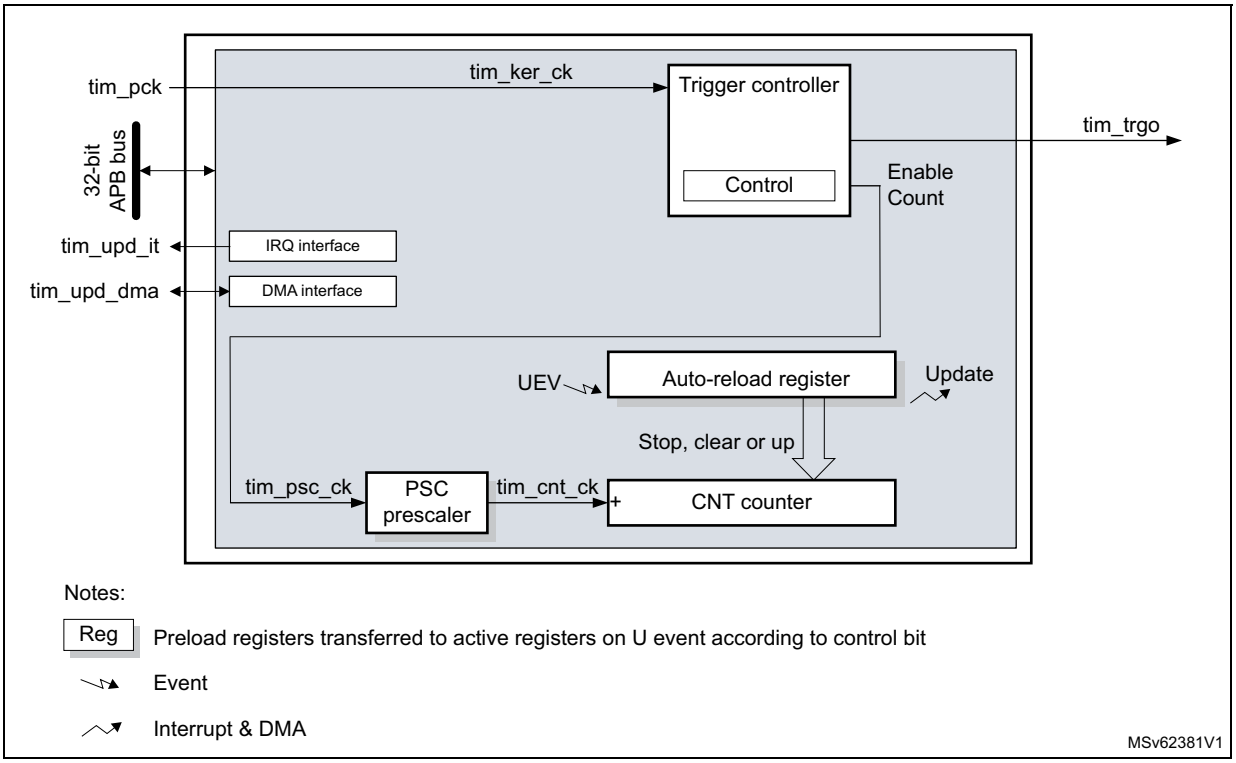
Basic timer (TIM6/TIM7) features include:

- 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.
- Synchronization circuit to trigger the DAC.
- Interrupt/DMA generation on the update event: counter overflow.

28.3 TIM6/TIM7 functional description

28.3.1 TIM6/TIM7 block diagram

Figure 333. Basic timer block diagram



28.3.2 TIM6/TIM7 internal signals

The table in this section summarizes the TIM inputs and outputs.

Table 205. TIM internal input/output signals

Internal signal name	Signal type	Description
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_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals (DAC).
tim_upd_it	Output	Timer update event interrupt
tim_upd_dma	Output	Timer update dma request

28.3.3 TIM6/TIM7 clocks

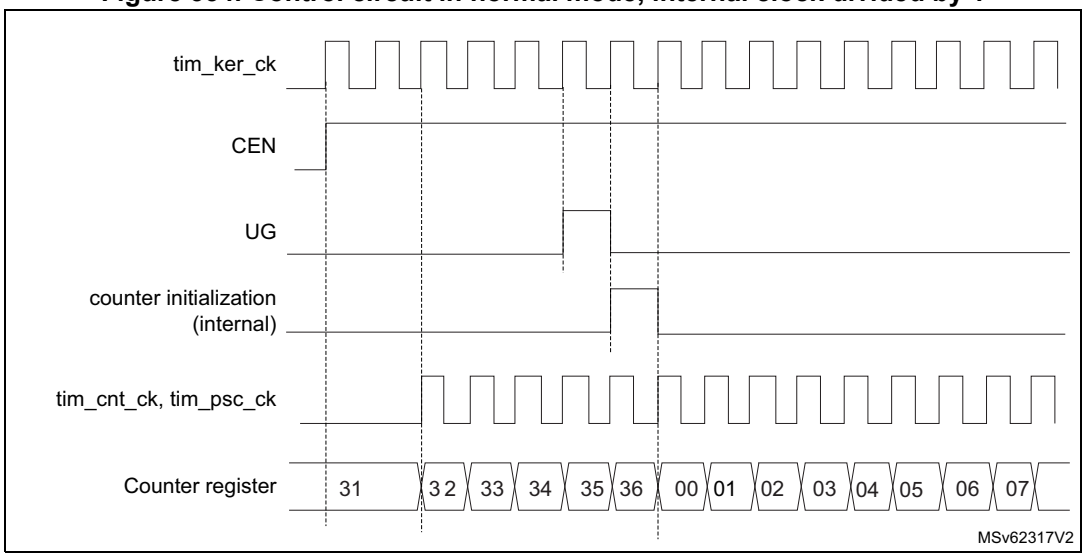
The timer bus interface is clocked by the tim_pclk APB clock.

The counter clock tim_ker_ck is connected to the tim_pclk input.

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 for UG that 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 334 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 334. Control circuit in normal mode, internal clock divided by 1



28.3.4 Time-base unit

The main block of the programmable 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)
- Auto-Reload register (TIMx_ARR).

The autoreload register is preloaded. The preload register is accessed each time an attempt is made to write or read the autoreload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the autoreload preload enable bit (ARPE) in the TIMx_CR1 register. The update event is sent when the counter reaches the overflow value 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 the TIMx_CR1 register is set.

Note that the actual counter enable signal tim_cnt_en is set one clock cycle after CEN bit set.

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 the TIMx_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 335 and *Figure 336* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 335. Counter timing diagram with prescaler division change from 1 to 2

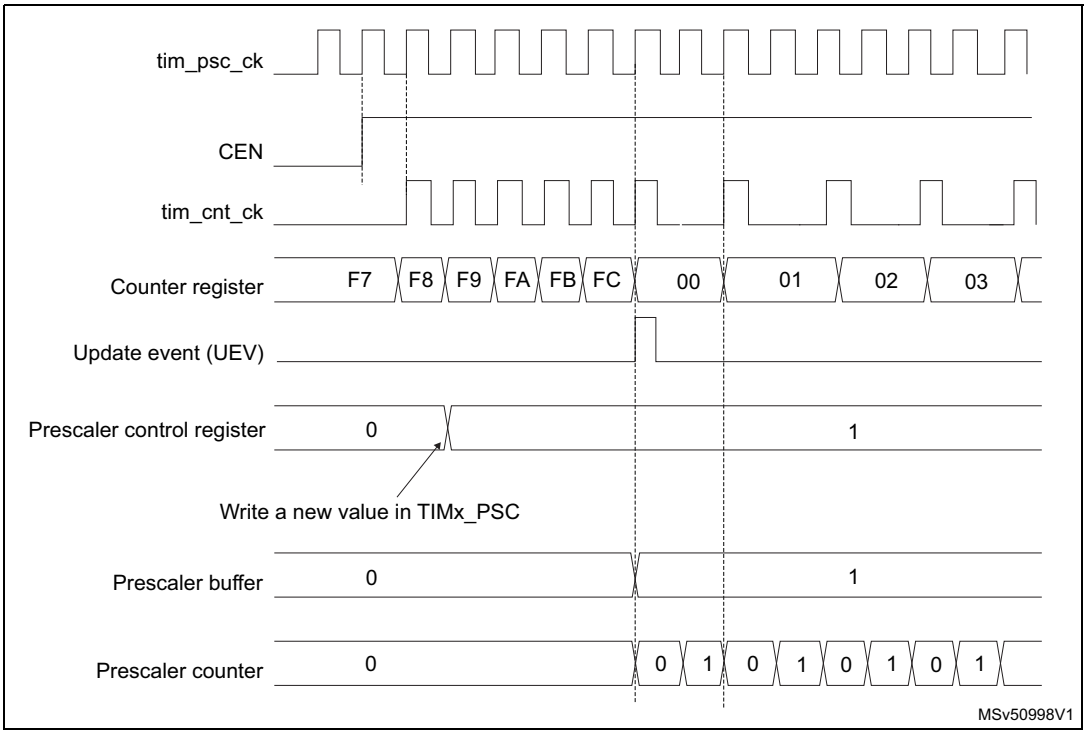
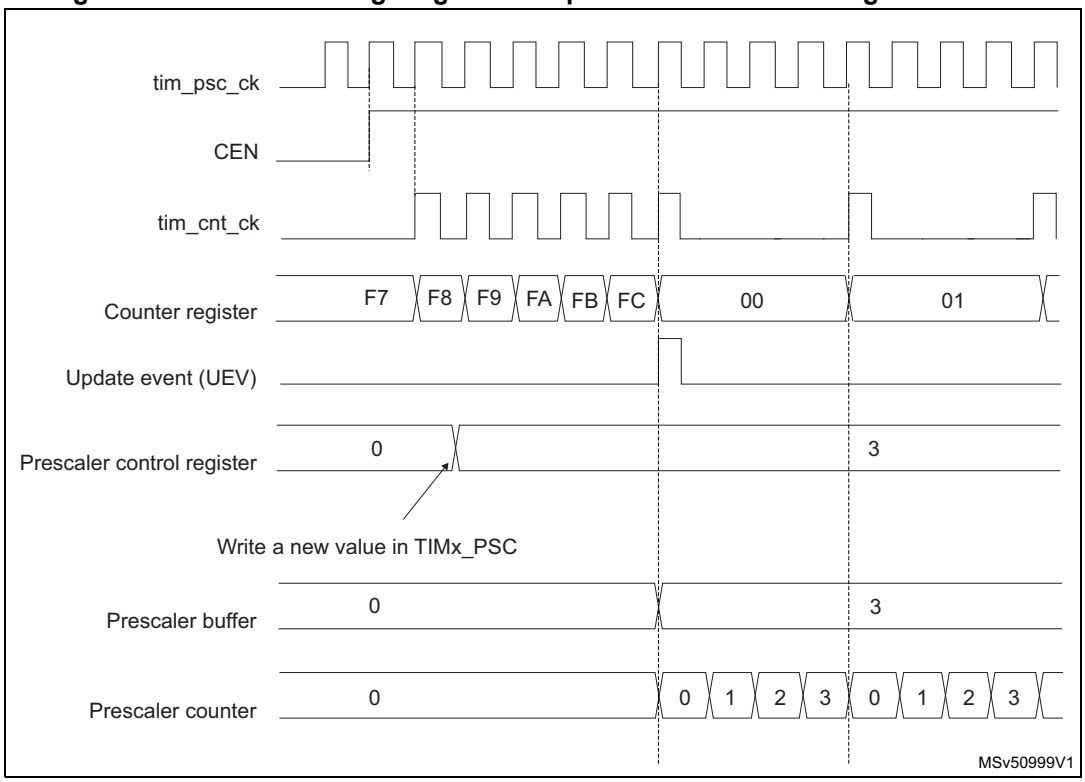


Figure 336. Counter timing diagram with prescaler division change from 1 to 4



28.3.5 Counting mode

The counter counts from 0 to the autoreload value (contents of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generate 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 the TIMx_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIMx_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents 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 337. Counter timing diagram, internal clock divided by 1

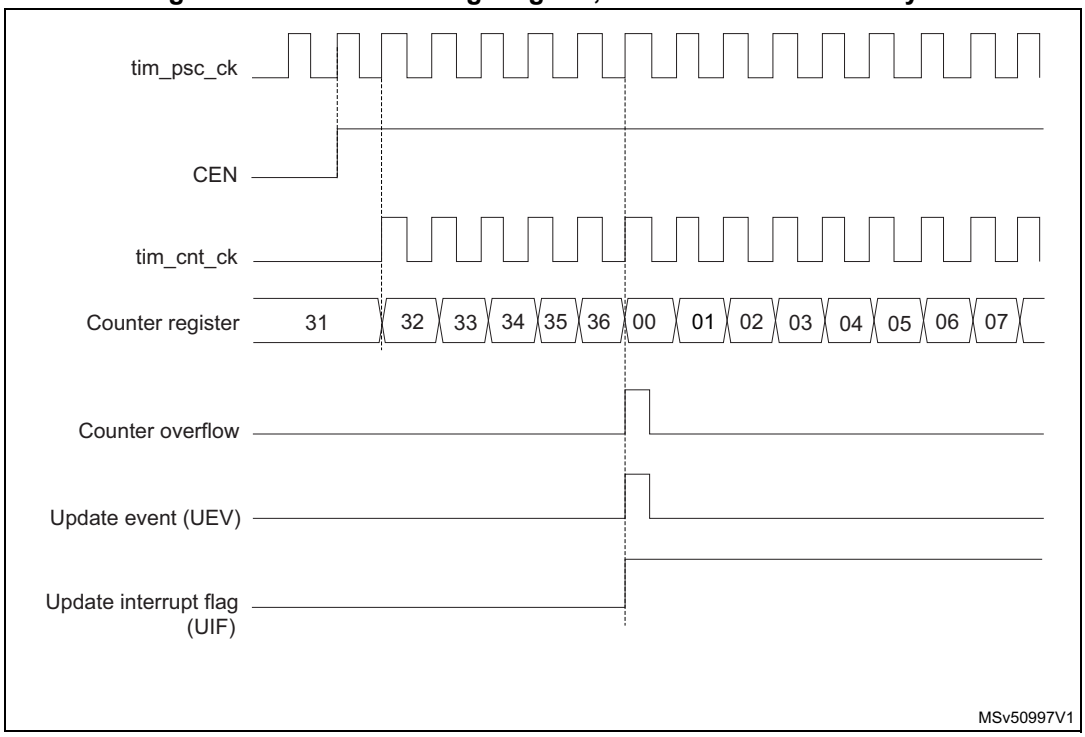


Figure 338. Counter timing diagram, internal clock divided by 2

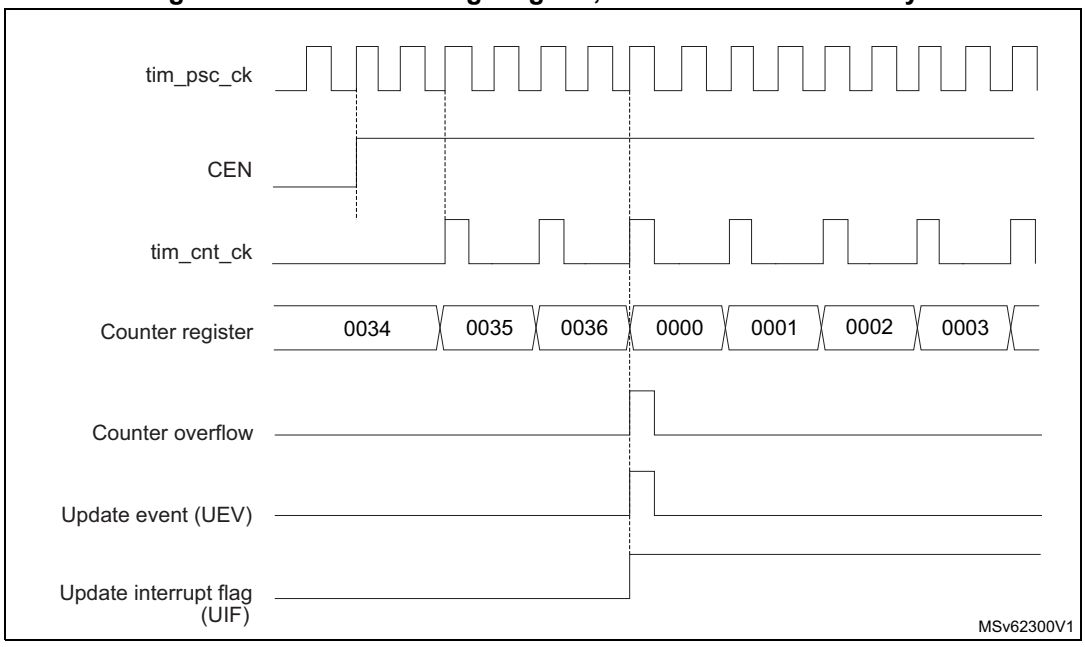


Figure 339. Counter timing diagram, internal clock divided by 4

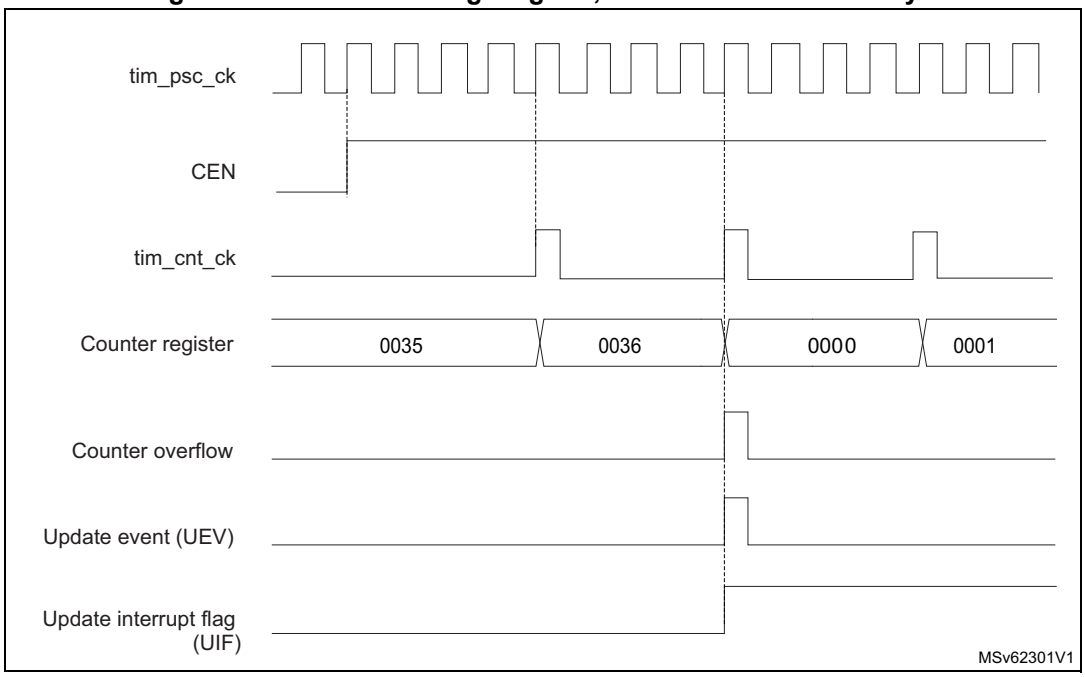


Figure 340. Counter timing diagram, internal clock divided by N

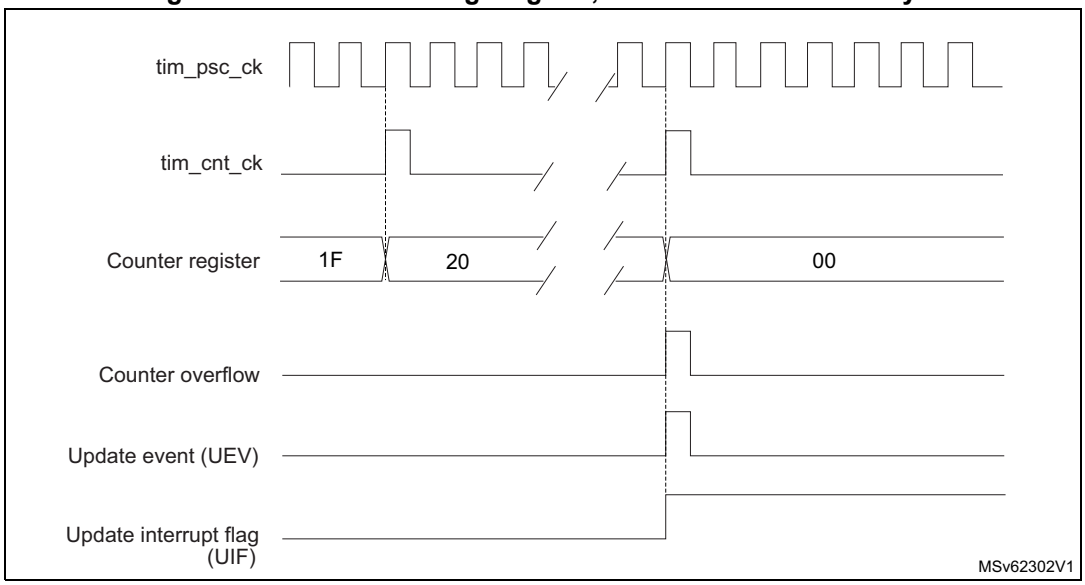


Figure 341. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

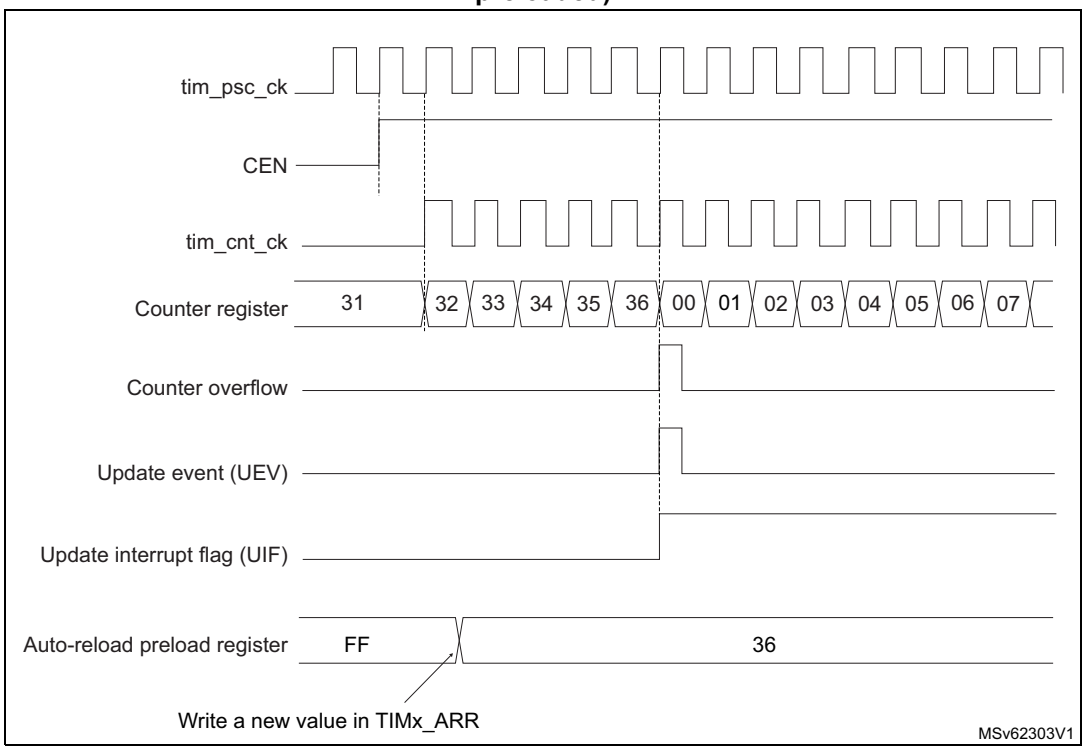
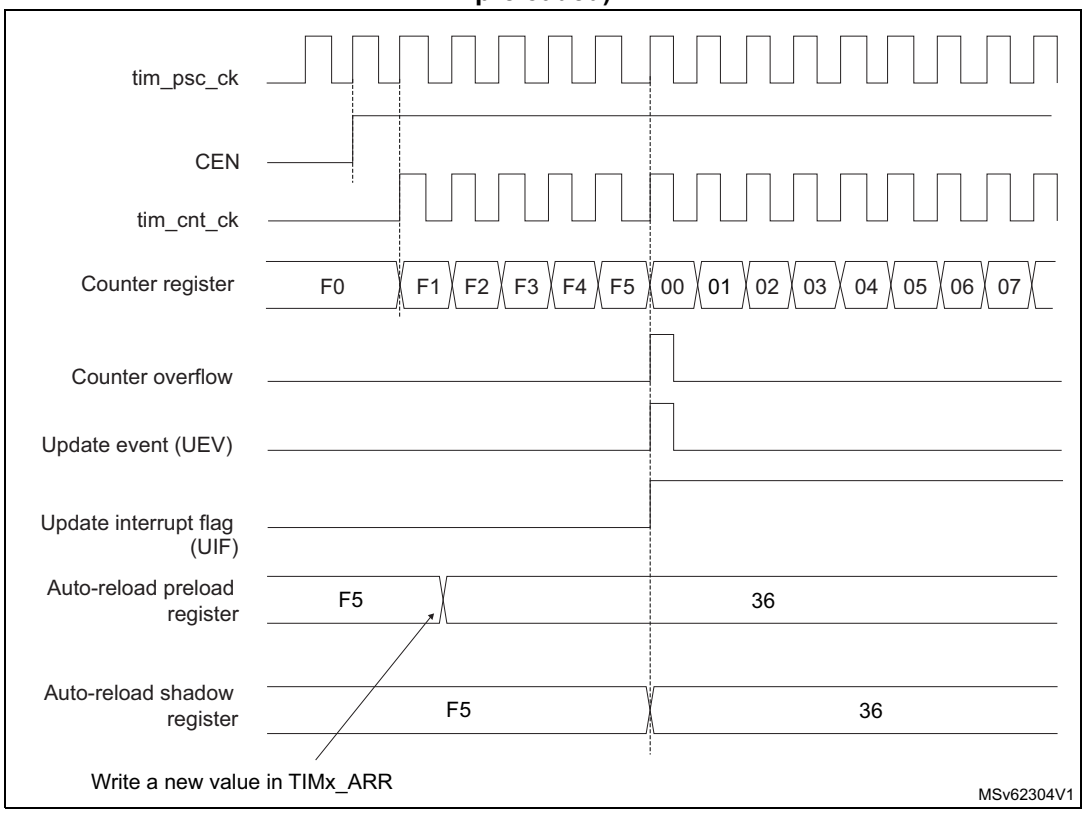


Figure 342. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)



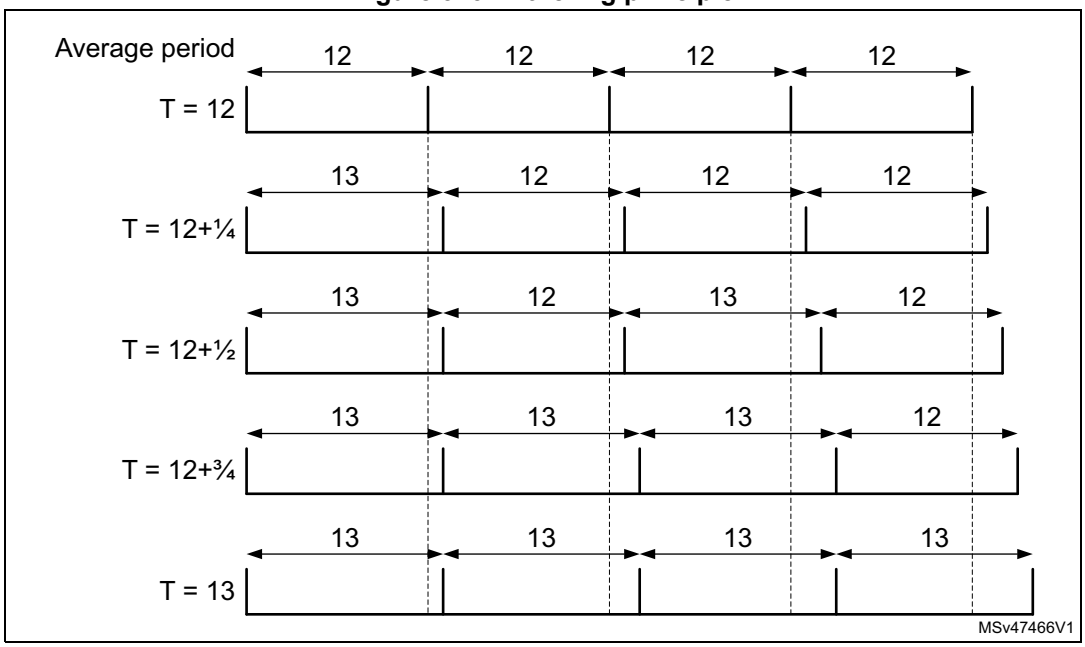
Dithering mode

The time base effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This affects the way the TIMx_ARR is behaving, and is useful for adjusting the average counter period when the timer is used as a trigger (typically for a DAC).

The operating principle is to have the actual ARR value slightly changed (adding or not one timer clock period) over 16 consecutive counting periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average counting period.

Figure 343 presents the dithering principle applied to four consecutive counting periods.

Figure 343. Dithering principle

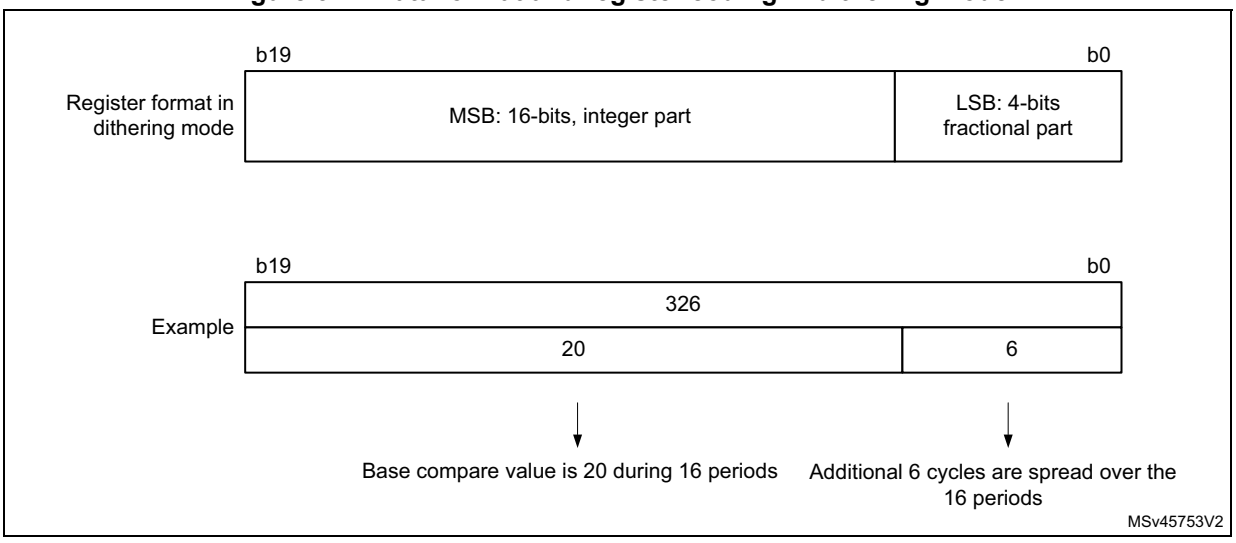


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 344* for example):

- 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 DITHEN bit must be reset
3. The CEN bit can be set (eventually with ARPE = 1).

Figure 344. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

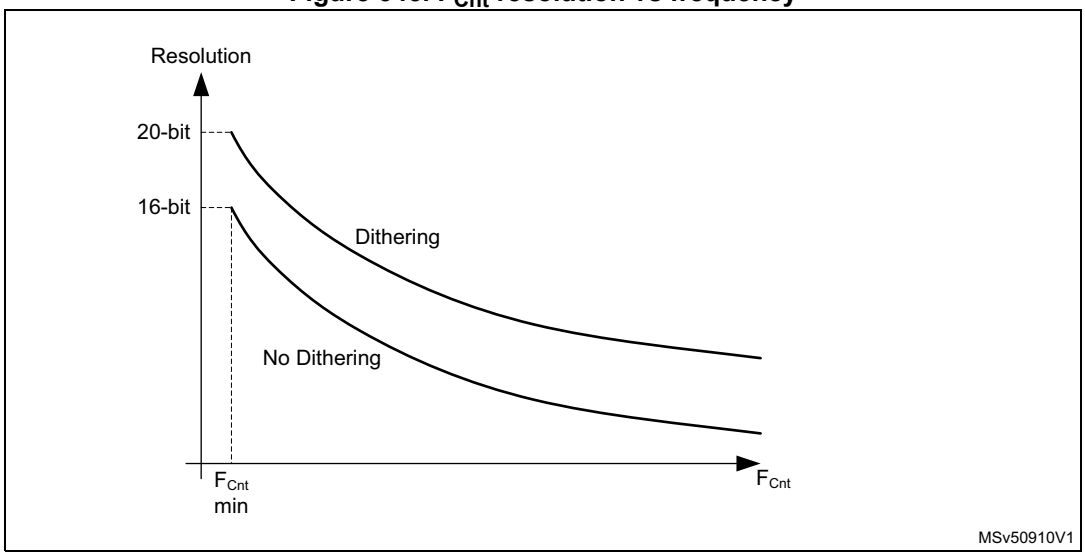
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR value is limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

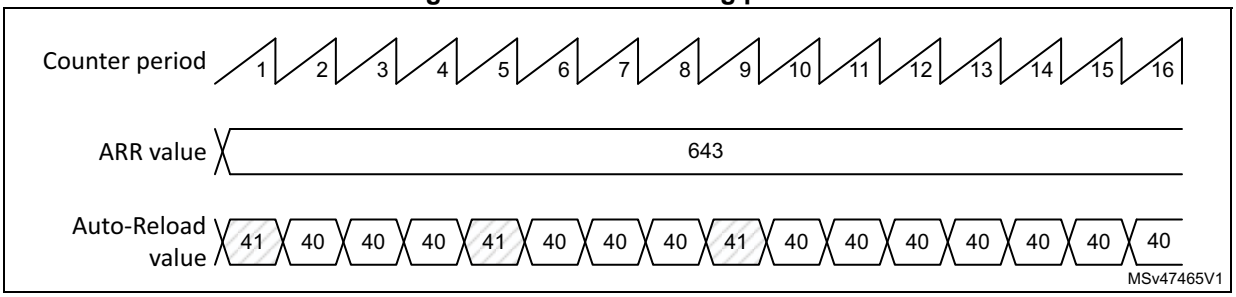
As shown on *Figure 345*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 345. F_{Cnt} resolution vs frequency



The period changes are spread over 16 consecutive periods, as described in *Figure 346*.

Figure 346. PWM dithering pattern



The autoreload and compare values increments are spread following the specific patterns described in *Table 206*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 206. TIMx_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	-
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	-

28.3.6 UIF bit remapping

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 (TIMxCNT[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.

28.3.7 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 receiving 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.

28.3.8 TIM6/TIM7 DMA requests

The TIM6/TIM7 can generate a single DMA request, as shown in *Table 207*.

Table 207. DMA request

DMA acronym	DMA request	Enable control bit
tim_upd_dma	Update	UDE

28.3.9 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or be stopped.

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).

28.3.10 TIM6/TIM7 low-power modes

Table 208. Effect of low-power modes on TIM6/TIM7

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

28.3.11 TIM6/TIM7 interrupts

The TIM6/TIM7 can generate a single interrupt, as shown in *Table 209*.

Table 209. Interrupt request

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM6 TIM7	Update	UIF	UIE	write 0 in UIF	Yes	No

28.4 TIM6/TIM7 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).

28.4.1 TIMx control register 1 (TIMx_CR1)(x = 6 to 7)

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.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			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.

Bits 10:8 Reserved, must be kept at reset value.

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 CEN bit).

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 generates 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). 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

CEN is cleared automatically in one-pulse mode, when an update event occurs.

28.4.2 TIMx control register 2 (TIMx_CR2)(x = 6 to 7)

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.	Res.	Res.	Res.	MMS[2:0]			Res.	Res.	Res.	Res.
									rw	rw	rw				

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000:**Reset** - the UG bit from the TIMx_EGR register is used as a trigger output (tim_trgo).

001:**Enable** - the Counter enable signal, tim_cnt_en, is used as a 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 when the CEN control bit is written.

010:**Update** - The update event is selected as a trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

Note: The clock of the slave timer or he peripheral receiving the tim_trgo 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.

Bits 3:0 Reserved, must be kept at reset value.

28.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7)

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.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
							rw								rw

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **UDE**: Update DMA request enable

0: Update DMA request disabled.
1: Update DMA request enabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled.
1: Update interrupt enabled.

28.4.4 TIMx status register (TIMx_SR)(x = 6 to 7)

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
															rc_w0

Bits 15:1 Reserved, must be kept at reset value.

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:

–On counter overflow if UDIS = 0 in the TIMx_CR1 register.

–When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

28.4.5 TIMx event generation register (TIMx_EGR)(x = 6 to 7)

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
															w

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

28.4.6 TIMx counter (TIMx_CNT)(x = 6 to 7)

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 and read as 0.

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.

28.4.7 TIMx prescaler (TIMx_PSC)(x = 6 to 7)

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 $f_{tim\_cnt\_ck}$ is equal to $f_{tim\_psc\_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIMx_EGR register.

28.4.8 TIMx autoreload register (TIMx_ARR)(x = 6 to 7)

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 into the actual auto-reload register.

Refer to *Section 28.3.4: Time-base unit* 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 reserved.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

28.4.9 TIMx register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

Table 210. TIMx 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.	DITHEN	UIFREMA	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																				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.	Res.	Res.	Res.	MMS [2:0]			Res.	Res.	Res.	Res.
	Reset value																										0	0	0				
0x08	Reserved																																
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.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
	Reset value																								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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
	Reset value																																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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
	Reset value																																0
0x18-0x20	Reserved																																
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
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

Refer to *Section 2.2* for the register boundary addresses.

29 Low-power timer (LPTIM)

29.1 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.

29.2 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

29.3 LPTIM implementation

Table 211 describes LPTIM implementation on STM32H503xx devices.

Table 211. STM32H503xx LPTIM features

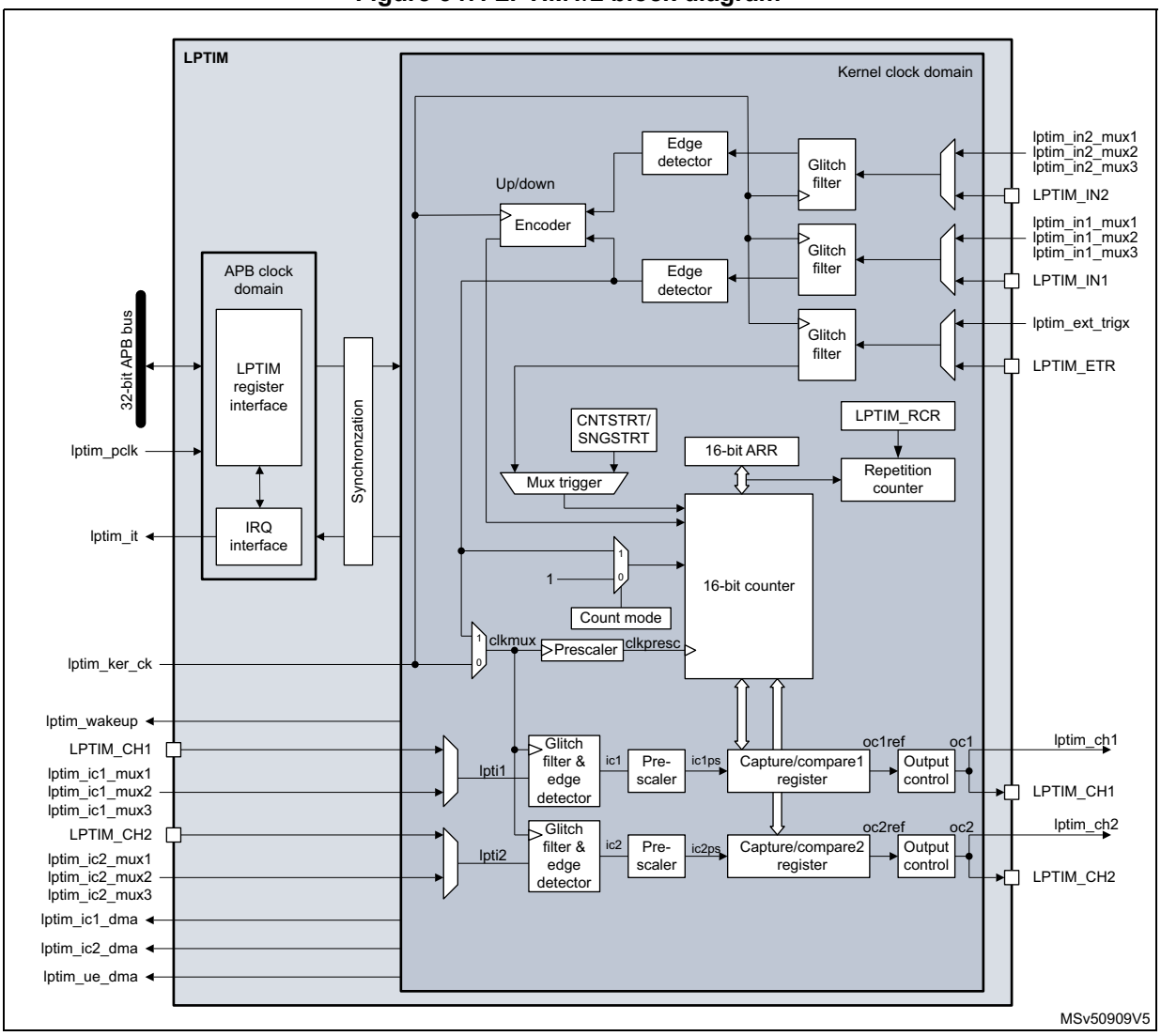
LPTIM modes/features[1]	LPTIM1	LPTIM2
Encoder mode	X	X
PWM mode	X	X
Input Capture	X	X
Number of channels	2	2
Number of DMA requests	3	3
Wake-up in Stop mode	X	X

1. X = supported.

29.4 LPTIM functional description

29.4.1 LPTIM block diagram

Figure 347. LPTIM1/2 block diagram(1)



1. Some I/Os may not be available, refer to *Section 29.4.2: LPTIM pins and internal signals*.

29.4.2 LPTIM pins and internal signals

The following tables provide the list of LPTIM pins and internal signals, respectively.

Table 212. 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 212. 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 213. 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

29.4.3 LPTIM input and trigger mapping

The LPTIM external trigger and input connections are detailed hereafter.

Table 214. LPTIM1/2 external trigger connections

TRIGSEL	External trigger	
	LPTIM1	LPTIM2
lptim_ext_trig0	GPIO	
lptim_ext_trig1	rtc_alra_trg	
lptim_ext_trig2	rtc_alrb_trg	
lptim_ext_trig3	tamp_trg1	
lptim_ext_trig4	tamp_trg2	gpdma_ch0_tcf
lptim_ext_trig5	gpdma_ch1_tcf	gpdma_ch4_tcf
lptim_ext_trig6	comp1_out	
lptim_ext_trig7	EVENTOUT	

Table 215. LPTIM1/2 input 1 connections

lptim_in1_mux	LPTIM1 input 1 connected to	LPTIM2 input 1 connected to
lptim_in1_mux0	GPIO	
lptim_in1_mux1	comp1_out	
lptim_in1_mux2	lptim2_ch1	lptim1_ch2
lptim_in1_mux3	Reserved	Reserved

Table 216. LPTIM1/2 input 2 connections

lptim_in2_mux	LPTIM1/2 input 2 connected to
lptim_in2_mux0	GPIO
lptim_in2_mux1	Reserved
lptim_in2_mux2	Reserved
lptim_in2_mux3	Reserved

Table 217. LPTIM1/2 input capture 1 connections

lptim_ic1_mux	LPTIM1 input capture 1 connected to	LPTIM2 input capture 1 connected to
lptim_ic1_mux0	GPIO	
lptim_ic1_mux1	comp1_out	
lptim_ic1_mux2	EVENTOUT	
lptim_ic1_mux3	RCC_MCO1	RCC_MCO2

Table 218. LPTIM1/2 input capture 2 connections

lptim_ic2_mux	LPTIM1 input capture 2 connected to	LPTIM2 input capture 2 connected to
lptim_ic2_mux0	I/O	
lptim_ic2_mux1	LSI	HSI/1024
lptim_ic2_mux2	LSE	CSI/128
lptim_ic2_mux3	RCC_HSE_1MHZ	HSI/8

29.4.4 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.

29.4.5 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.

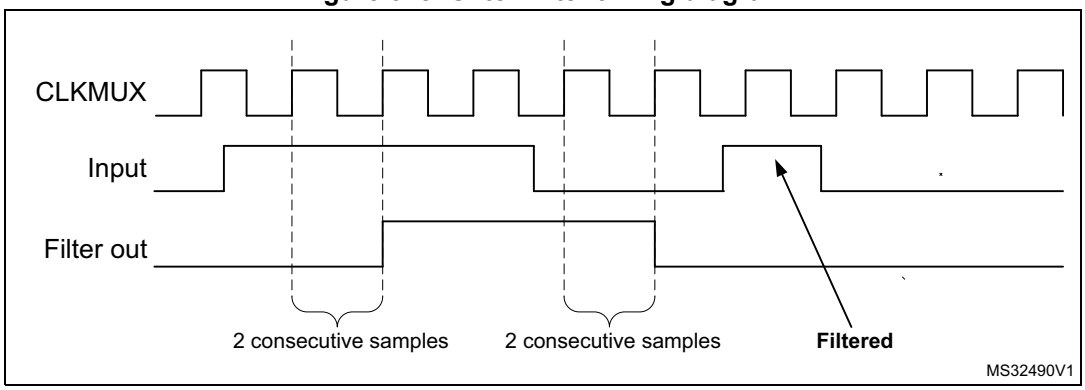
The digital filters are divided into three groups:

- 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 348* shows an example of glitch filter behavior in case of a two consecutive samples programmed.

Figure 348. 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.

29.4.6 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 219. Prescaler division ratios

Programming	Dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

29.4.7 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.

29.4.8 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.

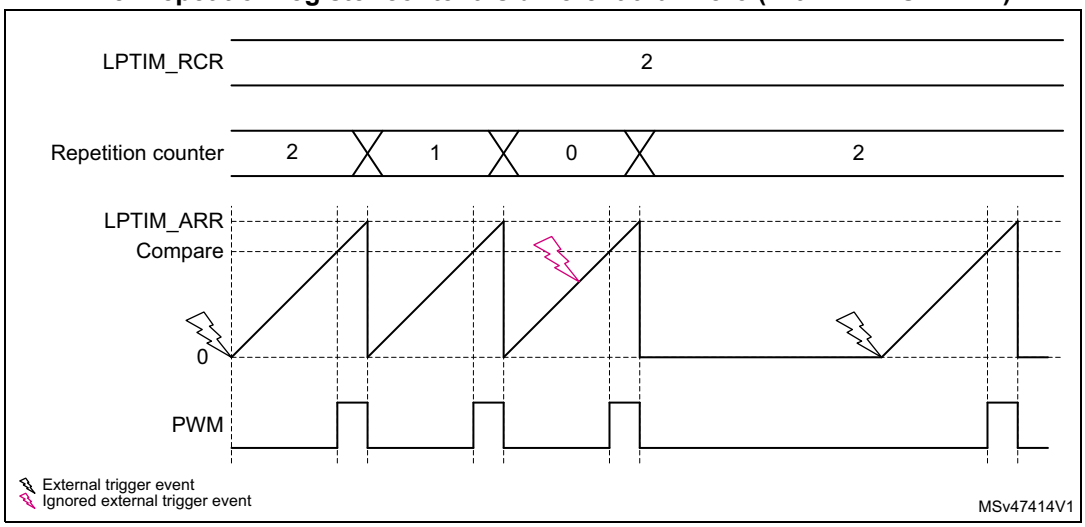
One-shot mode

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 349*.

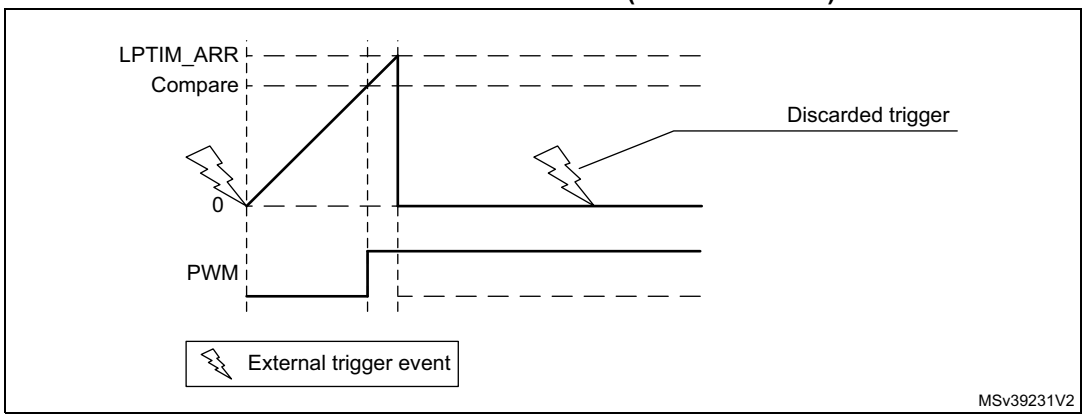
Figure 349. LPTIM output waveform, single-counting mode configuration when repetition register content is different than zero (with PRELOAD = 1)



- 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 350*.

Figure 350. 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.

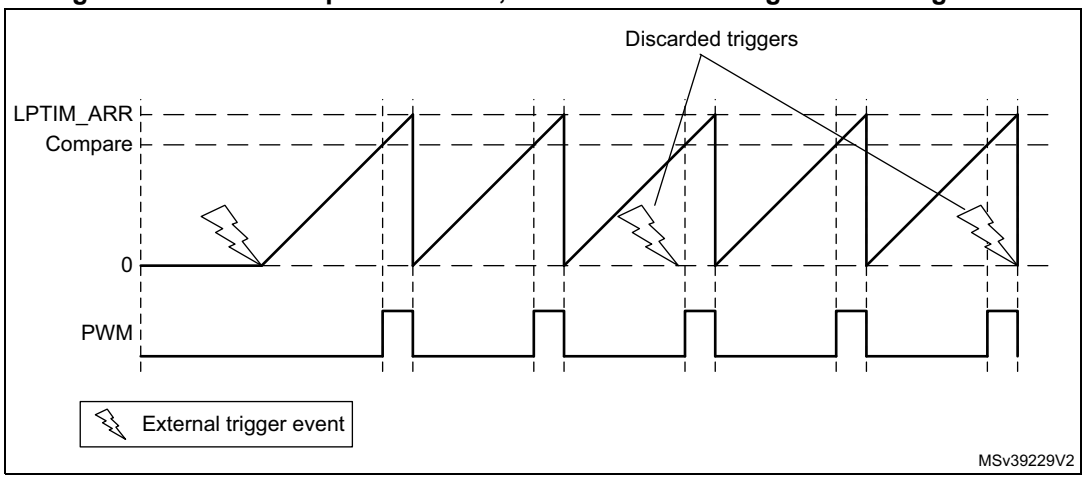
Continuous mode

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 351*.

In case of software start (TRIGEN[1:0] = 00), setting CNTSTRT starts the counter for continuous counting.

Figure 351. 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.

29.4.9 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.

29.4.10 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 29.4.19: 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.

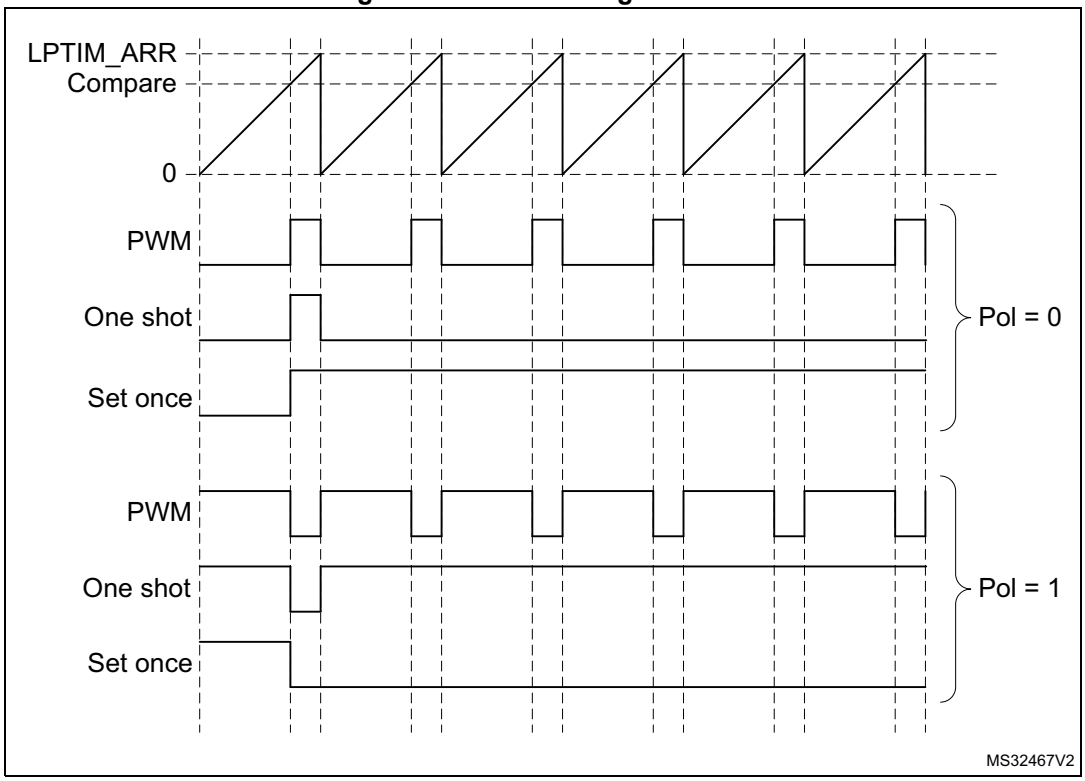
The LPTIM output waveform can be configured through the WAVE bit as follow:

- 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 352* 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 352. Waveform generation



29.4.11 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.

29.4.12 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.

29.4.13 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.

29.4.14 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 should ensure 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.

29.4.15 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.

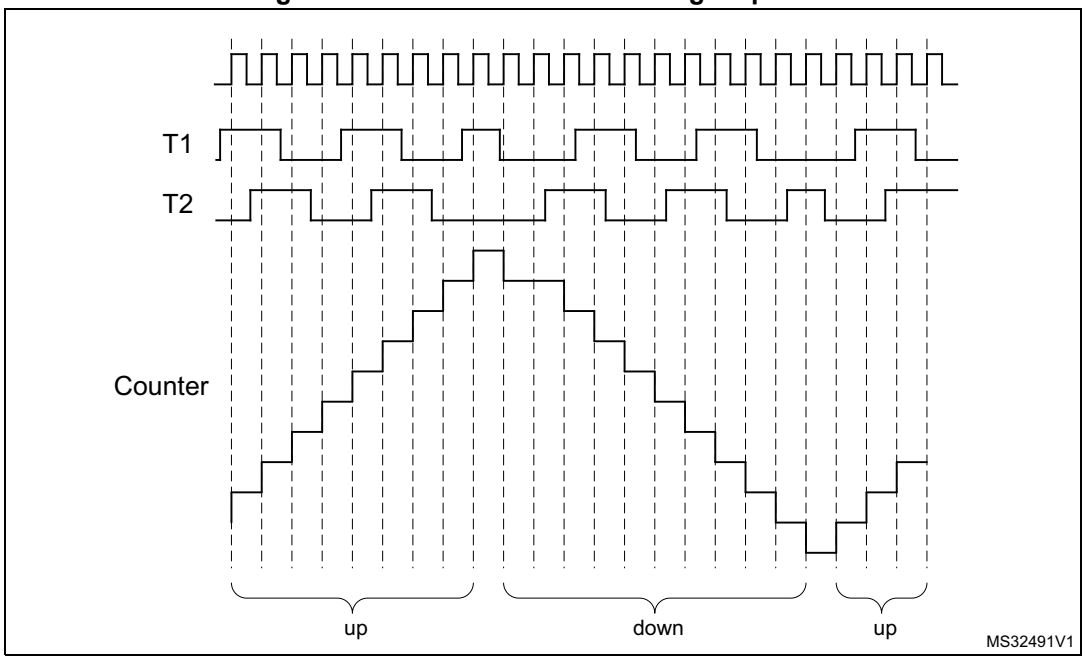
Table 220. Encoder counting scenarios

Active edge	Level on opposite signal (Input1 for Input2, Input2 for Input1)	Input1 signal		Input2 signal	
		Rising	Falling	Rising	Falling
Rising Edge	High	Down	No count	Up	No count
	Low	Up	No count	Down	No count
Falling Edge	High	No count	Up	No count	Down
	Low	No count	Down	No count	Up
Both Edges	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 353. Encoder mode counting sequence



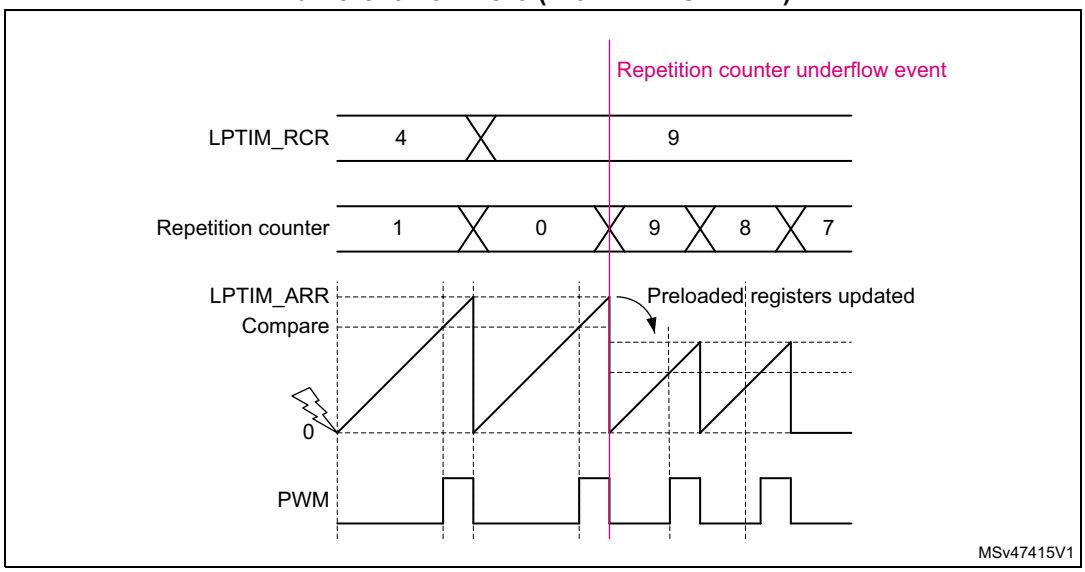
29.4.16 Repetition counter

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 354*.

Figure 354. 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 29.4.11: 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.

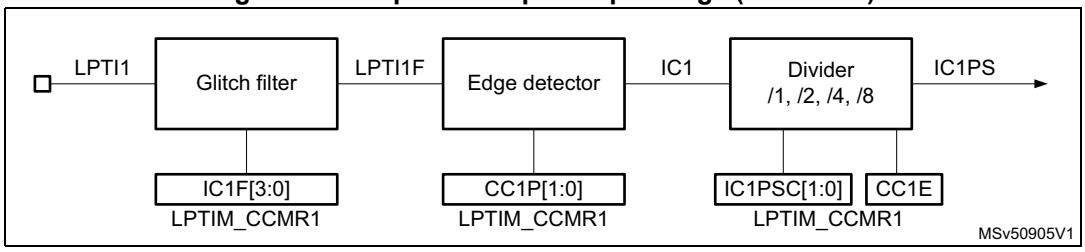
29.4.17 Capture/compare channels

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).

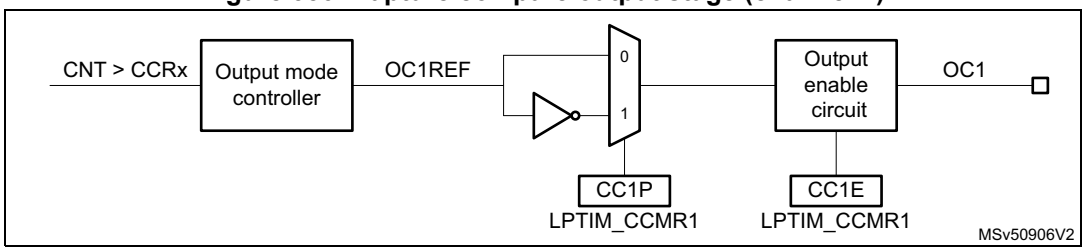
Figure 355. Capture/compare input stage (channel 1)



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 356. Capture/compare output stage (channel 1)



29.4.18 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 221*) 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 221*, 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 221. 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 221. 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

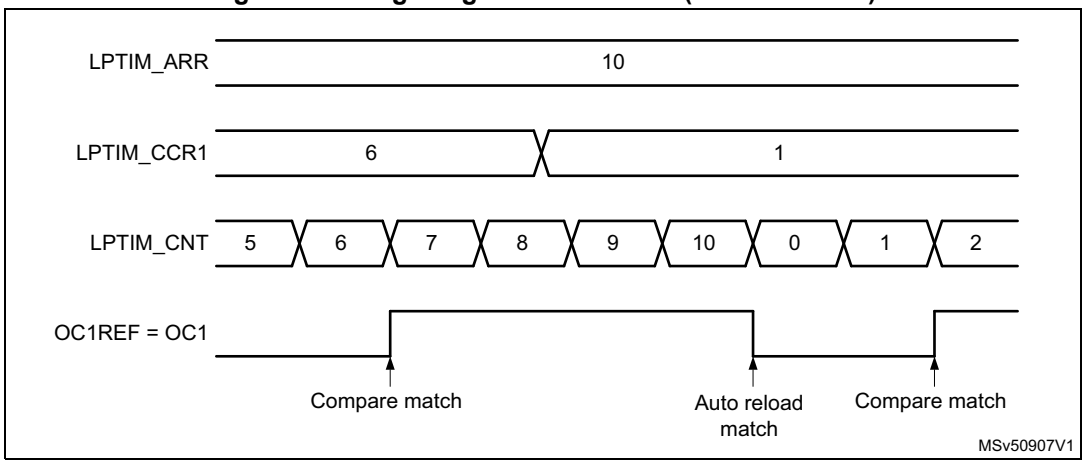
29.4.19 PWM mode

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 357 gives an example where the LPTIM channel 1 is configured in PWM mode with LPTIM_CCR1 = 6 then 1 and LPTIM_ARR=10.

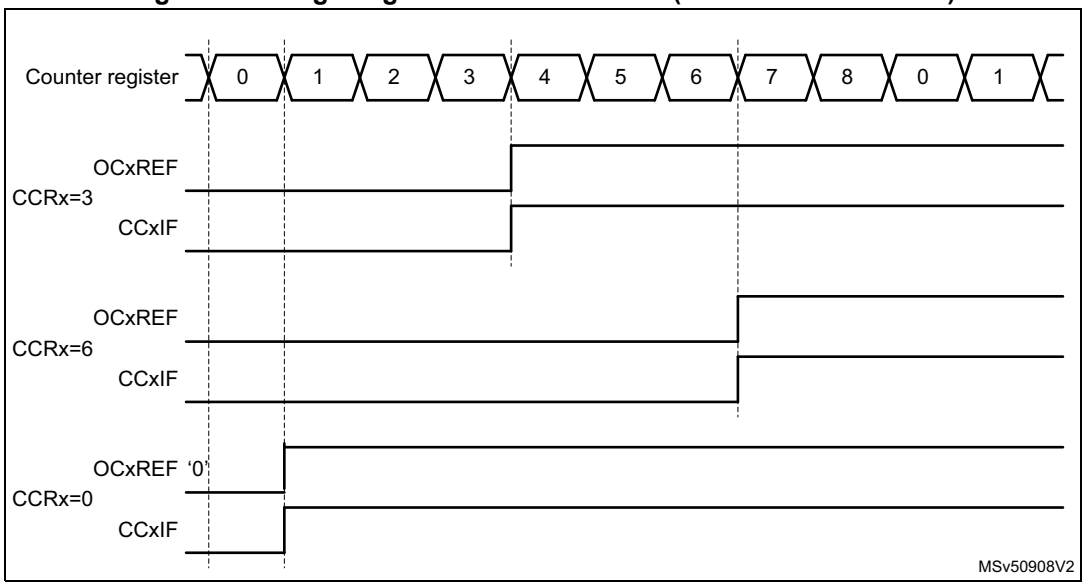
Figure 357. Edge-aligned PWM mode (PRELOAD = 1)



In the following example the reference PWM signal OCxREF is low as long as LPTIM_CNT ≤ LPTIM_CCRx else it becomes high.

Figure 358 shows some edge-aligned PWM waveforms in an example where LPTIM_ARR = 8.

Figure 358. 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.

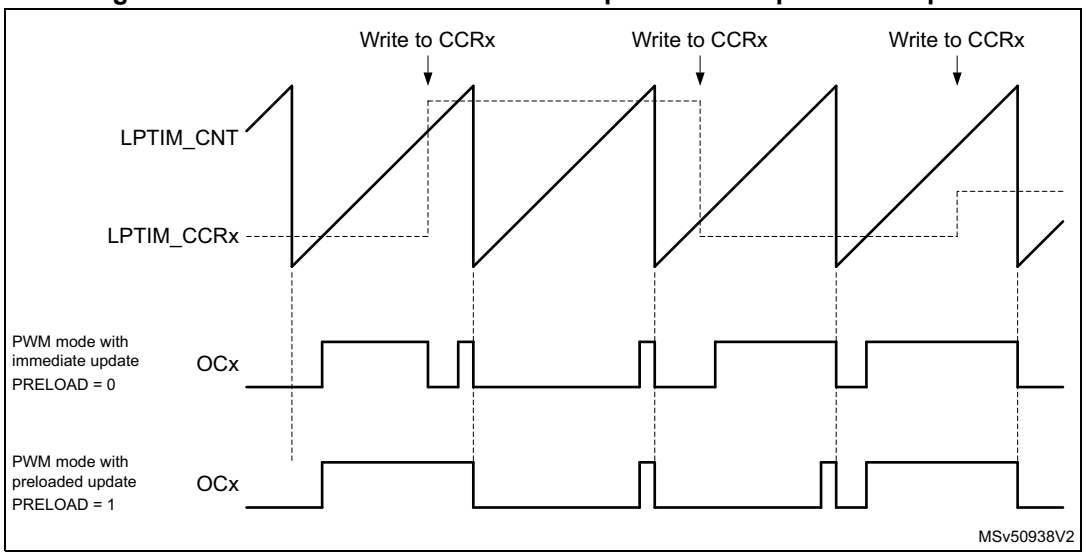
When the (PRELOAD = 0), the OCxREF signal level can be changed on-the-fly by software (or DMA) by updating the compare value in the LPTIM_CCRx register.

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 359* illustrates the behavior of the OCxREF signal level when PRELOAD = 0 and PRELOAD = 1.

Figure 359. 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 357 illustrates this behavior.

29.4.20 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*

29.4.21 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).

29.5 LPTIM low-power modes

Table 222. Effect of low-power modes on the LPTIM

Mode	Description
Sleep	No effect. LPTIM interrupts cause the device to exit Sleep mode.
Stop	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.
Standby	The LPTIM peripheral is powered down and must be reinitialized after exiting Standby mode.

Note: All DMA requests must be disabled (reset UEDE and CCxDE bits) before entering Sleep, Stop and Standby modes.

29.6 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 223. 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 29.3: LPTIM implementation*.
2. If LPTIM does not support encoder mode feature, this event does not exist. Refer to *Section 29.3: LPTIM implementation*.

29.7 LPTIM registers

Refer to *Section 1.2: List of abbreviations for registers on page 65* for a list of abbreviations used in register descriptions.

The peripheral registers can only be accessed by words (32-bit).

29.7.1 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 29.3.

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 29.3.

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 29.3.

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 29.3.

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

29.7.2 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 29.3.

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 29.3.

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 29.3.

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 29.3.

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 29.3.

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.

29.7.3 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 29.3.

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 29.3.

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 29.3.

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 29.3.

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.

29.7.4 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 29.3.

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 29.3.

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 29.3.

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 29.3.

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 29.3.

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.

29.7.5 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 29.3.

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 29.3.

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 29.3.

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 29.3.

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 29.3.

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

29.7.6 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 29.3.

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 29.3.

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 29.3.

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 29.3.

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 29.3.

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 29.3.

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 29.3.

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 29.3.

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

29.7.7 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 29.3.

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 29.4.3: 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 29.4.15: Encoder mode* for more details about Encoder mode 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).

29.7.8 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

29.7.9 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).

29.7.10 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).

29.7.11 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.

29.7.12 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 29.4.3: 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 29.4.3: 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 29.4.3: 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 29.4.3: LPTIM input and trigger mapping*.

29.7.13 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.

29.7.14 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).

29.7.15 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 29.3: LPTIM implementation*.

29.7.16 LPTIM register map

The following table summarizes the LPTIM registers.

Table 224. 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 224. 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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	LPTIM_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x01C	LPTIM_CNT	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
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	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
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)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 29.3: LPTIM implementation*.
2. If LPTIM does not support encoder mode feature, this bit is reserved. Refer to *Section 29.3: LPTIM implementation*.
3. If the LPTIM implements less than 2 channels this register is reserved. Refer to *Section 29.3: LPTIM implementation*.

Refer to *Section 2.2 on page 70* for the register boundary addresses.

30 Independent watchdog (IWDG)

30.1 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 30.3* 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.

30.2 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

30.3 IWDG implementation

Table 225. 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	-
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.

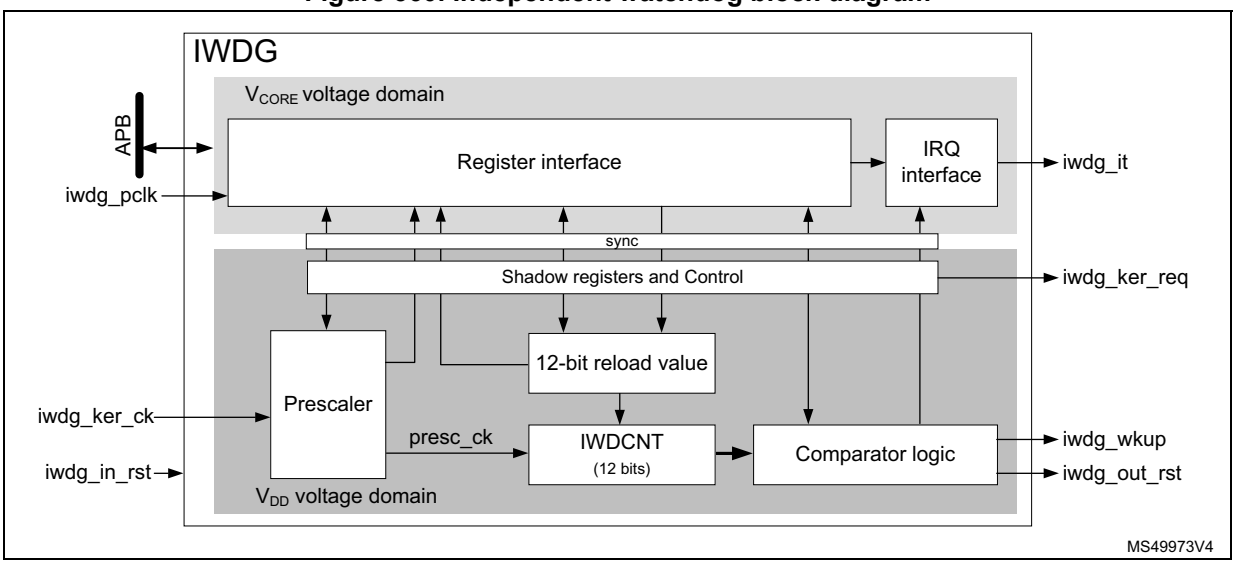
4. Controlled via the option byte IWDG_STOP in FLASH section.
5. Controlled via the option byte IWDG_STDBY in FLASH section.
6. Controlled via the option byte IWDG_SW in FLASH section.

30.4 IWDG functional description

30.4.1 IWDG block diagram

Figure 360 shows the functional blocks of the independent watchdog module.

Figure 360. 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 30.3* 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 226. 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.

30.4.2 IWDG internal signals

The list of IWDG internal signals is detailed in *Table 227*.

Table 227. 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

30.4.3 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 30.3* 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).

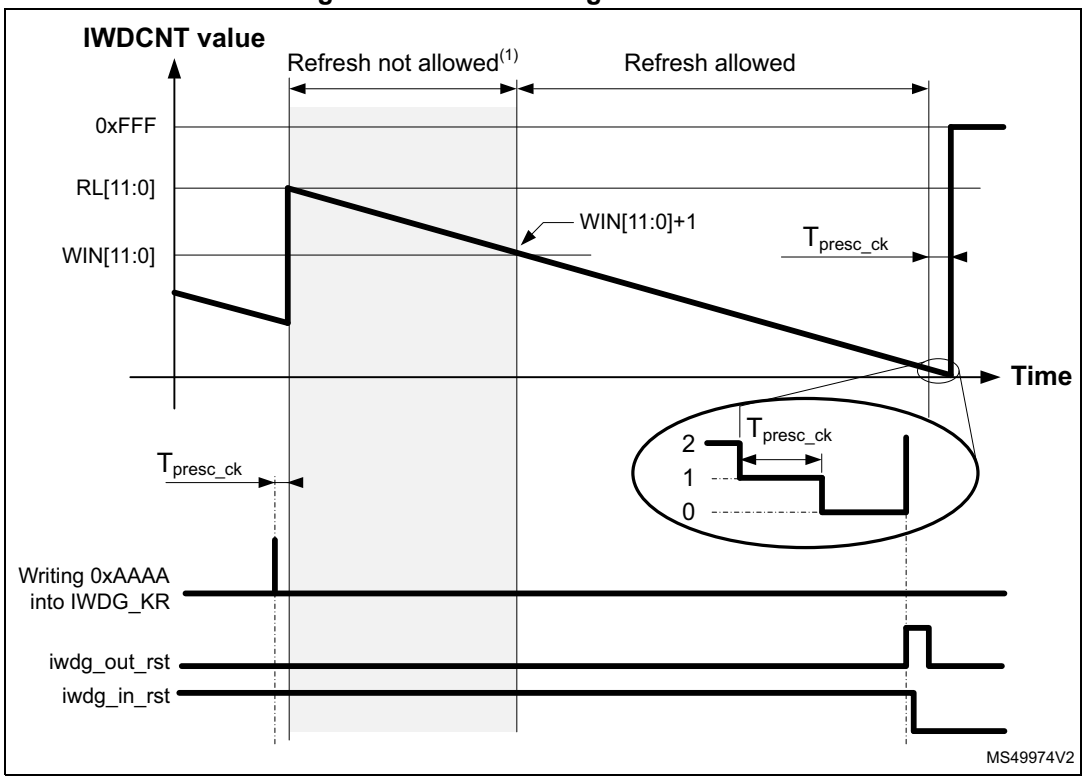
Whenever the key value 0x0000 AAAA is written in the *IWDG key register (IWDG_KR)*, the IWDG_RLR value is reloaded into the IWDCNT, and the watchdog reset is prevented.

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 361*, when the refresh command is executed, one period of presc_ck later, the IWDCNT is reloaded with the content of RL[11:0].

Figure 361. 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.

30.4.4 Window option

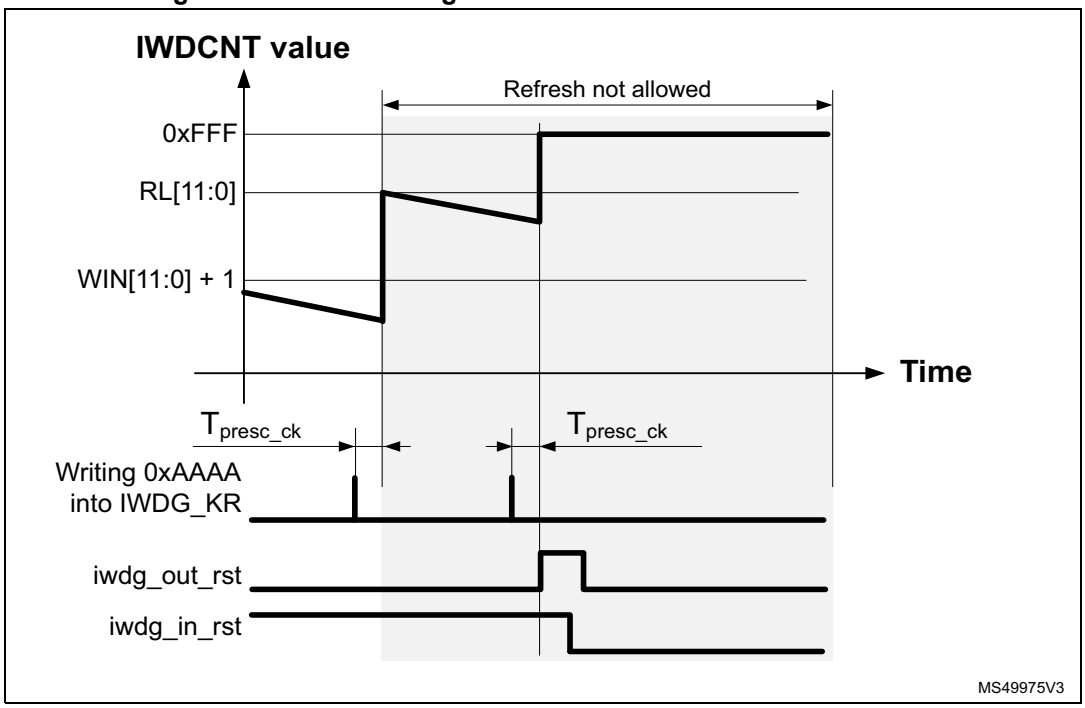
The IWDG can also work as a window watchdog, by setting the appropriate window in the *IWDG window register (IWDG_WINR)*.

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 362*, 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 362. 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.

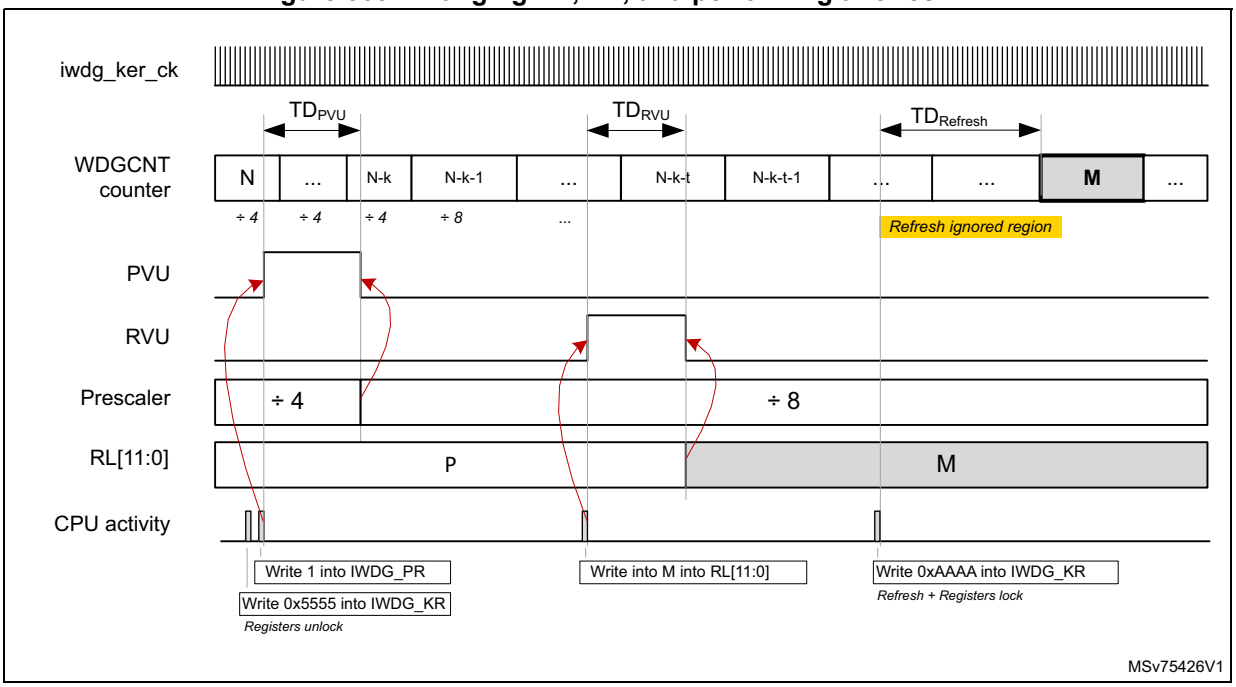
Configuring the IWDG when the window option is disabled

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 363. Changing PR, RL, and performing a refresh(1)



1. Refer to *Table 226: 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

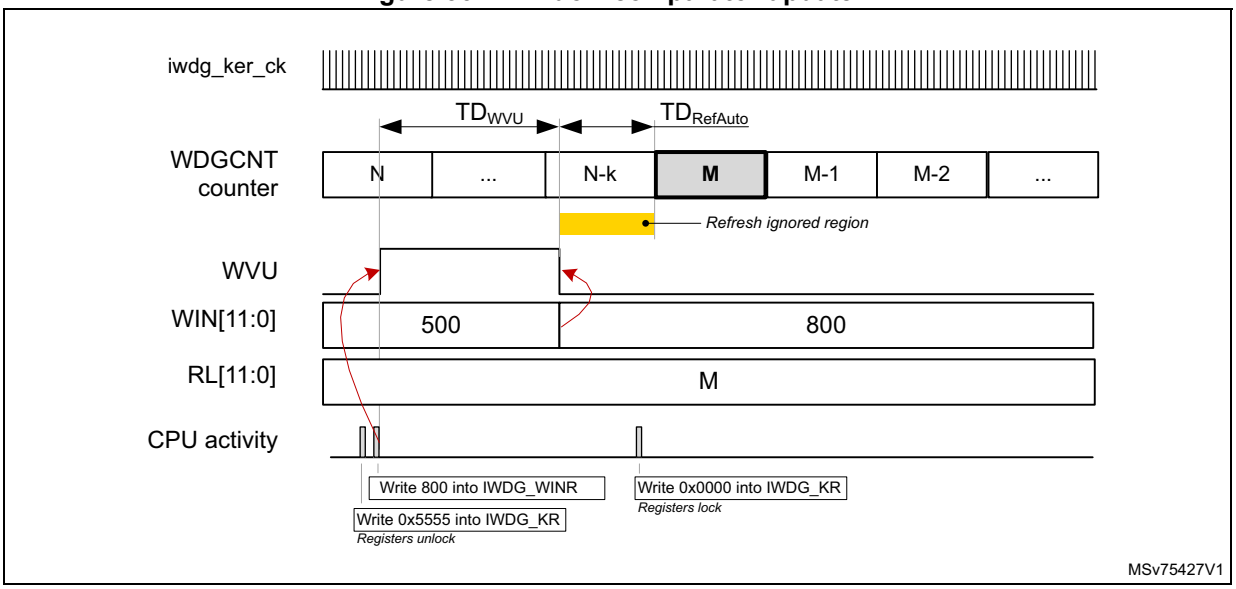
It is possible to update the window comparator when the IWDG is already running. The IWDCNT is reloaded as well. The following sequence can be performed to update the window comparator:

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.

Figure 364 shows this sequence. As soon as the IWDG_WINR register is written, the WVU flag goes high for TDwvu. After a window comparator update, a refresh is automatically performed. The refresh is effective in the worst case, on the next prescaler clock active edge after the falling edge of WVU (TDRefAuto)

Figure 364. Window comparator update(1)



1. Refer to *Table 226: IWDG delays versus actions* for details on timing values.

30.4.5 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 30.3* for details on the capabilities of this product.

30.4.6 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.

30.5 IWDG low power modes

Depending on option bytes configuration, the IWDG can continue counting or not during the low power modes. Refer to *Section 30.3* for details.

Table 228. 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 30.3* 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 30.3* for details. IWDG interrupts do not make the device to exit from Standby mode.

30.6 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.

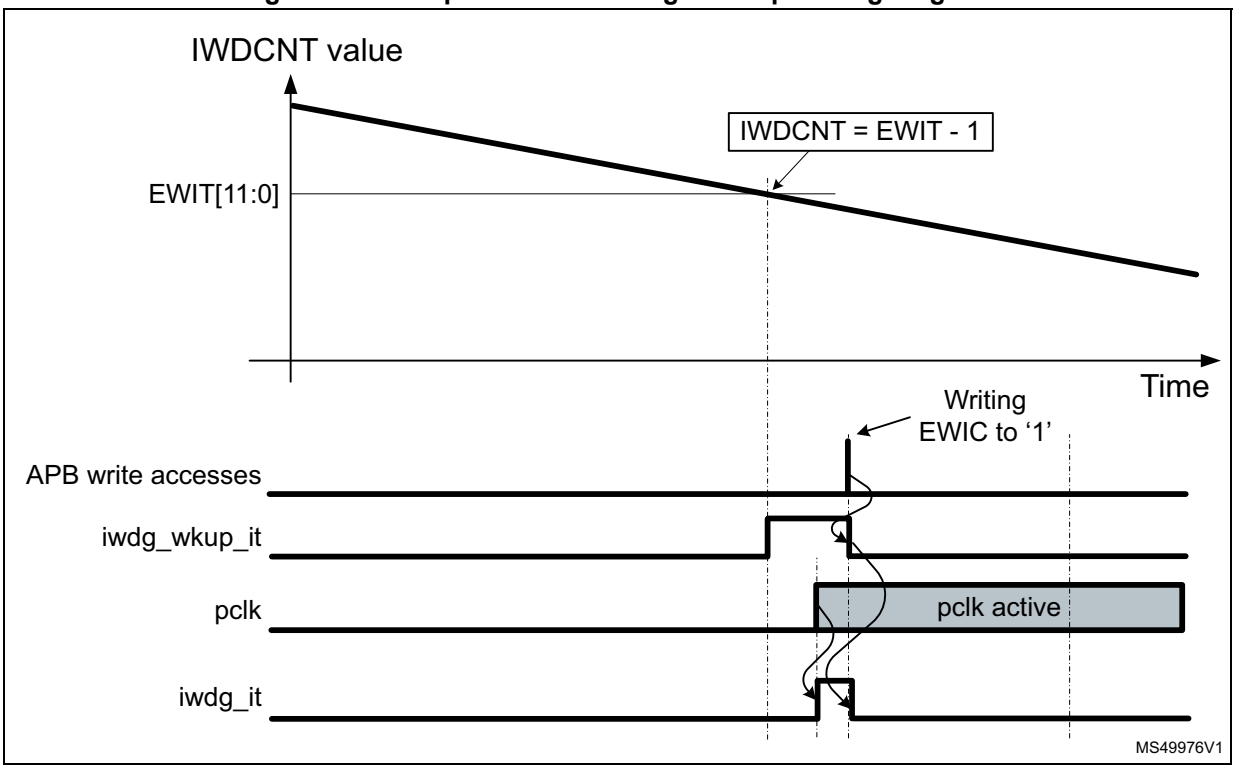
When the APB clock is available, the iwdg_it is activated as well.

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 365. 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

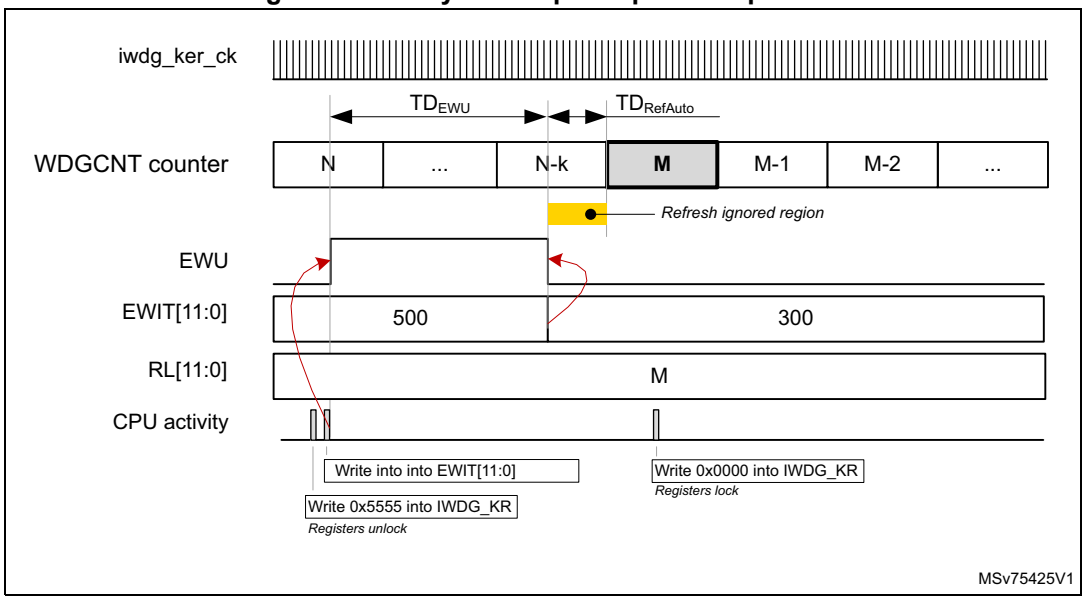
It is possible to change the early wake-up comparator value or to enable/disable the interrupt generation at any time, by performing the following sequence:

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.

Figure 365 shows this sequence. During the early wake-up comparator update operation the flag EWU remains to 1 for TDewu. A refresh is automatically performed. The refresh is effective on the worst case, on the next prescaler clock active edge after the falling edge of EWU (TDRefAuto).

Figure 366. Early wake-up comparator update(1)



1. Refer to *Table 226: IWDG delays versus actions* for details on timing values.

Table 229 summarizes the IWDG interrupt request.

Table 229. 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.

30.7 IWDG registers

Refer to *Section 1.2 on page 65* 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.

30.7.1 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.

30.7.2 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 30.4.6*. 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.

30.7.3 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 30.4.6*. 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.

30.7.4 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
Res.	EWIF	Res.	Res.	Res.	Res.	Res.	ONF	Res.	Res.	Res.	Res.	EWU	WVU	RVU	PVU
	r						r					r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **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 13:9 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 226: 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 226: 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 226: 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 226: 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.

30.7.5 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 30.4.6*.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.

30.7.6 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 window value

These bits are write access protected (see *Section 30.4.6*). 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.

30.7.7 IWDG register map

Table 230. 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.	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 2.2* for the register boundary addresses.

31 System window watchdog (WWDG)

31.1 Introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence.

The watchdog circuit generates a reset on expiry of a programmed time period, unless the program refreshes the contents of the down-counter before the T6 bit is cleared. A reset is also generated if the 7-bit down-counter value (in the control register) is refreshed before the down-counter reaches the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB clock and has a configurable time window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications requiring the watchdog to react within an accurate timing window.

31.2 WWDG main features

- Programmable free-running down-counter
- Conditional reset
  - Reset (if watchdog activated) when the down-counter value becomes lower than 0x40
  - Reset (if watchdog activated) if the down-counter is reloaded outside the window (see *Figure 368*)
- Early wake-up interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40

31.3 WWDG implementation

Table 231. WWDG features(1)

WWDG mode / feature	WWDG
Window function	X
Early wake-up interrupt generation	X
System reset generation(2)	X
Capability to work in system Stop	-
Capability to work in system Standby	-
Capability to be frozen when the microcontroller enters in Debug mode(3)	X
Option bytes to control the hardware mode(4)	X

1. “X” = supported, “-” = not supported.
2. Refer to the RCC section for additional information.
3. Controlled via WWDG_STOP in DBG block.

4. Controlled via the option byte WWDG_SW. When WWDG_SW is set in HW mode the WWDG is running as soon as the CPU is in Run or Sleep modes.

31.4 WWDG functional description

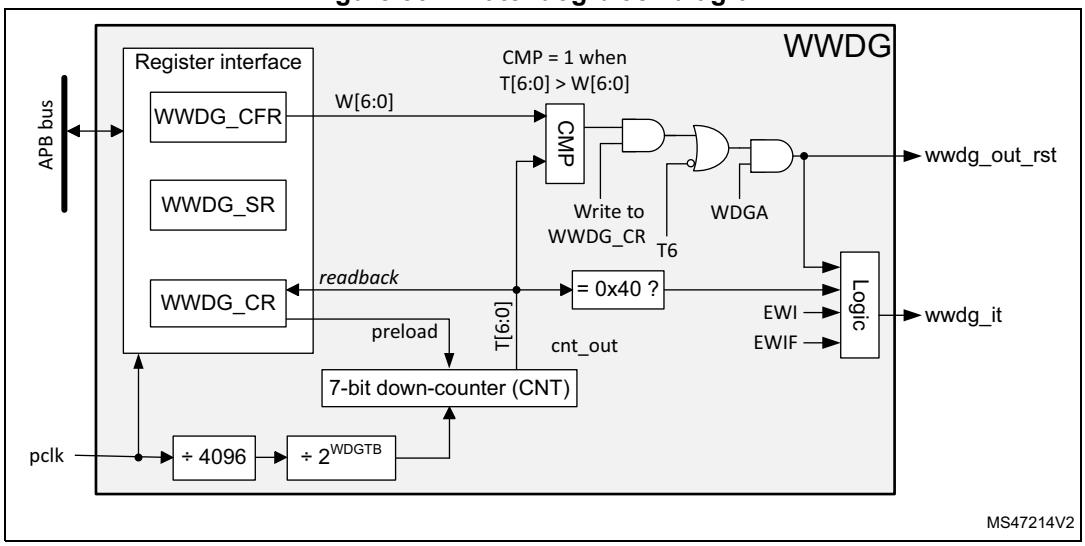
If the watchdog is activated (the WDGA bit is set in the WWDG_CR register), and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent a reset. This operation can take place only when the counter value is lower than or equal to the window register value, and higher than 0x3F. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

Refer to *Figure 367* for the WWDG block diagram.

31.4.1 WWDG block diagram

Figure 367. Watchdog block diagram



31.4.2 WWDG internal signals

Table 232 gives the list of WWDG internal signals.

Table 232. WWDG internal input/output signals

Signal name	Signal type	Description
pclk	Digital input	APB bus clock
wwdg_out_rst	Digital output	WWDG reset signal output
wwdg_it	Digital output	WWDG early interrupt output

31.4.3 Enabling the watchdog

When the user option WWDG_SW selects “Software window watchdog”, the watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again, except by a reset.

When the user option WWDG_SW selects “Hardware window watchdog”, the watchdog is always enabled after a reset, it cannot be disabled.

31.4.4 Controlling the down-counter

This down-counter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments that represent the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value, due to the unknown status of the prescaler when writing to the WWDG_CR register (see *Figure 368*). The *WWDG configuration register (WWDG_CFR)* contains the high limit of the window: to prevent a reset, the down-counter must be reloaded when its value is lower than or equal to the window register value, and greater than 0x3F. *Figure 368* describes the window watchdog process.

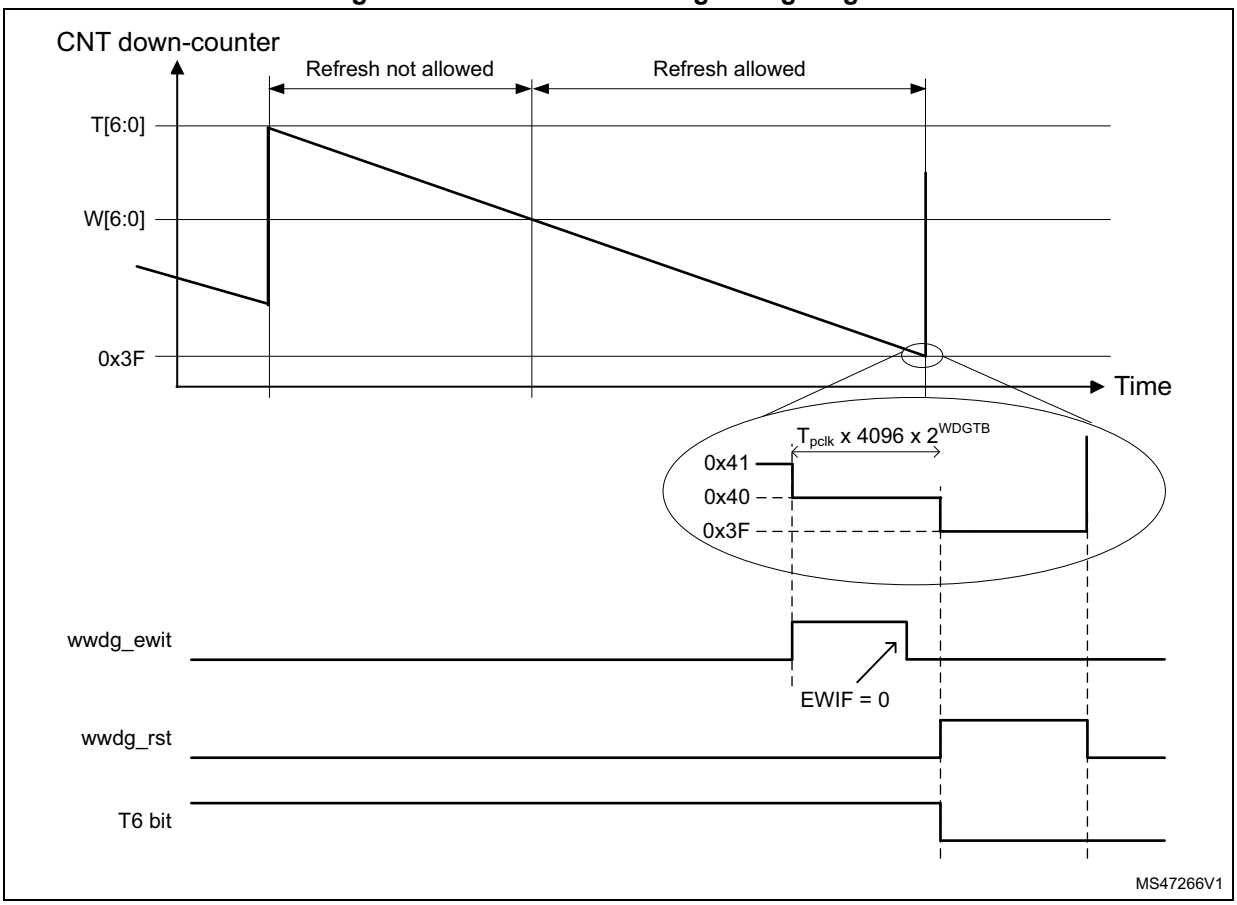
Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

31.4.5 How to program the watchdog timeout

Use the formula in *Figure 368* to calculate the WWDG timeout.

Warning: When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.

Figure 368. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WDGTB[2:0]} \times (T[5:0] + 1) \quad (ms)$$

where:

- t_{WWDG}: WWDG timeout
- t_{PCLK}: APB clock period measured in ms
- 4096: value corresponding to internal divider

As an example, if APB frequency is 48 MHz, WDGTB[2:0] is set to 3, and T[5:0] is set to 63:

$$t_{WWDG} = (1/\ 48000) \times 4096 \times 2^{3} \times (63 + 1) = 43.69ms$$

Refer to the datasheet for the minimum and maximum values of t_{WWDG}.

31.4.6 Debug mode

When the device enters debug mode (processor halted), the WWDG counter either continues to work normally or stops, depending on the configuration bit in DBG module. For more details, refer to .

31.5 WWDG interrupts

The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the reset is generated. To enable the early wake-up interrupt, the application must:

- Write EWIF bit of WWDG_SR register to 0, to clear unwanted pending interrupt
- Write EWI bit of WWDG_CFR register to 1, to enable interrupt

When the down-counter reaches the value 0x40, a watchdog interrupt is generated, and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case the corresponding ISR must reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The watchdog interrupt is cleared by writing 0 to the EWIF bit in the WWDG_SR register.

Note: When the watchdog interrupt cannot be served (for example due to a system lock in a higher priority task), the WWDG reset is eventually generated.

Table 233. WWDG interrupt requests

Interrupt		Event flag	Enable control bit	Interrupt clearing method	Exit from mode		
Vector	Event				Sleep	Stop(1)	Standby(1)
WWDG(2)	Early wake-up interrupt	EWIF	EWI	Write EWIF flag to 0	Yes	No	No

1. The WWDG interrupt can have additional capabilities, refer to *Section 31.3* for details.
2. WWDG vector corresponds to the assertion of the wwdg_it signal.

31.6 WWDG registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by halfwords (16-bit) or words (32-bit).

31.6.1 WWDG control register (WWDG_CR)

Address offset: 0x000

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	WDGA	T[6:0]						
								rs	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **WDGA:** Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]:** 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter, decremented every (4096 x $2^{WDGTB[2:0]}$) PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

31.6.2 WWDG configuration register (WWDG_CFR)

Address offset: 0x004

Reset value: 0x0000 007F

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

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:11 **WDGTB[2:0]:** Timer base

The timebase of the prescaler can be modified as follows:

000: CK counter clock (PCLK div 4096) div 1

001: CK counter clock (PCLK div 4096) div 2

010: CK counter clock (PCLK div 4096) div 4

011: CK counter clock (PCLK div 4096) div 8

100: CK counter clock (PCLK div 4096) div 16

101: CK counter clock (PCLK div 4096) div 32

110: CK counter clock (PCLK div 4096) div 64

111: CK counter clock (PCLK div 4096) div 128

Bit 10 Reserved, must be kept at reset value.

Bit 9 **EWI:** Early wake-up interrupt enable

Set by software and cleared by hardware after a reset. When set, an interrupt occurs whenever the counter reaches the value 0x40.

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:0 **W[6:0]:** 7-bit window value

These bits contain the window value to be compared with the down-counter.

31.6.3 WWDG status register (WWDG_SR)

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.	Res.	Res.	EWIF
															rc_w0

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

Bit 0 **EWIF:** Early wake-up interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing 0. Writing 1 has no effect. This bit is also set if the interrupt is not enabled.

31.6.4 WWDG register map

The following table gives the WWDG register map and reset values.

Table 234. WWDG 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	WWDG_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.	WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x004	WWDG_CFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGTB [2:0]			Res.	EWI	Res.	Res.	W[6:0]						
	Reset value																			0	0	0		0			1	1	1	1	1	1	1
0x008	WWDG_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.	Res.	Res.	EWIF
	Reset value																																0

Refer to *Section 2.2* for the register boundary addresses.

32 Real-time clock (RTC)

32.1 Introduction

The RTC provides an automatic wake-up to manage all low-power modes.

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 or under reset).

The RTC is functional in V_{BAT} mode.

32.2 RTC main features

The RTC supports the following features (see *Figure 369: 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, or by a switch to V_{BAT} mode.
- 17-bit auto-reload wake-up timer (WUT) for periodic events with programmable resolution and period.
- Alarm A, alarm B, wake-up Timer and timestamp individual privilege protection

The RTC is supplied through a switch that takes power either from the V_{DD} supply when present or from the VBAT pin.

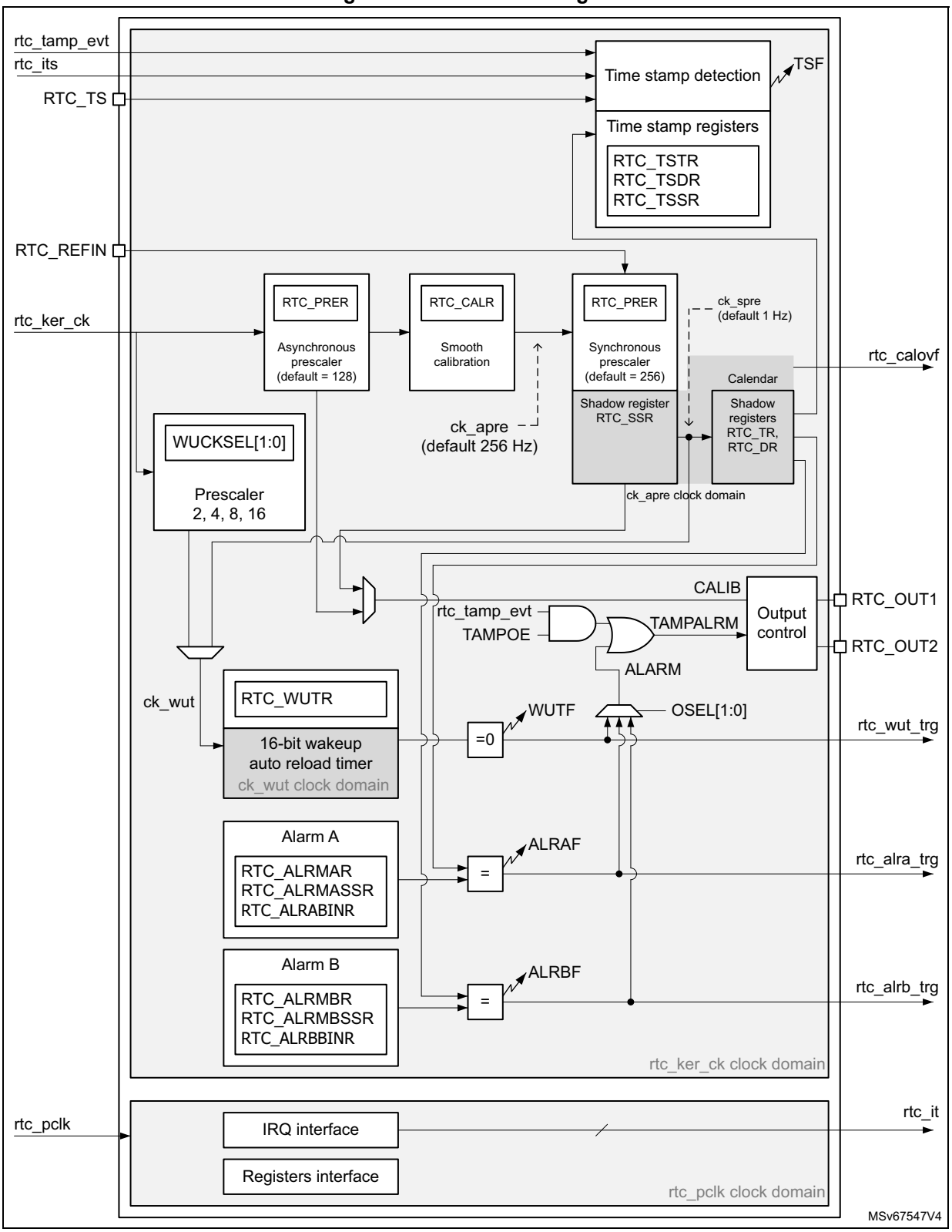
The RTC is functional in V_{BAT} mode and in all low-power modes when it is clocked by the LSE.

All RTC events (Alarm, wake-up Timer, Timestamp) can generate an interrupt and wake-up the device from the low-power modes.

32.3 RTC functional description

32.3.1 RTC block diagram

Figure 369. RTC block diagram



32.3.2 RTC pins and internal signals

Table 235. 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 236. 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_its	Input	RTC internal timestamp event
rtc_tamp_evt	Input	Tamper event (internal or external) detected in TAMP peripheral
rtc_it	Output	RTC interrupts (refer to *Section 32.5: 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 or V_{BAT} when the selected clock is not LSE. Refer to *Section 32.4: RTC low-power modes* for more details.

Table 237. RTC interconnection

Signal name	Source/destination
rtc_its	From power controller (PWR): main power loss/switch to V_{BAT} detection output
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.

32.3.3 GPIOs controlled by the RTC and TAMP

The GPIOs included in the Battery Backup Domain (V_{BAT}) are directly controlled by the peripherals providing functions on these I/Os, whatever the GPIO configuration.

RTC_OUT1, RTC_TS, TAMP_IN1 and TAMP_OUT2 are mapped on the same pin (PC13). The RTC and TAMP functions mapped on PC13 are available in all low-power modes and in V_{BAT} mode.

The output mechanism follows the priority order shown in *Table 238*.

Table 238. 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	TAMP2E=TAMP2AM=1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE=1 and ATOSELx=1	TAMP1E (TAMP_IN1 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 238. 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	TAMP2E=TAMP2AM=1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE=1 and ATOSELx=1	TAMP1E (TAMP_IN1 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_OUT2 output PP		00	0	0	0	Don't care	Don't care	1	Don't care	Don't care
TAMP_IN1 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_IN1 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		

Table 238. 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	TAMP2E=TAMP2AM=1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE=1 and ATOSELx=1	TAMP1E (TAMP_IN1 input enable)	TSE (RTC_TS input enable)
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		
Wakeup pin	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		
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 PA0 or 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 *Table 239*.

Table 239. RTC_OUT mapping

OSEL[1:0] bits ALARM output enable)	COE bit (CALIB output enable)	OUT2EN bit	RTC_OUT1 on PC13	RTC_OUT2 on PA0 or 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

32.3.4 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 non-privileged 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_PRIVCFGR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in privilege and non-privilege 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 non-privilege 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 can be written only in privilege mode.
  - These registers and control bits can be read in privilege and non-privilege mode.
- When CALPRIV is set:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and non-privilege 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 can be read and 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 can be read and 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 can be read and written only in privilege mode.
- When TSPRIV is set:
  - RTC_TSTR, RTC_TSDR and RTC_TSSSR registers, plus TAMPTS, ITSE, TSE, TSIE, TSEDGE control bits in the RTC_CR, CITSF, CTSOVF and CTSF bits in the RTC_SCR, TSF, TSOVF and ITSF in RTC_SR, and TSMF, TSOVMF and ITSMF in RTC_MISR can be read and written only in privilege mode.

A non-privileged 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.

32.3.5 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 369: 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 32.3.9: 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.

32.3.6 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 32.6.11: RTC shift control register (RTC_SHIFTR)*). The copy is not performed in Stop and Standby 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.

32.3.7 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.

32.3.8 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.

32.3.9 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 1085*), the timer starts counting down. When the wake-up function is enabled, the down-counting remains active in low-power modes. 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 32.3.1: 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.

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, and Standby) have no influence on the wake-up timer.

32.3.10 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 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 in order to enable RTC registers write access.

After Backup domain reset, some of the RTC registers are write-protected: RTC_TR, RTC_DR, RTC_PRER, RTC_CALR, RTC_SHIFTR, the bits INIT, BIN and BCDU in RTC_ICSR and the bits FMT, SUB1H, ADD1H, REFCKON in RTC_CR.

The following steps are required to unlock the write protection on the protected RTC registers.

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

The registers protected by INITPRIV are write-protected by the INIT KEY.

The registers protected by CALPRIV are write-protected by the CAL KEY.

In case PRIV or INITPRIV is set in the RTC_PRIVCFGR: the INIT KEY is unlocked and locked only if the write accesses into the RTC_WPR register are done in the privilege mode defined by PRIV, INITPRIV configuration.

In case PRIV or CALPRIV is set in the RTC_PRIVCFGR: the CAL KEY is unlocked and locked only if the write accesses into the RTC_WPR register are done in the privilege mode defined by PRIV, CALPRIV configuration.

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: 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. This step must be skipped in calendar initialization mode.
   – 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.

32.3.11 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 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 or Standby), 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 1084*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to *Section 32.3.13: 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 or Standby), 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 results to confirm that the data is coherent and correct. 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.

32.3.12 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.

32.3.13 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.

32.3.14 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 mode.

32.3.15 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.

32.3.16 Timestamp function

Timestamp is enabled by setting the TSE or ITSE bits 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 ITSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal timestamp event is detected. The internal timestamp event is generated by the switch to the V_{BAT} supply.

When a timestamp event occurs, due to internal or external event, the timestamp flag bit (TSF) in RTC_SR register is set. In case the event is internal, the ITSF flag is also set in RTC_SR register.

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 or from rtc_its internal signal 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.

32.3.17 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 fRTCCLK/(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.

32.3.18 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.*

32.4 RTC low-power modes

Table 240. Effect of low-power modes on RTC

Mode	Description
Sleep	No effect RTC interrupts cause the device to exit the Sleep mode.
Stop	The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Stop mode.
Standby	The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Standby mode.

The table below summarizes the RTC pins and functions capability in all modes.

Table 241. RTC pins functionality over modes

Functions	Functional in all low-power modes except Standby mode	Functional in Standby mode	Functional in V_{BAT} mode
RTC_TS	Yes	Yes	Yes
RTC_REFIN	Yes	No	No
RTC_OUT1	Yes	Yes	Yes
RTC_OUT2	Yes	No	No

32.5 RTC interrupts

The interrupt channel is set in the masked interrupt status register. The interrupt output is also activated.

Table 242. 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	write 1 in CALRAF	Yes(3)
	Alarm B	ALRBF	ALRBIE	write 1 in CALRBF	Yes(3)
	Timestamp	TSF	TSIE	write 1 in CTSF	Yes(3)
	Wake-up timer	WUTF	WUTIE	write 1 in CWUTF	Yes(3)
	SSR underflow (reload)	SSRUF	SSRUIE	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.

32.6 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).

32.6.1 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 1084* and *Reading the calendar on page 1086*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1084*.

This register can be write-protected against non-privileged access. Refer to *Section 32.3.4: 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

32.6.2 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 1084* and *Reading the calendar on page 1086*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1084*.

This register can be write-protected against non-privileged access. Refer to *Section 32.3.4: 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.

32.6.3 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.

32.6.4 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 1084*.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 32.3.4: 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

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.

It is cleared by hardware in initialization mode.

0: Wake-up timer configuration update not allowed except in initialization mode

1: Wake-up timer configuration update allowed

Bits 1:0 Reserved, must be kept at reset value.

32.6.5 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 1084*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1084*.

This register can be write-protected against non-privileged access. Refer to *Section 32.3.4: 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)

32.6.6 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 non-privileged access. Refer to *Section 32.3.4: 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.

32.6.7 RTC control register (RTC_CR)

This register is write protected. The write access procedure is described in RTC register write protection on page 1084.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 32.3.4: 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	ITSE	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
rw	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 **ITSE**: timestamp on internal event enable
0: internal event timestamp disabled
1: internal event timestamp enabled

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 32.3.17: 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**: Subtract 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

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.

32.6.8 RTC privilege mode control register (RTC_PRIVCFGR)

This register can be written only when the APB access is privileged.

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 non-privileged, 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 non-privileged.
1: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is privileged.

Bit 13 **CALPRIV**: Shift register, Delight saving, calibration and reference clock privilege protection
0: Shift register, Delight saving, calibration and reference clock can be written when the APB access is privileged or non-privileged.
1: Shift register, Delight 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 non-privileged.
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 non-privileged.
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 non-privileged.
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 non-privileged.
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 32.3.4: RTC privilege protection modes for details on the read protection.

This register can be globally write-protected, or each bit of this register can be individually write-protected against non-privileged access depending on the RTC_PRIVCFGR configuration (refer to *Section 32.3.4: RTC privilege protection modes*).

32.6.9 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.

32.6.10 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1084*.

This register can be write-protected against non-privileged access. Refer to *Section 32.3.4: 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 32.3.15: 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 32.3.15: 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 32.3.15: 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 32.3.15: RTC smooth digital calibration on page 1089*.

32.6.11 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1084*.

This register can be protected against non-privileged access. Refer to *Section 32.3.4: 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.

32.6.12 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 non-privileged access. Refer to *Section 32.3.4: 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.

32.6.13 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 non-privileged access. Refer to *Section 32.3.4: 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

32.6.14 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 non-privileged access. Refer to *Section 32.3.4: 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.

32.6.15 RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register can be protected against non-privileged access. Refer to *Section 32.3.4: 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.

32.6.16 RTC alarm A subsecond register (RTC_ALRMASSR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register can be protected against non-privileged access. Refer to *Section 32.3.4: 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.

32.6.17 RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register can be protected against non-privileged access. Refer to *Section 32.3.4: 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

32.6.18 RTC alarm B subsecond register (RTC_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register can be protected against non-privileged access. Refer to *Section 32.3.4: 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.

32.6.19 RTC status register (RTC_SR)

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 32.3.4: 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	ITSF	TSOVF	TSF	WUTF	ALRBF	ALRAF
									r	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.

Bit 5 **ITSF**: Internal timestamp flag

This flag is set by hardware when a timestamp on the internal event occurs.

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.

If ITSF flag is set, TSF must be cleared together with ITSF.

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.

32.6.20 RTC masked interrupt status register (RTC_MISR)

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 32.3.4: 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	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUMF**: SSR underflow masked flag
This flag is set by hardware when the SSR underflow interrupt occurs.

Bit 5 **ITSMF**: Internal timestamp masked flag
This flag is set by hardware when a timestamp on the internal event occurs and timestampinterrupt is raised.

Bit 4 **TSOVMF**: Timestamp overflow 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 masked flag
This flag is set by hardware when a timestamp interrupt occurs.
If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2 **WUTMF**: Wake-up timer masked flag
This flag is set by hardware when the wake-up timer 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 masked flag
This flag is set by hardware when the alarm B interrupt occurs.

Bit 0 **ALRAMF**: Alarm A masked flag
This flag is set by hardware when the alarm A 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.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 32.3.4: RTC privilege protection modes*.

32.6.21 RTC status clear register (RTC_SCR)

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 32.3.4: 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	CITSF	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
									w	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 **CITSF**: Clear internal timestamp flag
Writing 1 in this bit clears the ITSF bit in the RTC_SR register.

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.
If ITSF flag is set, TSF must be cleared together with ITSF by setting CRSF and CITSF.

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.

32.6.22 RTC alarm A binary mode register (RTC_ALRABINR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register can be protected against non-privileged access. Refer to *Section 32.3.4: 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 in Binary mode

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_ALRMASSRR, 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.

32.6.23 RTC alarm B binary mode register (RTC_ALRBBINR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register can be protected against non-privileged access. Refer to *Section 32.3.4: 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 in Binary mode

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_ALRMBSSRR, 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.

32.6.24 RTC register map

Table 243. 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	ITSE	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	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
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
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

Table 243. 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
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	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	ITSF	TSOVF	TSF	WUTF	ALRBF	ALRAF
	Reset value																										0	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	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																										0	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	CITSF	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
	Reset value																										0	0	0	0	0	0	0
0x60	RTC_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUT2_RMP
	Reset value																																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 2.2* for the register boundary addresses.

33 Tamper and backup registers (TAMP)

33.1 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 and also in V_{BAT} mode. The backup registers, as well as other secrets in the device, are protected by this anti-tamper detection circuit with 2 tamper pins and 13 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.

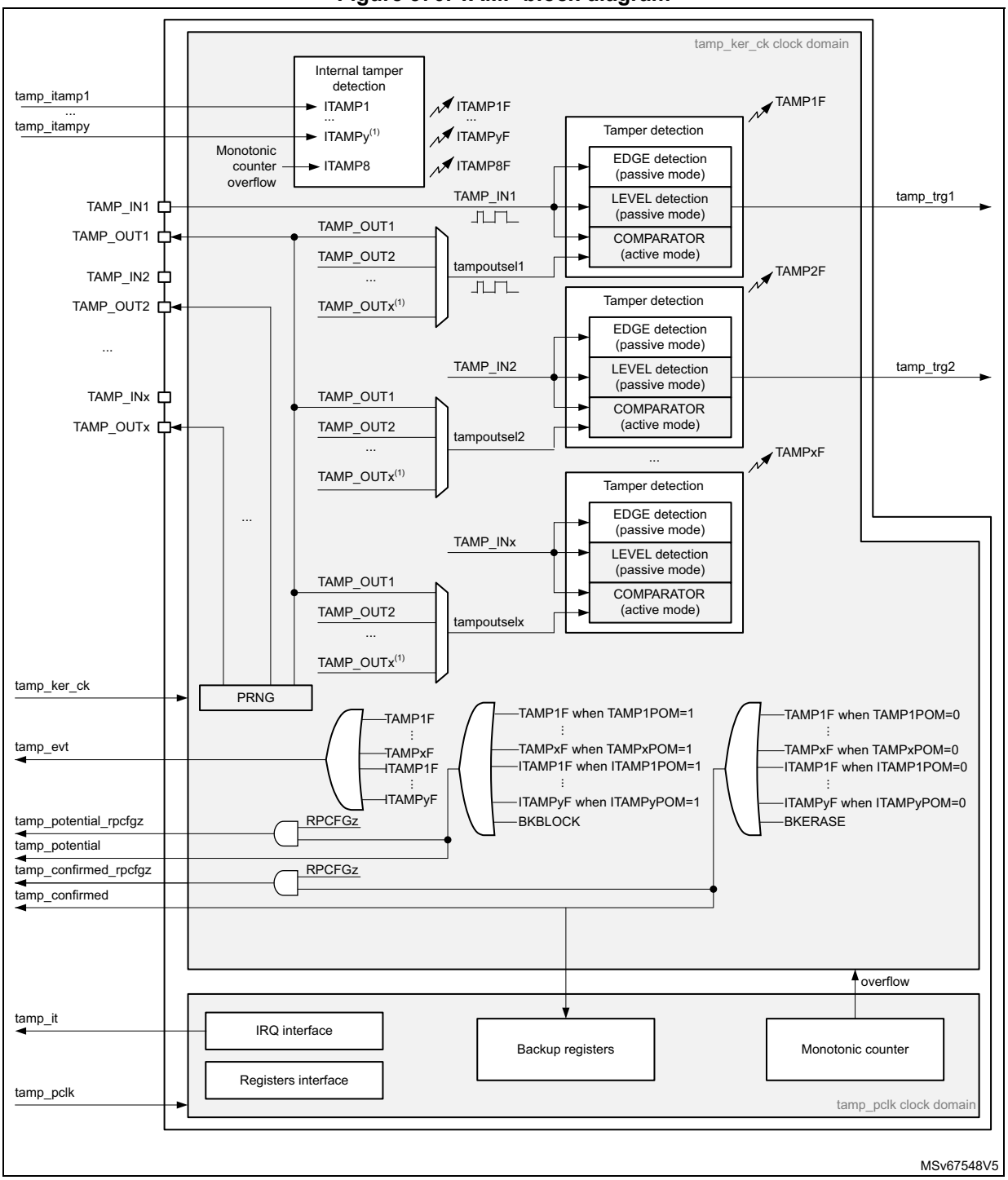
33.2 TAMP main features

- A tamper detection can optionally erase the backup registers, backup SRAM, SRAM2, caches. The device resources protected by tamper are named “device secrets”.
- 32 32-bit backup registers:
  - The backup registers (TAMP_BKPxR) are implemented in the backup domain that remains powered-on by V_{BAT} when the V_{DD} power is switched off.
- Up to 2 tamper pins for 2 external tamper detection events:
  - Active tamper mode: continuous comparison between tamper output and input to protect from physical open-short attacks.
  - Passive tampers: ultra-low power edge or level detection with internal pull-up hardware management.
  - Configurable digital filter.
- 13 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.
- Tamper configuration and backup registers privilege protection
- Monotonic counter.

33.3 TAMP functional description

33.3.1 TAMP block diagram

Figure 370. TAMP block diagram



1. The number of external and internal tampers depends on products.

33.3.2 TAMP pins and internal signals

Table 244. 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 245. 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_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 245. 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 33.5: TAMP interrupts* for details)
tamp_trg[x] (x = signal index)	Output	Tamper detection trigger

1. Refer to *Table 246: 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 or V_{BAT} depending on the selected clock (refer to *Section 33.4: TAMP low-power modes* for more details.

Table 246. TAMP interconnection

Signal name	Source/Destination
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 – SRAM2 The tamp_potential signal is used to erase the device secrets listed hereafter: – ICACHE content The device secrets access is blocked when erase is ongoing.
tamp_confirmed	The tamp_confirmed signal is used to erase the device secrets listed hereafter: – backup registers – SRAM2 – ICACHE content The device secrets access is blocked when erase is ongoing.

Table 246. TAMP interconnection (continued)

Signal name	Source/Destination
tamp_potential_rpcfg0	When the bit RPCFG0 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg0 signal is used to block the read and write accesses to the device secrets listed hereafter: – Backup SRAM
tamp_confirmed_rpcfg0	When the bit RPCFG0 is set in the TAMP_RPCFGR, the tamp_confirmed_rpcfg0 signal is used to erase the device secrets listed hereafter: – Backup SRAM The device secrets access is blocked when erase is on-going.
tamp_itamp1	Backup domain voltage threshold monitoring(1)
tamp_itamp2	Temperature monitoring(1)
tamp_itamp3	LSE monitoring (LSECSS)(2)
tamp_itamp4	HSE monitoring (rcc_hsecss_fail)
tamp_itamp5	RTC calendar overflow (rtc_calovf)
tamp_itamp6	JTAG/SWD access when NVSTATE ≠ OPEN
tamp_itamp7	ADC1 watchdog monitoring 1
tamp_itamp8(3)	Monotonic counter 1 overflow
tamp_itamp9	TRNG fault
tamp_itamp11	IWDG reset when tamper flag is set (potential tamper timeout)
tamp_itamp12	ADC1 analog watchdog monitoring 2
tamp_itamp13	ADC1 analog watchdog monitoring 3
tamp_itamp15	System fault

1. This monitoring must be enabled by setting MONEN in *PWR backup domain control register (PWR_BDCR)*.
2. This monitoring must be enabled by setting LSEON in *PWR backup domain control register (PWR_BDCR)*.
3. This signal is generated in the TAMP peripheral.

33.3.3 GPIOs controlled by the RTC and TAMP

Refer to *Section 32.3.3: GPIOs controlled by the RTC and TAMP*.

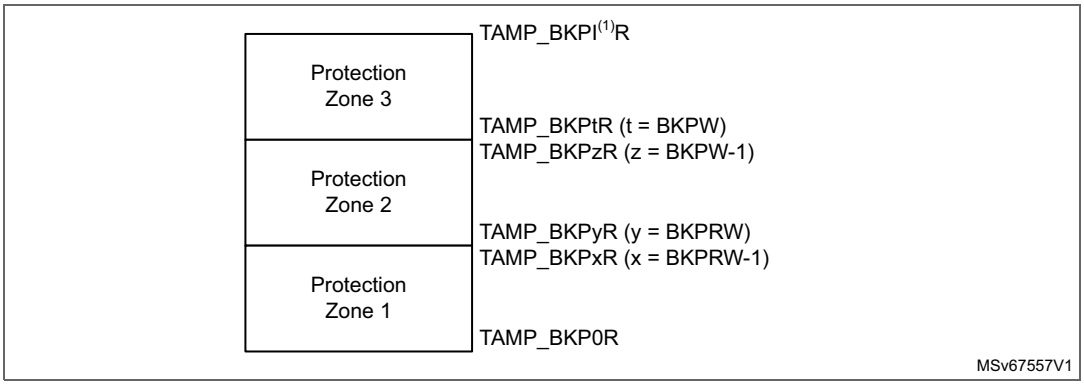
33.3.4 TAMP register write protection

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.

33.3.5 Backup registers protection zones

The backup registers protection is configured thanks to BKPRW[7:0] and BKPW[7:0] (refer to the figure below):

Figure 371. Backup registers protection zones



1. I = last backup register index

33.3.6 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 non-privileged 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 counters 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.
- Reading TAMP_CFGR, TAMP_PRIVCFGR is always possible in privilege and non-privilege modes. All the other TAMP registers can be read only in privileged mode, except for the backup registers and the monotonic counters which have their own protection setting.

The backup registers protection is configured thanks to BKPRW[7:0] and BKPRWPRIV for the protection zone 1, and thanks to BKPRW[7:0], BKPW[7:0] and BKPWPRIV for the protection zone 2 (refer to *Figure 371*).

A non-privileged 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.

33.3.7 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 246: 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 246: 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.

33.3.8 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 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 product state is changed from Closed to Open.

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 246: 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 246: 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 246: 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 246: TAMP interconnection*
- It is affected by potential tamper detection and BKBLOCK software command, as described with tamp_potential_rpcfgz signal in *Table 246: TAMP interconnection*

Table 247. 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 246: 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.

33.3.9 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 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.

Note: Tamper detection is still active when V_{DD} power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the TAMPx is mapped should be externally tied to the correct level.

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 372. Tamper sampling with precharge pulse

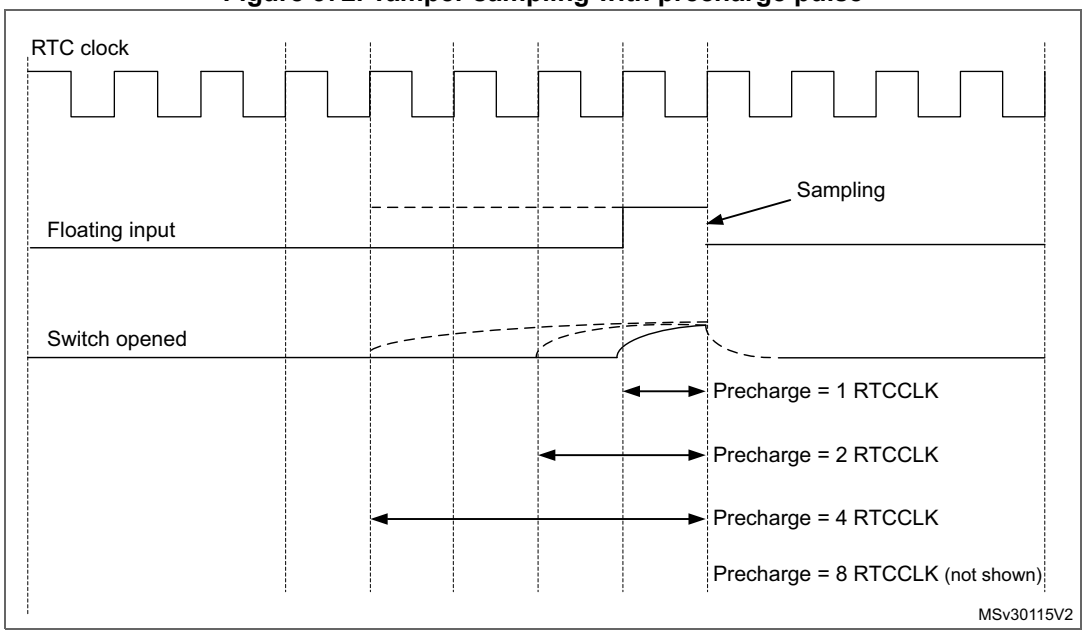
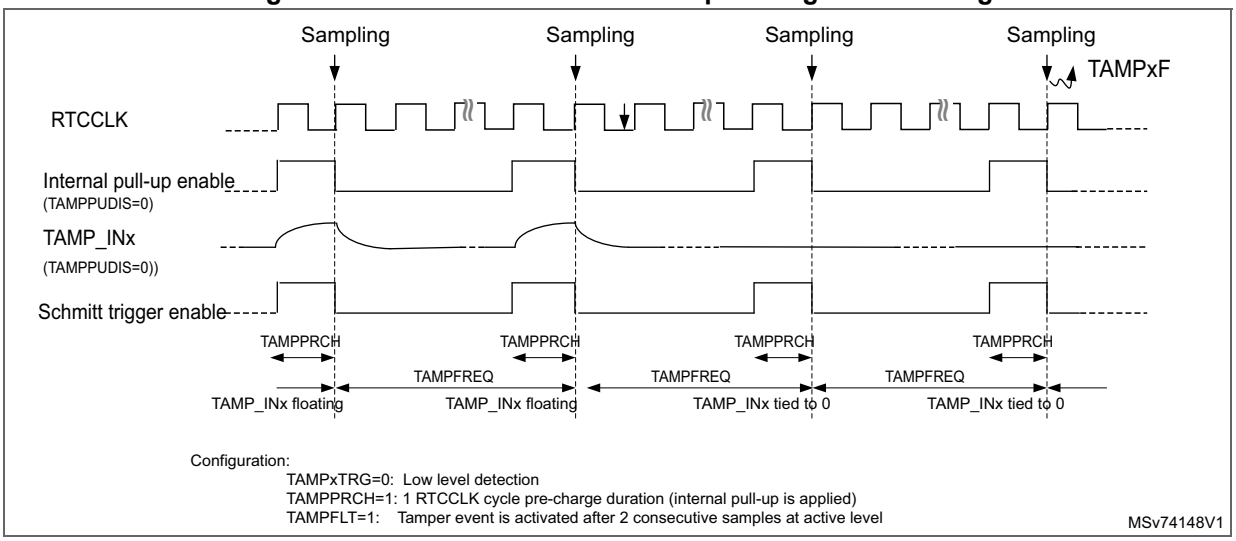


Figure 373. Low level detection with precharge and filtering



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 248. 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 249. Minimum ATPER value

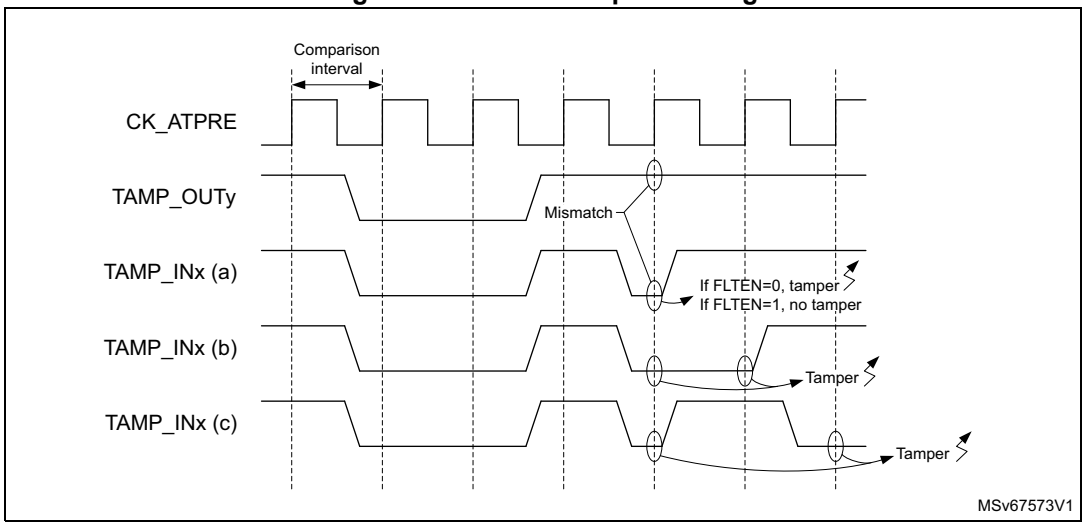
Number of selected outputs	Minimum ATPER
1	0
2	1
3 or 4	2
5 or more	3

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 374. Active tamper filtering



As illustrated in *Figure 374*, 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 250* for each possible setting of ATCKSEL, assuming f_{RTCCLK} = 32.768 kHz.

Table 250. 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: 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.

33.4 TAMP low-power modes

Table 251. Effect of low-power modes on TAMP

Mode	Description
Sleep	No effect. TAMP interrupts cause the device to exit the Sleep mode.
Stop	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.
Standby	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 Standby mode.

Table 252. TAMP pins functionality over modes

Pin name	Functional in all low-power modes	Functional in V_{BAT} mode
TAMP_IN[2:1]	Yes(1)	Yes(1)
TAMP_OUT[2:1]	Yes(2)	Yes(2)

1. Only PC13 and PA0 are functional in Standby and V_{BAT} modes.
2. Only PC13 is functional in Standby and V_{BAT} modes.

33.5 TAMP interrupts

Table 253. 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 251: Effect of low-power modes on TAMP* for more details about available features in the low-power modes.

33.6 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).

33.6.1 TAMP control register 1 (TAMP_CR1)

This register can be protected against non-privileged access. Refer to *Section 33.3.6: 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.	ITAMP15E	Res.	ITAMP13E	ITAMP12E	ITAMP11E	Res.	ITAMP9E	ITAMP8E	ITAMP7E	ITAMP6E	ITAMP5E	ITAMP4E	ITAMP3E	ITAMP2E	ITAMP1E
	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.	TAMP2E	TAMP1E
														rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ITAMP15E**: Internal tamper 15 enable
0: Internal tamper 15 disabled.
1: Internal tamper 15 enabled.

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 **ITAMP4E**: Internal tamper 4 enable
0: Internal tamper 4 disabled.
1: Internal tamper 4 enabled.

Bit 18 **ITAMP3E**: Internal tamper 3 enable
0: Internal tamper 3 disabled.
1: Internal tamper 3 enabled.

Bit 17 **ITAMP2E**: Internal tamper 2 enable
0: Internal tamper 2 disabled.
1: Internal tamper 2 enabled.

Bit 16 **ITAMP1E**: Internal tamper 1 enable
0: Internal tamper 1 disabled.
1: Internal tamper 1 enabled.

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 Reserved, must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

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.

33.6.2 TAMP control register 2 (TAMP_CR2)

This register can be protected against non-privileged access. Refer to *Section 33.3.6: 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.	Res.	Res.	TAMP2 TRG	TAMP1 TRG	BK ERASE	BK BLOCK	Res.	Res.	Res.	Res.	TAMP2 MSK	TAMP1 MSK
						rw	rw	w	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.	TAMP2 POM	TAMP1 POM
														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 Reserved, must be kept at reset value.

Bit 26 Reserved, must be kept at reset value.

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 Reserved, must be kept at reset value.

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 Reserved, must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

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 246: TAMP interconnection*.
2. The effects of tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in *Table 246: TAMP interconnection*.

33.6.3 TAMP control register 3 (TAMP_CR3)

This register can be protected against non-privileged access. Refer to *Section 33.3.6: 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.	ITAMP 15POM	Res.	ITAMP 13POM	ITAMP 12POM	ITAMP 11POM	Res.	ITAMP9 POM	ITAMP8 POM	ITAMP7 POM	ITAMP6 POM	ITAMP5 POM	ITAMP4 POM	ITAMP3 POM	ITAMP2 POM	ITAMP1 POM
	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 Reserved, must be kept at reset value.

Bit 14 **ITAMP15POM**: Internal tamper 15 potential mode
0: Internal tamper 15 event detection is in confirmed mode(1).
1: Internal tamper 15 event detection is in potential mode(2).

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 **ITAMP4POM**: Internal tamper 4 potential mode
0: Internal tamper 4 event detection is in confirmed mode(1).
1: Internal tamper 4 event detection is in potential mode(2).

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 **ITAMP2POM**: Internal tamper 2 potential mode
0: Internal tamper 2 event detection is in confirmed mode(1).
1: Internal tamper 2 event detection is in potential mode(2).

Bit 0 **ITAMP1POM**: Internal tamper 1 potential mode
0: Internal tamper 1 event detection is in confirmed mode(1).
1: Internal tamper 1 event detection is in potential mode(2).

1. The effects of internal tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in *Table 246: TAMP interconnection*
2. The effects of internal tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in *Table 246: TAMP interconnection*.

33.6.4 TAMP filter control register (TAMP_FLTCR)

This register can be protected against non-privileged access. Refer to *Section 33.3.6: 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.

33.6.5 TAMP active tamper control register 1 (TAMP_ATCR1)

This register can be protected against non-privileged access. Refer to *Section 33.3.6: 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
Res.	Res.	Res.	Res.	ATOSEL2[1:0]		ATOSEL1[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	TAMP2 AM	TAMP1 AM
				rw	rw	rw	rw							rw	rw
ATOSEL4[1:0]		ATOSEL3[1:0]		ATOSEL2[1:0]		ATOSEL1[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	TAMP2 AM	TAMP1 AM
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 249: 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 15:12 Reserved, must be kept at reset value.

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 Reserved, must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

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.

33.6.6 TAMP active tamper seed register (TAMP_ATSEEDR)

This register can be protected against non-privileged access. Refer to *Section 33.3.6: 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.

33.6.7 TAMP active tamper output register (TAMP_ATOR)

This register can be protected against non-privileged access. Refer to *Section 33.3.6: 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.

33.6.8 TAMP active tamper control register 2 (TAMP_ATCR2)

This register can be protected against non-privileged access. Refer to *Section 33.3.6: 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.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ATOSEL2[2:0]			ATOSEL1[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		rw	rw	rw	rw	rw	rw								

Bits 31:14 Reserved, must be kept at reset value.

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.

33.6.9 TAMP configuration register (TAMP_CFGR)

This register can be globally write-protected, or each bit of this register can be individually write-protected against non-privileged access depending on the TAMP_PRIVCFGR configuration (refer to *Section 33.3.6: 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPW[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.	BKPRW[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 Reserved, must be kept at reset value.

Bits 23:16 **BKPW[7:0]**: Backup registers write protection offset

BKPW value must be from 0 to 32.

Protection zone 2 is defined for backup registers from TAMP_BKPyR (y = BKPRW) to TAMP_BKPzR (z = BKPW-1, with BKPW > BKPRW):
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 = BKPW if BKPWSEC ≥ BKPRWSEC, else t = BKPRWSEC).
If BKPWSEC = 32: there is no protection zone 3.

Refer to *Figure 371: Backup registers protection zones*.

Note: If BKPWPRIV is set, BKPRW[7:0] can be written only in privileged mode.

Bit 15 Reserved, must be kept at reset value.

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

Bits 7:0 **BKPRW[7:0]**: Backup registers read/write protection offset

BKPRW value must be from 0 to 32.

Protection zone 1 is defined for backup registers from TAMP_BKP0R to TAMP_BKPxR (x = BKPRW-1, with BKPRW ≥ 1).
If BKPRW = 0: there is no protection zone 1.

Refer to *Figure 371: Backup registers protection zones*.

Note: If BKPRWPRIV is set, BKPRW[7:0] can be written only in privileged mode.

33.6.10 TAMP privilege configuration register (TAMP_PRIVCFGR)

This register can be written only when the APB access is privileged.

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 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 33.3.6: 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 non-privileged.

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.

33.6.11 TAMP interrupt enable register (TAMP_IER)

This register can be protected against non-privileged access. Refer to *Section 33.3.6: 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.	ITAMP15IE	Res.	ITAMP13IE	ITAMP12IE	ITAMP11IE	Res.	ITAMP9IE	ITAMP8IE	ITAMP7IE	ITAMP6IE	ITAMP5IE	ITAMP4IE	ITAMP3IE	ITAMP2IE	ITAMP1IE
	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.	TAMP2IE	TAMP1IE
														rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ITAMP15IE**: Internal tamper 15 interrupt enable
0: Internal tamper 15 interrupt disabled.
1: Internal tamper 15 interrupt enabled.

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 **ITAMP4IE**: Internal tamper 4 interrupt enable
0: Internal tamper 4 interrupt disabled.
1: Internal tamper 4 interrupt enabled.

Bit 18 **ITAMP3IE**: Internal tamper 3 interrupt enable
0: Internal tamper 3 interrupt disabled.
1: Internal tamper 3 interrupt enabled.

Bit 17 **ITAMP2IE**: Internal tamper 2 interrupt enable
0: Internal tamper 2 interrupt disabled.
1: Internal tamper 2 interrupt enabled.

Bit 16 **ITAMP1IE**: Internal tamper 1 interrupt enable
0: Internal tamper 1 interrupt disabled.
1: Internal tamper 1 interrupt enabled

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 Reserved, must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

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.

33.6.12 TAMP status register (TAMP_SR)

This register can be protected against non-privileged access. Refer to *Section 33.3.6:*

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.	ITAMP15F	Res.	ITAMP13F	ITAMP12F	ITAMP11F	Res.	ITAMP9F	ITAMP8F	ITAMP7F	ITAMP6F	ITAMP5F	ITAMP4F	ITAMP3F	ITAMP2F	ITAMP1F
	rw		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.	TAMP2F	TAMP1F
														r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ITAMP15F**: Internal tamper 15 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 15.

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 **ITAMP4F**: Internal tamper 4 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 4.

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 **ITAMP2F**: Internal tamper 2 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 2.

Bit 16 **ITAMP1F**: Internal tamper 1 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 1.

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 Reserved, must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

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.

33.6.13 TAMP masked interrupt status register (TAMP_MISR)

This register can be protected against non-privileged access. Refer to *Section 33.3.6: 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.	ITAMP15MF	Res.	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF
	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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2MF	TAMP1MF
														r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ITAMP15MF**: internal tamper 15 interrupt masked flag
This flag is set by hardware when the internal tamper 15 interrupt is raised.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **ITAMP13MF**: internal tamper 13 interrupt masked flag
This flag is set by hardware when the internal tamper 13 interrupt is raised.

Bit 27 **ITAMP12MF**: internal tamper 12 interrupt masked flag
This flag is set by hardware when the internal tamper 12 interrupt is raised.

Bit 26 **ITAMP11MF**: internal tamper 11 interrupt masked flag
This flag is set by hardware when the internal tamper 11 interrupt is raised.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9MF**: internal tamper 9 interrupt masked flag
This flag is set by hardware when the internal tamper 9 interrupt is raised.

Bit 23 **ITAMP8MF**: Internal tamper 8 interrupt masked flag
This flag is set by hardware when the internal tamper 8 interrupt is raised.

Bit 22 **ITAMP7MF**: Internal tamper 7 tamper interrupt masked flag
This flag is set by hardware when the internal tamper 7 interrupt is raised.

Bit 21 **ITAMP6MF**: Internal tamper 6 interrupt masked flag
This flag is set by hardware when the internal tamper 6 interrupt is raised.

Bit 20 **ITAMP5MF**: Internal tamper 5 interrupt masked flag
This flag is set by hardware when the internal tamper 5 interrupt is raised.

Bit 19 **ITAMP4MF**: Internal tamper 4 interrupt masked flag
This flag is set by hardware when the internal tamper 4 interrupt is raised.

Bit 18 **ITAMP3MF**: Internal tamper 3 interrupt masked flag
This flag is set by hardware when the internal tamper 3 interrupt is raised.

Bit 17 **ITAMP2MF**: Internal tamper 2 interrupt masked flag
This flag is set by hardware when the internal tamper 2 interrupt is raised.

Bit 16 **ITAMP1MF**:Internal tamper 1 interrupt masked flag
This flag is set by hardware when the internal tamper 1 interrupt is raised.

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 Reserved, must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **TAMP2MF**: TAMP2 interrupt masked flag
This flag is set by hardware when the tamper 2 interrupt is raised.

Bit 0 **TAMP1MF**: TAMP1 interrupt masked flag
This flag is set by hardware when the tamper 1 interrupt is raised.

33.6.14 TAMP status clear register (TAMP_SCR)

This register can be protected against non-privileged access. Refer to *Section 33.3.6: 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.	CITAMP15F	Res.	CITAMP13F	CITAMP12F	CITAMP11F	Res.	CITAMP9F	CITAMP8F	CITAMP7F	CITAMP6F	CITAMP5F	CITAMP4F	CITAMP3F	CITAMP2F	CITAMP1F
	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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP2F	CTAMP1F
														w	w

Bit 31 Reserved, must be kept at reset value.

Bit 30 **CITAMP15F**: Clear ITAMP15 detection flag
Writing 1 in this bit clears the ITAMP15F bit in the TAMP_SR register.

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 **CITAMP4F**: Clear ITAMP4 detection flag
Writing 1 in this bit clears the ITAMP4F bit in the TAMP_SR register.

Bit 18 **CITAMP3F**: Clear ITAMP3 detection flag
Writing 1 in this bit clears the ITAMP3F bit in the TAMP_SR register.

Bit 17 **CITAMP2F**: Clear ITAMP2 detection flag
Writing 1 in this bit clears the ITAMP2F bit in the TAMP_SR register.

Bit 16 **CITAMP1F**: Clear ITAMP1 detection flag
Writing 1 in this bit clears the ITAMP1F bit in the TAMP_SR register.

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 Reserved, must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

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.

33.6.15 TAMP monotonic counter 1 register (TAMP_COUNT1R)

This register can be protected against non-privileged access. Refer to *Section 33.3.6: 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]															
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
COUNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **COUNT[31:0]:**

This register is read-only only and is incremented by one when a write access is done to this register. This register cannot roll-over and is frozen when reaching the maximum value.

33.6.16 TAMP resources protection configuration register (TAMP_RPCFGR)

This register can be protected against non-privileged access. Refer to *Section 33.3.6: 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.	Res.	Res.	Res.	Res.	Res.	RP CFG0
															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 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **RPCFG0**: Configurable resource 0 protection(1)

0: Resource 0 is not included in the device secrets protected by TAMP peripheral

1: Resource 0 is included in the device secrets protected by TAMP peripheral

1. Refer to tamp_confirmed_rpcfg0 and tamp_potential_rpcfg0 signals in *Table 244: TAMP input/output pins* and *Table 246: TAMP interconnection*.

33.6.17 TAMP backup x register (TAMP_BKPxR)

This register can be protected against non-privileged access. Refer to *Section 33.3.6: 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 product state is opened.

33.6.18 TAMP register map

Table 254. 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.	ITAMP15E	Res.	ITAMP13E	ITAMP12E	ITAMP11E	Res.	ITAMP9E	ITAMP8E	ITAMP7E	ITAMP6E	ITAMP5E	ITAMP4E	ITAMP3E	ITAMP2E	ITAMP1E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2E	TAMP1E
	Reset value		0		0	0	0		0	0	0	0	0	0	0	0	0															0	0
0x04	TAMP_CR2	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2TRG	TAMP1TRG	BKERASE	BKBLOCK	Res.	Res.	Res.	Res.	TAMP2MSK	TAMP1MSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2POM	TAMP1POM
	Reset value							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.	ITAMP15POM	Res.	ITAMP13POM	ITAMP12POM	ITAMP11POM	Res.	ITAMP9POM	ITAMP8POM	ITAMP7POM	ITAMP6POM	ITAMP5POM	ITAMP4POM	ITAMP3POM	ITAMP2POM	ITAMP1POM
	Reset value																		0		0	0	0		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.	ATCK SEL[3:0]				Res.	Res.	Res.	Res.	ATO SEL2 [1:0]		ATO SEL1 [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	TAMP2AM	TAMP1AM
	Reset value	0	0				0	0	0					0	1	1	1					0	0	0	0							0	0
0x10	TAMP_ATCR1	FLTEN	ATOSHARE	Res.	Res.	Res.	AT PER[2:0]			Res.	Res.	Res.	Res.	ATCK SEL[3:0]				ATO SEL4 [1:0]		ATO SEL3 [1:0]		ATO SEL2 [1:0]		ATO SEL1 [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	TAMP2AM	TAMP1AM
	Reset value	0	0				0	0	0					0	1	1	1	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.	Res.	Res.	Res.	Res.	Res.	Res.	ATO SEL2 [2:0]			ATO SEL1 [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																			0	0	0	0	0	0								
0x20	TAMP_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRW[7:0]							
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0

Table 254. 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
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															
0x2C	TAMP_IER	Res.	ITAMP15IE	Res.	ITAMP13IE	ITAMP12IE	ITAMP11IE	Res.	ITAMP9IE	ITAMP8IE	ITAMP7IF	ITAMP6IE	ITAMP5IE	ITAMP4IE	ITAMP3IE	ITAMP2IE	ITAMP1IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2IE	TAMP1IE
	Reset value		0		0	0	0		0	0	0	0	0	0	0	0	0														0	0
0x30	TAMP_SR	Res.	ITAMP15F	Res.	ITAMP13F	ITAMP12F	ITAMP11F	Res.	ITAMP9F	ITAMP8F	ITAMP7F	ITAMP6F	ITAMP5F	ITAMP4F	ITAMP3F	ITAMP2F	ITAMP1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2F	TAMP1F
	Reset value		0		0	0	0		0	0	0	0	0	0	0	0	0														0	0
0x34	TAMP_MISR	Res.	ITAMP15MF	Res.	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2MF	TAMP1MF
	Reset value		0		0	0	0		0	0	0	0	0	0	0	0	0														0	0
0x3C	TAMP_SCR	Res.	CITAMP15F	Res.	CITAMP13F	CITAMP12F	CITAMP11F	Res.	CITAMP9F	CITAMP8F	CITAMP7F	CITAMP6F	CITAMP5F	CITAMP4F	CITAMP3F	CITAMP2F	CITAMP1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP2F	CTAMP1F
	Reset value		0		0	0	0		0	0	0	0	0	0	0	0	0														0	0
0x40	TAMP_ COUNTR	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
0x50	TAMP_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN4_RMP	IN3_RMP	IN2_RMP	Res.	Res.	Res.	Res.	OUT5_RMP	OUT3_RMP		Res.
	Reset value																						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.	Res.	Res.	Res.	Res.	Res.	RPCFG0
	Reset value																																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 2.2* for the register boundary addresses.

34 Inter-integrated circuit interface (I2C)

34.1 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.

34.2 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 on address match

For information on I2C instantiation, refer to *Section 34.3: I2C implementation*.

34.3 I2C implementation

This section provides an implementation overview with respect to the I2C instantiation.

Table 255. I2C implementation

I2C features(1)	I2C1	I2C2
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
Wake-up from Stop mode only (no autonomous mode)	X	X
SMBus/PMBus	X	X

1. X = supported.

34.4 I2C functional description

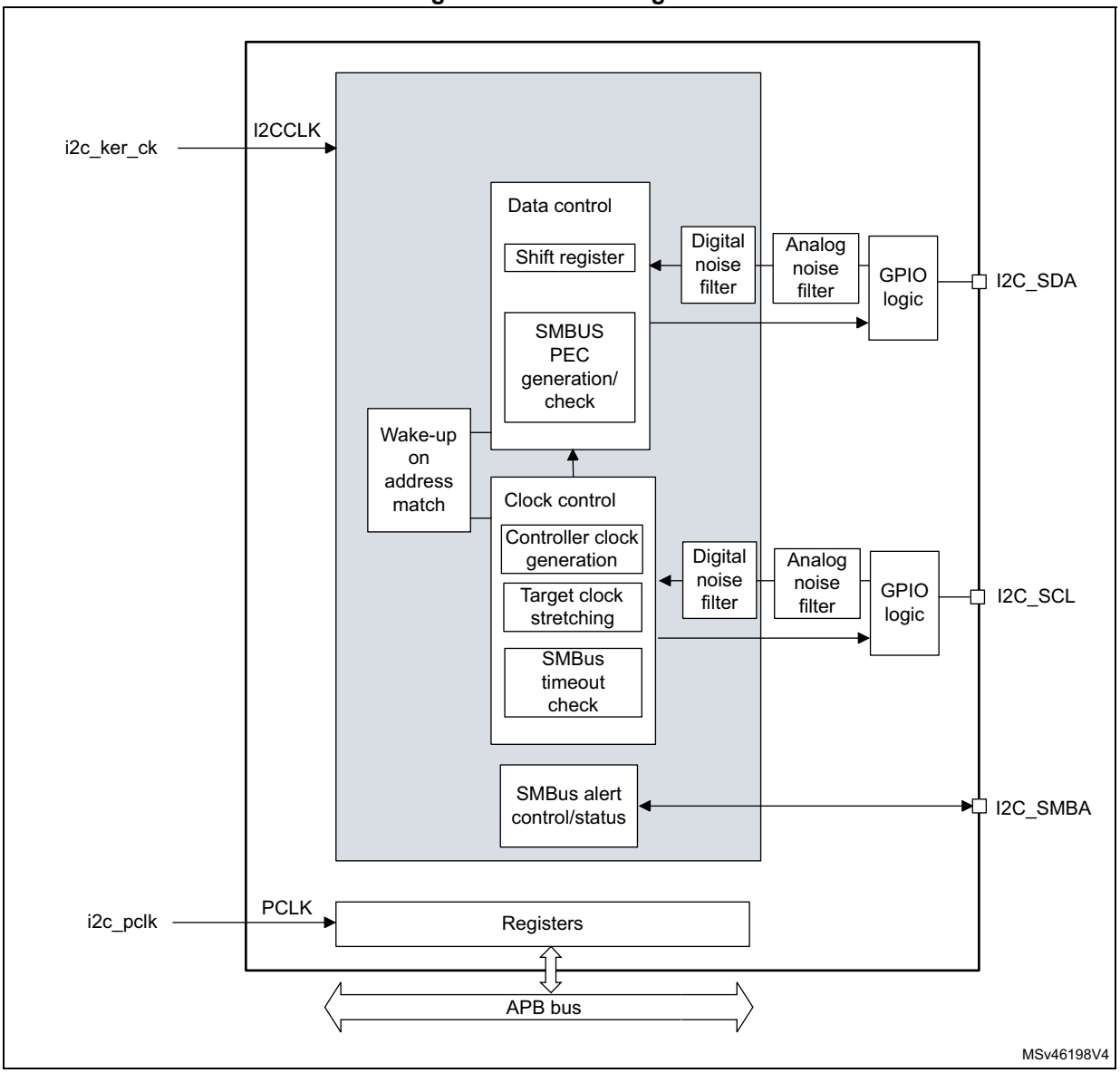
In addition to receiving and transmitting data, the peripheral converts them from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The peripheral is connected to the I²C-bus through a data pin (SDA) and a clock pin (SCL). It supports Standard-mode (up to 100 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.

34.4.1 I2C block diagram

Figure 375. Block diagram



34.4.2 I2C pins and internal signals

Table 256. 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 257. 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_it	Output	I2C interrupts, refer to *Table 270* 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)

34.4.3 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}.

34.4.4 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

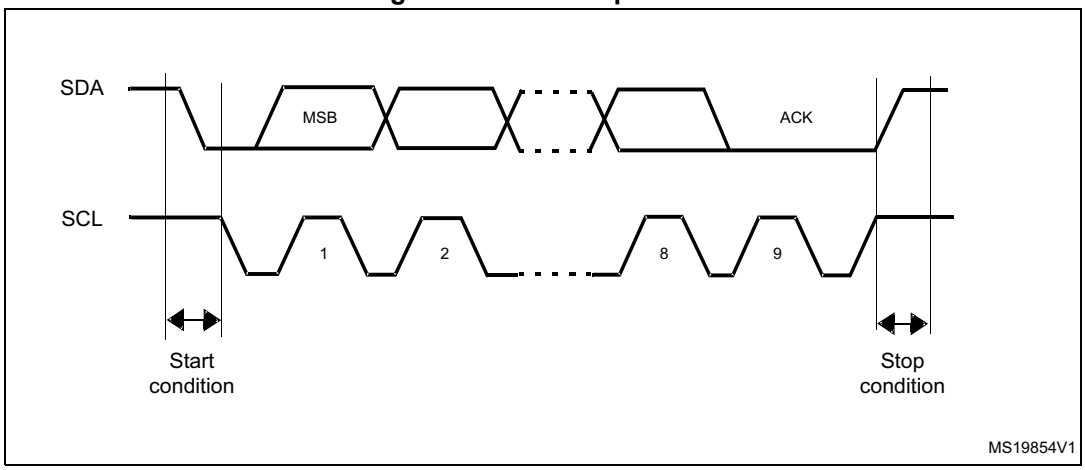
In controller mode, the I2C peripheral initiates a data transfer and generates the clock signal. Serial data transfers always begin with a START condition and end with a STOP condition. Both START and STOP conditions are generated in controller mode by software.

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 376. I²C-bus protocol



The acknowledge can be enabled or disabled by software. The own addresses of the I2C peripheral can be selected by software.

34.4.5 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 34.4.6* 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 258. 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	Wake-up from Stop mode on address match not supported when the digital filter is enabled

1. Maximum duration of spikes that the filter can suppress

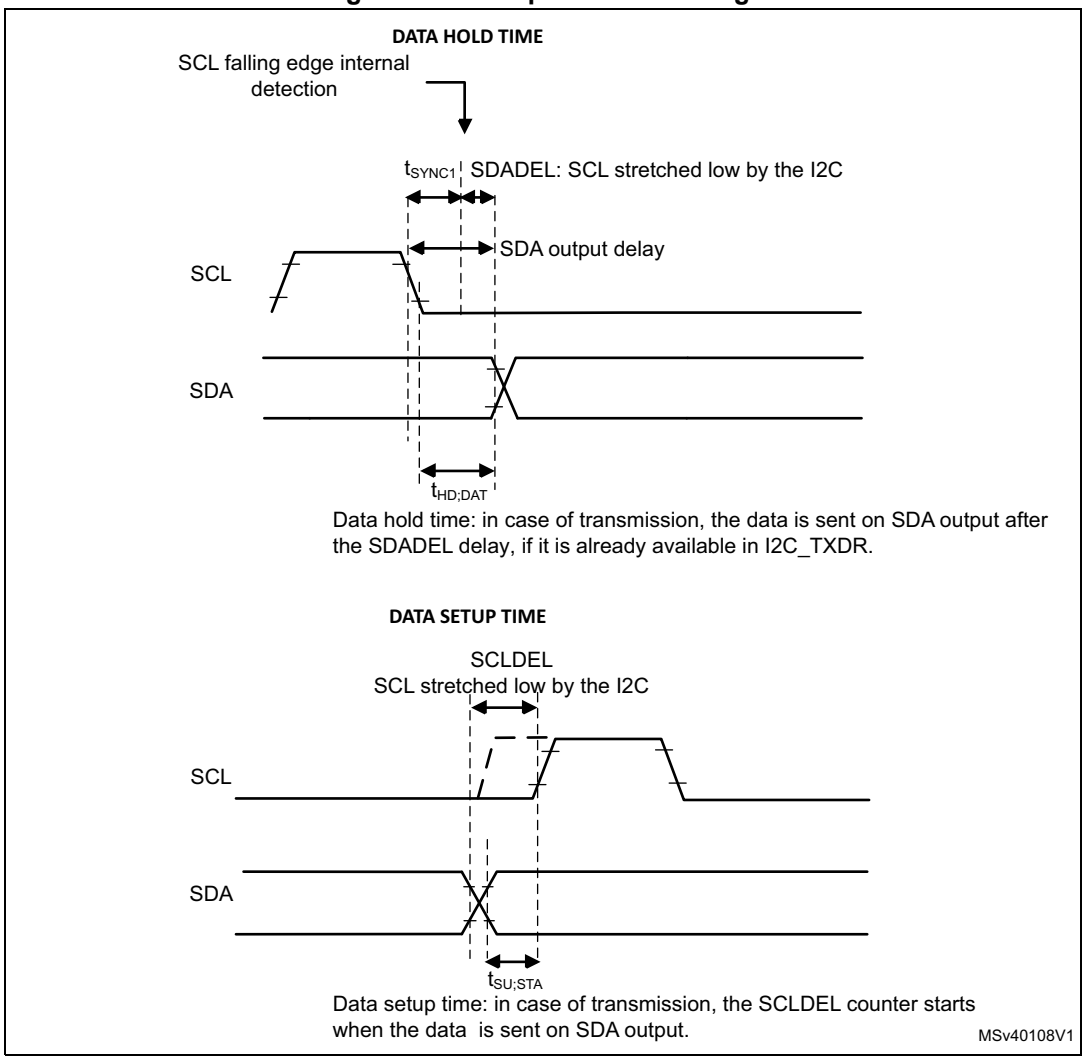
Caution: The filter configuration cannot be changed when the I2C peripheral is enabled.

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 377. 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 259. 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.

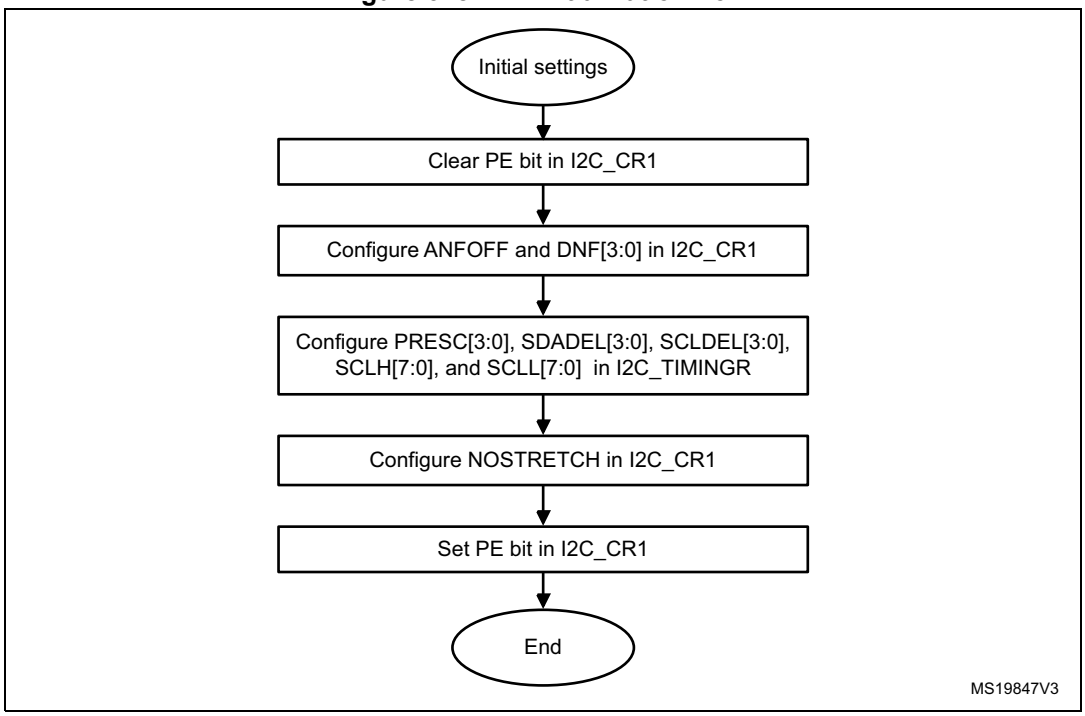
When the SCL falling edge is internally detected, the I2C peripheral releasing the SCL output after the delay t_{SCLL} = (SCLL + 1) x t_{PRESC}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}. The t_{SCLL} delay impacts the SCL low time t_{LOW}.

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 378. I2C initialization flow



34.4.6 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

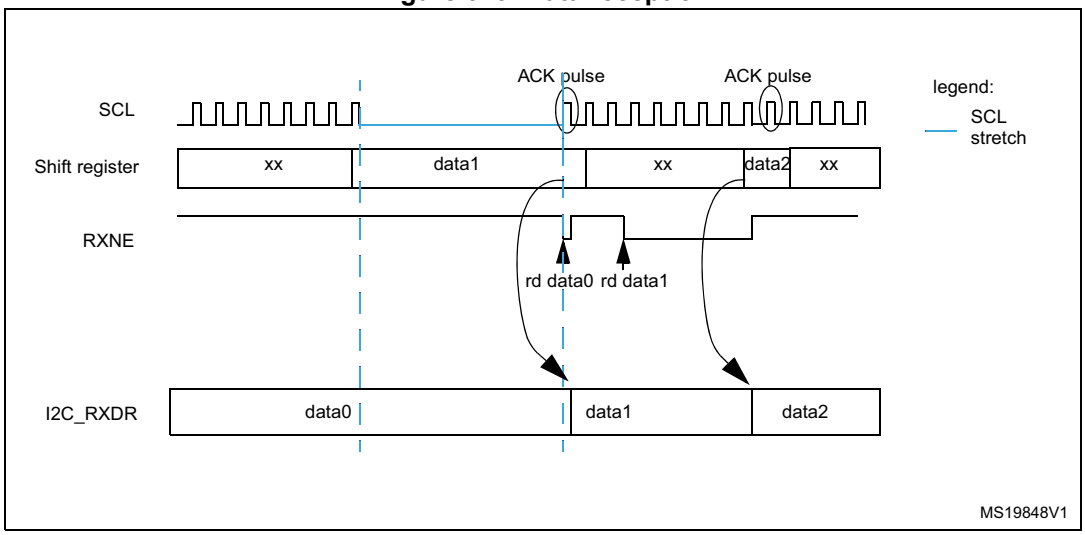
34.4.7 I2C data transfer

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).

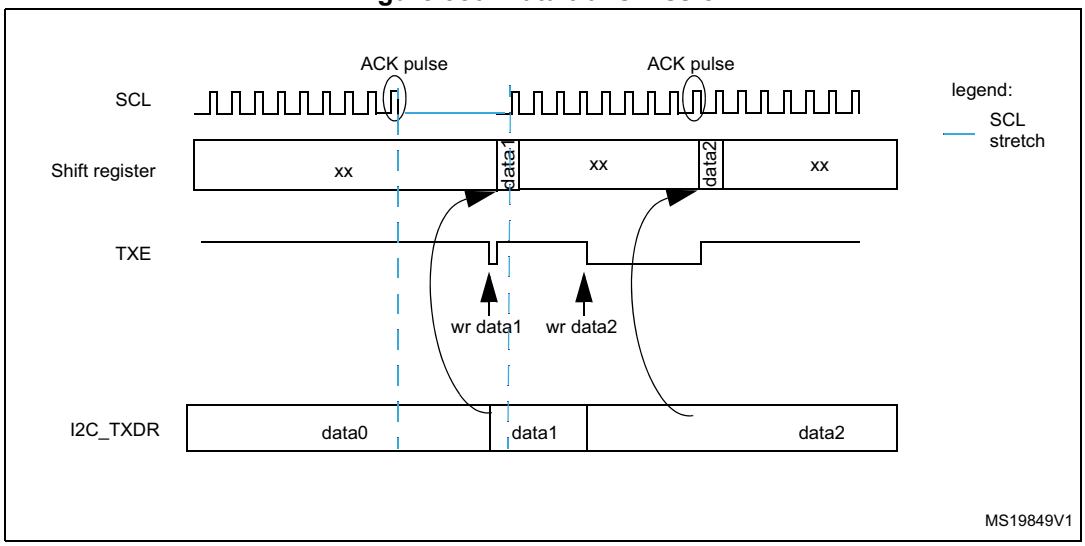
Figure 379. Data reception



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 380. 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 260. 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

34.4.8 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.

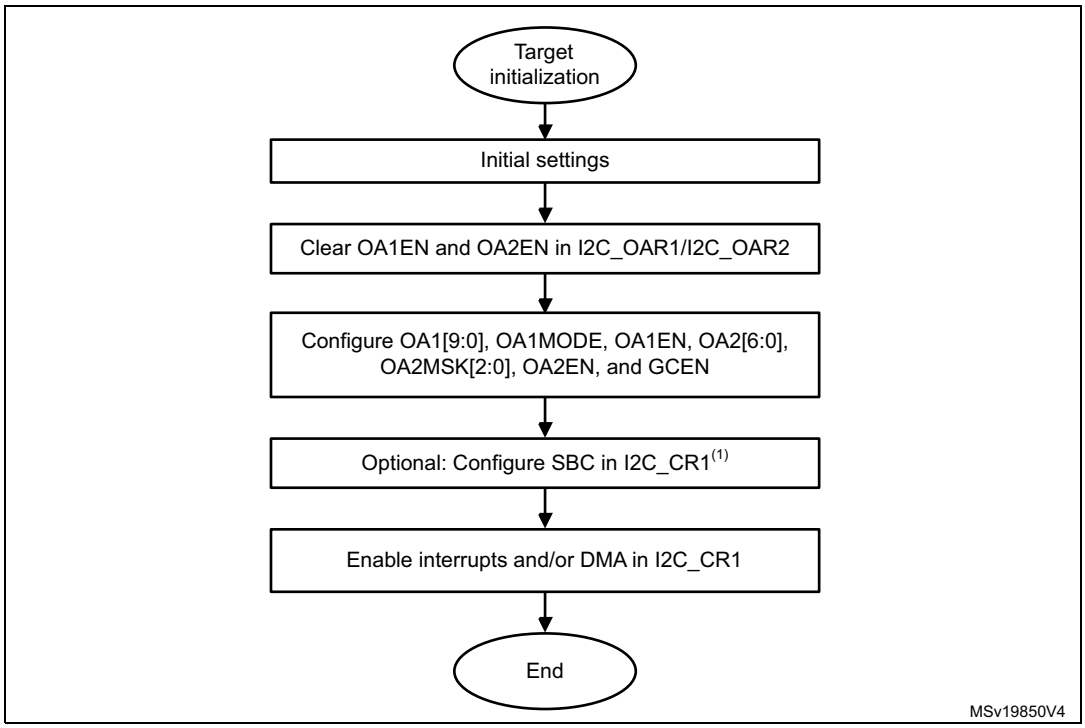
NBYTES[7:0] can be loaded with a value greater than 0x1. Receiving then continues until the corresponding number of bytes are received.

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 381. 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 382. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0

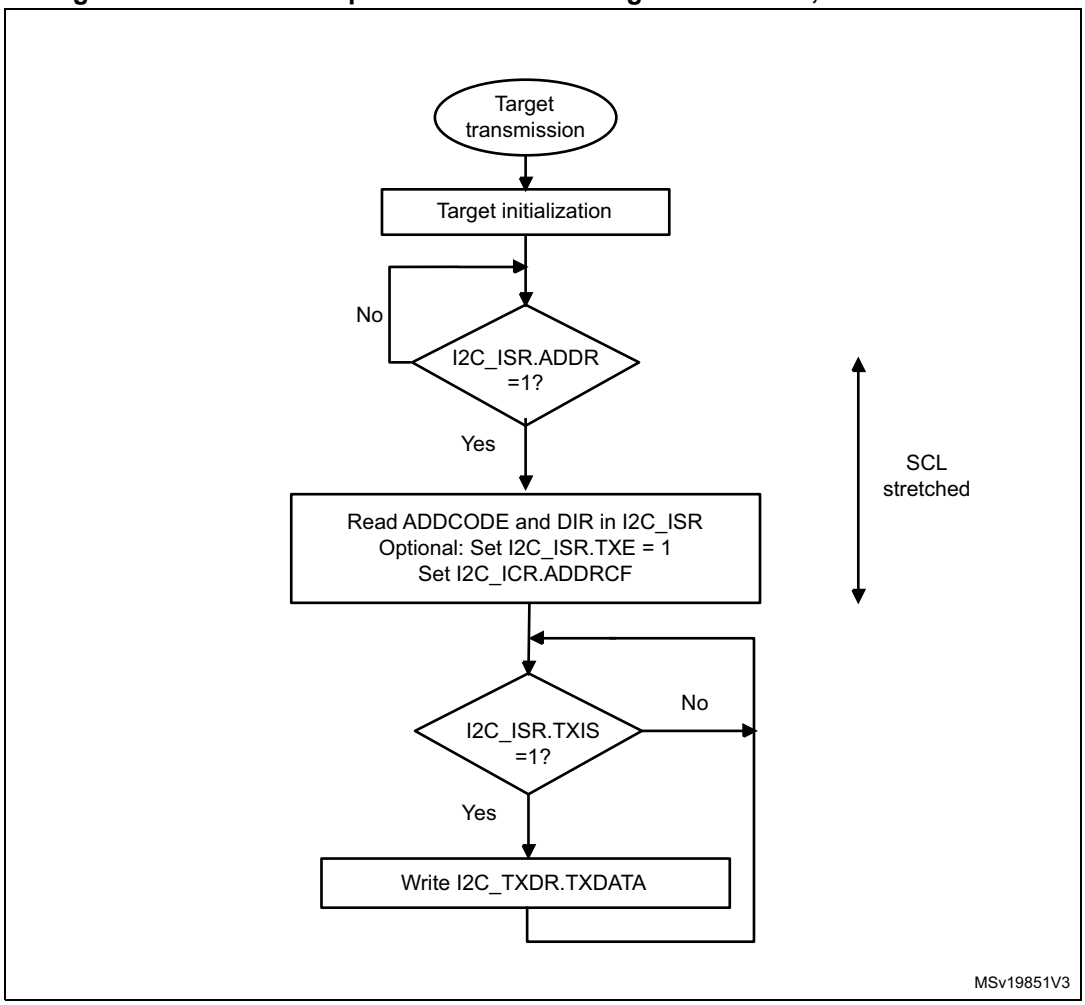


Figure 383. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1

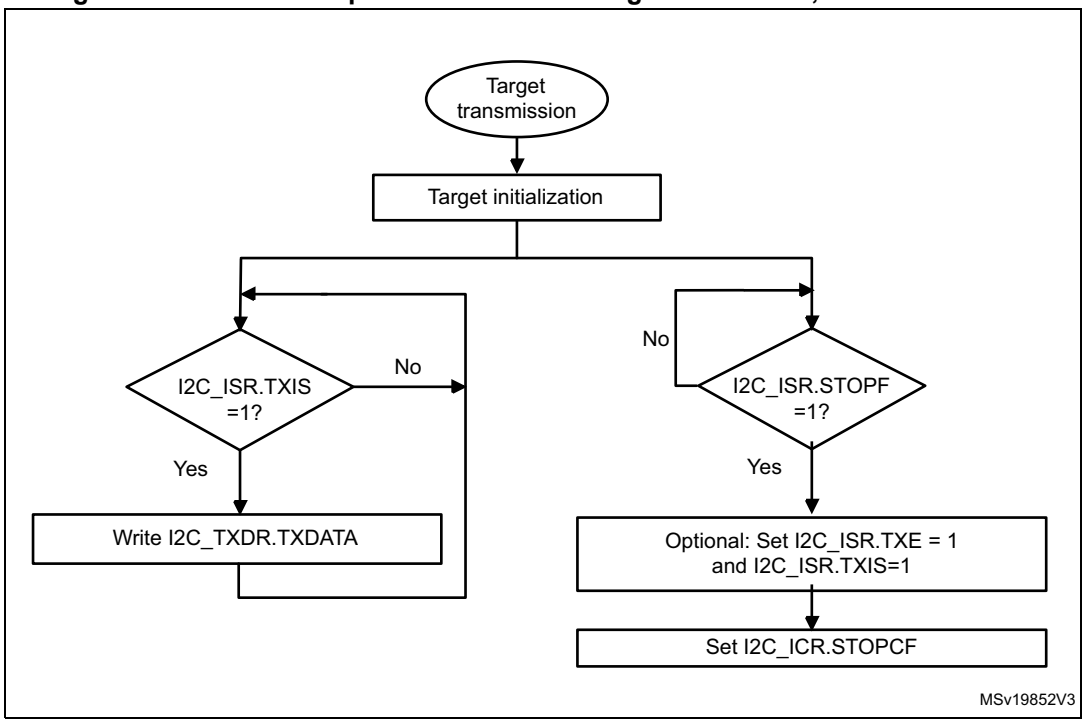
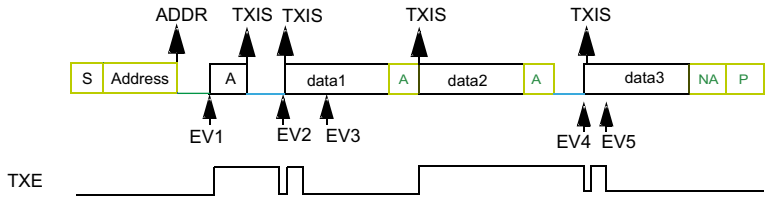


Figure 384. Transfer bus diagrams for I2C target transmitter (mandatory events only)

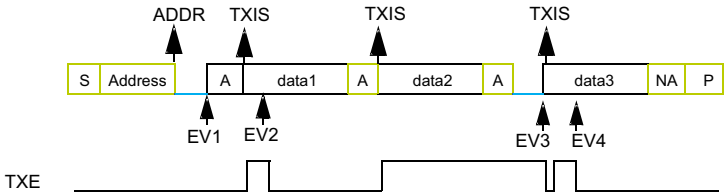
Example I2C target transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:



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)

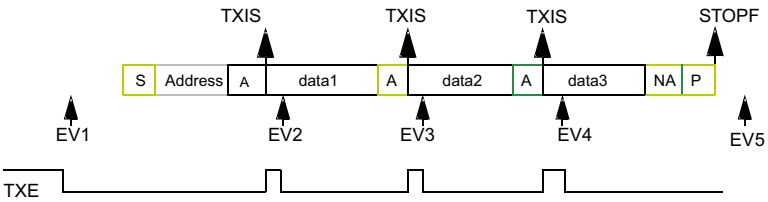
Example I2C target transmitter 3 bytes without 1st data flush, NOSTRETCH=0:



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 385. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0

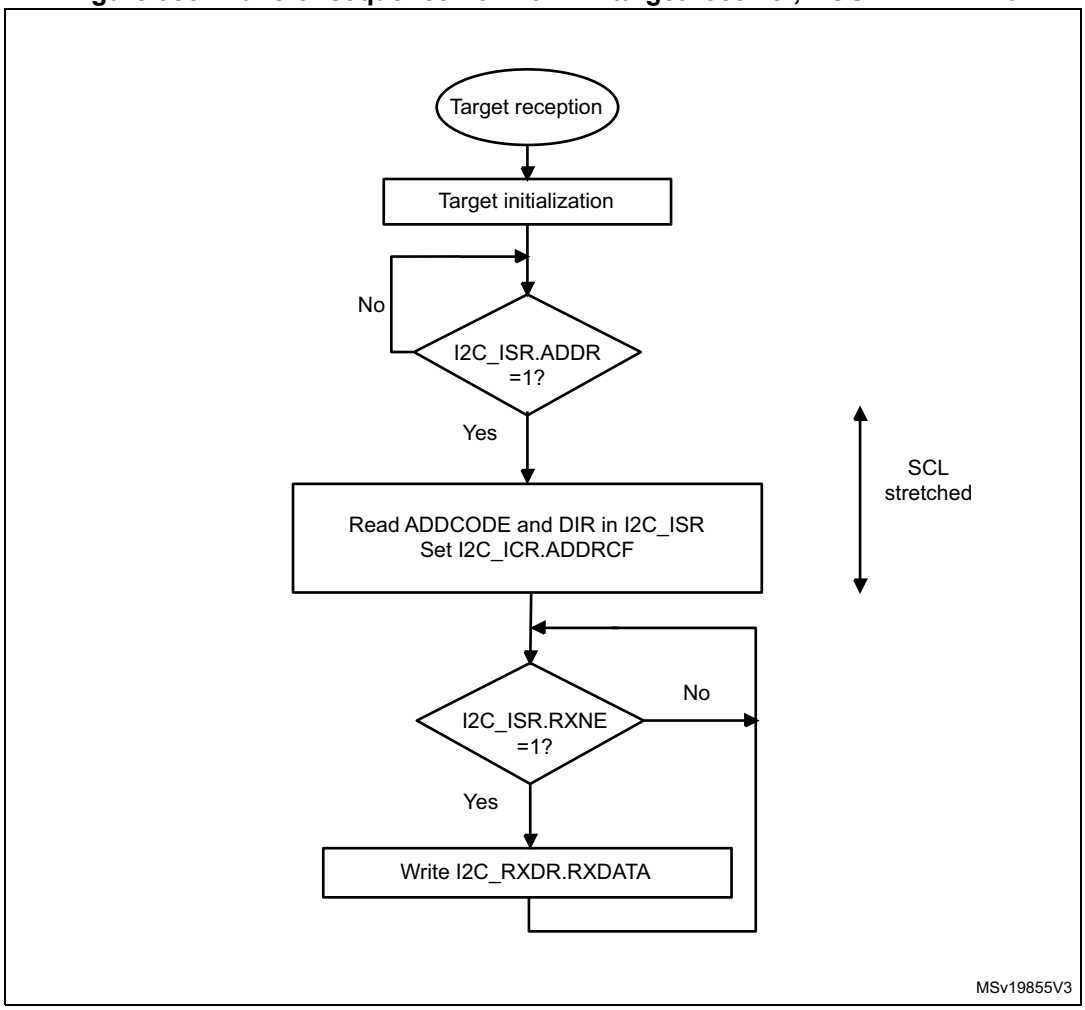


Figure 386. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1

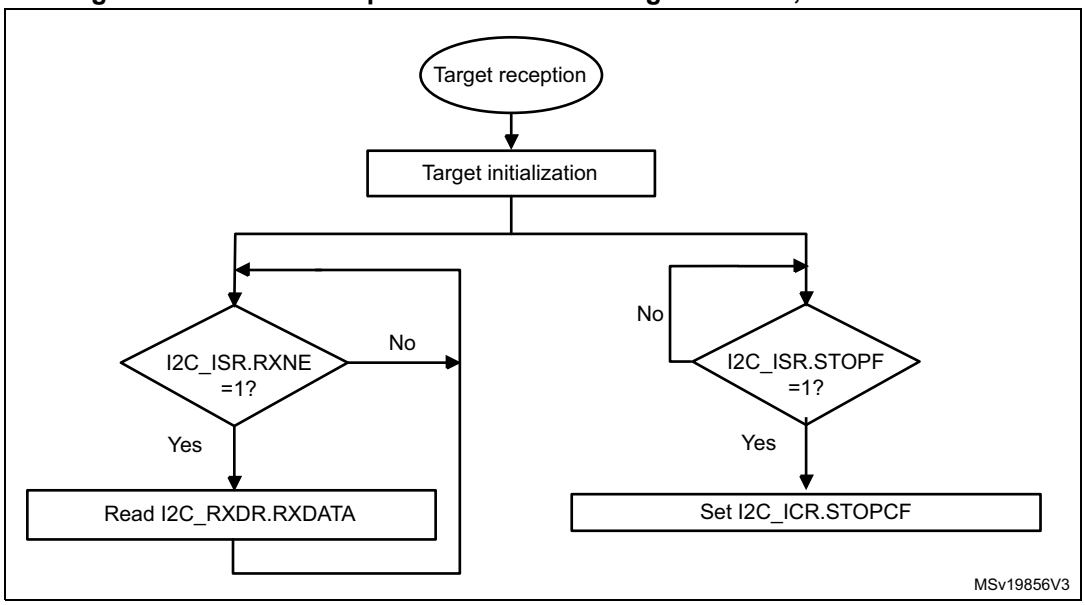
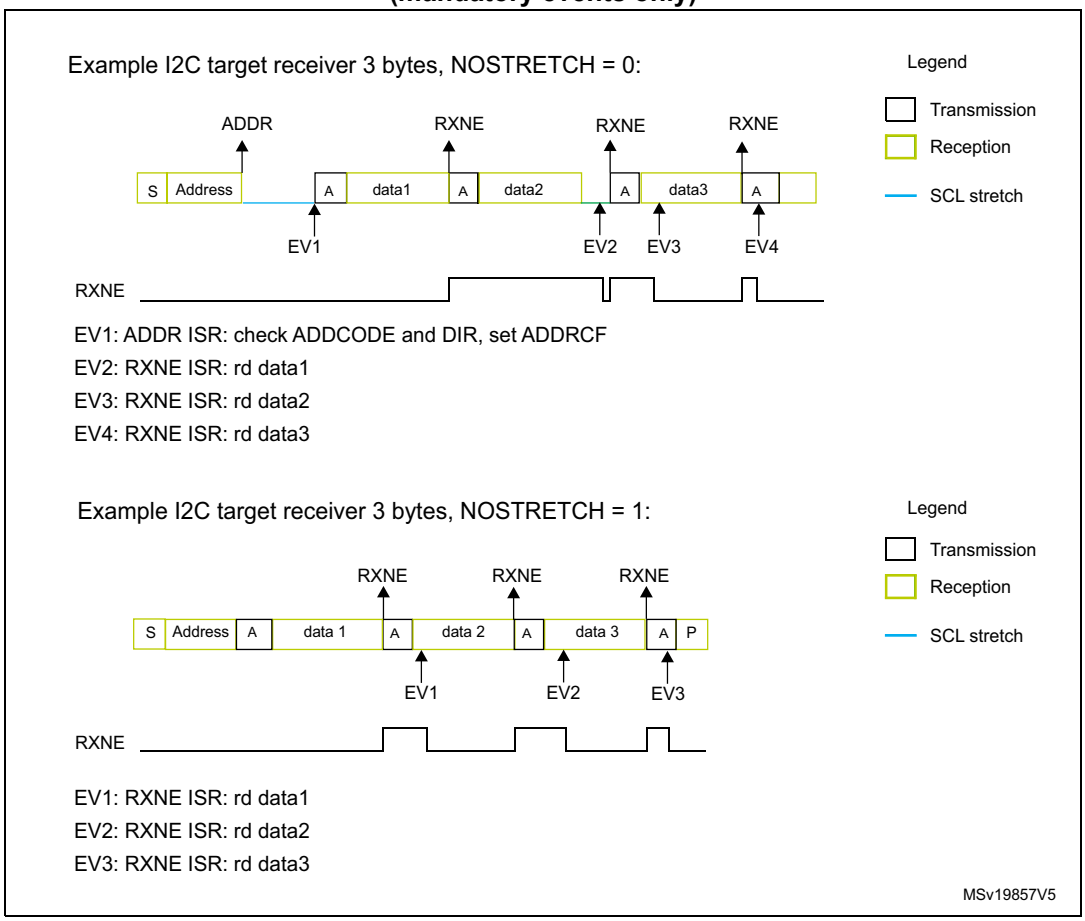


Figure 387. Transfer bus diagrams for I2C target receiver (mandatory events only)



34.4.9 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:

$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+ 1) + (SCLL+ 1)] \times (PRESC+ 1) \times t_{I2CCLK}\}$

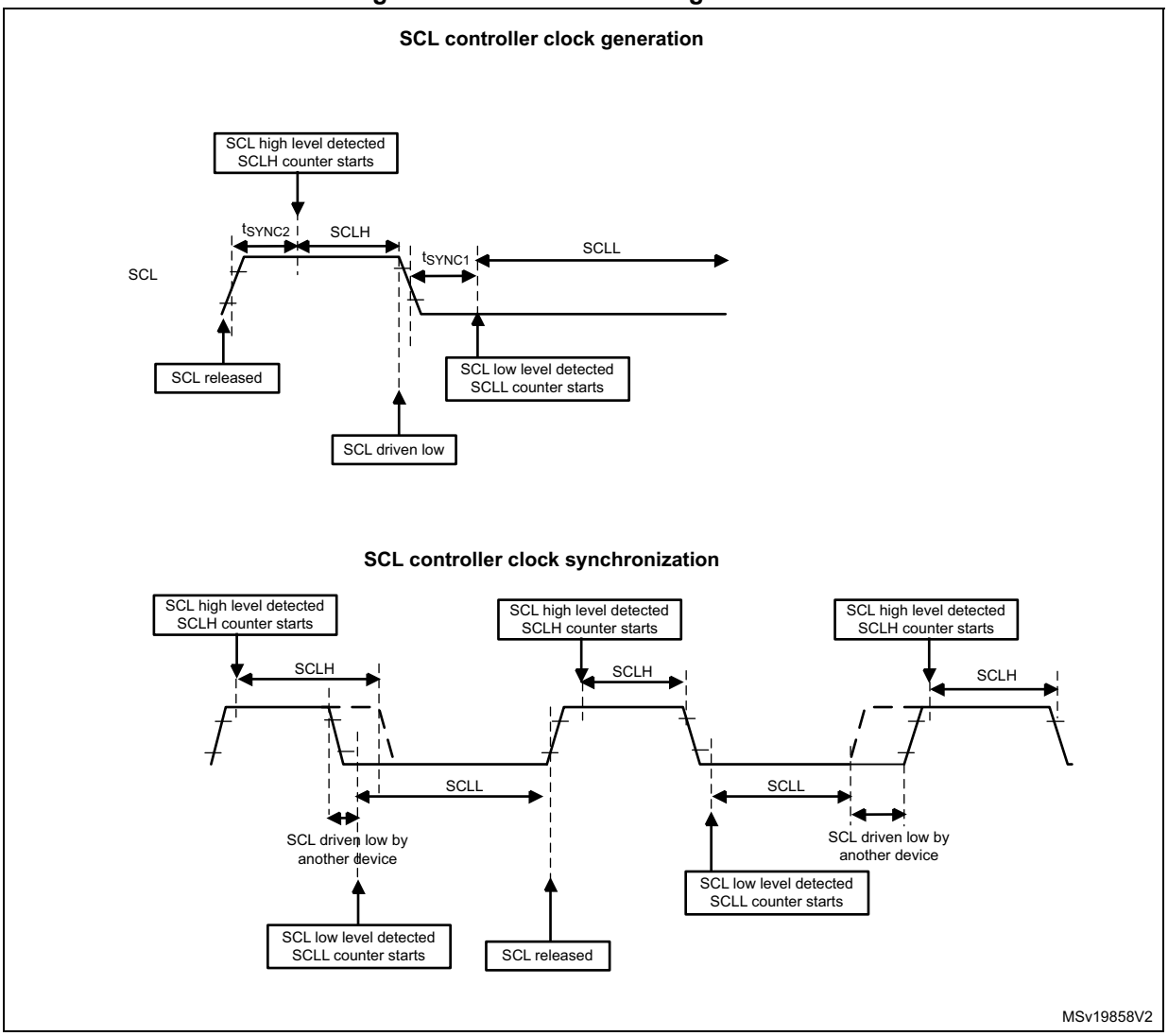
The duration of t_{SYNC1} depends upon:

- 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 388. 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 261. 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 34.4.10* 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.

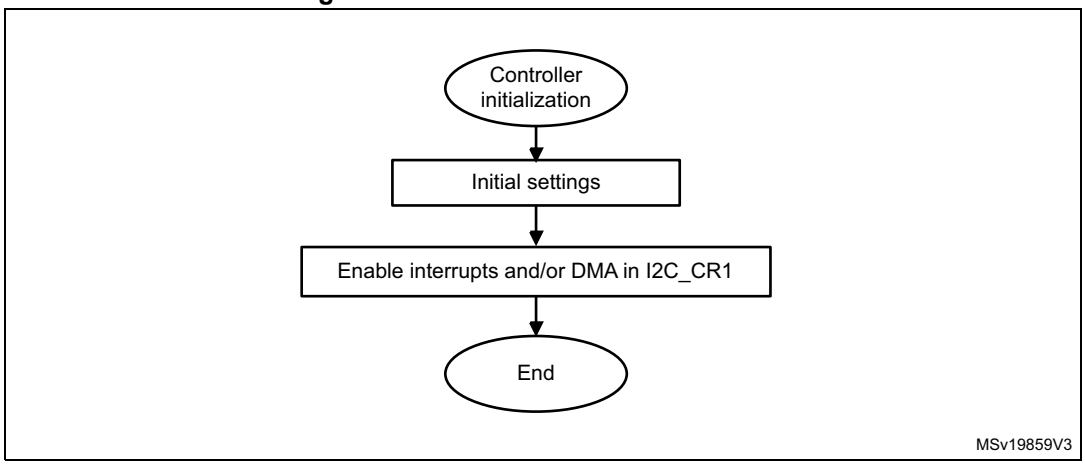
In case of an arbitration loss, the controller automatically switches back to target mode and can acknowledge its own address if it is addressed as a target.

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 389. Controller initialization flow

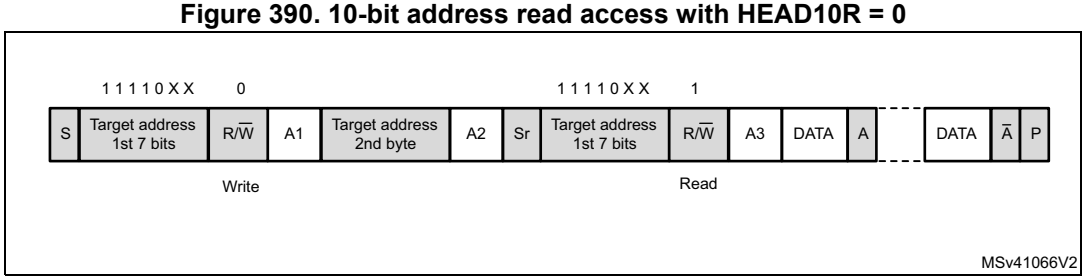


Initialization of a controller receiver addressing a 10-bit address target

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.

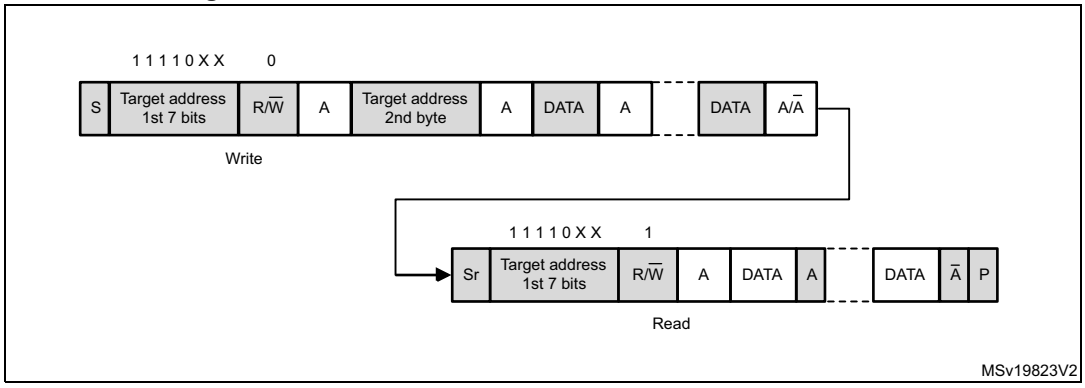
Figure 390. 10-bit address read access with HEAD10R = 0



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 391. 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 392. Transfer sequence flow for I2C controller transmitter, N ≤ 255 bytes

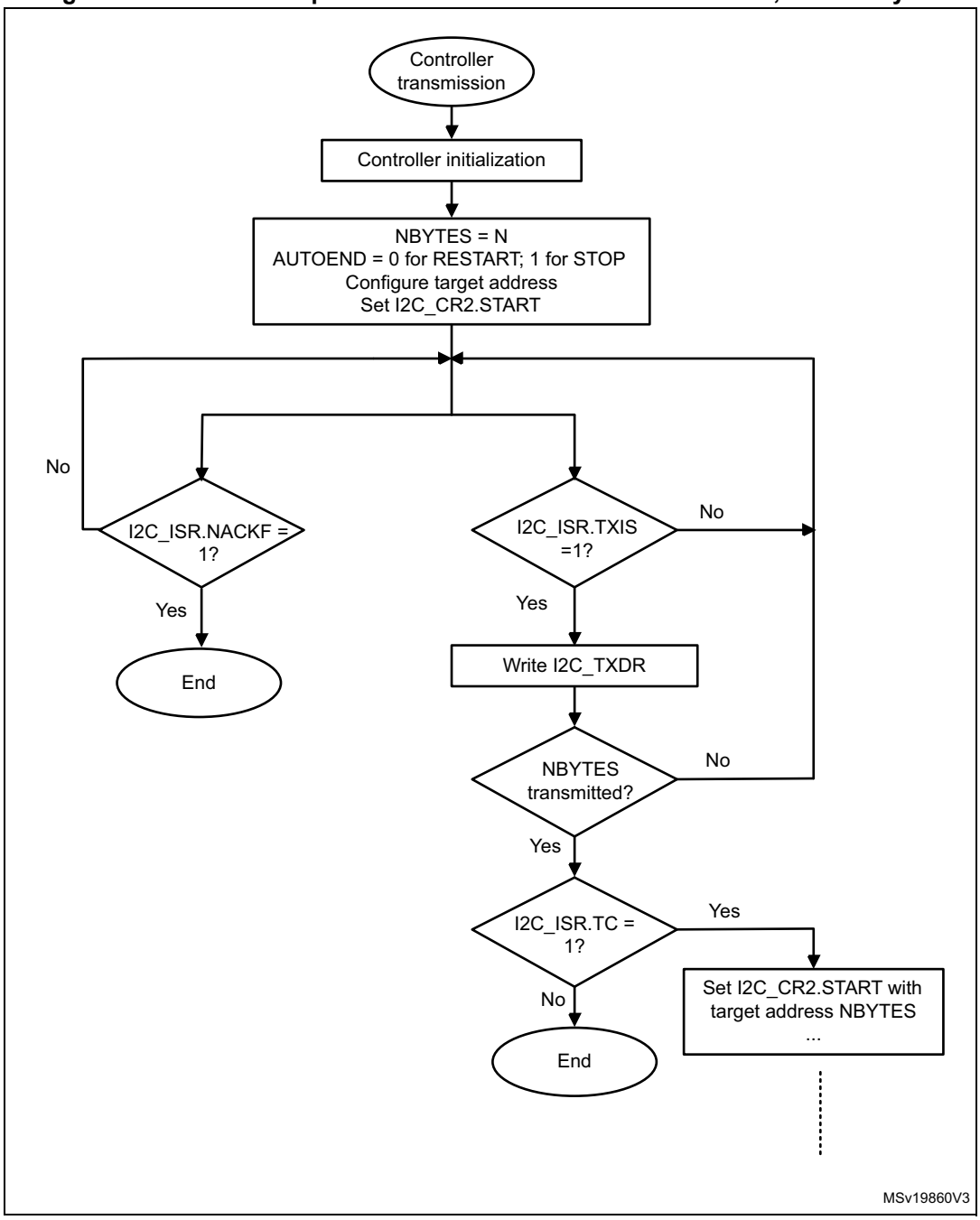


Figure 393. Transfer sequence flow for I2C controller transmitter, N > 255 bytes

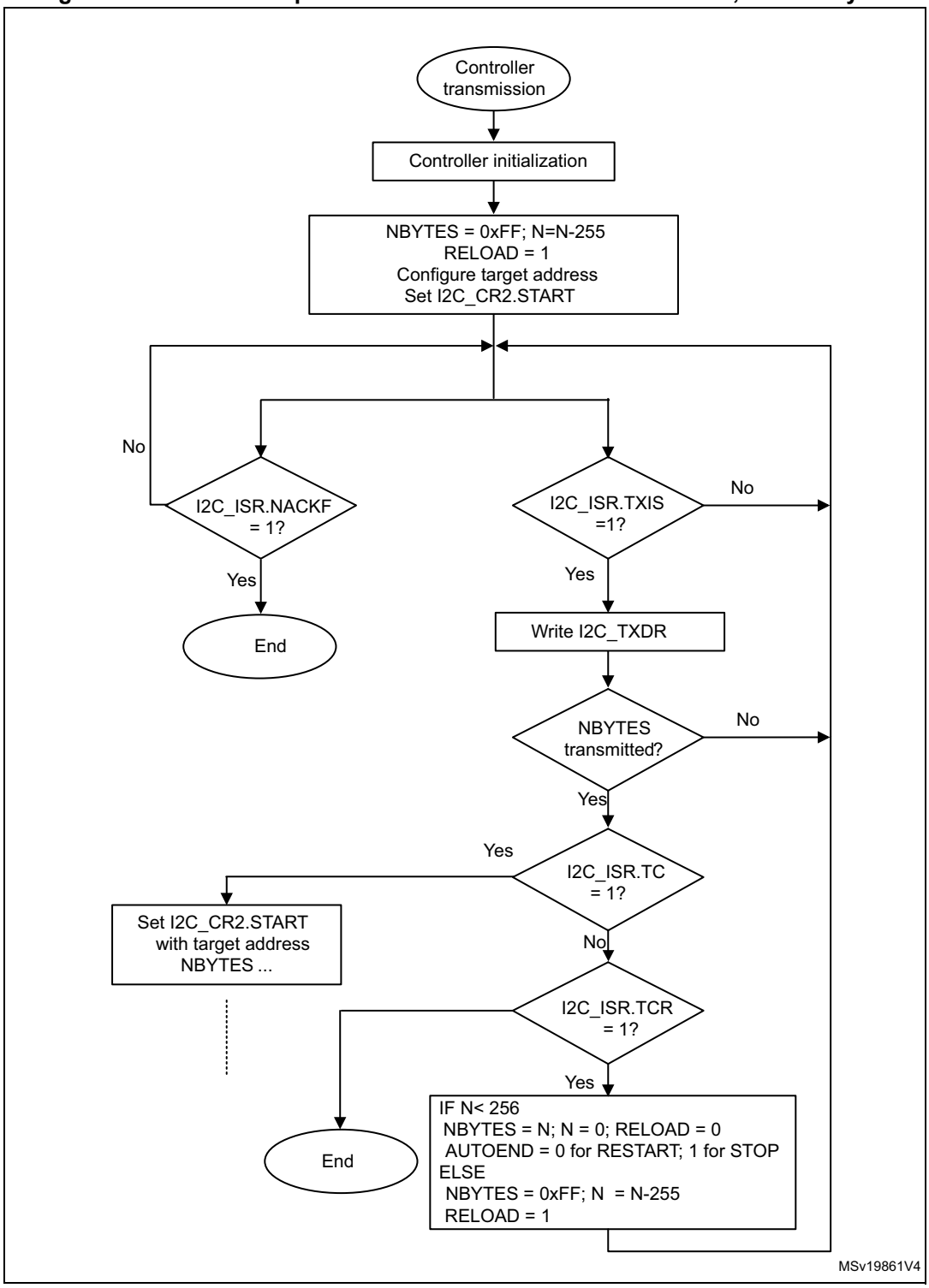
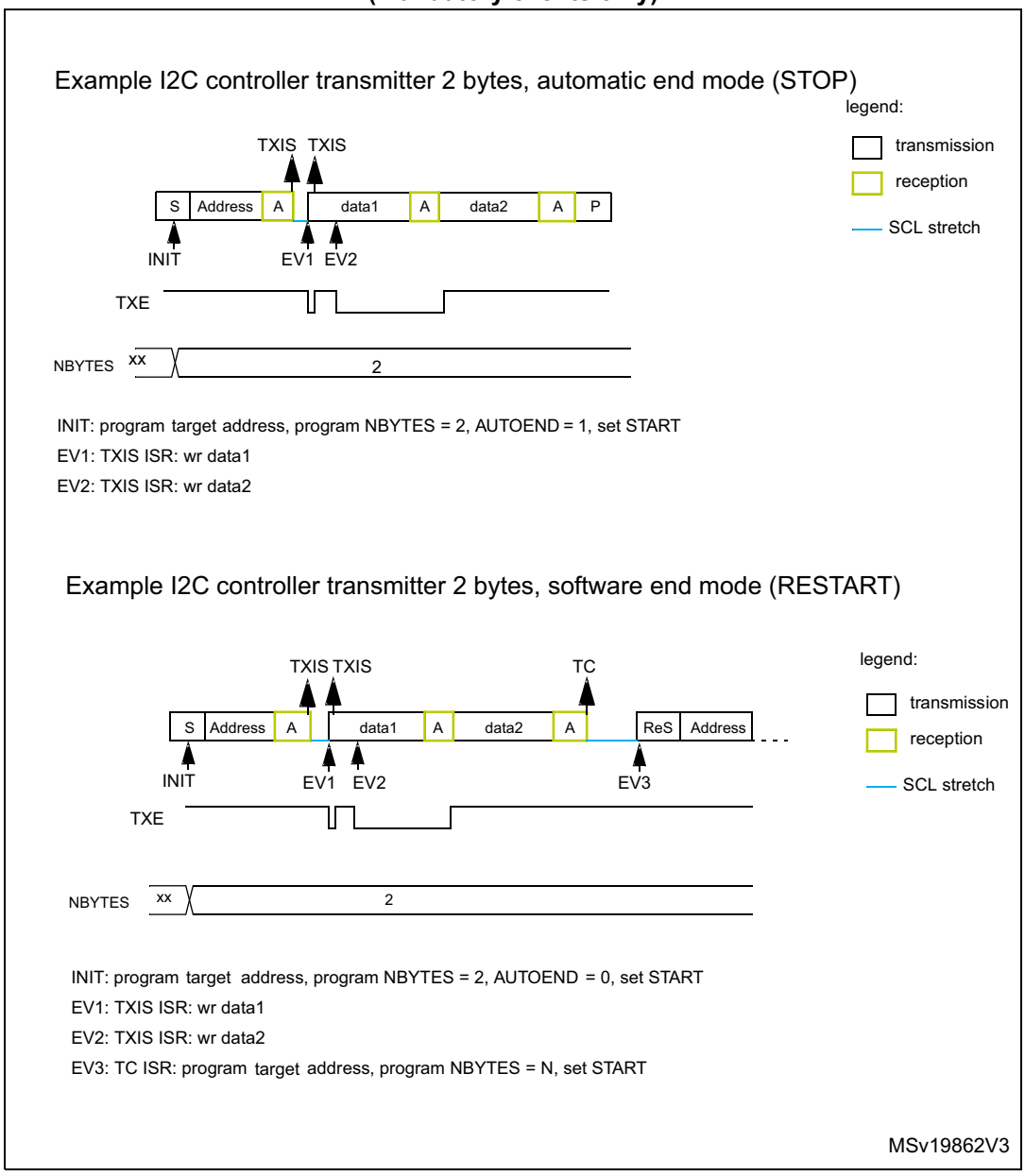


Figure 394. Transfer bus diagrams for I2C controller transmitter (mandatory events only)



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 395. Transfer sequence flow for I2C controller receiver, N ≤ 255 bytes

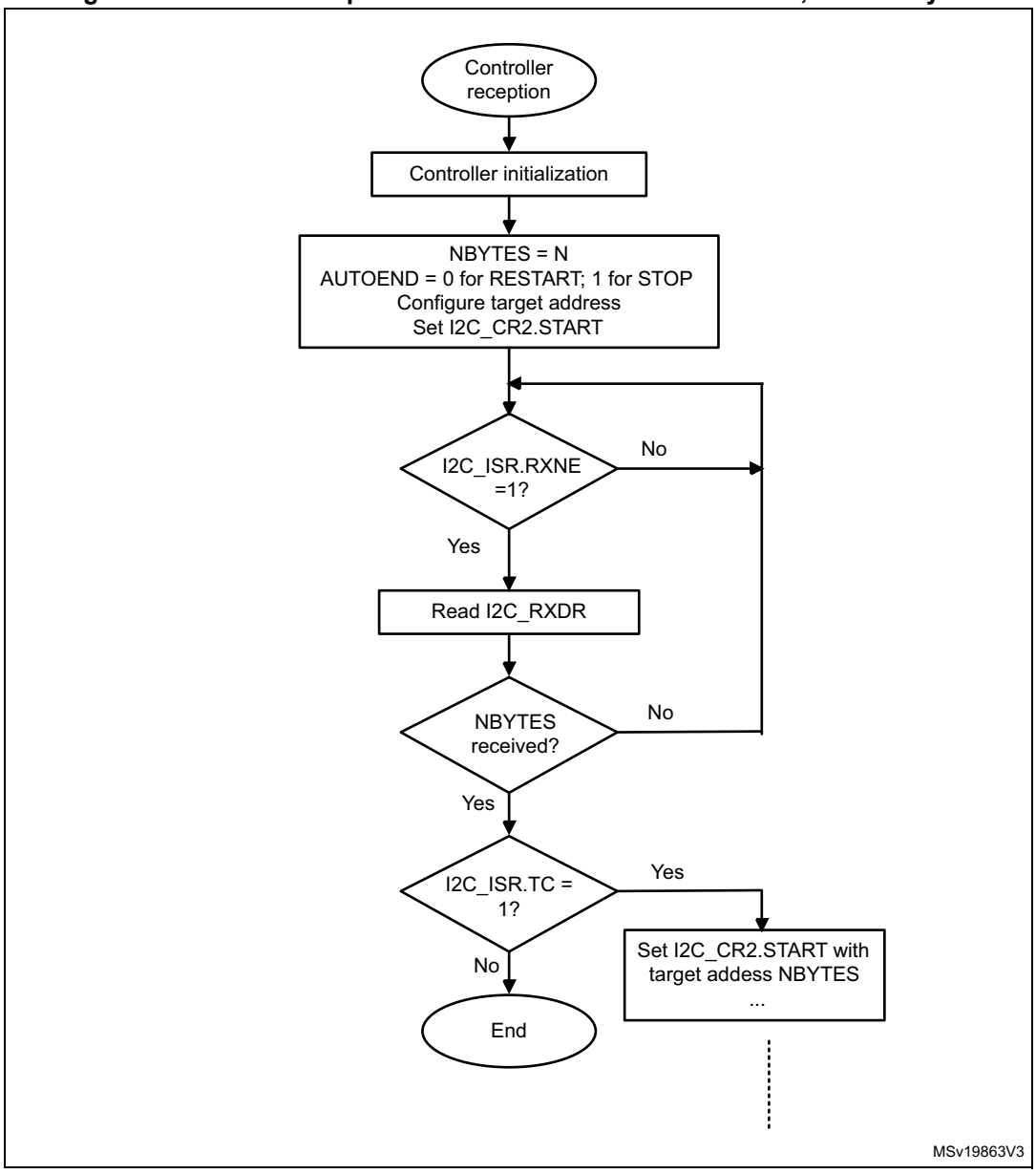


Figure 396. Transfer sequence flow for I2C controller receiver, N > 255 bytes

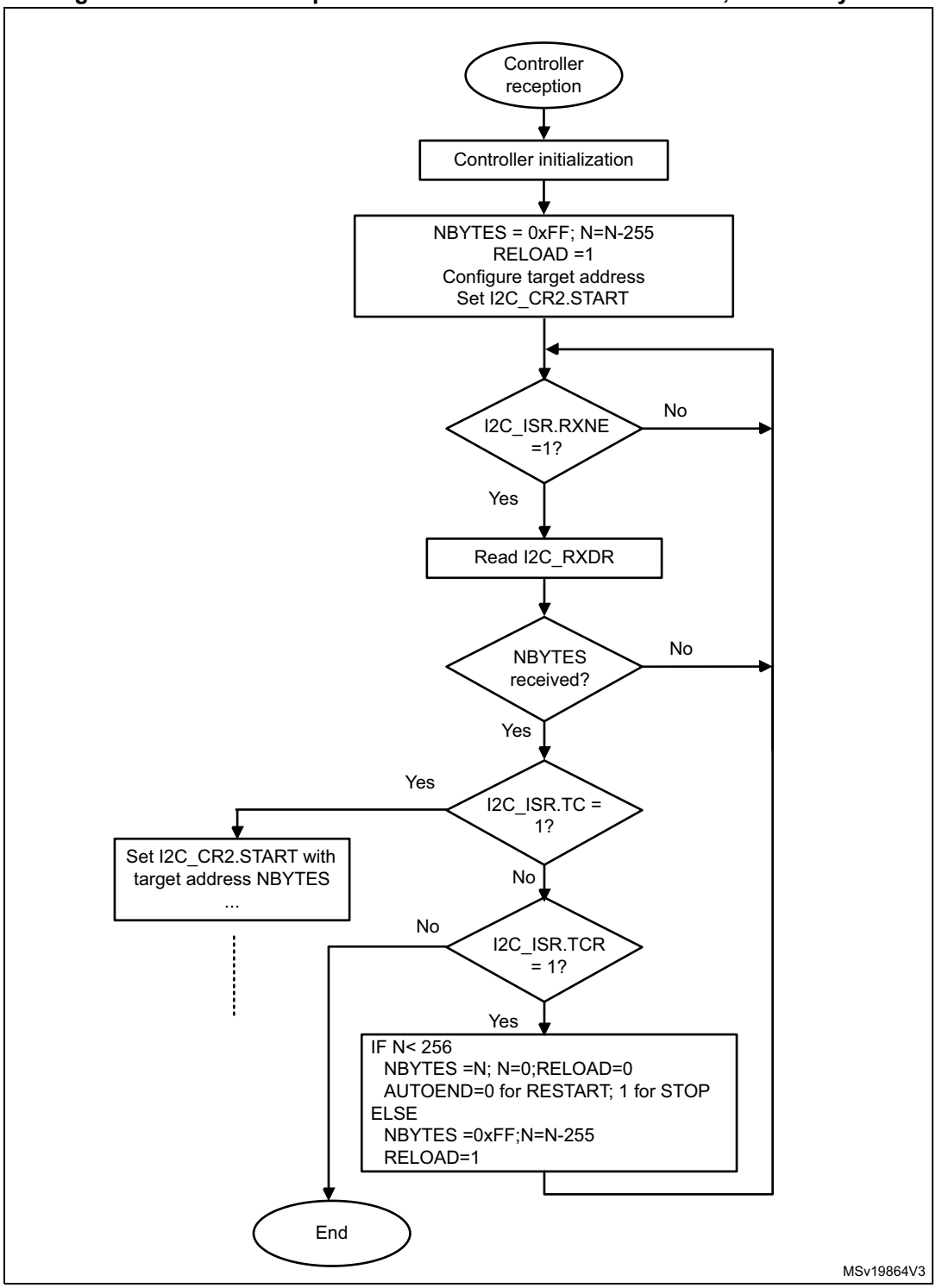
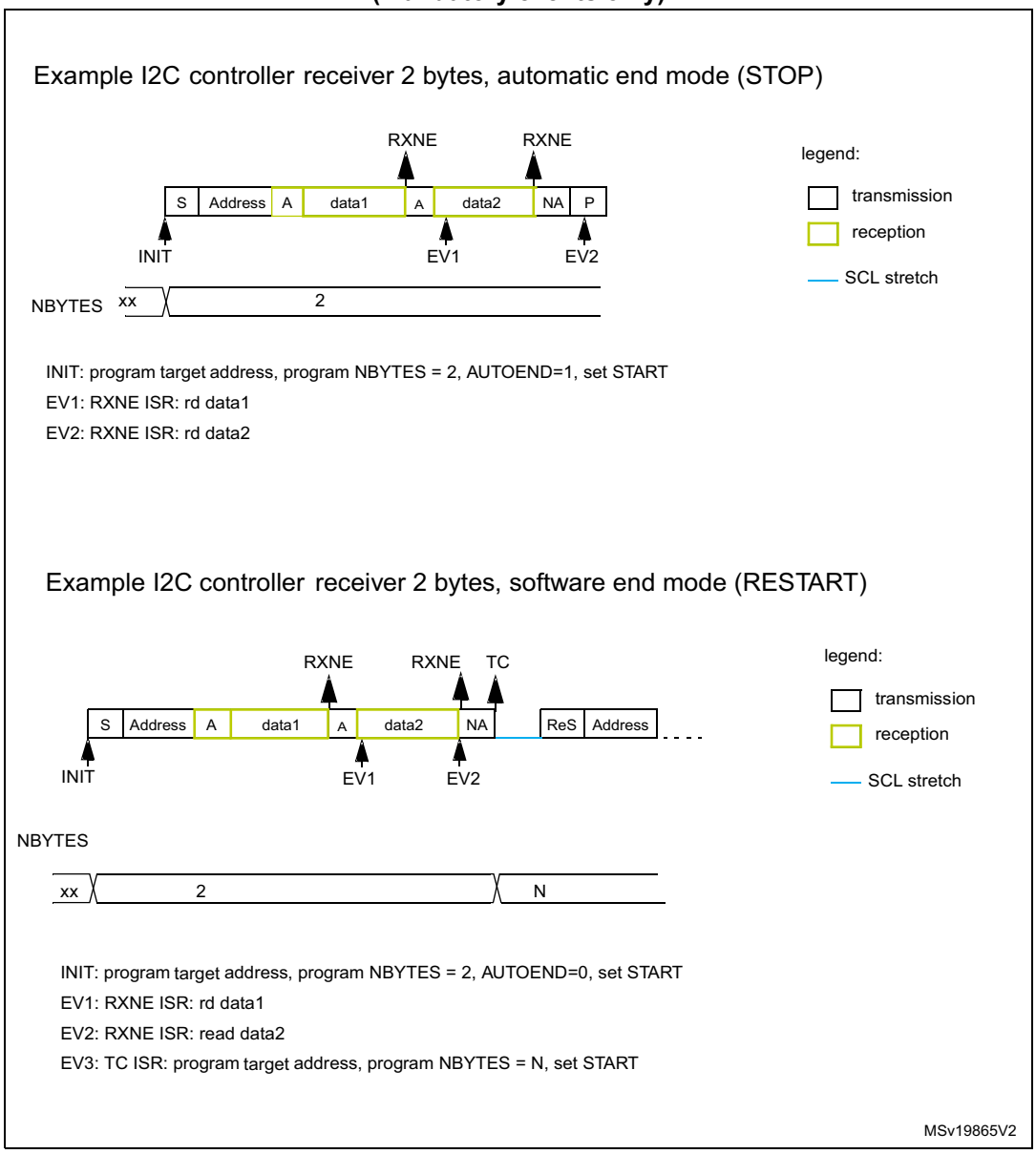


Figure 397. Transfer bus diagrams for I2C controller receiver (mandatory events only)



34.4.10 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 262. 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 263. 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.

34.4.11 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.

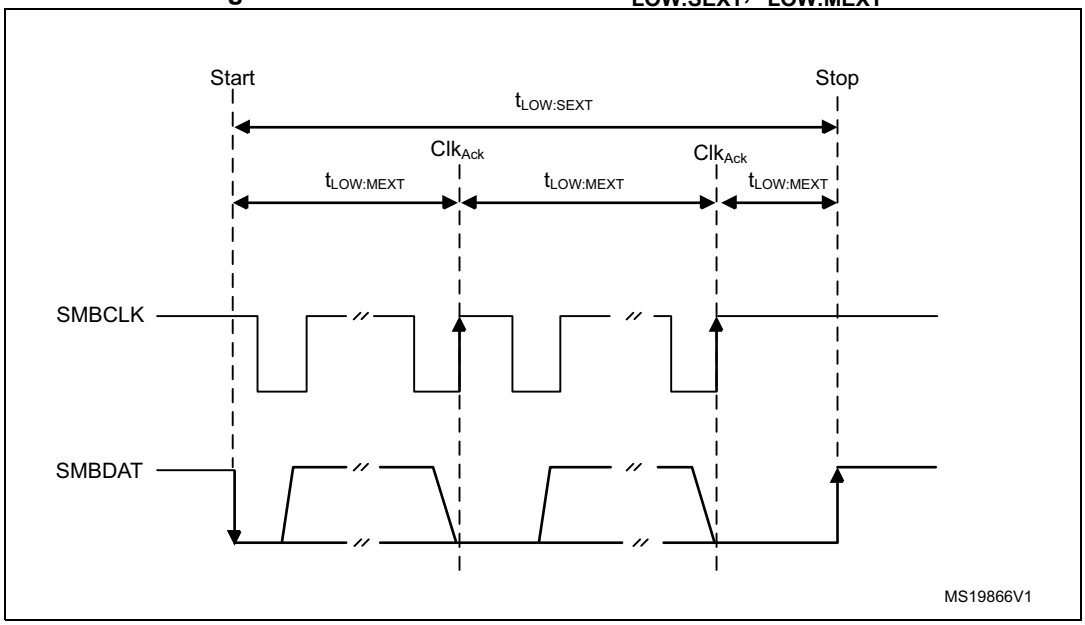
Table 264. SMBus timeout specifications

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 398. 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 34.4.9*).

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.

34.4.12 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 265. 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 266*.

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 267*.

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 268*.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

34.4.13 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 266. 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 267. 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 268. 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

34.4.14 SMBus target mode

In addition to I2C target transfer management (refer to *Section 34.4.8: 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 399. Transfer sequence flow for SMBus target transmitter N bytes + PEC

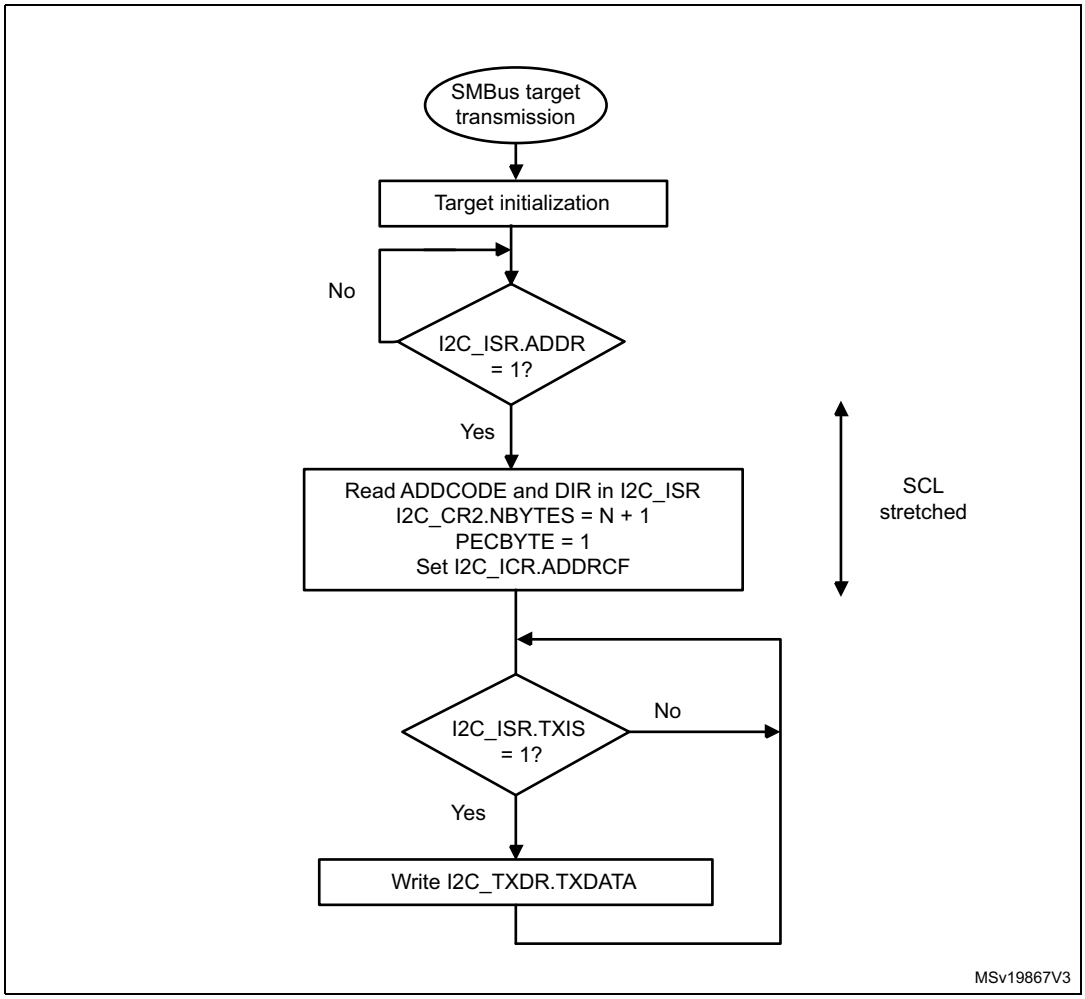
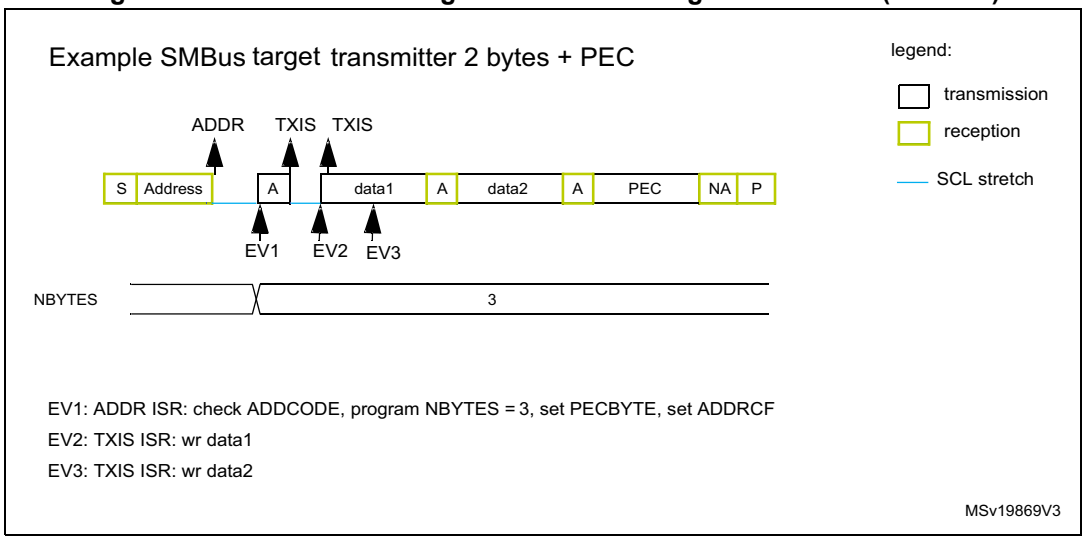


Figure 400. Transfer bus diagram for SMBus target transmitter (SBC = 1)



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 401. Transfer sequence flow for SMBus target receiver N bytes + PEC

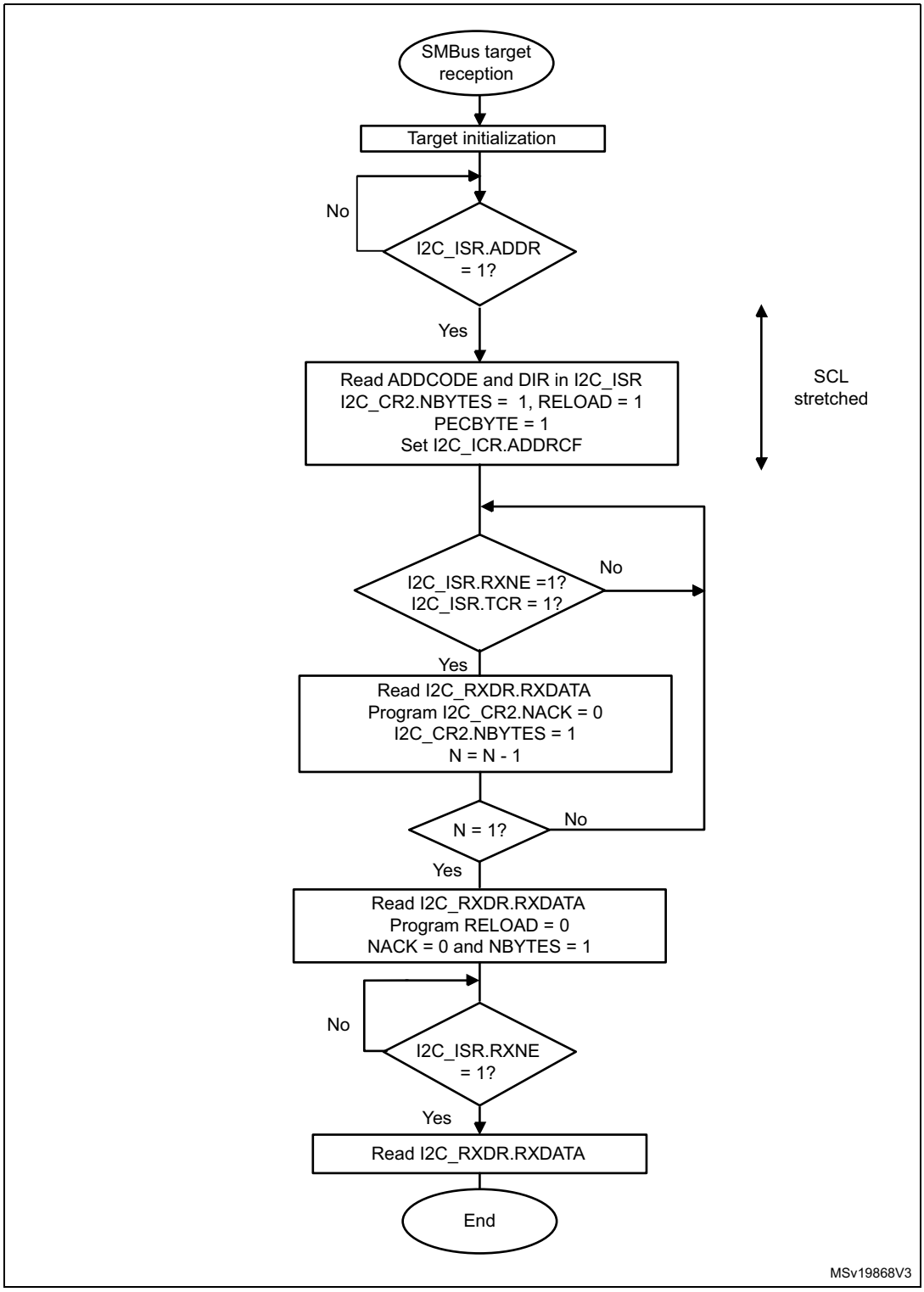
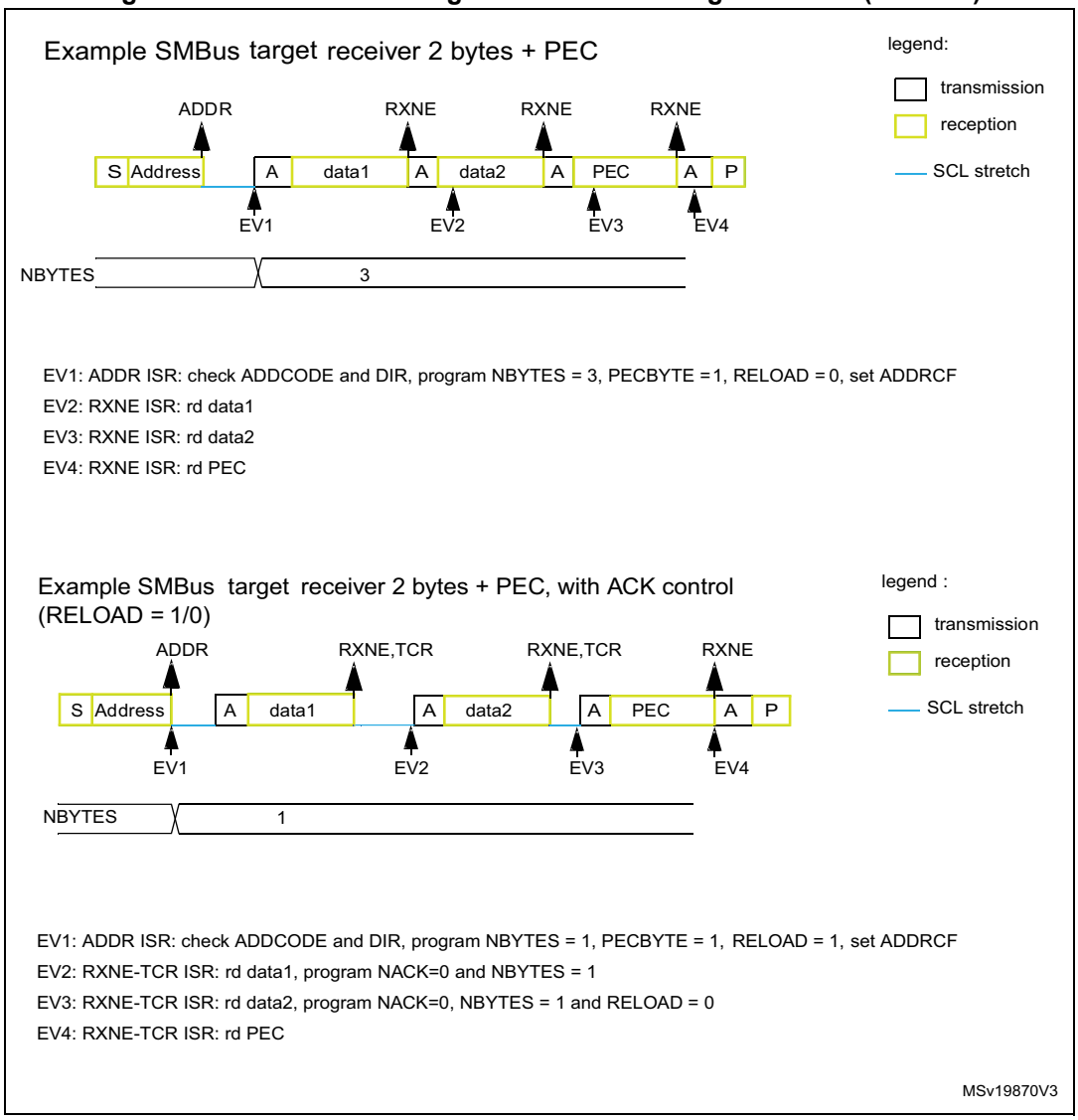


Figure 402. Bus transfer diagrams for SMBus target receiver (SBC = 1)



34.4.15 SMBus controller mode

In addition to I2C controller transfer management (refer to *Section 34.4.9: I2C controller mode*), this section provides extra software flowcharts to support SMBus.

SMBus controller transmitter

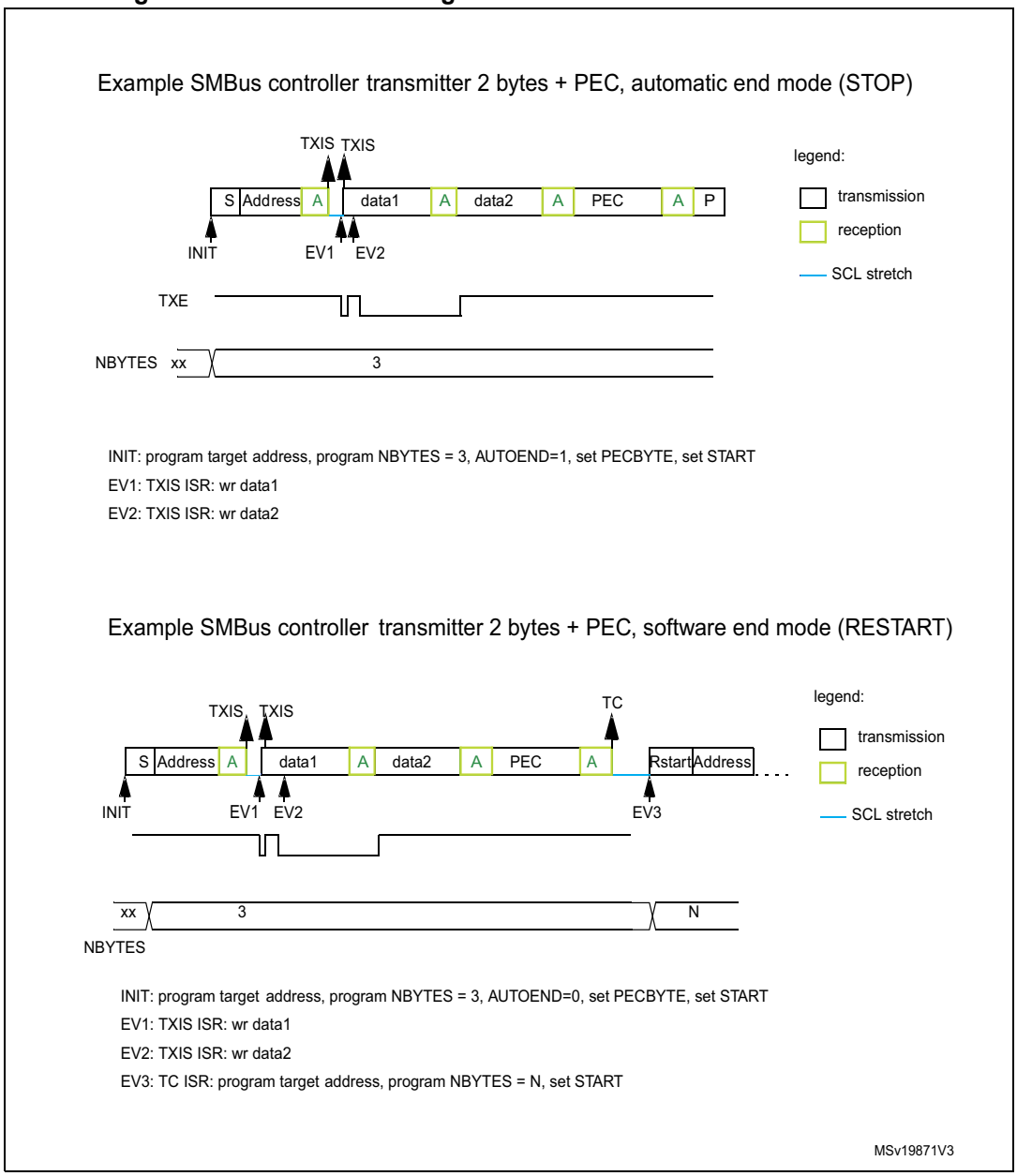
When the SMBus controller wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be loaded in the NBYTES[7:0] bitfield, before setting the START bit. In this case, the total number of TXIS interrupts is NBYTES[7:0] - 1. So if the PECBYTE bit is set when NBYTES[7:0] = 0x1, 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 403. Bus transfer diagrams for SMBus controller transmitter



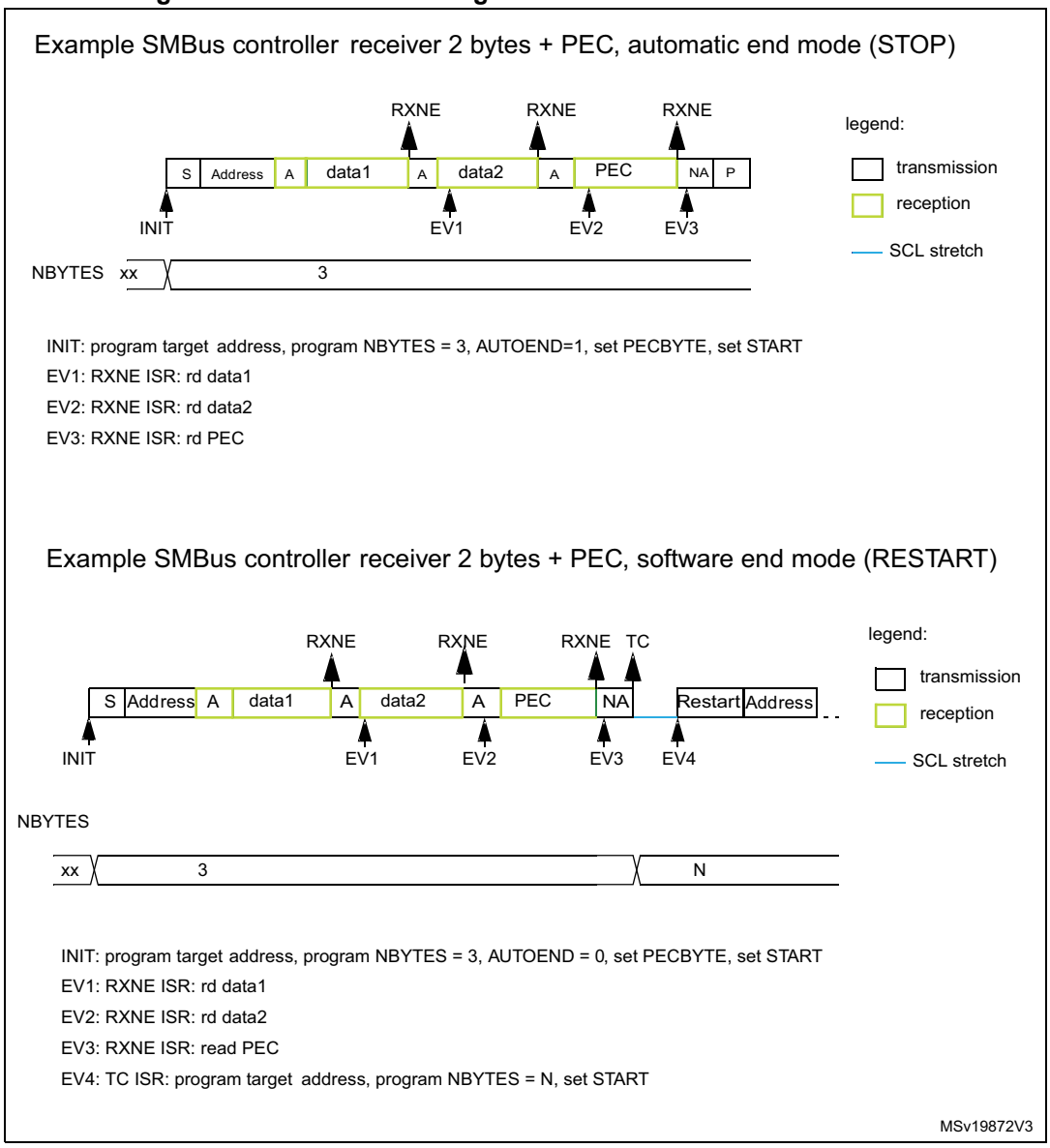
SMBus controller receiver

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 404. Bus transfer diagrams for SMBus controller receiver



34.4.16 Wake-up from Stop mode on address match

The I2C peripheral is able to wake up the device from Stop mode (APB clock is off), when the device is addressed. All addressing modes are supported.

The wake-up from Stop mode is enabled by setting the WUPEN bit of the I2C_CR1 register. The HSI and CSI only oscillator must be selected as the clock source for i2c_ker_ck to allow the wake-up from Stop mode.

In Stop mode, the HSI and CSI only oscillator is stopped. Upon detecting START condition, the I2C interface starts the HSI and CSI only oscillator and stretches SCL low until the oscillator wakes up.

HSI and CSI only is then used for the address reception.

If the received address matches the device own address, I2C stretches SCL low until the device wakes up. The stretch is released when the ADDR flag is cleared by software. Then the transfer goes on normally.

If the address does not match, the HSI and CSI only oscillator is stopped again and the device does not wake up.

Note: When the system clock is used as I2C clock, or when WUPEN = 0, the HSI and CSI only oscillator does not start upon receiving START condition.

Only an ADDR interrupt can wake the device up. Therefore, do not enter Stop mode when I2C is performing a transfer, either as a controller or as an addressed target after the ADDR flag is set. This can be managed by clearing the SLEEPDEEP bit in the ADDR interrupt routine and setting it again only after the STOPF flag is set.

Caution: The digital filter is not compatible with the wake-up from Stop mode feature. Before entering Stop mode with the WUPEN bit set, deactivate the digital filter, by writing zero to the DNF[3:0] bitfield.

Caution: The feature is only available when the HSI and CSI only oscillator is selected as the I2C clock.

Caution: Clock stretching must be enabled (NOSTRETCH = 0) to ensure proper operation of the wake-up from Stop mode feature.

Caution: If the wake-up from Stop mode is disabled (WUPEN = 0), the I2C peripheral must be disabled before entering Stop mode (PE = 0).

34.4.17 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.

34.5 I2C in low-power modes

Table 269. Effect of low-power modes to I2C

Mode	Description
Sleep	No effect. I2C interrupts cause the device to exit the Sleep mode.
Stop(1)	The contents of I2C registers are kept. – WUPEN = 1 and I2C is clocked by an internal oscillator (HSI and CSI only). The address recognition is functional. The I2C address match condition causes the device to exit the Stop mode. – WUPEN = 0: the I2C must be disabled before entering Stop mode.
Standby	The I2C peripheral is powered down. It must be reinitialized after exiting Standby mode.

1. Refer to *Section 34.3: I2C implementation* for information about the Stop modes supported by each instance. If the wake-up from a specific stop mode is not supported, the instance must be disabled before entering that specific Stop mode.

34.6 I2C interrupts

The following table gives the list of I2C interrupt requests.

Table 270. I2C interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop modes	Exit Standby modes
I2C_EV	Receive buffer not empty	RXNE	RXIE	Read I2C_RXDR register	Yes	No	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		Yes(1)	
	NACK reception	NACKF	NACKIE	Write NACKCF = 1		No	
I2C_ERR	Bus error	BERR	ERRIE	Write BERRCF = 1	Yes	No	No
	Arbitration loss	ARLO		Write ARLOCF = 1			
	Overrun/underrun	OVR		Write OVRCF = 1			
I2C_ERR	PEC error	PECERR	ERRIE	Write PECERRCF = 1	Yes	No	No
	Timeout/ t_{LOW} error	TIMEOUT		Write TIMEOUTCF = 1			
	SMBus alert	ALERT		Write ALERTCF = 1			

1. The ADDR match event can wake up the device from Stop mode only if the I2C instance supports the wake-up from Stop mode feature. Refer to *Section 34.3: I2C implementation*.

34.7 I2C DMA requests

34.7.1 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 15: General purpose direct memory access controller (GPDMA)*) 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.

34.7.2 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 15: General purpose direct memory access controller (GPDMA)*) 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.

34.8 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.

34.9 I2C registers

Refer to *Section 1.2* for the list of abbreviations used in register descriptions.

The registers are accessed by words (32-bit).

34.9.1 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 20 mA drive enable

0: 20 mA I/O drive disabled

1: 20 mA 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.

34.9.2 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.

34.9.3 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.

34.9.4 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.

34.9.5 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.

34.9.6 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.

34.9.7 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.

34.9.8 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.

34.9.9 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.

34.9.10 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.

34.9.11 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.

34.9.12 I2C register map

The table below provides the I2C register map and the reset values.

Table 271. 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

Refer to *Section 2.2* for the register boundary addresses.

35 Improved inter-integrated circuit (I3C)

35.1 Introduction

The I3C interface handles communication between this device and others, such as sensors and host processor, connected on an I3C bus.

An I3C bus is a two-wire, serial single-ended, multidrop bus, intended to improve a legacy I^2C bus.

The I3C SDR-only peripheral implements all the features required by the MIPI® I3C specification v1.1. It can control all I3C bus-specific sequencing, protocol, arbitration, and timing, and can act as controller (formerly known as master), or as target (formerly known as slave).

When acting as controller, the I3C peripheral improves the features of the I^2C interface preserving some backward compatibility: it allows an I^2C target to operate on an I3C bus in legacy I^2C fast-mode (Fm) or legacy I^2C fast-mode plus (Fm+), provided that the latter does not perform clock stretching.

The I3C peripheral can be used with DMA, to off-load the CPU.

35.2 I3C main features

The I3C peripheral supports:

- MIPI® I3C specification v1.1 (see details in *Table 275*), as:
  - I3C SDR-only primary controller
  - I3C SDR-only secondary controller
  - I3C SDR-only target
- I3C SCL bus clock frequency up to 12.5 MHz
- Registers configuration from the host application via the APB slave port
- Queued data transfers:
  - Transmit FIFO (TX-FIFO) for data bytes/words to be transmitted on the I3C bus
  - Receive FIFO (RX-FIFO) for received data bytes/words on the I3C bus
  - For each FIFO, optional DMA mode with a dedicated DMA channel
- Queued control/status transfers, when controller:
  - Control FIFO (C-FIFO) for control words to be sent on the I3C bus
  - Optional status FIFO (S-FIFO) for status words as received on the I3C bus
  - For each FIFO, optional DMA mode with a dedicated DMA channel
- Messages:
  - Legacy I^2C read/write messages to legacy I^2C targets in Fm/Fm+
  - I3C SDR read/write private messages
  - I3C SDR broadcast CCC messages (see details in *Table 281*)
  - I3C SDR read/write direct CCC messages (see details in *Table 281*)

- Frame-level management, when controller:
  - Optional C-FIFO and TX-FIFO preload
  - Multiple messages encapsulation
  - Optional arbitrable header generation on the I3C bus
  - HDR exit pattern generation on the I3C bus for error recovery
- Programmable bus timing, when controller:
  - SCL high and low period
  - SDA hold time
  - Bus free (minimum) time
  - Bus available/idle condition time
  - Clock stall time
- Target-initiated requests management:
  - Simultaneous support up to four targets, when controller
  - In-band interrupts, with programmable IBI payload (up to 4 bytes), with pending read notification support
  - Bus control request, with recovery flow support and hand-off delay
  - Hot-join mechanism
- HDR exit pattern detection, when target
- Bus error management:
  - CEx with x = 0, 1, 2, 3 when controller
  - TEx with x = 0, 1, ... , 6 when target
  - Bus control switch error and recovery
  - Target reset
- Individual programmable event-based management:
  - Per-event identification with flag reporting and clear control
  - Host application notification via flag polling, and/or via interrupt with a per-event programmable enable
  - Error type identification
- Wake-up from Stop mode(s), as controller (see *Section 35.3.2*):
  - On an in-band interrupt without payload
  - On a hot-join request
  - On a controller-role request
- Wake-up from Stop mode(s), as target (see *Section 35.3.2*):
  - On a reset pattern
  - On a missed start
- Multiclock domain management:
  - Separate APB clock and kernel clock, driven from independently programmed clock sources via the RCC, in addition to SCL clock
  - Minimum operating frequency for the kernel clock and the APB clock vs. the application-driven SCL clock (see clocks constraints in *Section 35.6.2*)

35.3 I3C implementation

35.3.1 I3C instantiation

There is a single I3C instance in the device.

35.3.2 I3C wake-up from low-power mode(s)

The I3C peripheral can wake up the device from a low-power mode, as detailed in *Table 272*. For more details about the wake-up capabilities, refer to *Section 35.13*.

Table 272. I3C wake-up

Wake-up
From Stop mode with SVOS3

35.3.3 I3C FIFOs

The FIFOs are implemented as defined in *Table 273*.

Table 273. I3C FIFOs implementation

FIFO	Content	Unit	Size (in unit)	Used as controller/target (rationale)
C-FIFO	(32-bit) Control words	Word	2	Controller (a frame can be based on multiple control words; this is not the case as target)
S-FIFO	(32-bit) Status words			Controller (target: status only in register mode)
TX-FIFO	Transmitted data	Byte	8	Controller and target
RX-FIFO	Received data			

35.3.4 I3C triggers

This feature is not available in this product: no hardware trigger signal is connected as an input to the I3C peripheral.

35.3.5 I3C interrupt(s)

The interrupt mapping is implemented as detailed in *Table 274*.

Table 274. I3C interrupt(s)

Signal name	Signal type	Description
i3c_err_it	O	Error interrupt line
i3c_evt_it	O	Event interrupt line

35.3.6 I3C MIPI® support

The I3C peripheral supports the MIPI specification v1.1, as defined in *Table 275*.

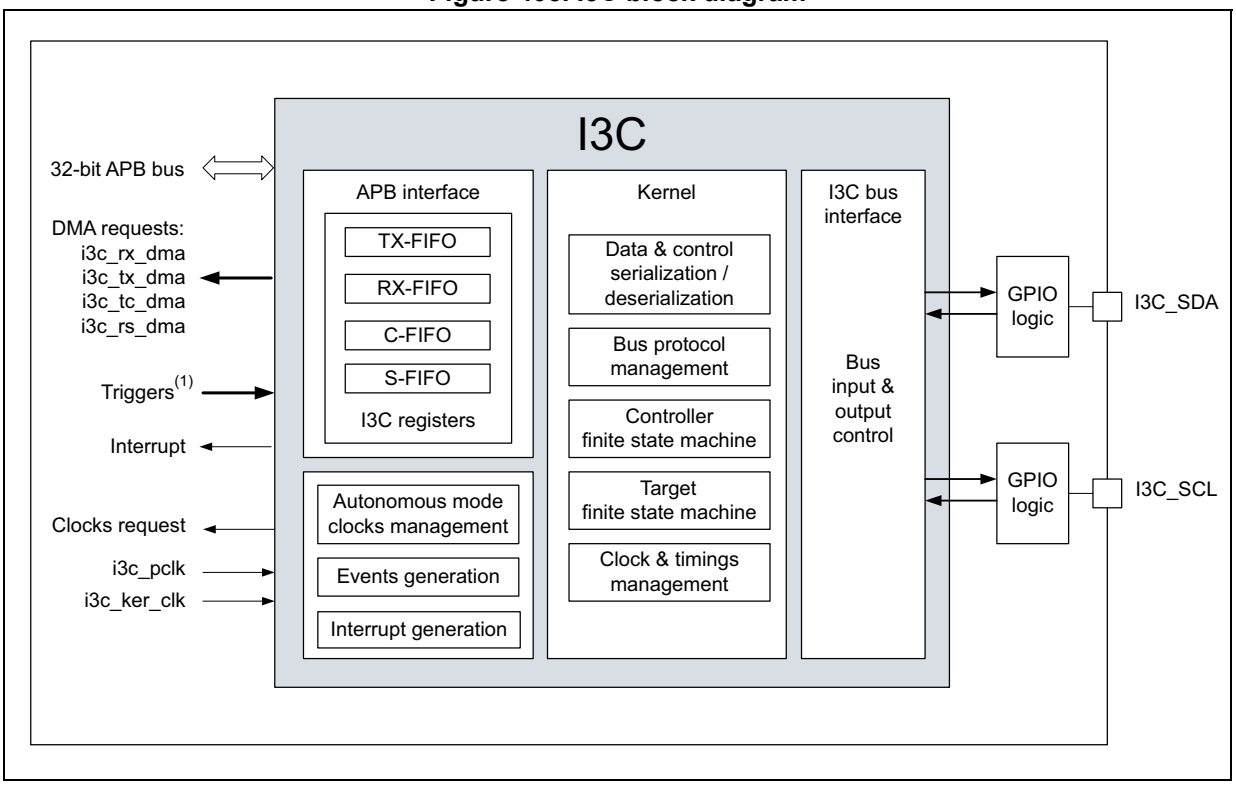
Table 275. I3C peripheral controller/target features versus MIPI v1.1

Feature	MIPI I3C v1.1	I3C peripheral		Comments
		When controller	When target	
I3C SDR message	X	X	X	-
Legacy I^2C message (Fm/Fm+)	X	X	-	Mandatory when controller, and the I3C bus is mixed with (external) legacy I^2C target(s). Optional in MIPI v1.1 when target.
HDR DDR message	X	-	-	Optional in MIPI v1.1
HDR-TSL/TSP, HDR-BT	X	-	-	Optional in MIPI v1.1
Dynamic address assignment	X	X	X	-
Static address	X	X	-	No (intended) support of the I3C peripheral as a target on an I^2C bus.
Grouped addressing	X	X	-	Optional in MIPI v1.1
CCCs	X	X	X	Mandatory and some optional CCCs supported (refer to *Table 281* when controller/target).
Error detection and recovery	X	X	X	-
In-band interrupt (with MDB)	X	X	X	-
Secondary controller	X	X	X	-
Hot-join mechanism	X	X	X	-
Target reset	X	X	X	-
Synchronous timing control	X	X	-	Optional in MIPI v1.1
Asynchronous timing control 0	X	X	-	Mandatory in MIPI v1.1 when controller. Optional in MIPI v1.1 when target.
Asynchronous timing control 1, 2, 3	X	-	-	Optional in MIPI v1.1
Device to device tunneling	X	X	-	Optional in MIPI v1.1
Multi-lane data transfer	X	-	-	Optional in MIPI v1.1
Monitoring device early termination	X	-	-	Optional in MIPI v1.1

35.4 I3C block diagram

The I3C block diagram is illustrated in *Figure 405*.

Figure 405. I3C block diagram



1. Refer to *Section 35.3.4: I3C triggers*.

35.5 I3C pins and internal signals

Table 276. I3C input/output pins

Pin name	Pin type	Description
I3C_SDA	Input/output	I3C bus serial data line
I3C_SCL	Input/output	I3C bus serial clock line

Table 277. I3C internal input/output signals

Signal name	Signal type	Description
i3c_pclk	I	APB clock
i3c_ker_clk	I	Kernel clock (also named as I3CCLK)
i3c_pclk_req	O	APB clock request
i3c_ker_clk_req	O	Kernel clock request
i3c_it(1)	O	Global interrupt line

Table 277. I3C internal input/output signals (continued)

Signal name	Signal type	Description
i3c_err_it(1)	O	Error interrupt line
i3c_evt_it(1)	O	Event interrupt line
i3c_rx_dma	O	DMA request for reading received bytes/words from RX-FIFO
i3c_tx_dma	O	DMA request for writing to be transmitted bytes/words to TX-FIFO
i3c_tc_dma	O	DMA request for writing to be transmitted control words to C-FIFO, when the I3C peripheral acts as controller
i3c_rs_dma	O	DMA request for reading status words from S-FIFO, when the I3C peripheral acts as controller

1. Refer to *Section 35.3.5: I3C interrupt(s)*.

35.6 I3C reset and clocks

35.6.1 I3C reset

On a system reset, the I3C peripheral is reset.

Alternatively, the software can reset specifically the I3C peripheral by writing the corresponding reset control bit (I3CxRST) of the reset and clock controller (RCC). Refer to the RCC section of this document for more details.

Additionally, when acting as target, the enabled I3C peripheral (EN = 1 in the I3C_CFGR register) can receive an in-band reset pattern on the I3C bus from the controller. The software is then notified (when RSTF = 1 in the I3C_EVR register and/or the corresponding interrupt is enabled) to perform the requested action, as registered in RSTACT[1:0] of the I3C_DEVR0 register, on the former reception of the broadcast or direct RSTACT CCC. Refer to *Table 281* and *Section 35.16.16* for more details.

This reset interrupt notification can be used to wake up from a low power mode, where the I3C peripheral (typically in the V_{CORE} domain) is active.

For more details about the corresponding low-power mode(s), refer to the power management in the PWR section.

35.6.2 I3C clocks and requirements

As indicated in *Figure 405*, the I3C peripheral is implemented with several clock domains:

- SCL bus clock: for the I3C bus interface
  - When controller: the user must set and can adjust SCL/SDA timings by programming *I3C timing register 0 (I3C_TIMINGR0)*, *I3C timing register 1 (I3C_TIMINGR1)* and *I3C timing register 2 (I3C_TIMINGR2)*, as summarized in *Controller initialization* and *Updating the configuration for a transfer, as controller*.
  - When target: the user must set and comply with the bus available condition (t_{AVAL} for an in-band interrupt or controller-role request), and the bus idle condition (t_{IDLE} for a hot-join request), by programming *I3C timing register 1 (I3C_TIMINGR1)*, as summarized in *Target initialization*.

- I3CCLK kernel clock: for the I3C protocol management, data and control serialization/deserialization, controller and target finite state machines, bus clock and timings management
- APB clock: for the APB interface, DMA interface, events, and interrupt generation

APB clock and kernel clocks are driven from independently programmed clock sources via the RCC (refer to *Section 10: Reset and clock control (RCC)*).

I3C kernel clock requirement, as controller

According to the intended value of the SCL clock on the bus, the application must guarantee that the frequency of the I3CCLK kernel clock be at least 2x the frequency of the SCL clock

Note: Sustaining $F_{SCL\ max}$ = 12.9 MHz means a frequency of the I3CCLK kernel clock > 25.8 MHz.

I3C kernel clock requirements, as target

According to the intended value of the SCL clock on the bus, the application must guarantee a minimum operating frequency for the I3CCLK kernel clock, meeting the following constraints:

1. Period of the I3CCLK kernel clock < t_{HIGH} (SCL clock high period)
   - $t_{HIGH\ min}$ = 24 ns. A frequency higher than 41.7 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
2. Period of the I3CCLK kernel clock < t_{CASr} (clock after repeated start condition)
   - $t_{CASr\ min}$ = $t_{CAS\ min}$ / 2 = 19.2 ns. A frequency higher than 52 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
3. Two periods of the I3CCLK kernel clock < t_{LOW_OD} (SCL clock low period in open drain)
   - $t_{LOW\_OD\ min}$ = 200 ns. A frequency higher than 10 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
4. Frequency of the I3CCLK kernel clock > 2.5x frequency of the SCL clock
   - $F_{SCL\ max}$ = 12.9 MHz. A frequency higher than 32.3 MHz guarantees this constraint, which can be relaxed, depending on the I3C controller

APB clock requirement

According to the intended value of the SCL clock on the bus, the application must guarantee a minimum operating frequency for the APB clock:

APB clock period < 3x (SCL clock period) - I3CCLK kernel clock period

This means that $F_{APB} > [F_{SCL} \times F_{I3CCLK} / (3x\ F_{I3CCLK} - F_{SCL})]$

Note: This equation can be simplified to a minimum value of 5 MHz for the APB frequency.

35.7 I3C peripheral state and programming

35.7.1 I3C peripheral state

The I3C peripheral plays the role of I3C bus controller, or the role of an I3C target. In any case (see *Figure 406* and *Figure 407*), the peripheral is in one of the following states:

- Disabled state:
  - After an I3C reset (system reset or I3C reset from RCC), the I3C peripheral is in disabled state.
  - When the software sets to 1 bit EN in the I3C_CFGR register, the I3C peripheral takes into account the value of the different configuration registers, and switches to the (enabled and) idle state.
- Idle state:
  - After being enabled (EN = 1), the I3C peripheral activates its I3CCLK and SCL clock domains, and is able to communicate on the I3C bus.
  - The software can partly update the I3C peripheral configuration, see *Updating the configuration for a transfer, as controller* and *Updating the configuration of the I3C peripheral, as target* for more details.
  - The I3C peripheral switches to the (enabled and) active state in either one of the following conditions:

    Once the software initiates a transfer (as controller: when the software initiates a frame transfer; as target: when the software initiates an IBI/CR/HJ request).

    Once the hardware receives a request from another I3C device on the bus (as controller: after a start request from a target and a maximum T_{CAS} time; as target: when receiving a broadcast/direct CCC or a private read/write).
- Active state:
  - The I3C peripheral executes the transfer(s) on the bus.
  - When the requested transfer(s) is(are) completed, the software is notified by an event from the *I3C event register (I3C_EVR)*, and the corresponding interrupt is enabled by *I3C interrupt enable register (I3C_IER)*. The I3C peripheral switches to idle state, is still able to communicate on the bus, and can be (partly) reconfigured:

    As controller: the raised event/flag can be frame completed (FCF), IBI/controller-role/hot-join request completed (IBIF/CRF/HJF), or transfer error (ERRF).

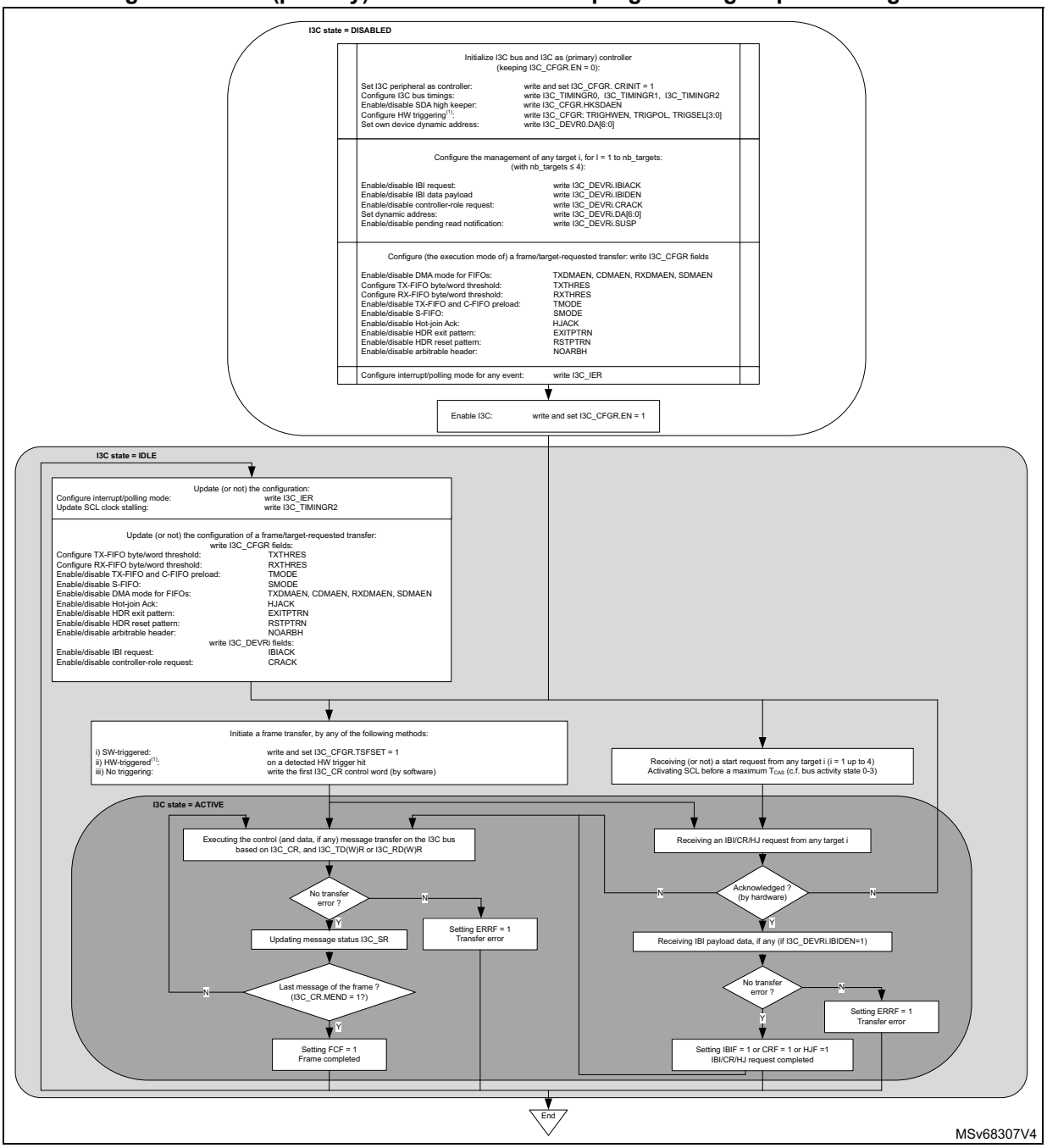
    As target: the raised event/flag can be dynamic address assignment completed (DAUPDF), IBI completed (IBIENDF), controller-role gaining completed (CRUPDF), broadcast/direct CCC completed (xxUPDF/RSTF/GETF/STAF), private read/write completed (FCF), or transfer error (ERRF).

Note: The software can disable the I3C peripheral *(write EN = 0), partially resetting it (subparts within the SCL clock domain and the I3CCLK kernel clock domain). Event, interrupt, and clock request generation are also impacted. The previously written configuration of the APB registers is kept and not modified.*

35.7.2 I3C controller state and programming sequence

Figure 406 illustrates the overall programming sequence of the I3C peripheral acting as (primary) controller, including state transitions, main subtasks, and conditions, as explained in this section.

Figure 406. I3C (primary) controller state and programming sequence diagram



1. This feature is implementation-dependent and can be unavailable. Refer to *Section 35.3.4: I3C triggers*.

Controller initialization

When the controller is in disabled state (EN = 0 in the I3C_CFGR register), the software must initialize as follows:

- Configure *I3C configuration register (I3C_CFGR)* with the following fields:
  - CRINIT = 1: as I3C bus controller
  - HKSDAEN: high keeper on SDA enable/disable
- Configure I3C bus timings:
  a) *I3C timing register 0 (I3C_TIMINGR0)*:
  - SCL clock high time period (t_{DIG_H}, $t_{DIG\_H\_MIXED}$) in legacy I^2C and I3C open-drain/push-pull
  - SCL clock low time period (t_{DIG_L}, $t_{DIG\_OD\_L}$) in legacy I^2C and I3C open-drain/push-pull

  b) *I3C timing register 1 (I3C_TIMINGR1)*:
  - SDA hold time in push-pull (t_{HD_PP})
  - bus free condition time (I3C $t_{CAS,}$ legacy I^2C t_{BUF})
  - I3C repeated start timing (t_{CASr}, t_{CBSr})
  - I3C stop timing (t_{CBP})
  - SCL clock low maximum stalling on the ENTDAA CCC ($t_{STALLDAA}$), or on the ACK/NACK of a legacy I^2C or address phase of an I3C transfer, or the parity bit of a write data transfer, or on the ACK/NACK data phase of a legacy I^2C write, or on the transition bit of an I3C read transfer, or on the ACK/NACK phase of a legacy I^2C write (t_{STALL}), to adjust SCL clock low stalling, if needed by the I3C peripheral itself, when used as controller
  - $t_{NEWCRLock}$ for controller-role hand-off procedure (after GETACCCR CCC)

  c) *I3C timing register 2 (I3C_TIMINGR2)*:
  - SCL clock low stalling time, with separated enable/disable for each phase, to adjust SCL clock low stalling, if needed on the SDA hand-off with the addressed I3C target or legacy I^2C target
- Configure its own dynamic address: DA [6:0] field of the *I3C own device characteristics register (I3C_DEVR0)*
- Configure the management of any device target x: *I3C device x characteristics register (I3C_DEVRx)*, for x = 1 to x ≤ 4
- Configure the execution mode of a frame transfer or a target-requested transfer: *I3C configuration register (I3C_CFGR)*, with the following fields:
  - TXDMAEN, CDMAEN, RXDMAEN, SDMAEN: DMA mode enable/disable for respectively, TX-FIFO, C-FIFO, RX-FIFO, S-FIFO
  - TXTHRES, RXTHRES: respectively TX-FIFO and RX-FIFO byte/world threshold
  - TMODE: transmit mode (enable/disable for both TX-FIFO and C-FIFO preload)
  - SMODE: S-FIFO enable/disable
  - EXITPTRN, RSTPTRN: exit, reset pattern enable/disable
  - HJACK: hot-join acknowledge enable/disable
  - NOARBH: arbitrable header disable/enable
- Configure interrupt generation or polling mode from any event: *I3C interrupt enable register (I3C_IER)*

Then, the software can enable the I3C peripheral (set EN = 1).

Note: The software can write once all the fields of the I3C_CFGR while enabling it.

Start a controller-initiated frame transfer

When the controller is in enabled state (EN = 1 in the I3C_CFGR register), the software can initiate a frame transfer by any of the following configuration methods:

- Software-triggering: on a write and set TSFSET = 1 in the I3C_CFGR register
  - This causes the hardware to raise the flag CFNFF = 1 in the I3C_EVR register, to request a first control word I3C_CR to be written.
- No triggering: on a write of the first control word I3C_CR by software.

Then, regardless of the frame starting method, the I3C peripheral switches to active state. While the control word is not the last message of the I3C frame (while MEND = 0 in the I3C_CR register), and while there is no transfer error (while ERRF = 1 in the I3C_EVR register), the hardware keeps requesting a next control word, and continuing the frame transfer:

- If the C-FIFO in not configured in DMA mode (CDMAEN = 0 in the I3C_CFGR register), the software writes a next control word following the flag CFNFF = 1 in the I3C_EVR register, or the corresponding interrupt if enabled (if CFNFIE = 1 in I3C_IER register)
- If the C-FIFO in configured in DMA mode (CDMAEN = 1), a next control word is automatically pushed and written by the allocated DMA channel consequently to the asserted I3C DMA request (i3c_tc_dma).

Start and receiving a target-initiated transfer

When the controller is in enabled state (EN = 1 in the I3C_CFGR register), concurrently to a possible controller-initiated transfer, a target can initiate a transfer by issuing a start request (drive SDA low), provided the controller has allowed a hot-join request, an IBI request, or a controller-role request via the I3C_DEVR0 register.

In this case, even though the controller software has no intent to start a frame transfer, the hardware switches to active state (activates the SCL clock before a maximum t_{CAS} time defined as 1 µs, 100 µs, 2 ms, or 50 ms, depending on, respectively, the bus activity state 0, 1, 2, or 3) to receive the hot-join/in-band interrupt/controller-role request from the target.

For more information about the execution of target-initiated I3C bus transfer and its related programming as a controller, refer to the relevant figures in *Section 35.9*:

- *Figure 418: IBI transfer, as controller/target*
- *Figure 419: Hot-join request transfer, as controller/target*
- *Figure 420: Controller-role request transfer, as controller/target*

Executing a (controller-initiated) frame transfer

The controller executes on the bus the frame transfer until the completion of the last message (FCF = 1 in the I3C_EVR register), or a transfer error (ERRF = 1 in the I3C_EVR register), and the corresponding interrupt, if enabled. This is based on I3C_CR and I3C_TD(W)R registers, written explicitly by software or pushed by the allocated DMA channel, and based on the I3C_RD(W)R, read explicitly by the software or by the allocated DMA channel. Then the I3C controller switches back to idle state.

For more information about the execution of controller-initiated I3C bus transfer and its related programming as a controller, refer to figures in *Section 35.9*:

- *Figure 408: I3C CCC messages, as controller*
- *Figure 409: I3C broadcast ENTDAA CCC, as controller*
- *Figure 410: I3C broadcast, direct read and direct write RSTACT CCC, as controller*
- *Figure 415: I3C private read/write messages, as controller*
- *Figure 417: Legacy I2C read/write messages, as controller*

Figure 406 does not include the management of the FIFOs (TX-FIFO, RX-FIFO, C-FIFO, and S-FIFO). This is detailed in *Section 35.10*.

For each completed message without transfer error, the hardware reports the exchanged transfer on the I3C bus by updating *I3C status register (I3C_SR)*, which can be read or not by the software when the S-FIFO is disabled (SMODE = 0 in the I3C_CFGR register).

- In the case of a direct CCC read or a private read transfer, in addition to the completion of the last message (FCF = 1 in the I3C_EVR register) or a transfer error (ERRF = 1 in the I3C_EVR register) and the corresponding interrupt if enabled, and provided that the S-FIFO is disabled for the status register I3C_SR (SMODE = 0 in the I3C_CFGR register), the software is notified if the read transfer is ended prematurely by the target by RXTGTENDF = 1 in the I3C_EVR register, and the corresponding interrupt, if enabled. The software can then read I3C_SR, to get more information about the executed transfer.

Alternatively, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), the status register I3C_SR must be read for each executed message, either directly by the software (notified by SFNEF = 1 in the I3C_EVR register and the corresponding interrupt, if enabled), or via the DMA (if SDMAEN = 1 in the I3C_CFGR register), no matter if a read is prematurely ended by the target or not. Frame completion (FCF = 1 in the I3C_EVR register) occurs only after reading the status of the last message (S-FIFO is empty). For more information, refer to *Section 35.10.4*.

Updating the configuration for a transfer, as controller

Back in idle state, the software can update the configuration of the I3C peripheral before the next transfer:

- Modify SCL clock stalling via *I3C timing register 2 (I3C_TIMINGR2)*
- Modify the interrupt/polling mode policy via *I3C interrupt enable register (I3C_IER)*
- Modify the following fields of the *I3C configuration register (I3C_CFGR)*:
  - TXTHRES, RXTHRES
  - TMODE, SMODE
  - TXDMAEN, CDMAEN, RXDMAEN, SDMAEN
  - EXITPTRN, RSTPTRN
  - NOARBH

- Modify/prepare the control words, status words, read/write data of the next frame transfer to be executed, by software and/or DMA
  - *I3C message control register (I3C_CR)*, *I3C message control register [alternate] (I3C_CR)*
  - *I3C status register (I3C_SR)*
  - *I3C transmit data byte register (I3C_TDR)*, *I3C transmit data word register (I3C_TDWR)*
  - *I3C receive data byte register (I3C_RDR)*, *I3C receive data word register (I3C_RDWR)*
- Typically after having issued and completed a broadcast/direct DISEC/ENEC CCC:
  - Modify the hot-join acknowledge policy via bit HJACK in the I3C_CFGR register
  - Modify IBI/CR acknowledge policy to any target x, via *I3C device x characteristics register (I3C_DEVRx)*

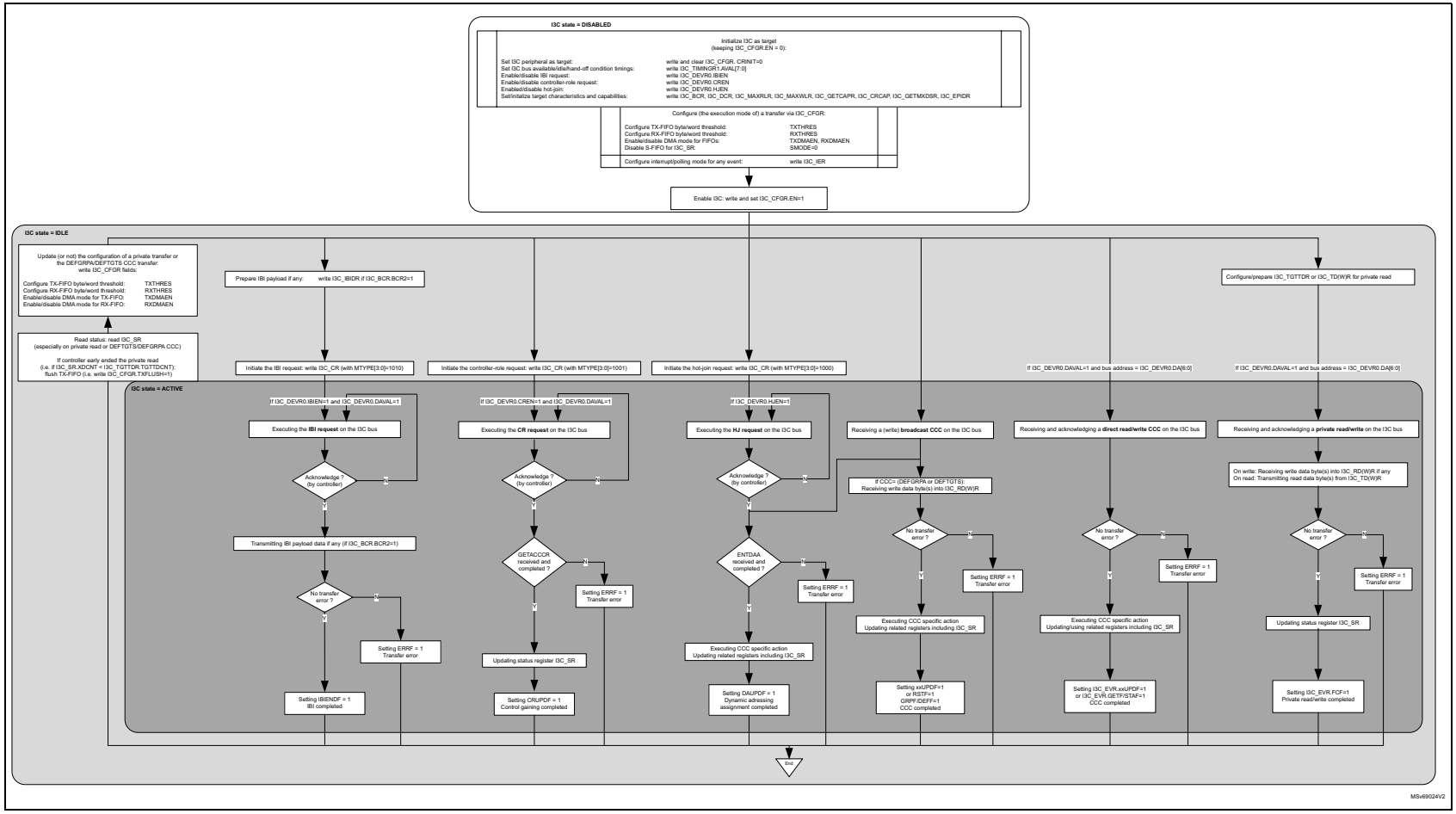
The registers usage versus the I3C peripheral role as controller is summarized in *Section 35.8.1*.

The static/dynamic registers fields usage when acting as controller is summarized in *Table 279*.

35.7.3 I3C target state and programming sequence

Figure 407 illustrates the overall programming sequence of the I3C peripheral acting as target, including state transitions, main subtasks, and conditions, as explained in this section.

Figure 407. I3C target state and programing sequence diagram



Target initialization

When the target is in disabled state (EN = 0 in the I3C_CFGR register), the software must initialize as follows:

- Set the I3C peripheral as target: write and clear CRINIT = 0 in the I3C_CFGR register
- Set the I3C bus timings via *I3C timing register 1 (I3C_TIMINGR1)*: write AVAL[7:0] for:
  - Bus available condition time (t_{AVAL}) for IBI or controller-role request
  - Bus idle condition time (t_{IDLE}) for hot-join request
  - $t_{NEWCRLock}$ for controller-role hand-off procedure (after GETACCCR CCC)
- Configure target-initiated requests: write *I3C own device characteristics register (I3C_DEVR0)* with
  - IBIEN: in-band interrupt (also known as IBI) request enable/disable
  - CREN: controller-role request enable/disable
  - HJEN: hot-join request enable/disable
- Initialize target characteristics and capabilities: write
  - *I3C bus characteristics register (I3C_BCR)*
  - *I3C device characteristics register (I3C_DCR)*
  - *I3C maximum read length register (I3C_MAXRLR)*
  - *I3C maximum write length register (I3C_MAXWLR)*
  - *I3C get capability register (I3C_GETCAPR)*
  - *I3C controller-role capability register (I3C_CRCAPR)*
  - *I3C get max data speed register (I3C_GETMXDSR)*
  - *I3C extended provisioned ID register (I3C_EPIDR)*
- Configure the execution mode of a transfer: *I3C configuration register (I3C_CFGR)*, with the following fields:
  - TXDMAEN, RXDMAEN: DMA mode enable/disable for, respectively, TX-FIFO and RX-FIFO
  - TXTHRES, RXTHRES: respectively TX-FIFO and RX-FIFO byte/world threshold
  - Disable S-FIFO: SMODE = 0 (default/reset value)
- Configure interrupt generation or polling mode from any event: *I3C interrupt enable register (I3C_IER)*

Then, the software can enable the I3C peripheral (write and set EN = 1).

Note: The software can write once all the fields of the I3C_CFGR register while enabling it.

Receiving a (broadcast CCC, direct read/write CCC or private read/write) message from the controller

When the target is in idle state (EN = 1 in the I3C_CFGR register), the target is ready to receive a communication message on the I3C bus from the controller, and is ready to switch to the active state.

Typically, first, the active target is receiving a broadcast ENTDAA CCC, possibly after optional received broadcast ENEC/DISEC CCC(s), and is then assigned a dynamic address. The event DAUPF in the I3C_EVR register is raised to 1, the related interrupt is generated if enabled, and the target goes back to idle state.

After that, the idle target is ready to receive any other broadcast CCC message, or direct read/write CCC, or private read/write message from the controller.

For more information about the execution of controller-initiated I3C bus transfers and its related programming as a target, including the updated I3C registers and fields, refer to figures in *Section 35.9*:

- *Figure 411: I3C CCC messages, as target*
- *Figure 412: I3C broadcast ENTDAA CCC, as target*
- *Figure 413: I3C broadcast DEFTGTS CCC, as target*
- *Figure 414: I3C broadcast DEFGRPA CCC, as target*
- *Figure 416: I3C private read/write messages, as target*

Figure 407 does not include the FIFOs management (TX-FIFO, RX-FIFO). This is detailed in *Section 35.10*.

Read the message status register

For each received and completed message without transfer error, the hardware reports the exchanged transfer on the I3C bus by updating the *I3C status register (I3C_SR)*, which can be read by the software after being notified by the corresponding flag in the *I3C event register (I3C_EVR)*, or by the corresponding interrupt if enabled in the *I3C interrupt enable register (I3C_IER)*.

I3C status register (I3C_SR) must be read by the software after the following messages:

- A private read: to get the number of exchanged data bytes, as the controller can have ended the transfer earlier than expected by the target (if XDCNT[15:0] in the I3C_SR register is lower than TGTTDCNT[15:0] in the I3C_TGTTDR register). If so, software must flush the TX-FIFO (write TXFLUSH = 1 in the I3C_CFGR register).
- A DEFTGTS CCC or a DEFGRPA CCC: to get the number of received data bytes in the RX-FIFO

Start a (target-initiated) transfer

When the target goes first from disabled to idle state (software writes EN = 1 in the I3C_CFGR register), concurrently to be able to receive a broadcast CCC from the controller, the software can initiate a hot-join request (the software writes MTYPE[3:0] = 1000 in the I3C_CR register) to be eligible to participate to a next ENTDAA CCC, provided it is allowed to do so (HJEN = 1 in the I3C_DEVR0 register).

Once a dynamic address is assigned (DAUPF = 1 in the I3C_EVR register), more generally and possibly concurrently to a frame transfer emitted by the controller, the software can initiate an IBI (in-band interrupt request) to the controller, or a controller-role request by writing the related control word into the I3C_CR register.

For more information about the execution of target-initiated I3C bus transfer, and its related programming as a target, refer to figures in *Section 35.9*:

- *Figure 418: IBI transfer, as controller/target*
- *Figure 419: Hot-join request transfer, as controller/target*
- *Figure 420: Controller-role request transfer, as controller/target*

Updating the configuration of the I3C peripheral, as target

Back in idle state, the software can update the configuration of the I3C target before a next transfer:

- Modify the interrupt/polling mode policy via *I3C interrupt enable register (I3C_IER)*
- Modify following fields of the *I3C configuration register (I3C_CFGR)*:
  - TXTHRES, RXTHRES
  - TXDMAEN, RXDMAEN
- Modify/prepare the *I3C IBI payload data register (I3C_IBIDR)*, if any payload (if BCR2 = 1 in the I3C_BCR register), before initiating an IBI transfer (write *I3C message control register [alternate] (I3C_CR)* with MTYPE[3:0] = 1010)
- Modify/prepare *I3C target transmit configuration register (I3C_TGTTDR)*, to disable or enable the TX-FIFO to be preloaded with a defined number of data bytes to be transmitted, before receiving a private read or a direct CCC read (out of the GETSTATUS CCC) from the controller

The registers usage vs. the I3C peripheral role as target is summarized in *Section 35.8.1*.

The static/dynamic registers fields usage, when acting as target, is summarized in *Table 280*.

35.8 I3C registers and programming

35.8.1 I3C register set, as controller/target

Table 278 lists the registers and their usage versus the I3C peripheral role.

Table 278. I3C register usage

Register	Used as controller	Used as target
I3C_CR	X	X
I3C_CFGR	X	X
I3C_RDR	X	X
I3C_RDWR	X	X
I3C_TDR	X	X
I3C_TDWR	X	X
I3C_IBIDR	X	X
I3C_TGTTDR	-	X
I3C_SR	X	X
I3C_SER	X	X
I3C_RMR	X	X
I3C_EVR	X	X
I3C_IER	X	X
I3C_CEVR	X	X
I3C_DEVR0	X	X

Table 278. I3C register usage (continued)

Register	Used as controller	Used as target
I3C_DEVRx (x = 1 to 4)	X	-
I3C_MAXRLR	-	X
I3C_MAXWLR	-	X
I3C_TIMINGR0	X	-
I3C_TIMINGR1	X	X
I3C_TIMINGR2	X	-
I3C_BCR	-	X
I3C_DCR	-	X
I3C_GETCAPR	-	X
I3C_CRCAPR	-	X
I3C_GETMXDSR	-	X
I3C_EPIDR	-	X

35.8.2 I3C registers and fields use versus peripheral state, as controller

When the I3C peripheral acts as controller, *Table 279* lists the registers and their usage versus the controller state (disabled, idle, and active).

Table 279. I3C registers/fields usage versus controller state

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CR	X	-	-	X

Table 279. I3C registers/fields usage versus controller state (continued)

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CFGR	X	CRINIT HKSDAEN EN(1)	Write: any used field except {CRINIT, HKSDAEN}, namely {CDMAEN SDMAEN TXDMAEN RXDMAEN TMODE SMODE TXTHRES RXTHRES HJACK EXITPTRN RSTPTRN NOARBH}(2) CFLUSH SFLUSH TXFLUSH RXFLUSH TSFSET	-
I3C_RDR	X	-	-	Read
I3C_RDWR	X	-	-	Read
I3C_TDR	X	-	-	Write
I3C_TDWR	X	-	-	Write
I3C_IBIDR	X	-	-	Read
I3C_TGTTDR	-			
I3C_SR	X	-	-	Read
I3C_SER	X	-	Read	-
I3C_RMR	X	-	-	Read RADD[6:0] IBIRDCNT[2:0]

Table 279. I3C registers/fields usage versus controller state (continued)

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_EVR	X	-	-	Read (controller-role fields): HJF CRF IBIF FCF ERRF RXTGTENDF RXFNEF TXFNFF SFNEF CFNFF RXLASTF TXLASTF TXFEF CFEF
I3C_IER	X	Write xIE with x = HJ, CR, IBI, FC, ERR, RXTGTEND, RXFNE, TXFNF, SFNE, CFNF(2)		
I3C_CEVR	X	-	-	Write (controller-role fields, refer to I3C_EVR)
I3C_DEVR0	X	Write DA[6:0](2)		-
I3C_DEVRx x = 1..4	X	Write SUSP, IBIDEN, IBIACK, CRACK, DA[6:0](2)		-
I3C_MAXRLR	-			
I3C_MAXWLR	-			
I3C_TIMINGR0	X	X	-	-
I3C_TIMINGR1	X	X	-	-
I3C_TIMINGR2	X	Write(2)		
I3C_BCR	-			
I3C_DCR	-			
I3C_GETCAPR	-			
I3C_CRCAPR	-			
I3C_GETMXDSR	-			
I3C_EPIDR	-			

1. Bit EN in the I3C_CFGR register is written and set in disabled state (when the same bit is 0). This field can be also written and de-asserted in idle state.
2. These fields are typically written and initialized in disabled state during bus configuration. They are not write-protected when EN = 0, and can be also written and updated in other state(s).

35.8.3 I3C registers and fields usage versus peripheral state, as target

When the I3C peripheral acts as target, *Table 280* lists the registers and their usage versus the I3C target state (disabled, idle, and active).

Table 280. I3C registers/fields usage versus target state

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CR	X	-	MTYPE[3:0] DCNT[2:0]	-
I3C_CFGR	X	CRINIT EN(1)	Write: any used field except CRINIT, namely: {TXDMAEN RXDMAEN TXTHRES RXTHRES}(2) TXFLUSH RXFLUSH	-
I3C_RDR	X	-	-	Read
I3C_RDWR	X	-	-	Read
I3C_TDR	X	-	-	Write
I3C_TDWR	X	-	-	Write
I3C_IBIDR	X	-	Write	-
I3C_TGTTDR	X	-	Write	-
I3C_SR	X	-	-	Read DIR, XDCNT[15:0]
I3C_SER	X	-	Read DOVR, STALL, PERR, CODERR[3:0]	-
I3C_RMR	X	-	-	Read RCODE[7:0]

Table 280. I3C registers/fields usage versus target state (continued)

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_EVR	X	-	-	Read (target-role fields): GRPF DEFF INTUPDF ASUPDF RSTF MRLUPDF MWLUPDF DAUPDF STAF GETF WKPF CRUPDF IBIENDF ERRF FCF RXFNEF TXFNFF TXLASTF TXFEF
I3C_IER	X	Write xIE with x = GRP, DEF, INTUPD, ASUPD, RST, MRLUPD, MWLUPD, DAUPD, STA, GET, WKP, CRUPD, IBIEND, ERR, FC, RXFNE, TXFNF(2)		
I3C_CEVR	X	-	-	Write (target-role fields, refer to I3C_EVR)
I3C_DEVR0	X	HJEN CREN IBIEN	Read RSTVAL, RSTACT[1:0], and AS[1:0]	-
I3C_DEVRx x = 1..4	-			
I3C_MAXRLR	X	X	-	-
I3C_MAXWLR	X	X	-	-
I3C_TIMINGR0	-			
I3C_TIMINGR1	X	AVAL[7:0]	-	-
I3C_TIMINGR2	-			
I3C_BCR	X	X	-	-
I3C_DCR	X	X	-	-
I3C_GETCAPR	X	X	-	-
I3C_CRCAPR	X	X	-	-

Table 280. I3C registers/fields usage versus target state (continued)

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_GETMXDSR	X	X	-	-
I3C_EPIDR	X	X	-	-

1. Bit EN in the I3C_CFGR register is written and set in disabled state (when the same bit is 0). This field can be also written and de-asserted in idle state.
2. These fields are typically written and initialized in disabled state during I3C bus configuration. They are not write-protected when EN = 0.

35.9 I3C bus transfers and programming

35.9.1 I3C command set (CCCs), as controller/target

The list of the supported I3C command set (for example, list of CCCs, common command codes) and the overview of how they are handled by the I3C peripheral acting as controller or target, is specified in *Table 281*.

Table 281. List of supported I3C CCCs, as controller/target

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
Broadcast CCCs							
ENEC	0x00	Write	No defining/sub-command byte	With one data byte (enable target events byte)	X	X, INTUPDF	Update and enable I3C_DEVR0: HJEN, CREN, IBIEN if any
DISEC	0x01			With one data byte (disable target events byte)	X	X, INTUPDF	Update and disable I3C_DEVR0: HJEN, CREN, IBIEN if any
ENTASx x = 0...3	0x02 ... 0x05			No data byte	X	X, ASUPDF	Update I3C_DEVR0.AS[1:0]
RSTDAA	0x06			-	X	X, DAUPDF	Clear I3C_DEVR0.DAVAL = 0
ENTDAA	0x07			-	X	X, DAUPDF	Update I3C_DEVR0: DA[6:0] and set DAVAL = 1
DEFTGTS	0x08			With [1+ 4x (1+ number_of_targets)] x data bytes	X	X, DEFF	Update I3C_RDR/ I3C_RDWR. Refer to *Figure 413*.
SETMWL	0x09			With two data byte	X	X, MWLUPDF	Update I3C_MAXWLR
SETMRL	0x0A			With 2 or 3 data bytes	X	X, MRLUPDF	Update I3C_MAXRLR
ENTTM	0x0B			With one data byte	X	-	
SETXTIME	0x28		With sub-command byte	Without or with one or more data bytes	X		
SETAASA	0x29		No defining/sub-command byte	No data byte	X		
RSTACT	0x2A		With defining byte (0x00, 0x01 or 0x02)		X	X, RSTF after detected reset pattern	Update I3C_DEVR0: RSTACT[1:0] and set RSTVAL = 1
DEFGRPA	0x2B		No defining/sub-command byte	With several data bytes	X	X, GRPF	Update I3C_RDR/ RDWR. Refer to *Figure 414*.
RSTGRPA	0x2C			No data byte	X	-	-

Table 281. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
Direct CCCs							Action if ACK (if I3C target Address = I3C_DEVR0.DA[6:0] and I3C_DEVR0.DAVAL = 1) (else NACK)
ENEC	0x80	Write	No defining/sub-command byte	With one data byte (enable target events byte)	X	X, INTUPDF	Update and enable I3C_DEVR0: HJEN, CREN, IBIEN if any
DISEC	0x81			With one data byte (disable target events byte)	X	X, INTUPDF	Update and disable I3C_DEVR0: HJEN, CREN, IBIEN if any
ENTASx x = 0...3	0x82...0x85			No data byte	X	X, ASUPDF	Update I3C_DEVR0.AS[1:0]
SETDASA	0x87			No data byte	X	-	-
SETNEWDA	0x88			With one data byte	X	X, DAUPDF	Update I3C_DEVR0: DA[6:0] (and set DAVAL = 1)
SETMWL	0x89			With two data bytes	X	X, MWLUPDF	Update I3C_MAXWLR
SETMRL	0x8A			With two or three data bytes	X	X, MRLUPDF	Update I3C_MAXRLR
GETMWL	0x8B	Read	No defining/sub-command byte	With two data bytes	X	X, GETF	Return data bytes from I3C_MAXWLR[15:0]. Refer to *Section 35.16.19*.
GETMRL	0x8C			With two or three data bytes	X	X, GETF	Return data bytes from I3C_MAXRLR[15:0] and if I3C_BCR.BCR2 = 1 return third byte from I3C_MAXRLR.IBIP[2:0]. Refer to *Section 35.16.18*.

Table 281. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
GETPID	0x8D	Read	No defining/sub-command byte	With six data bytes	X	X, GETF	Return data bytes from I3C_EPIDR. Refer to *Section 35.16.28*.
GETBCR	0x8E			With one data byte	X	X, GETF	Return data byte from I3C_BCR[7:0]. Refer to *Section 35.16.23*.
GETDCR	0x8F				X	X, GETF	Return I3C_DCR[7:0]. Refer to *Section 35.16.24*.
GETSTATUS	0x90		With or without defining byte (TGTSTAT, PRECR)	With two data bytes (format 1 or format 2 with PRECR)	X	X, STAF if format 1 X, GETF if format 2	Return 2 data bytes, as detailed in *Section 35.9.9*.
GETACCCR	0x91		No defining/sub-command byte	With one data byte	X	X, CRUPDF	Return data byte from I3C_DEVR0.DA[6:0] with parity bit
GETMXDS	0x94		With or without defining byte (WRRDTURN, CRHDLY)	With two data bytes (format 1) or 5 data bytes (format 2 or format 3 with WRRDTURN) or 1 data byte (format 3 with CRHDLY)	X	X, GETF	Return data byte(s) from I3C_GETMXDSR. Refer to *Section 35.16.27*.
GETCAPS	0x95		With or without defining byte (TGTSTAT, CRCAPS)	With 3 data bytes (format 1 or format 2 with TGTSTAT) or two data bytes (format 2 with CRCAPS)	X	X, GETF	Return 3 GETCAPx data bytes from I3C_GETCAPR (refer to *Section 35.16.25*) or Return 2 CRCAPx data bytes from I3C_CRCAPR (refer to *Section 35.16.26*)
D2DXFER	0x97	Write	With defining byte	With defining byte	X	-	-
SETXTIME	0x98		With sub-command byte	With sub-command byte	X		
GETXTIME	0x99	Read	No defining/sub-command byte	No defining/sub-command byte	X		

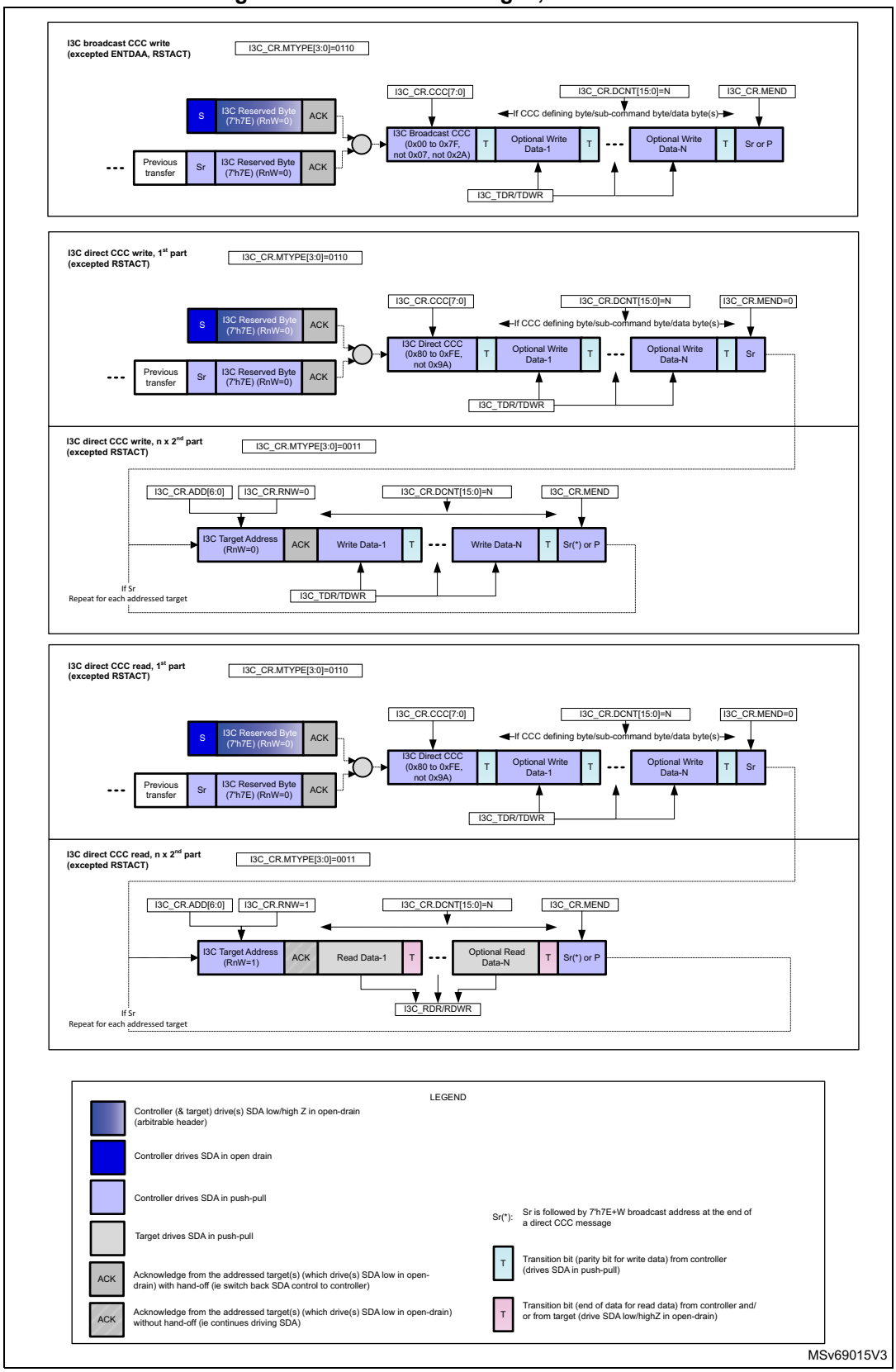
Table 281. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
RSTACT	0x9A	Read/ Write	With defining byte (0x00, 0x01, or 0x02)	With defining byte (0x00, 0x01, or 0x02)	X	X, RSTF if detected reset pattern	Read: return data byte from RSTACT[1:0] in the I3C_DEVR0 register. Write: update I3C_DEVR0: RSTACT[1:0] and set RSTVAL = 1
SETGRPA	0x9B	Write	No defining/sub-command byte	No defining/sub-command byte	X	-	-
RSTGRPA	0x9C				X		

35.9.2 I3C broadcast/direct CCC transfer (except ENTDAA, RSTACT), as controller

Figure 408 illustrates I3C broadcast CCC write transfer (except ENTDAA, RSTACT), and direct CCC read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

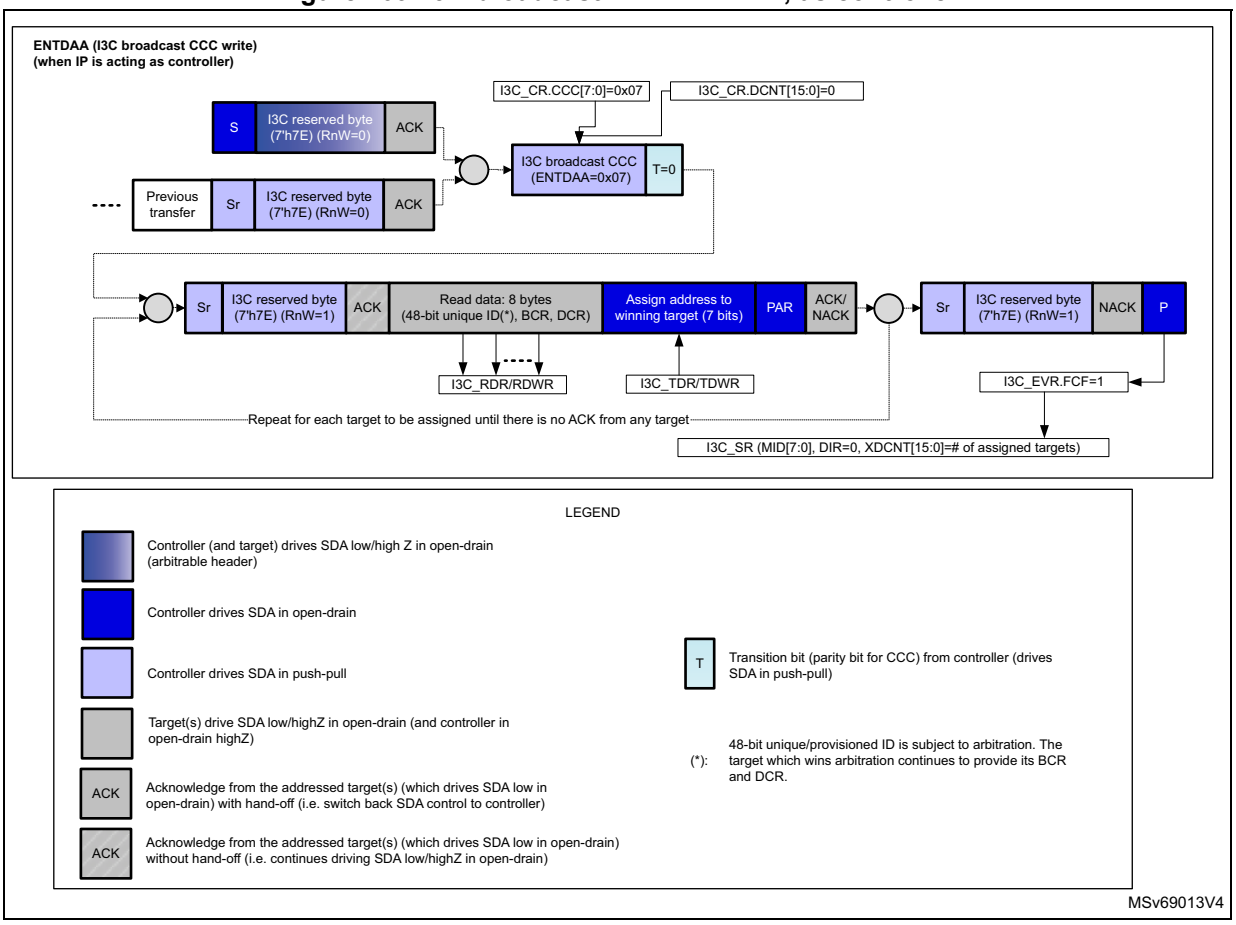
Figure 408. I3C CCC messages, as controller



35.9.3 I3C broadcast ENTDAA CCC transfer, as controller

Figure 409 illustrates I3C broadcast ENTDAA CCC, as communicated on the I3C bus, and as programmed when acting as controller.

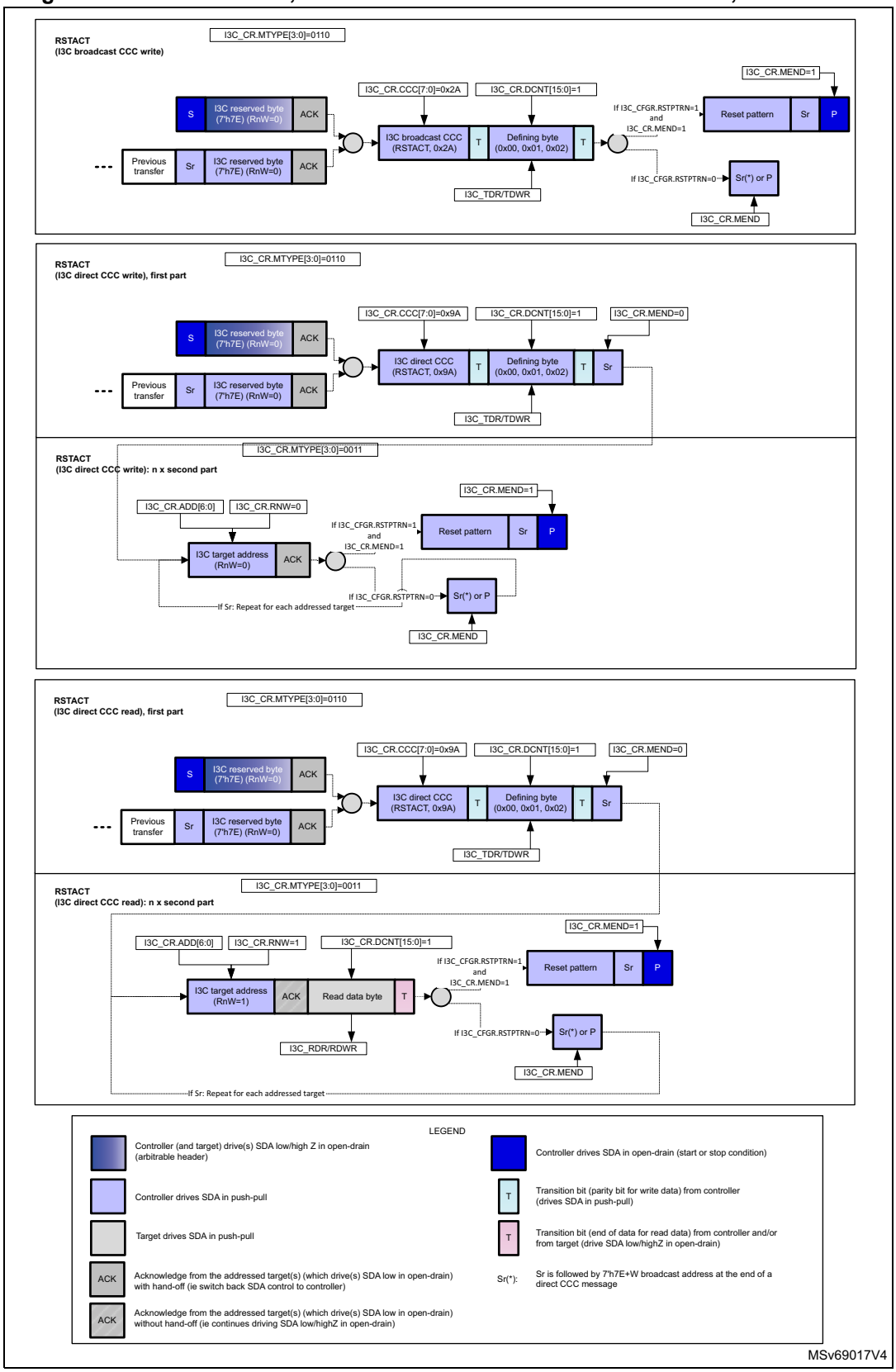
Figure 409. I3C broadcast ENTDAA CCC, as controller



35.9.4 I3C broadcast/direct RSTACT CCC transfer, as controller

Figure 410 illustrates I3C broadcast (write), direct write and read RSTACT CCC, as communicated on the I3C bus, and as programmed when acting as controller.

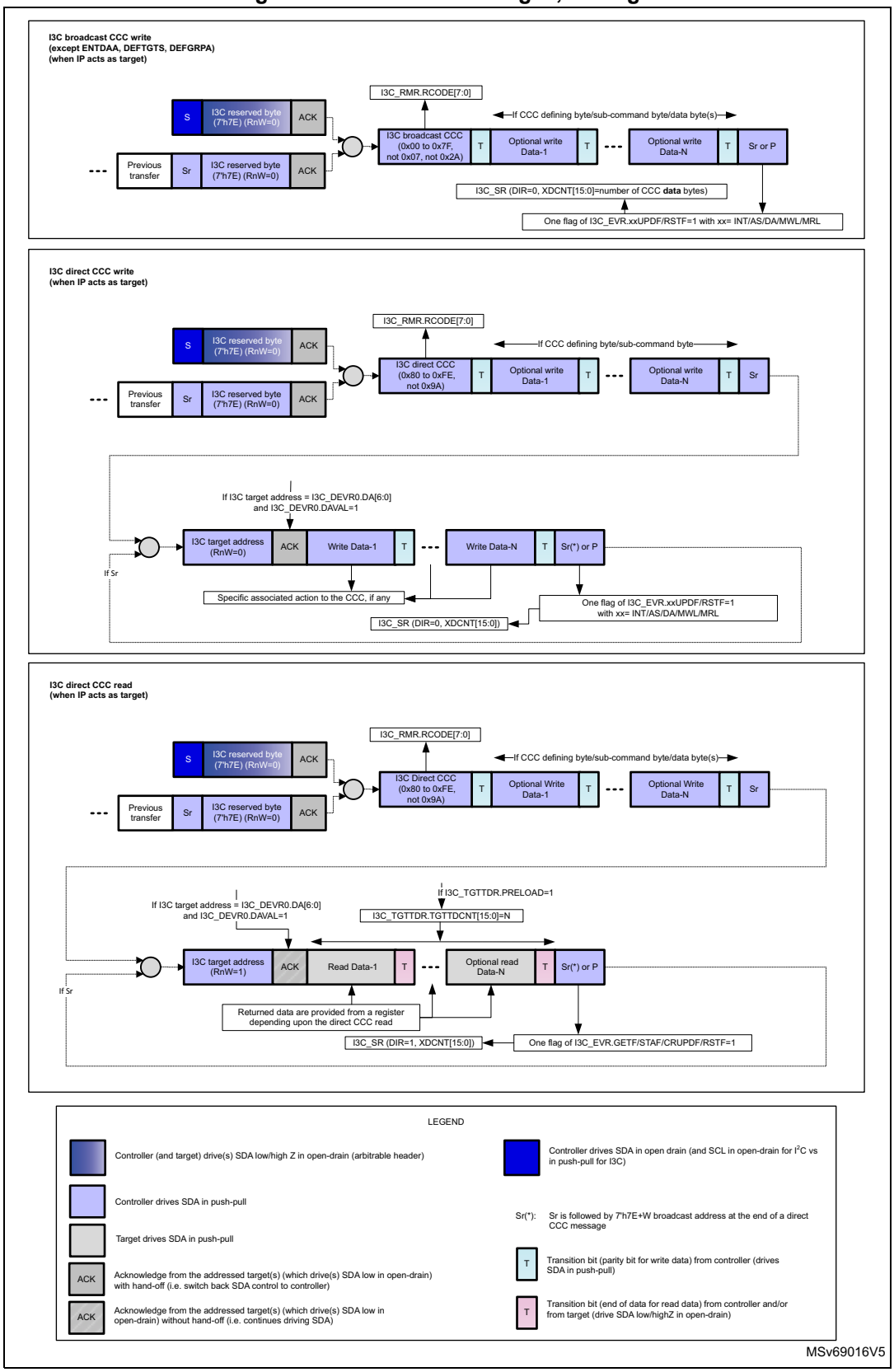
Figure 410. I3C broadcast, direct read and direct write RSTACT CCC, as controller



35.9.5 I3C broadcast/direct CCC transfer (except ENTDAA, DEFTGTS, DEFGRPA), as target

Figure 411 illustrates I3C broadcast CCC write transfer (except ENTDAA, DEFTGTS, DEFGRPA), direct CCC read/write transfer, as communicated on the I3C bus, and as programmed when acting as target.

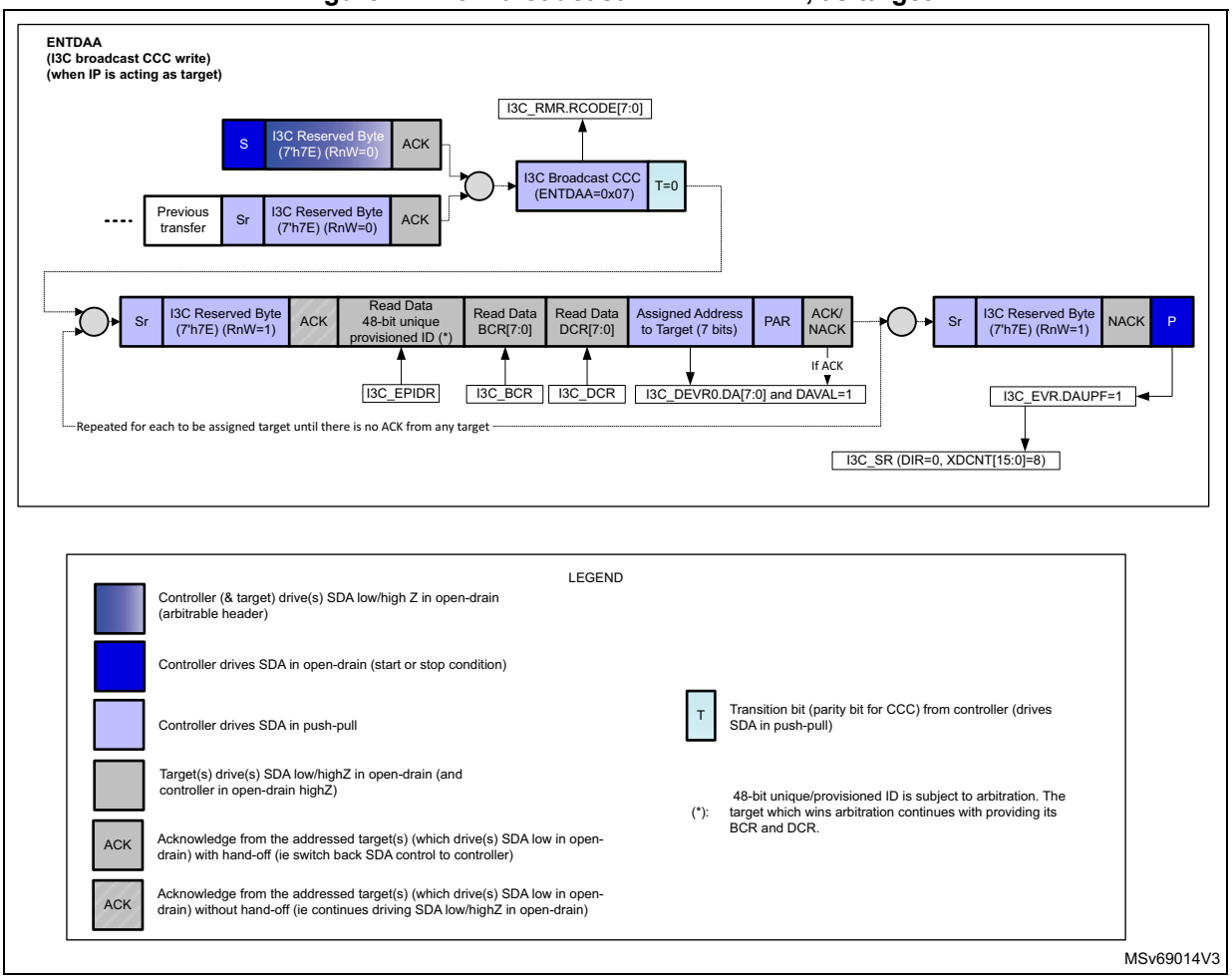
Figure 411. I3C CCC messages, as target



35.9.6 I3C broadcast ENTDAA CCC transfer, as target

Figure 412 illustrates I3C broadcast ENTDAA CCC, as communicated on the I3C bus, and as programmed when acting as target.

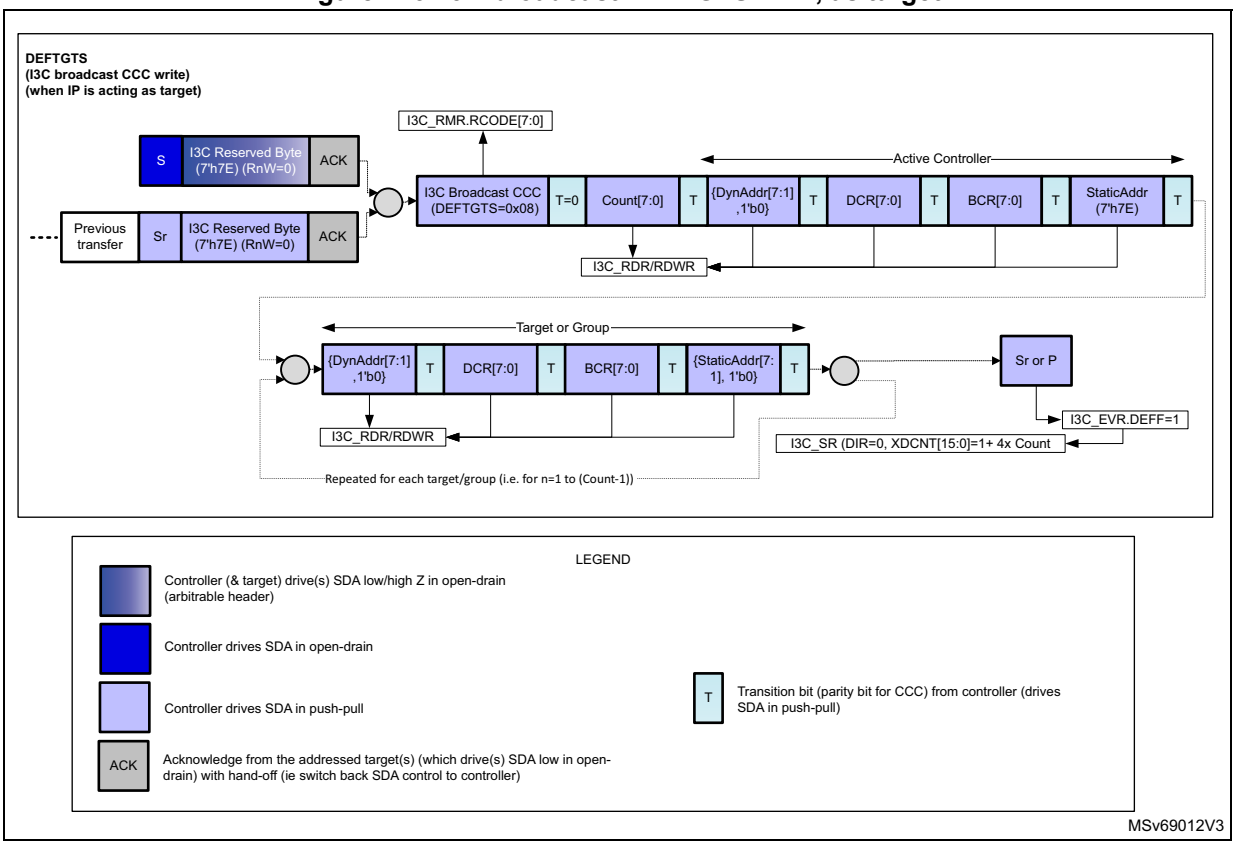
Figure 412. I3C broadcast ENTDAA CCC, as target



35.9.7 I3C broadcast DEFTGTS CCC transfer, as target

Figure 413 illustrates I3C broadcast DEFTGTS CCC, as communicated on the I3C bus, and as programmed when acting as target.

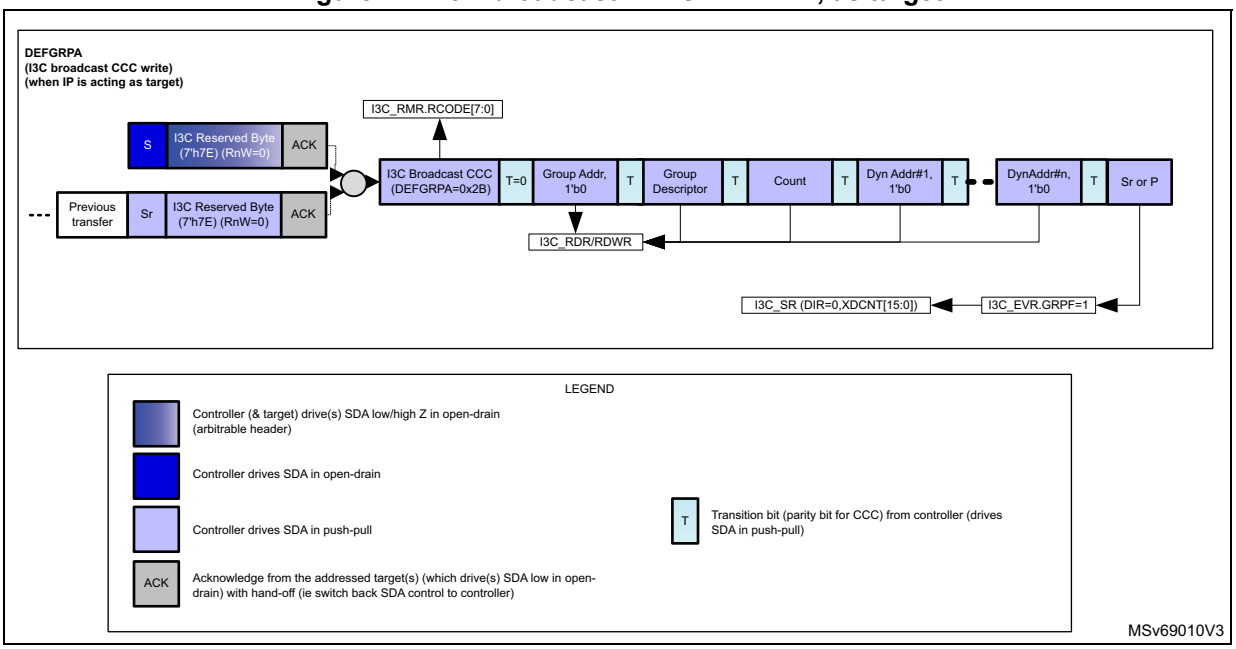
Figure 413. I3C broadcast DEFTGTS CCC, as target



35.9.8 I3C broadcast DEFGRPA CCC transfer, as target

Figure 414 illustrates I3C broadcast DEFGRPA CCC, as communicated on the I3C bus, and as programmed when acting as target.

Figure 414. I3C broadcast DEFGRPA CCC, as target



35.9.9 I3C direct GETSTATUS CCC response, as target

When the I3C acts as target, the hardware returns two data bytes on reception of GETSTATUS CCC, with format 1 (without defining byte or with defining byte TGTSTAT = 0x00), or format 2 (with defining byte PRECR = 0x91).

The returned 2-byte STATUS[15:0] with format 1 on the I3C bus is then as follows:

- STATUS[15:14] = 00 (unused)
- STATUS[13] = 1 if a missed start was detected since the former GETSTATUS CCC, else 0
- STATUS[12] = 1 if an overrun/underrun error was detected since the former GETSTATUS CCC, else 0
- STATUS[11] = 1 if an SCL stable for more than 125 µs was detected during an SDR read since the former GETSTATUS CCC, else 0
- STATUS[10:8] = 000 to 110: encoded value x = 0 to 6, corresponding to a target error TEx if a protocol error was detected since the former GETSTATUS CCC (if STATUS[5] = 1), else 000
- STATUS[7:6] = 00 (ready to prepare for hand-off procedure)
- STATUS[5] = 1 if a protocol error was detected since the former GETSTATUS CCC, else 0
- STATUS[4] = 0 (reserved)
- STATUS[3:1] = 000 (unused)
- STATUS[0] = 1 if there is a pending interrupt (if an IBI is configured in the I3C_CR register, and IBIEN = 1 and DAVAL = 1 in the I3C_DEVR0.register, and the IBI is not yet acknowledged by the controller neither disabled via DISEC), else 0

The returned 2-byte STATUS[15:0] with format 2 on the I3C bus is then as follows:

- STATUS[15:8] = 0000 0000 (unused)
- STATUS[7:2] = 00000 (unused)
- STATUS[1] = 1 if a received DEFTGTS or a received DEFGRPA CCC is still under software processing, and the related event is not yet cleared by software (DEFF = 1 or GRPF = 1 in the I3C_EVR register); the controller must wait before issuing a GETACCR CCC (else it is not acknowledged)
- STATUS[0] = 1 if a DEFTGTS or DEFGRPA CCC may have been missed. This bit is asserted if a missed start is detected (WKPF = 1 in the I3C_EVR register), de-asserted if DEFF = 1 or GRPF = 1 in the I3C_EVR register.

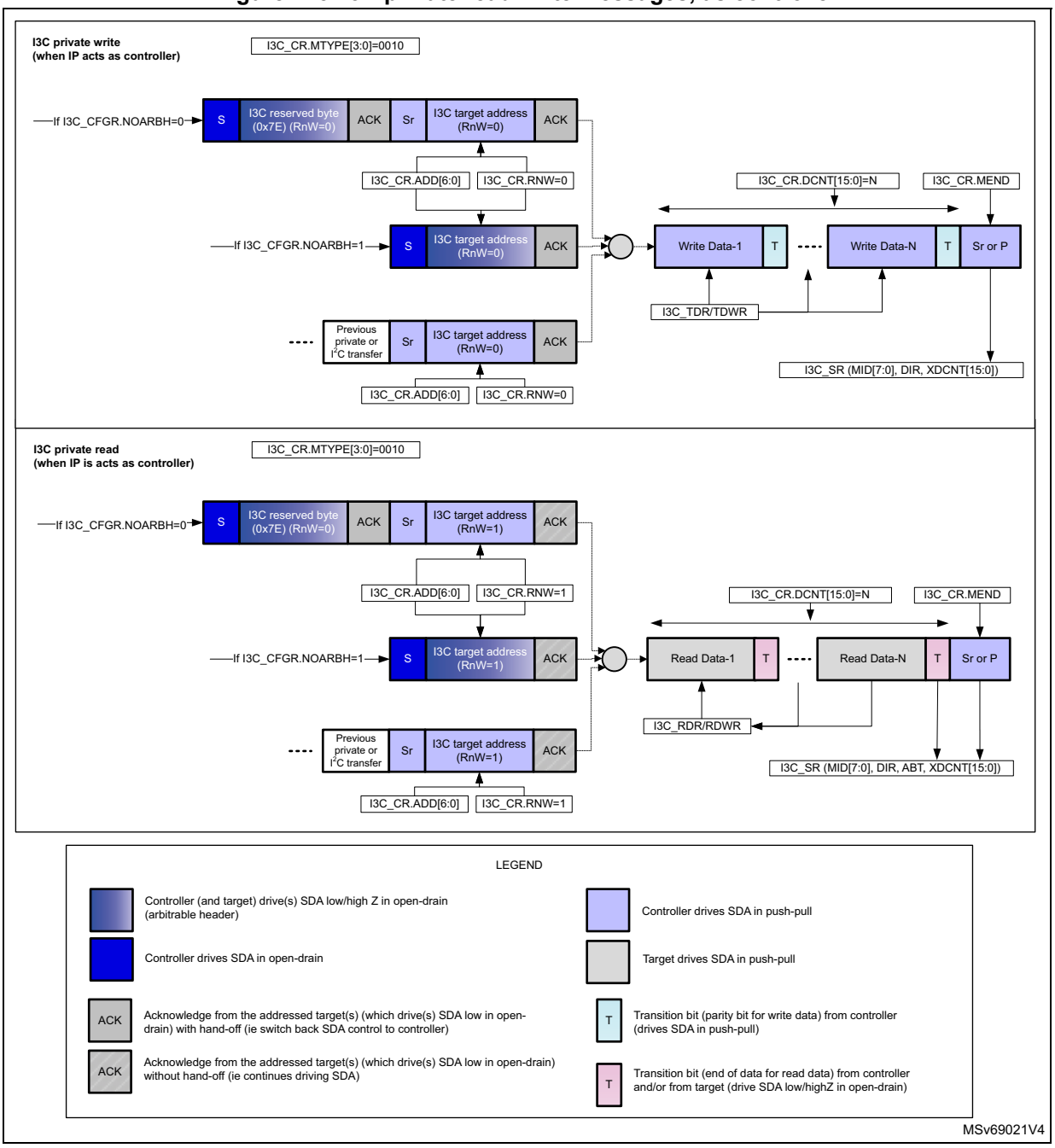
Completion of a GETSTATUS CCC of format 1 is reported by STAF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled (if STAIE = 1 in the I3C_IER register).

Completion of a GETSTATUS CCC of format 2 is reported by GETF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled (if GETIE = 1 in the I3C_IER register).

35.9.10 I3C private read/write transfer, as controller

Figure 415 illustrates private read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

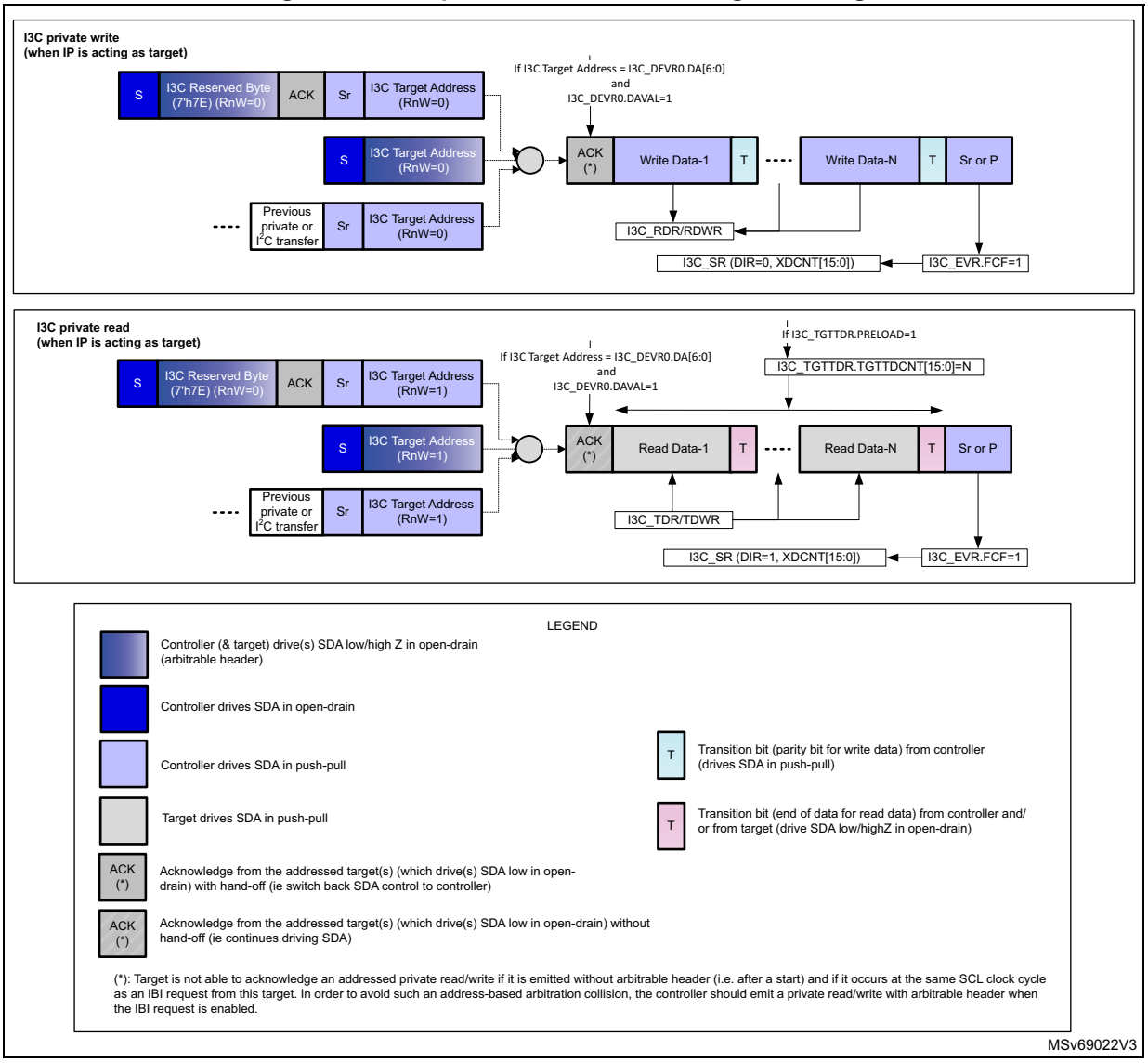
Figure 415. I3C private read/write messages, as controller



35.9.11 I3C private read/write transfer, as target

Figure 416 illustrates I3C private read/write transfer, as communicated on the I3C bus, and as programmed when acting as target.

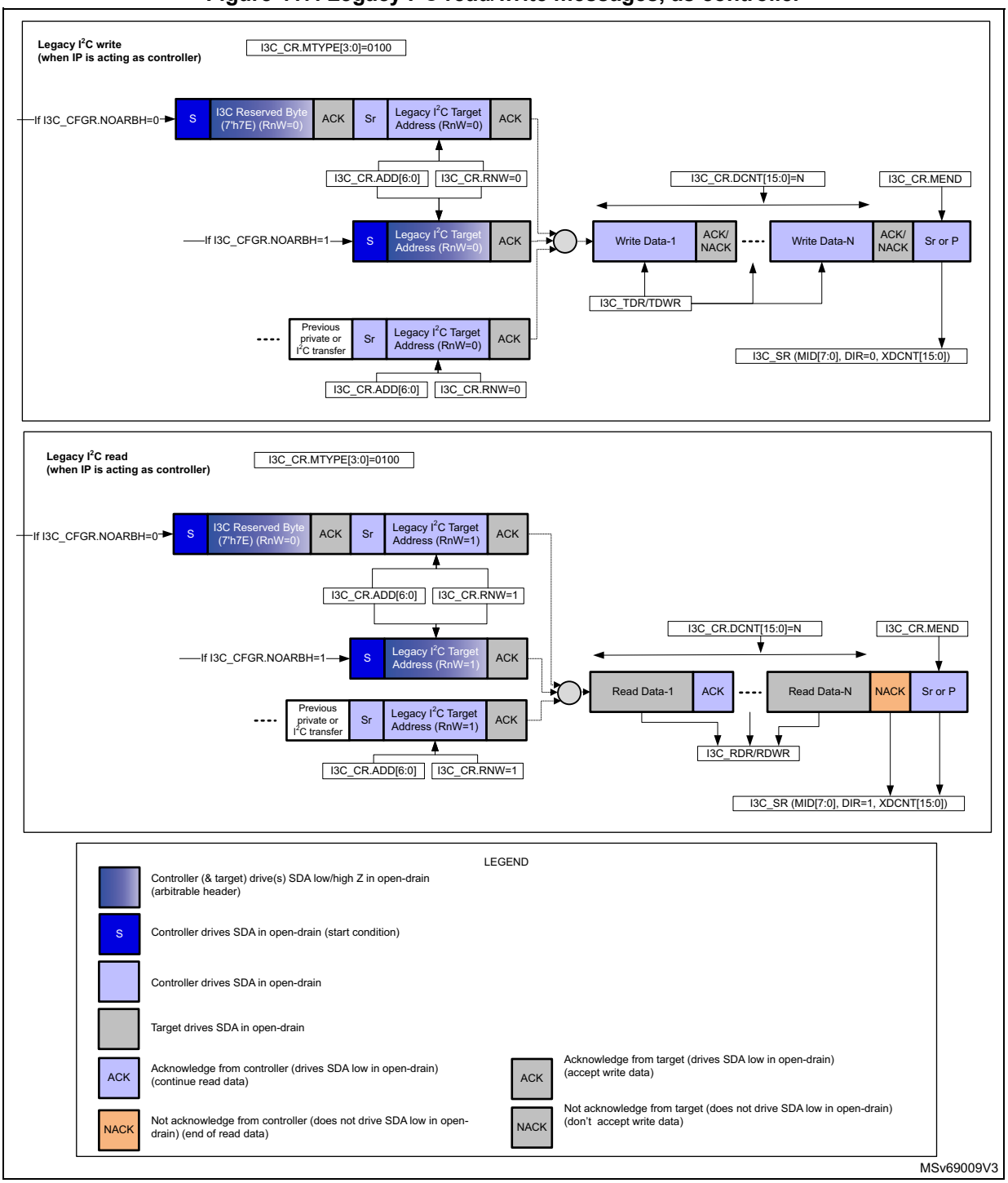
Figure 416. I3C private read/write messages, as target



35.9.12 Legacy I²C read/write transfer, as controller

Figure 417 illustrates legacy I²C read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

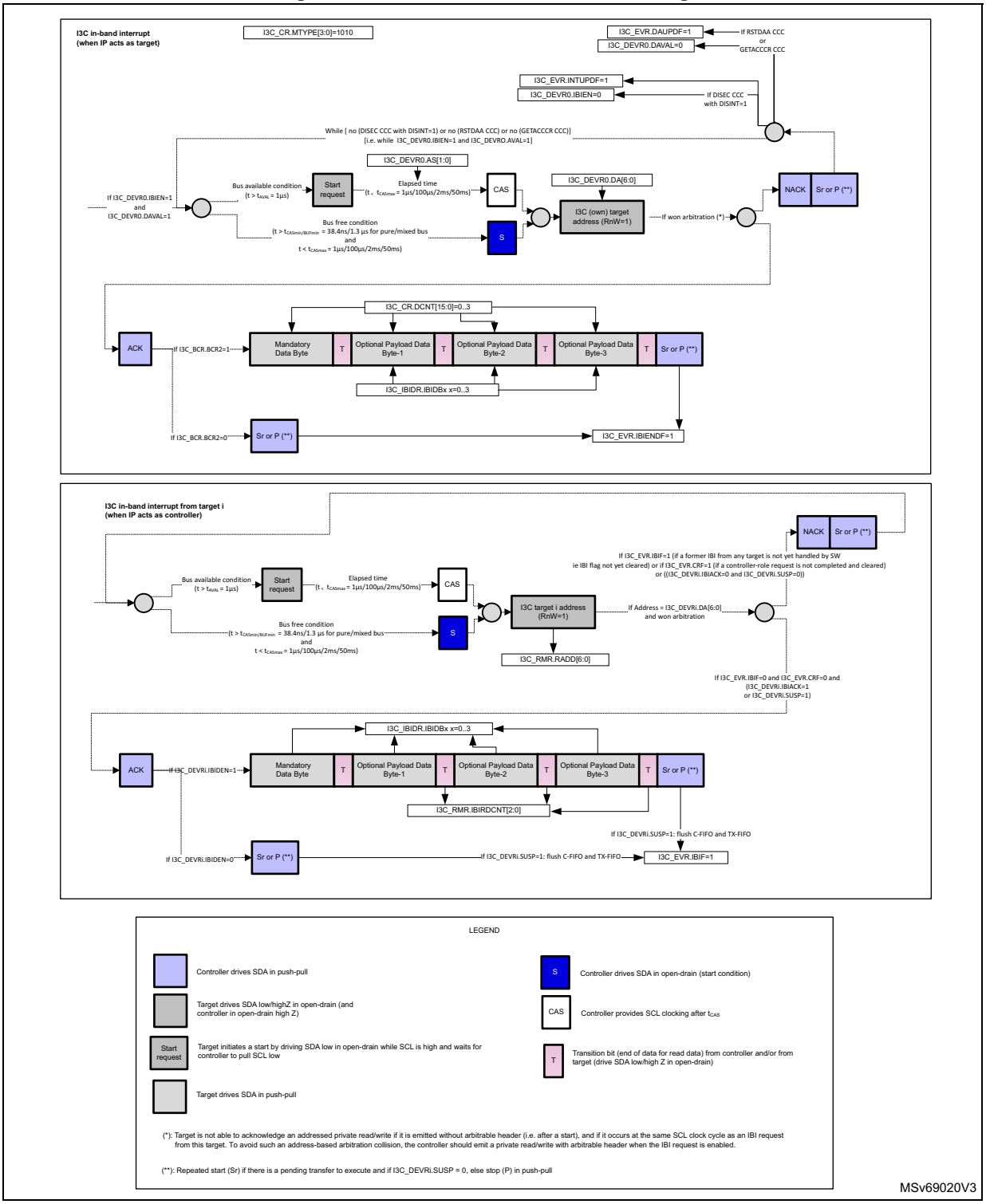
Figure 417. Legacy I²C read/write messages, as controller



35.9.13 I3C IBI transfer, as controller/target

Figure 418 illustrates IBI (in-band interrupt) transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

Figure 418. IBI transfer, as controller/target

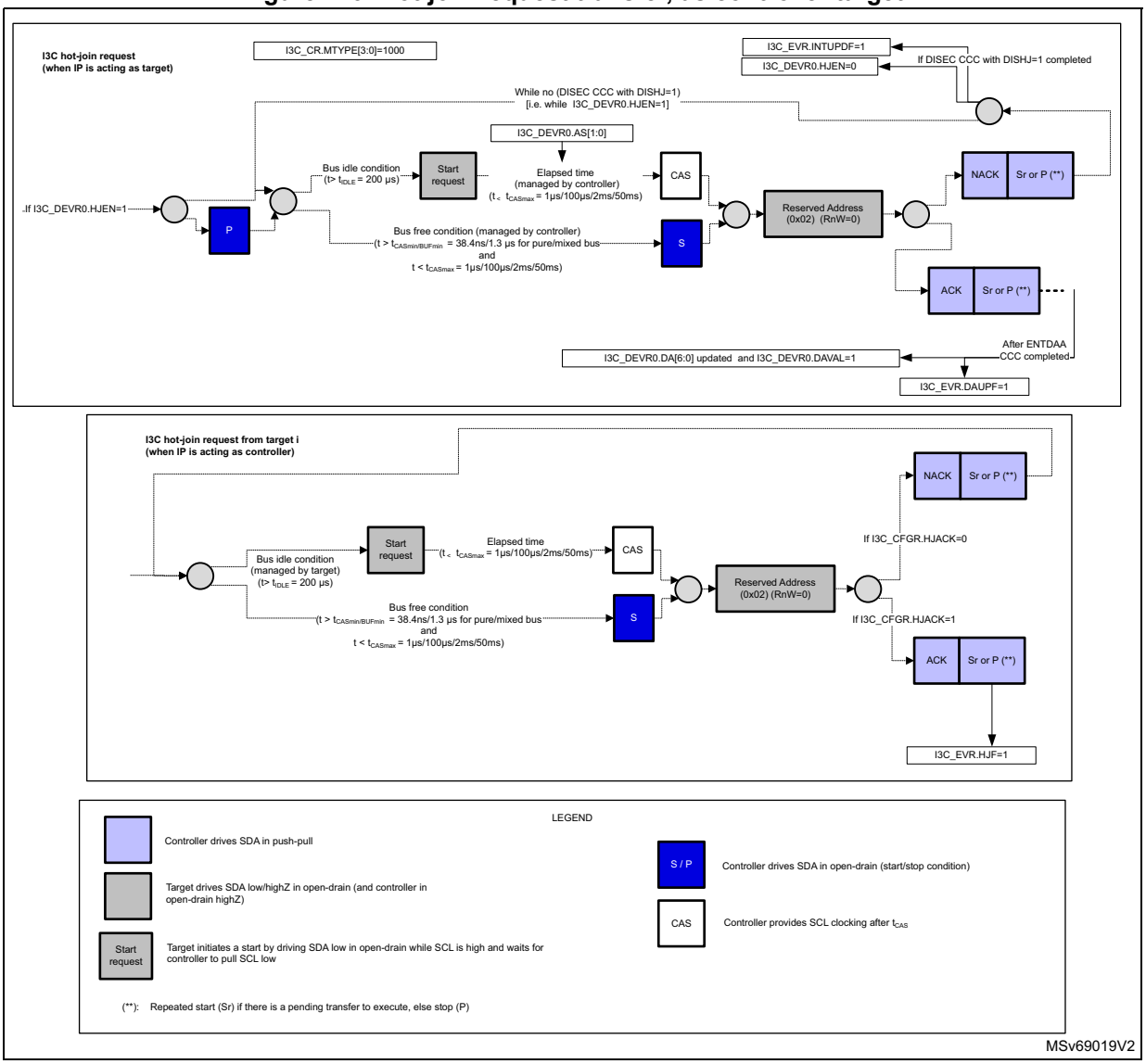


When the I3C peripheral acts as controller, the I3C_IBIDR register is used to receive the IBI data payload. Consequently, the IBI request from the target must not exceed a 4-byte data payload. If there is more information to be exchanged in the context of this in-band interrupt, the controller software must issue a private read.

35.9.14 I3C hot-join request transfer, as controller/target

Figure 419 illustrates hot-join request transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

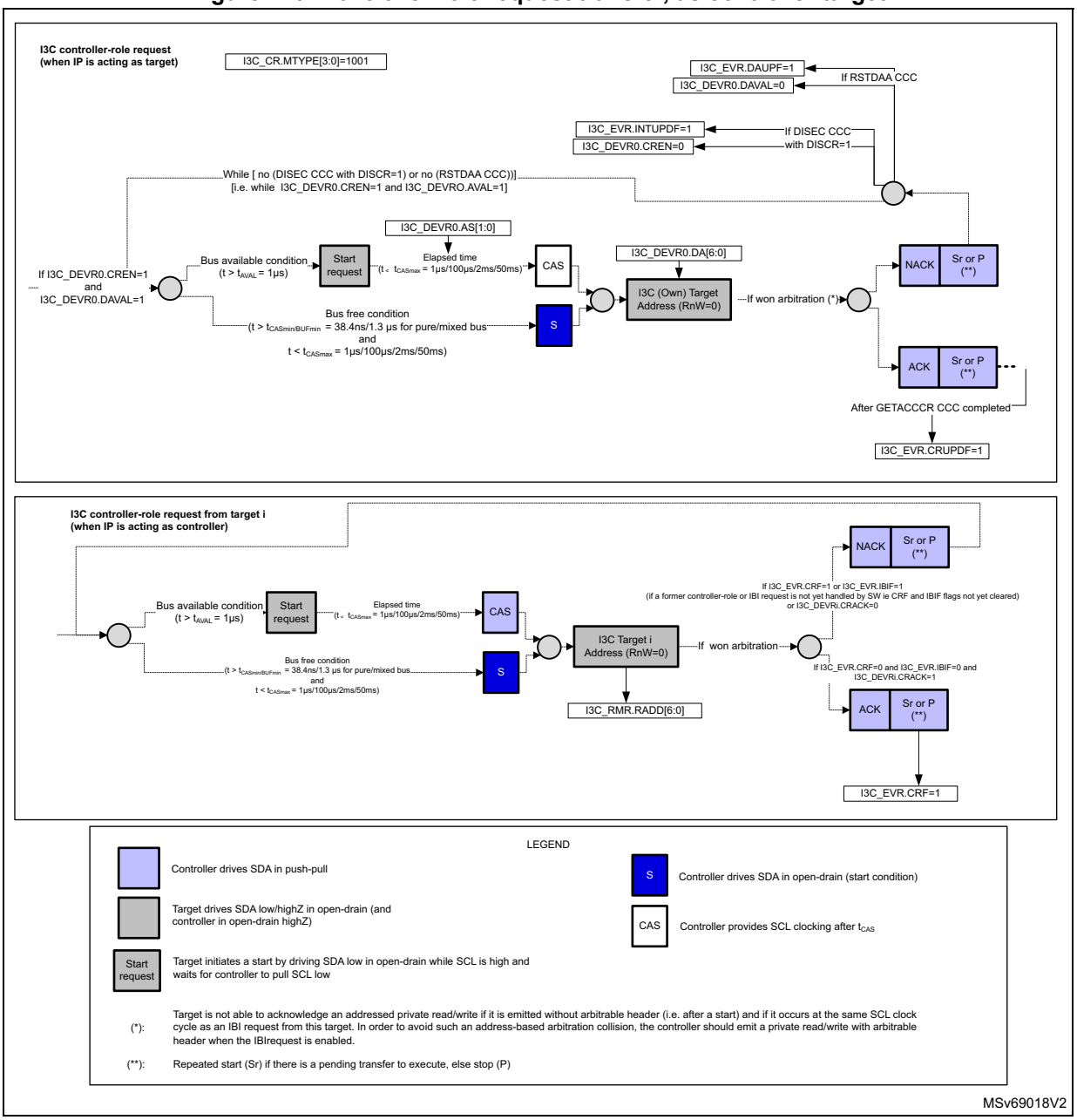
Figure 419. Hot-join request transfer, as controller/target



35.9.15 I3C controller-role request transfer, as controller/target

Figure 420 illustrates controller-role request transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

Figure 420. Controller-role request transfer, as controller/target



35.10 I3C FIFOs management, as controller

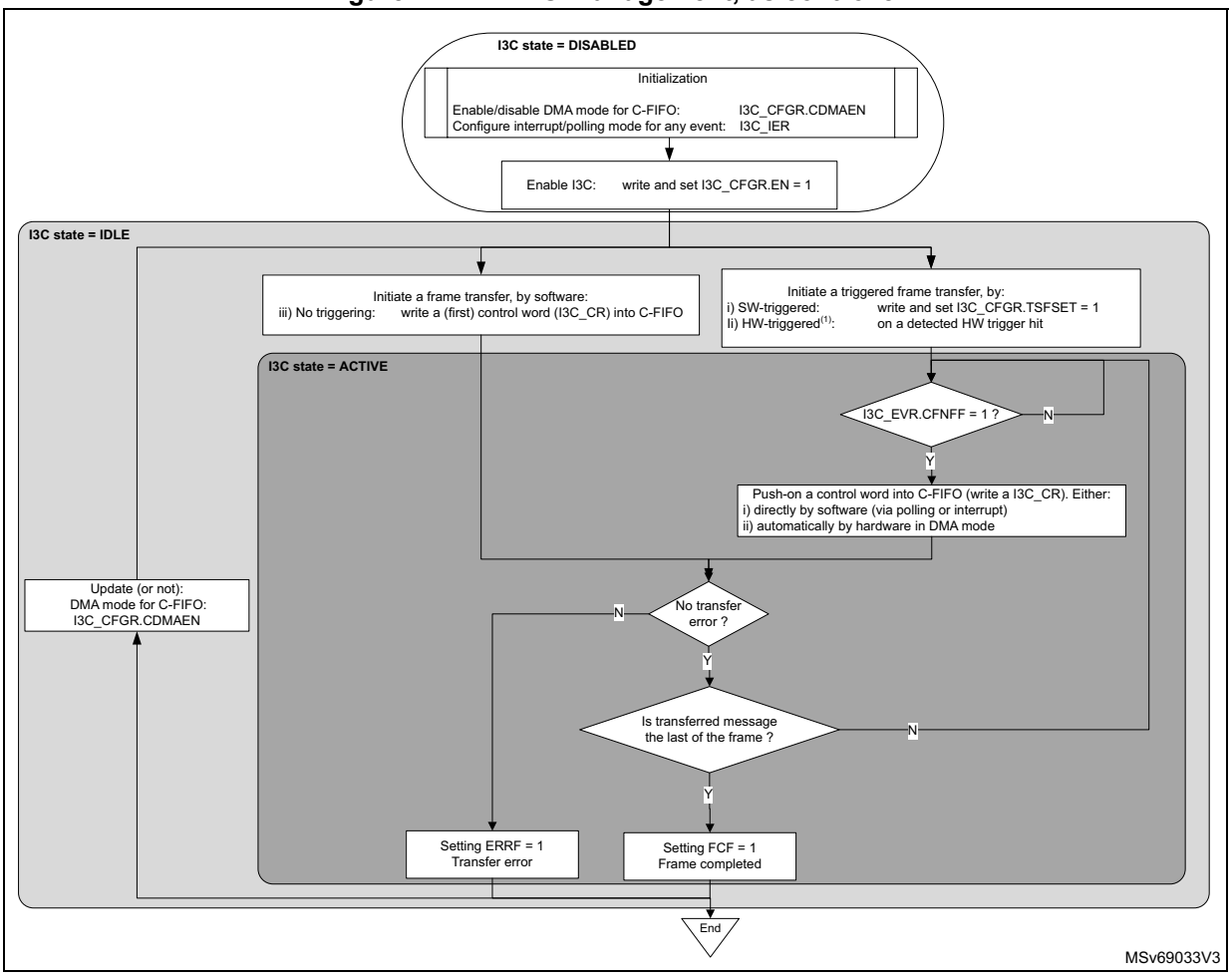
35.10.1 C-FIFO management, as controller

When controller, as illustrated in figures of *Section 35.9*, C-FIFO can be used during any of the following transfers:

- broadcast CCC (*Figure 408*, *Figure 409*, *Figure 410*)
- direct read/write CCC (*Figure 410*)
  - command part, first message
  - data part, next message(s)
- private read/write (*Figure 415*)
- legacy I2C read/write (*Figure 417*)
- software-initiated error recovery (SCL forced to be stopped until next header message followed by HDR exit pattern)

Figure 421 illustrates the management of the C-FIFO for queuing control word(s) on the I3C bus, when the I3C peripheral acts as controller.

Figure 421. C-FIFO management, as controller



1. This feature is implementation-dependent and can be unavailable. Refer to *Section 35.3.4*.

First, the software must initialize the C-FIFO management via CDMAEN in the I3C_CFGR register, to be written either:

- directly by the software (if CDMAEN = 0) at the control word level:
  - via polling mode (CFNFIE = 0 in the I3C_IER register): waiting for a next control word is requested by the hardware (CFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_CR register
  - via enabled interrupt notification if CFNFIE = 1
- by the allocated DMA channel (if CDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tc_dma):
  - as configured at block level, the DMA is automatically pushing-on/writing control word(s) into the I3C_CR register from its memory source buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

In any case, if C-FIFO is empty and a restart must be emitted with a new control word, a C-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and COVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The DMA mode for the C-FIFO management can be modified when the I3C peripheral is not in active state.

When controller, if a transfer error occurs (ERRF = 1 in the I3C_EVR register), the C-FIFO is flushed automatically by the hardware.

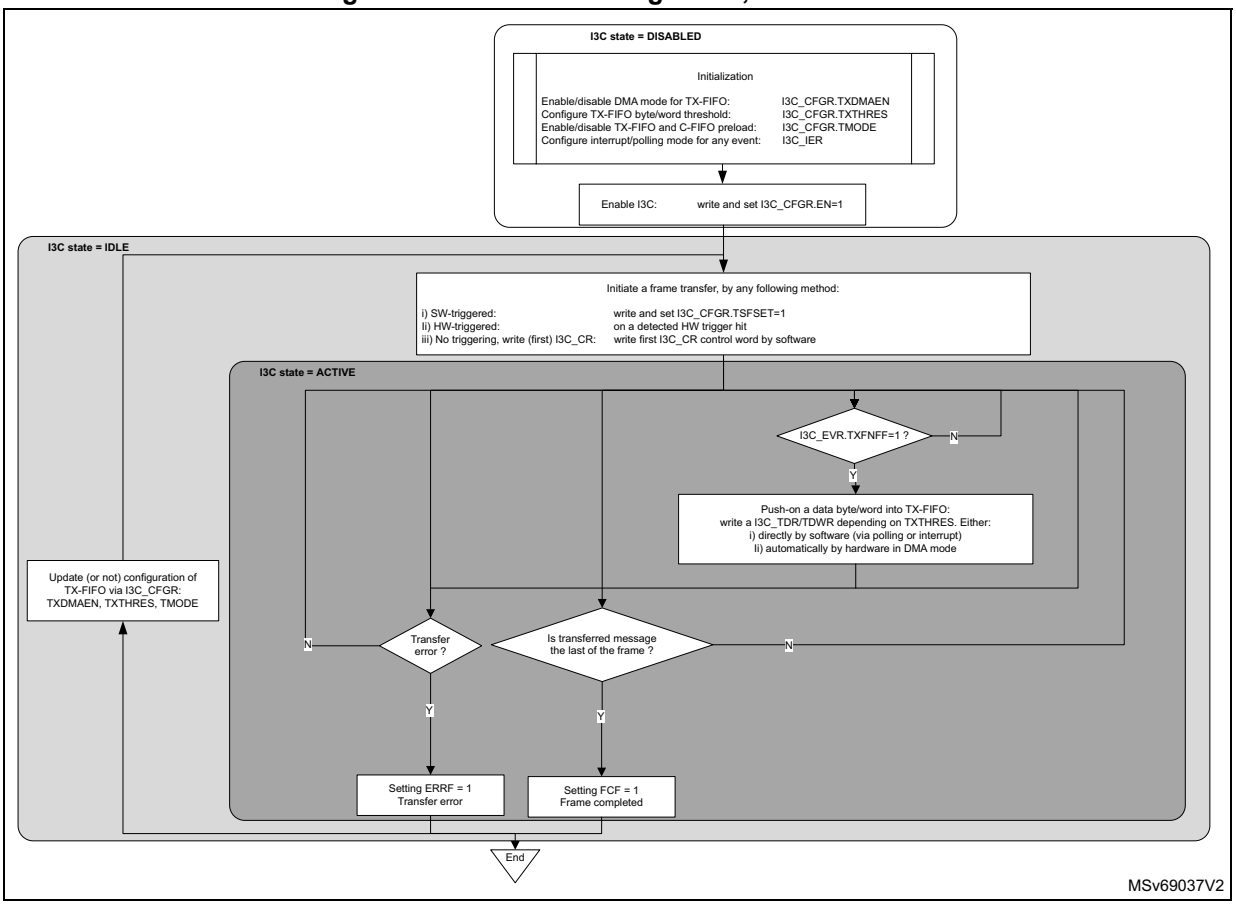
35.10.2 TX-FIFO management, as controller

When controller, as shown in figures of *Section 35.9*, TX-FIFO can be used during any of the following transfers:

- broadcast or direct CCC (including ENTDAA and RSTACT), if a defining/sub-command byte or if data byte(s) are present (*Figure 408*, *Figure 409*, *Figure 410*)
- private write (*Figure 415*)
- legacy I2C write (*Figure 417*)

Figure 422 illustrates the management of the TX-FIFO for queuing data bytes or word(s) to be transmitted on the I3C bus, when the I3C peripheral acts as controller.

Figure 422. TX-FIFO management, as controller



First, the software must initialize the TX-FIFO management via the following fields in the I3C_CFGR register:

- TXDMAEN: enable/disable DMA mode for TX-FIFO
- TXTHRES: push-on data byte(s) or word(s) into TX-FIFO
- TMODE: enable/disable TX-FIFO and C-FIFO preload

Then, depending upon bit TXDMAEN in the I3C_CFGR register, the TX-FIFO is written either:

- directly by the software (if TXDMAEN = 0) at the byte/word level:
  - via polling mode (TXFNFIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (TXFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_TDR or I3C_TDWR register, depending upon bit TXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if TXFNFIE = 1
- by the allocated DMA channel (if TXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tx_dma):
  - as configured at DMA block level, the DMA is automatically pushing-on/writing data bytes/words to the I3C_TDR or I3C_TDWR register (depending upon bit TXTHRES in the I3C_CGFR. register) from its memory source buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

An I3C message begins from a start or a repeated start, and ends with a stop or a repeated start. At message level, the last data byte/word to be transmitted is flagged by TXLASTF = 1 in the I3C_EVR register. When an I3C frame is described with multiple messages (separated by a repeated start), this event can be used by the software to update the pointer to the buffer where the byte(s)/word(s) of the next message is/are stored.

When frame completion is reported (FCF = 1 in the I3C_EVR register, and the corresponding interrupt is enabled), the TX-FIFO is empty.

If the TX-FIFO is empty and a data byte must be transmitted on the I3C bus, a TX-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The configuration for the TX-FIFO management can be modified when the I3C peripheral is not in active state.

When controller, if a transfer error occurs (ERRF = 1), the TX-FIFO is automatically flushed by the hardware.

No C-FIFO/TX-FIFO preload

As defined in *Table 273*, C-FIFO size is two words, TX-FIFO size is 8 bytes.

When no C-FIFO/TX-FIFO preload is configured (TMODE = 0 in the I3C_CFGR register), the I3C peripheral emits a start on the I3C bus as soon as the first control word is written into the C-FIFO. Then, it decodes the I3C_CR register, and requires a next data byte/word to be written, if needed within this message. As soon as a next control word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus or if the C-FIFO gets available room), this control word is requested to be written into the C-FIFO until the last message (MEND = 1 in the I3C_CR register).

Similarly, as soon as another data byte/word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus, and if TX-FIFO is not full, and if data byte(s)/word(s) must be transmitted during this I3C message), this data byte/word must be written in the TX-FIFO.

C-FIFO and TX-FIFO preload

When C-FIFO/TX-FIFO preload is configured (TMODE = 1 in the I3C_CFGR register), before emitting a start on the bus, the I3C peripheral waits for loading as much as possible both the C-FIFO and the TX-FIFO, as follows:

1. Wait for a first control word to be written into the C-FIFO.
2. Wait for data byte(s)/word(s) to be written in the TX-FIFO, if any, as defined by the first control word (if RNW = 0 and DCNT[15:0] = 0 in the I3C_CR register), and up to the TX-FIFO size.
3. If TX-FIFO is not full and if the first control word is not the last of the frame (MEND = 0 in the I3C_CR register):
   a) Wait for a second control word to be written into the C-FIFO, then C-FIFO is full.
   b) If TX-FIFO is not full, wait for data byte(s)/word(s) to be written in the TX-FIFO, if any, as defined by the second control word (RNW = 0 and DCNT[15:0] = 0 in the I3C_CR register), and up to the TX-FIFO size.

Then, as soon as a next control word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus), this control

word is requested to be written into the C-FIFO until the last message (MEND = 1 in the I3C_CR register).

Similarly, as soon as a next data byte/word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus and if TX-FIFO is not full and if data byte(s)/word(s) are to be transmitted during this I3C message), this data byte/word must be written in the TX-FIFO.

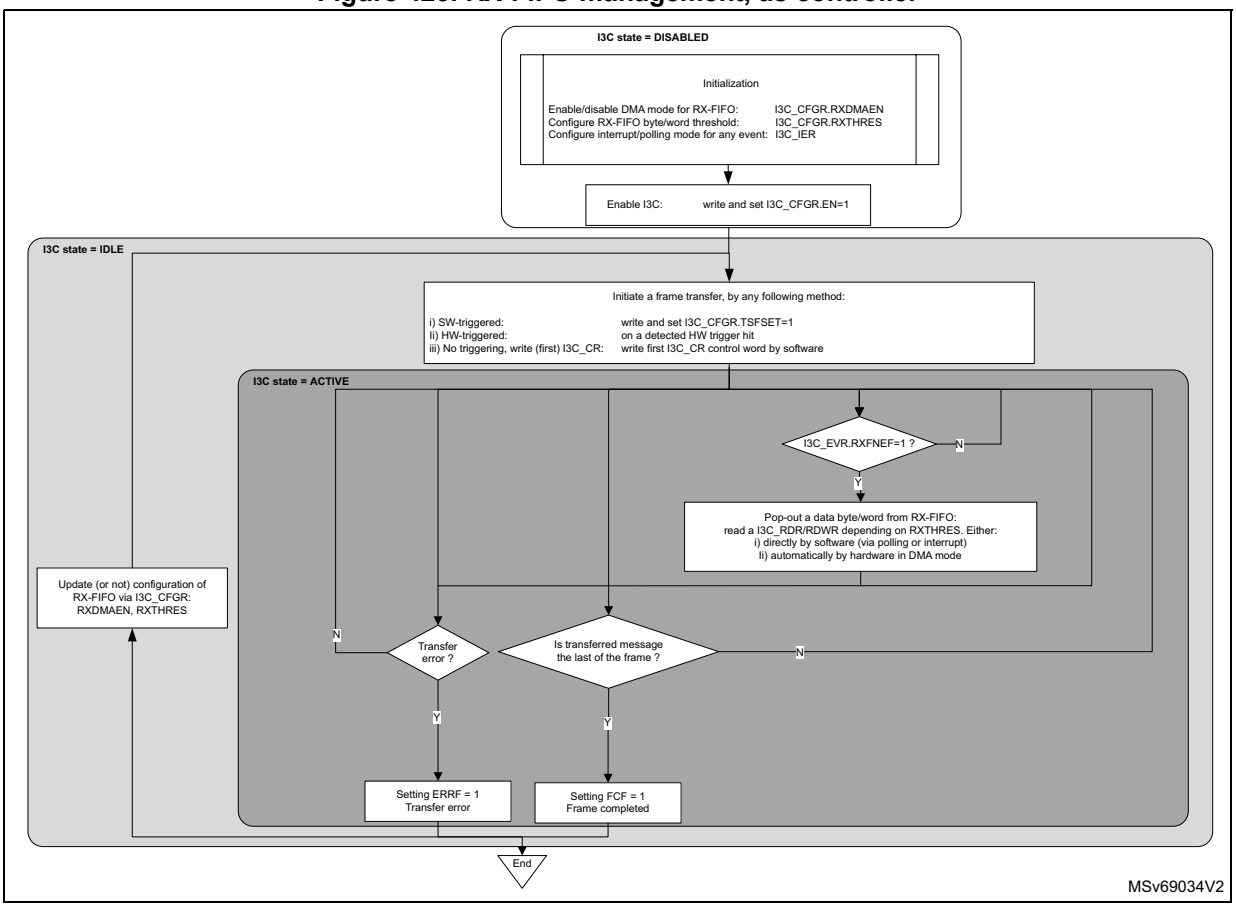
35.10.3 RX-FIFO management, as controller

When controller, as shown in figures of *Section 35.9*, RX-FIFO is used during any of the following transfers:

- broadcast ENTDAA CCC (*Figure 409*)
- direct CCC read (*Figure 408*), including direct RSTACT CCC read (*Figure 410*)
- private read (*Figure 415*)
- legacy I2C read (*Figure 417*)

Figure 423 illustrates the management of the RX-FIFO for queuing and popping-out data bytes or word(s) as received from the I3C bus, when the I3C peripheral acts as controller.

Figure 423. RX-FIFO management, as controller



First, the software must initialize the RX-FIFO management via the following fields of the I3C_CFGR register:

- RXDMAEN: enable/disable DMA mode for RX-FIFO
- RXTHRES: pop-out data byte(s) or word(s) from RX-FIFO

Then, depending on RXDMAEN, the RX-FIFO is read either:

- directly by the software (RXDMAEN = 0) at the byte/word level:
  - via polling mode (RXFNEIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (RXFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_RDR or I3C_RDWR register, depending upon bit RXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if RXFNEIE = 1 in the I3C_IER register
- by the allocated DMA channel (if RXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rx_dma):
  - as configured at DMA block level, the DMA automatically pops-out/reads data bytes/words from the I3C_RDR or I3C_RDWR register (depending upon bit RXTHRES), and writes them into its memory destination buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

An I3C message begins from a start or a repeated start, and ends with a stop or a repeated start. At message level, the last received data byte/word from the I3C bus is flagged by IRXLASTF = 1 in the I3C_EVR register. When an I3C frame is described with multiple messages (separated by a repeated start), this event can be used by the software for updating the pointer to the buffer where is/are stored the data byte(s)/word(s) of the next message.

If RX-FIFO is full and a data byte is received on the I3C bus, an RX-FIFO overrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The configuration for the RX-FIFO management can be modified when the I3C peripheral is not in active state.

Early read termination from the target

A private read message can be early completed (also known as prematurely ended) by the addressed target.

- If RXDMAEN = 1:
  - the software must allocate, for the DMA request i3c_rx_dma, a DMA channel x, capable of peripheral early termination (refer to DMA implementation section).
  - The software must configure the DMA channel x to enable the DMA peripheral-flow control mode via PFREQ = 1 in the DMA_CxTR2 register, to be able to perform a(n) (early or not) DMA block completion.
  - Then, on block completion, the software can read BR1.BNDT[15:0] in DMA_Cx and/or DMA_CxSAR register(s), to get the effective number of DMA transferred bytes.
- If RXDMAEN = 0:
  - at message level, the last received data byte/word from the I3C bus is flagged by RXFNEF = 1 and RXLASTF = 1 in the I3C_EVR register.

In any case, if the S-FIFO is disabled (if SMODE = 0 in the I3C_CFGR register), the software is notified that an early read termination occurs by RXTGTENDF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled. Then, software can read the status register I3C_SR to check information related to the last message ,and get the number of received data bytes on the prematurely ended read transfer (XDCNT[15:0] in the I3C_SR register).

In any case, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), the software or the DMA (depending upon SDMAEN in the I3C_CFGR register) must read for each message the status register I3C_SR. The number of effective received data bytes on the prematurely ended read message is reported by XDCNT[15:0] (and then ABT = 1) in the I3C_SR register.

For more information, refer to *I3C status register (I3C_SR)* and *Section 35.10.4*.

35.10.4 S-FIFO management, as controller

When controller, S-FIFO can be used by the software to be able to read the *I3C status register (I3C_SR)* for each transferred message.

Reading status register with disabled S-FIFO

If SMODE = 0 in the I3C_CFGR register, the S-FIFO is disabled and the status register can be read as a usual register:

- the register content is overwritten by hardware when is transferee a new message
- I3C_SR contains the status of the last transferred message
- SCL clock is not stalled if status register is not read.

If SMODE = 0, for the specific case of a private read prematurely ended by the target:

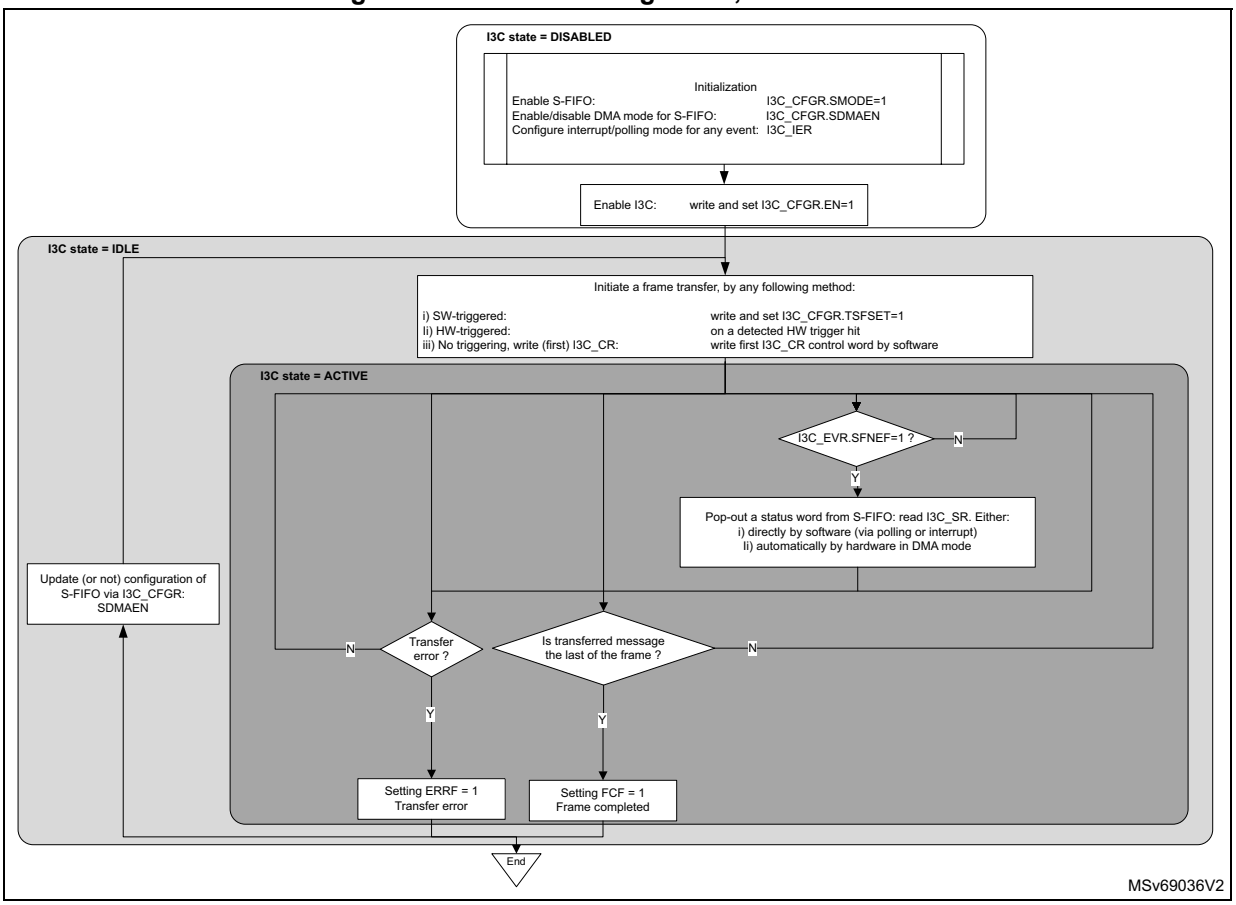
- the software is notified by the flag RXTGTENDF = 1 in the I3C_EVR register and the corresponding interrupt if enabled.
- Then, until that the software has cleared the event flag write and set CRXTGTENDF = 1 in the I3C_CEVR register:
  - no more data byte can be received on the I3C bus and written by the hardware into I3C_RDR/I3C_RDWR registers
  - I3C_SR cannot be updated
  - SCL clock is stalled if needed

Typically, I3C_SR can be read when FCF = 1, or ERRF = 1, or RXTGTENDF = 1 in the I3C_EVR register. XDCNT[15:0] in the I3C_SR register can be read to get the effective number of received data bytes on a private read, after an early termination from the target (refer to *Section 35.10.3*).

Reading status register with enabled S-FIFO

If SMODE = 1 in the I3C_CFGR register, the S-FIFO is enabled. *Figure 424* illustrates the management of the S-FIFO for queuing and popping-out a status word for each executed message on the I3C bus, when the I3C peripheral acts as controller.

Figure 424. S-FIFO management, as controller



First, the software must initialize the S-FIFO management via the SDMAEN field (enable/disable DMA mode for S-FIFO) in the I3C_CFGR register. Then, depending on SDMAEN, the S-FIFO is read either:

- directly by the software (if SDMAEN = 0):
  - via polling mode (SFNEIE = 0 in the I3C_IER register): waiting for a next status word is requested by the hardware (SFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_SR register
  - via enabled interrupt notification (SFNEIE = 1)
- by the allocated DMA channel (if SDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rs_dma):
  - as configured at DMA block level, the DMA is automatically popping-out/reading status words from the I3C_SR register and writing them into its memory destination buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

Each message status must be read, else an overrun error is asserted by the hardware (ERRF = 1 in the I3C_EVR register and COVR = 1 in the I3C_SER register) when the S-FIFO is full and a next message status must be written. The corresponding interrupt is raised if enabled by ERRIE = 1 in the I3C_IER register.

The frame completion (FCF = 1 in the I3C_EVR register) is reported only when the S-FIFO is empty.

The configuration for the S-FIFO management can be modified when the I3C peripheral is not in active state.

35.11 I3C FIFOs management, as target

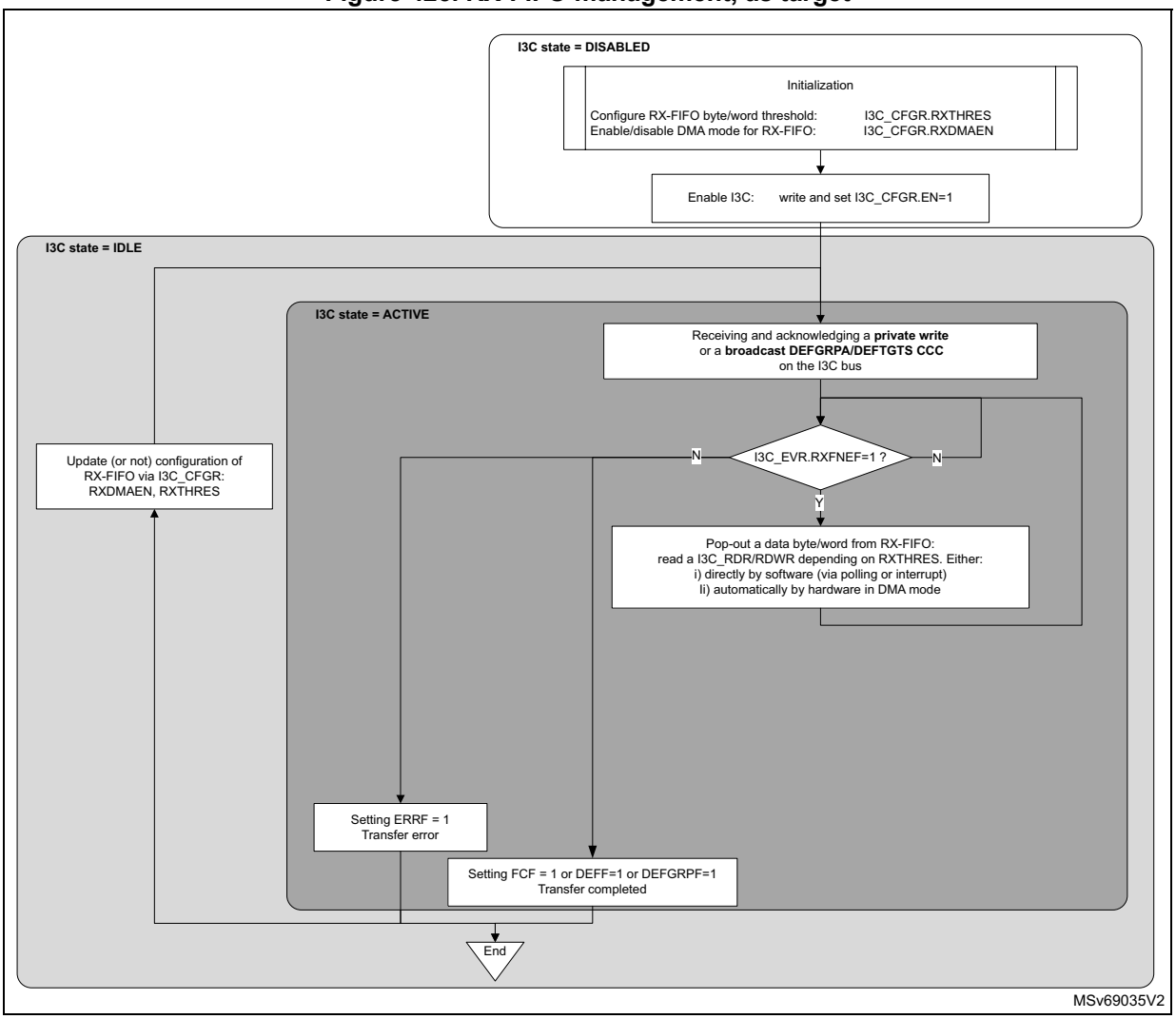
35.11.1 RX-FIFO management, as target

When target, as shown in figures of *Section 35.9*, the RX-FIFO is used during any of the following received and acknowledged transfers:

- broadcast DEFTGTS CCC (*Figure 413*)
- broadcast DEFGRPA CCC (*Figure 414*)
- private write (*Figure 416*)

Figure 425 illustrates the management of the RX-FIFO for queuing and popping-out data bytes or word(s) as received from the I3C bus, when the I3C peripheral acts as target.

Figure 425. RX-FIFO management, as target



First, the software must initialize the RX-FIFO management via the following I3C_CFGR register fields:

- RXDMAEN: enable/disable DMA mode for RX-FIFO
- RXTHRES: pop-out data byte(s) or word(s) from RX-FIFO

Then, depending on RXDMAEN, the RX-FIFO is read either:

- directly by the software (if RXDMAEN = 0) at the byte/word level:
  - via polling mode (RXFNEIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (RXFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_RDR or I3C_RDWR registers, depending upon RXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if RXFNEIE = 1
- by the allocated DMA channel (if RXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rx_dma):
  - as configured at DMA block level, the DMA is automatically popping-out/reading data bytes/words from the I3C_RDR or I3C_RDWR register (depending upon RXTHRES) and writing them into its memory destination buffer, until the transfer completion (in the I3C_EVR register, FCF = 1 in case of private write, or GRPF = 1 in case of a DEFGRPA CCC, or DEFF = 1 in case of DEFTGTS CCC), unless a transfer error occurs.

If RX-FIFO is full and a new data byte is received on the I3C bus, a RX-FIFO overrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register) and the corresponding interrupt if enabled.

When the transfer is completed (in the I3C_EVR register, FCF = 1, or GRPF = 1, or DEFF = 1):

- RX-FIFO is empty
- Until software has not processed the RX data buffer corresponding to the completed private write / DEFTGTS / DEFGRPA transfer (meaning until software has not cleared the corresponding flag write and set CFCF / CDEFF / CGRPF to 1 in the I3C_CEVR register), if a next private write / DEFTGTS CCC / DEFGRPA CCC is received and a data byte must be written by the hardware into the RX-FIFO, a transfer error is reported with a RX-FIFO overrun (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register) and the corresponding interrupt if enabled.

The configuration for the RX-FIFO management can be modified when the I3C peripheral is not in active state.

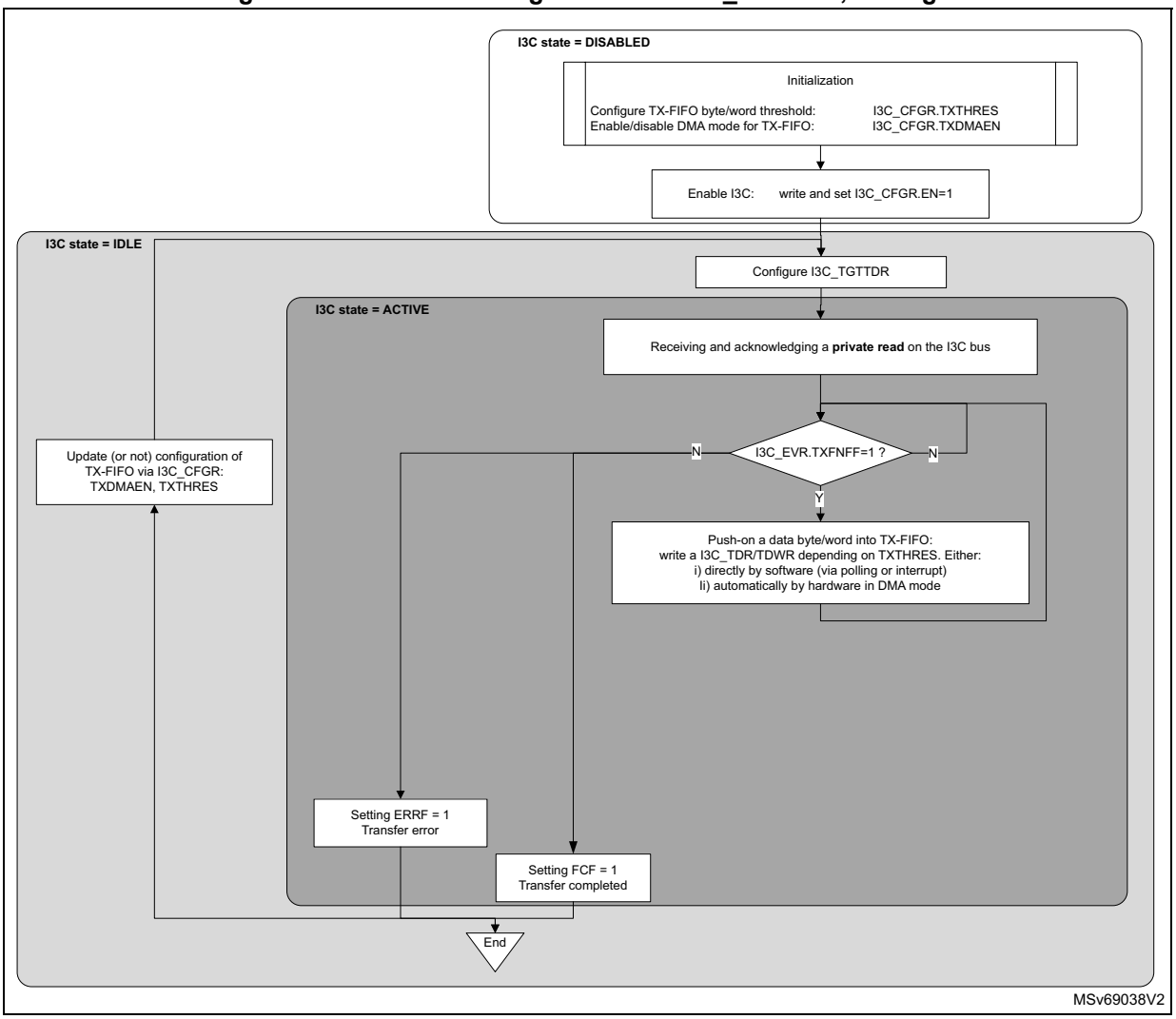
35.11.2 TX-FIFO management, as target

Main scheme

When target, as shown in figures of *Section 35.9*, the TX-FIFO is used only during a private read (*Figure 416*).

Figure 426 illustrates the management of the TX-FIFO for queuing and pushing-on data bytes or word(s) to be transmitted on the I3C bus, when the I3C peripheral acts as target.

Figure 426. TX-FIFO management with I3C_TGTTDR, as target



First, the software must initialize the TX-FIFO management and write I3C_CFGR with

- TXDMAEN: enable/disable DMA mode for TX-FIFO
- TXTHRES: push-on data byte(s) or word(s) to TX-FIFO

Then, before receiving a private read on the I3C bus, the software must configure the *I3C target transmit configuration register (I3C_TGTTDR)* to preload the TX-FIFO with a number of data bytes (write TGTTDCNT[15:0] and PRELOAD = 1 in a single access), so that data bytes from the target are ready to be transmitted on the I3C bus:

- If PRELOAD = 1 and TGTTDCNT[15:0] > TX-FIFO size, TX-FIFO is first preloaded up to the FIFO size.
- If PRELOAD = 1 and TGTTDCNT[15:0] ≤ TX-FIFO size, TX-FIFO is preloaded up to TGTTDCNT[15:0].

Note: TX-FIFO size is 8 bytes (refer to Table 273).

Depending upon TXDMAEN, the TX-FIFO is preloaded either:

- directly by the software (TXDMAEN = 0) at the byte/word level:
  - via polling mode (TXFNFIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (TXFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_TDR or I3C_TDWR register, depending upon TXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if TXFNFIE = 1
- by the allocated DMA channel (TXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tx_dma):
  - as configured at DMA block level, the DMA is automatically pushing-on/writing data bytes/words to the I3C_TDR or I3C_TDWR register (depending upon TXTHRES) from its memory source buffer, until the transfer completion (FCF = 1 in the IEC_EVR register), unless a transfer error occurs.

Then, in the same way, either directly by the software or by the allocated DMA channel, if there are remaining TGTTDCNT[15:0] in the I3C_TGTTDR register to be loaded into TX-FIFO for continuing the private read (set PRELOAD = 1 and TGTTDCNT[15:0] > TX-FIFO size), and provided that the private read is not yet completed by the controller:

- If TXTHRES = 0: when a byte is transmitted on the I3C bus, a next byte is preloaded into TX-FIFO
- If TXTHRES = 1: when four bytes are transmitted on the I3C bus, the next word is preloaded into TX-FIFO.

The private read transfer is completed (FCF = 1 in the I3C_EVR register) when either the target or the controller first terminates the data byte transfer.

On transfer completion, the software can:

- read XDCNT[15:0] in the I3C_SR register: the effective number of transmitted data bytes
- read TGTTDCNT[15:0] in the I3C_TGTTDR register: the remaining number of bytes to be loaded and transmitted on the I3C bus
- flush TX-FIFO: write and set (or not) TXFLUSH = 1 in the I3C_CFGR register, continue (or not) for another private read, depending upon the user application

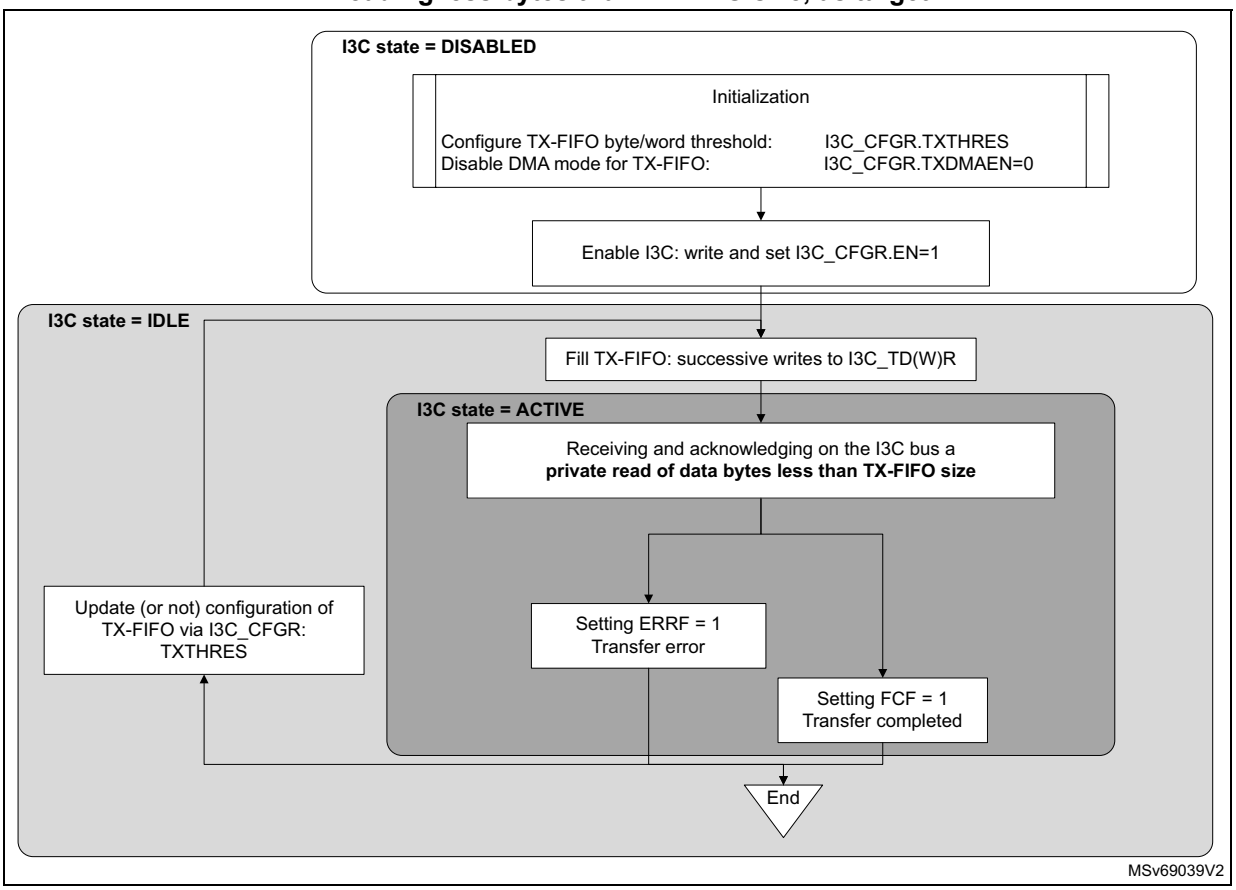
If TX-FIFO is empty and a data byte must be transmitted on the I3C bus, a TX-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_EVR register, an interrupt is generated.

The configuration for the TX-FIFO management can be modified when the I3C peripheral is not in active state.

Alternative without I3C_TGTTDR, if less bytes than the TX-FIFO size

Alternatively to the use of the I3C_TGTTDR register, as shown in the *Figure 427*, when the DMA is not used (TDMAEN = 0), provided that the number of data bytes to be read on the I3C bus is less that the TX-FIFO size, the software can directly prepare and fill the TX-FIFO with the number of bytes to be read directly by the software, by successive writes to the I3C_TDR or I3C_I3C_TDWR, depending upon TXTHRES.

Figure 427. TX-FIFO management by software without I3C_TGTTDR if reading less bytes than TX-FIFO size, as target



35.12 I3C error management

35.12.1 Controller error management

Table 282 enumerates the I3C bus error conditions, and, for each of them, the corresponding detection, action and reporting when the I3C peripheral acts as controller.

Table 282. I3C controller error management

Error type	Description	Error detection	Controller action	Reported error(1)
CE0	Illegally formatted CCC (e.g. less returned data byte(s))	Prematurely ended read data by target on a direct CCC read(2)	Hardware emits a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0000

Table 282. I3C controller error management (continued)

Error type	Description	Error detection	Controller action	Reported error(1)
CE1	Monitoring error	An incorrect ACK is detected at the end of a legacy I^2C read	Hardware keeps SCL running for nine clock cycles and another byte to be possibly exchanged, then emits again a NACK, followed by a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0001
		Restart or stop cannot be generated at the end of an I3C SDR read	SCL is kept running. Software can stop SCL by writing a control word with MTYPE[3:0] = 0000 in the I3C_CR register (header message), then must typically wait at least 150 µs before emitting another message.	
		After a CE1 error, a start cannot be generated		
CE2	No response to broadcast address (0b111_1110)	Header (0b111_1110 + RnW = 0) is detected as NACK-ed during a message different from escalation fault or reset pattern message	Hardware emits an HDR exit pattern followed by a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0010
CE3	Failed controller hand-off	New controller has not driven SCL low after having SDA low (after a start via a test header or a start request from a target) and after the delay time defined by its activity state has elapsed	Hardware emits a start + 0b111_1110 + RnW = 0, followed by ACK/NACK from target(s), then stops.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0011
-	Incorrect returned 7-bit target address with parity bit on a GETACCCR CCC	Incorrect dynamic address and/or parity bit is detected	Hardware cancels the GETACCCR CCC by emitting a Restart + 0b111_1110 + RnW = 0, followed by ACK/NACK from target(s), then stops.	ERRF = 1 and DERR = 1
-	Addressed target is NACK-ed on a direct CCC read for the first time	NACK is detected	Hardware performs a single-retry by emitting a Restart + 7-bit same target address + RnW = 1.	-
-	Addressed target is NACK-ed on either a direct CCC write, an I3C private read/write, a legacy I^2C, or a direct CCC read for the second time	NACK is detected	Hardware emits a stop.	ERRF = 1 and ANACK= 1

Table 282. I3C controller error management (continued)

Error type	Description	Error detection	Controller action	Reported error(1)
-	Assigned/transmitted dynamic address with parity bit is NACK-ed on a ENTDAA CCC for the first time	NACK is detected	Hardware performs a single-retry assignation loop by emitting a Restart + 0b111_1110 + RnW = 1, followed by an ACK and 8-byte read data from the (most-priority) target, followed by the assigned address + parity bit.	-
-	Assigned/transmitted dynamic address with parity bit is NACK-ed on a ENTDAA CCC for the second time	NACK is detected	Hardware emits a stop.	ERRF = 1 and DNACK= 1
-	Write data is NACK-ed on a legacy I2C write			
-	Control word, status word, transmitted data or read data is not written/read in time vs. I3C bus timings	SCL stall timeout	Hardware emits a stop.	ERRF = 1 and (COVR = 1 or DOVR = 1)

1. ERRF in the I3C_EVR register, PERR, CODERR[3:0], DERR, ANACK, DNACK, COVR, and DOVR in the I3C_SER register.
2. MIPI v1.1: on a GETCAPS CCC, the number of received data bytes can be 2, 3 or 4. However a target compliant with MIPI v1.0 can return only the first byte as per previously named GETHDRCAP CCC. As a result, CE0 is not generated if the number is lower than 4.
On a GETMXDS CCC, the number of received data bytes can be 2 or 5. CE0 is not generated if the number is lower than 5.

35.12.2 Target error management

Table 283 enumerates the I3C bus error conditions, the corresponding detection, action, and reporting when the I3C peripheral acts as target.

Table 283. I3C target error management

Error type	Description	Error detection	Next (when emitted by the controller)	Reported error(1)
TE0	Invalid broadcast address (0b111_1110+RnW = 0) or Invalid 7-bit dynamic address + RnW = 1 after DAA assignment	A forbidden address is detected after a start or a repeated start	Hardware waits for an HDR exit pattern	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1000
TE1	CCC code	A CCC code is detected with a parity error		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1001

Table 283. I3C target error management (continued)

Error type	Description	Error detection	Next (when emitted by the controller)	Reported error[1]
TE2	Write data	A write data byte is detected with a parity error on an I3C private write message	Hardware waits for a stop or a repeated start	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1010
TE3	Assigned address during dynamic address arbitration	Assigned dynamic address is detected with a parity error on a ENTDAA CCC		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1011
TE4	0b111_1110 + RnW = 1 missing after Sr during dynamic address arbitration	Missing {0b111_1110 + RnW = 1} after Sr during dynamic address arbitration is detected		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1100
TE5	Transaction after detecting CCC	Invalid CCC direction is detected vs. direction provided during address phase, on a CCC direct read/write		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1101
TE6	Monitoring error	SDA is detected as driven at unexpected value on an SDR data read (during a CCC direct read or a private read or an IBI)	Hardware releases SDA, then waits for a stop or a repeated start	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1110
-	Monitoring SCL on an SDR data read	SCL is detected as stable for more than 125 µs on an SDR data read (during a CCC direct read or a private read or an IBI)		ERRF = 1 and STALL = 1
-	Data to be transmitted not written/available in time vs. I3C bus timings	Data to be transmitted not written by software or DMA in due time on a private read	Hardware detects a stop or a repeated start	ERRF = 1 and DOVR = 1
-	Received data cannot be registered in time vs. I3C bus timings	Received data is not read by software or DMA in due time on a private write or a DEFTGTS or a DEFGRPA CCC		

1. ERRF in the I3C_EVR register, PERR, ODERR[3:0], STALL, and DOVR in the I3C_SER register.

35.13 I3C wake-up from low-power mode(s)

35.13.1 Wake-up from Stop

The user must first configure the reset and clock controller (RCC) to set up the clock data path down to the I3C peripheral: to select the source oscillator for the I3C kernel clock, the source oscillator for the I3C APB clock, and to set the clocks frequency. At initialization via the RCC, the user must enable the I3C clocks for a given I3C peripheral to be functional, separately for Run/Sleep mode and for Stop mode. For more details about RCC programming, refer to *Section 10: Reset and clock control (RCC)*.

The I3C hardware automatically manages its own clocks gating and generates a separated clock request output signal to the RCC for its kernel clock and its APB clock, whenever the device is in Run, Sleep or Stop mode.

When entering a Stop mode, the V_{CORE} domain is supplied, by default any clock oscillator is disabled, and in any case, neither the system clock nor a peripheral clock is running in the domain.

As controller: wake-up on an IBI without MDB, a hot-join request or a controller-role request

When the peripheral acts as controller, before the product enters a low-power mode, the software must issue an ENTASx CCC, generally with x = 0, 1, 2 or 3, to inform targets that the I3C controller is not expected to exit from idle state neither to communicate on the I3C bus before an interval of, respectively, 1 µs, 100 µs, 2 ms or 50 ms, has elapsed. This delay defines the T_{CAS} delay for the controller to set the SCL bus clock low and running, after a start condition. More specifically for a Stop mode, the CCC must be restricted to an ENTASx with the value x=1, 2, or 3.

The general scheme for the I3C wake-up from Stop is the following:

1. First the I3C peripheral requests its kernel clock to the system:
   - On a detected start condition on the I3C bus (SDA line is detected to be driven low while SCL is high).
   - As controller, as initiated by an external I3C target device for a hot-join/in-band interrupt/controller-role request. Once the kernel clock is provided and running, the I3C hardware uses an internal timer to wait for the corresponding T_{CAS} time to elapse, then drives low SCL and continues toggling SCL to let the target perform its hot-join/in-band interrupt/controller-role request on the I3C bus.
2. The system enables the source oscillator of the I3C kernel clock, and the clock gets ready (after few microseconds from HSI). The user can keep the source oscillator ON in Stop mode to reduce this startup latency, at the expense of power consumption.
   - The I3C peripheral maintains the kernel clock request until the generation of the Stop condition on the I3C bus.
3. After that the kernel clock is running, as controller the I3C peripheral requests its APB clock to the system:
   - On the ACKed address of a received IBI request (*Figure 418*), and it maintains the APB clock request until that IBIF is cleared.

     If the IBI is without MDB: a Stop is normally generated on the I3C bus, even if the APB clock is not yet provided.
   - On the ACKed address of a received controller-role request (*Figure 419*), and it maintains the APB clock request until that CRF is cleared.
   - On the ACKed address of a hot-join request (*Figure 420*), and it maintains the APB clock request until that HJF is cleared.
4. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken.
5. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
6. The system enables the system clock that drives the APB clock. There is an additional delay if the selected oscillator source for the system and APB clocks is not the same as the one driving the I3C kernel clock. If so, the system enables the source oscillator of the system clock, which gets ready after a delay.
7. With the APB clock running, the I3C peripheral can log the I3C transfer in its status and data registers. When the bus transfer is completed, the I3C peripheral generates the corresponding flag (IBIF/CRF/HJF), and the enabled interrupt can wake up the CPU.

As target: wake-up on a reset pattern

The target reset pattern is a specific scheme for a controller to wake up and possibly reset a target from a low power mode. It consists both in the RSTACT CCC and in the in-band reset pattern generation.

As target, the sequence for the I3C wake-up from Stop on a reset pattern is the following:

1. When the I3C peripheral detects a reset pattern on the bus (14 SDA transitions while SCL is kept low), it requests its APB clock to the system to set the RSTF flag of the I3C_EVR register
2. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken (after few microseconds).
3. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
4. The system enables the source oscillator of the system clock, which gets ready after few microseconds.
5. With the APB clock running, the I3C peripheral can raise the RSTF flag of the I3C_EVR register, and the enabled interrupt can wake up the CPU.

As target: wake-up on a missed start

1. The I3C peripheral requests its kernel clock to the system:
   - On a detected start condition on the I3C bus (SDA line is detected to be driven low while SCL is high)
   - As target, as initiated by an external I3C device, whatever controller or target.
   - If the kernel clock is not provided before that SCL is driven low by the external controller, then the I3C target detects that a I3C bus start is missed and waits for the kernel clock to be provided. It can be noted that an I3C controller message intended to be addressed to it may have been missed (and NAcked). Or may have been missed any I3C CCC or private message from the controller to another addressed target or an IBI/CR/HJF start request from another target.
2. The system enables the source oscillator of the I3C kernel clock, and the clock gets ready (after few microseconds if HSI). The user may keep the source oscillator ON in Stop mode to reduce this startup latency, at the expense of power consumption. Especially if the controller is in an Activity State of x = 0 (with a 1 µs T_{CAS} latency).
3. After that the kernel clock is running, as target, the I3C peripheral requests its APB clock to the system to raise the WKPF flag:
4. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken (after few microseconds).
5. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
6. The system enables the system clock, which drives the APB clock. There is an additional delay if the selected oscillator source for the system and APB clocks is not the same as the one driving the I3C kernel clock. If so, the system enables the source oscillator of the system clock, which gets ready after few microseconds.
7. With the APB clock running:
   - The missed start flag (WKPF of the I3C_EVR register) is raised if it occurred, and the enabled interrupt can wake up the CPU. It is then known that an I3C bus transaction was missed, but not whether the target was addressed or not. The software should use a timeout to return to Stop mode, if it is not addressed by the controller again within this interval.

35.14 I3C in low-power modes

Table 284. Effect of low-power modes

Mode	Description
Sleep	No effect. I3C interrupts cause the device to exit Sleep mode.
Stop[1]	The content of the I3C registers is kept when entering Stop mode. I3C hardware manages automatically its own clocks gating and generates, for its kernel clock and APB clock, a clock request output signal to the RCC. I3C interrupts can cause the device to wake up and exit Stop mode.
Standby	The I3C peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Table 272* to know if Stop mode is supported, and which one.

35.15 I3C interrupts

Table 285. I3C interrupt requests

Interrupt acronym		Interrupt event	Used as		Interrupt enable: field in I3C_IER	Event flag: field in I3C_EVR	Event clear method: write 1 to field in I3C_CEVR
			Controller	Target			
I3C	EVT	A control word is requested	X	-	CFNFIE	CFNFF	-
		A status word is available	X	-	SFNEIE	SFNEF	-
		Data to be transmitted are requested	X	X	TXFNFIE	TXFNFF	-
		Received data are available	X	X	RXFNEIE	RXFNEF	-
		Controller: frame transfer is completed Target: private transfer is completed	X	X	FCIE	FCF	CFCF
		A private read transfer is prematurely ended by the target (and SMODE = 0 in the I3C_CFGR register)	X	-	RXTGTENDIE	RXTGTENDF	CRXTGTENDF
		An IBI request is received	X	-	IBIIE	IBIF	CIBIF
		A controller-role request is received	X	-	CRIE	CRF	CCRF
		A hot-join request is received	X	-	HJIE	HJF	CHJF
		IBI request is completed	-	X	IBIENDIE	IBIENDF	CIBIENDF
		I3C bus start is missed	-	X	WKPIE	WKPF	CWKPF
		Direct GETACCR CCC is received	-	X	CRUPDIE	CRUPDF	CCRUPDF
		Direct GETSTATUS CCC is received	-	X	STAIE	STAF	CSTAF
		Any direct GETxxx CCC (except GETSTATUS) is received	-	X	GETIE	GETF	CGETF
		Dynamic address is updated (broadcast ENTDAA or RSTDAA, or direct SETNEWDA is received)	-	X	DAUPDIE	DAUPDF	CDAUPDF
		Direct SETMWL CCC is received	-	X	MWLUPDIE	MWLUPDF	CMWLUPDF
		Direct SETMRL CCC is received	-	X	MRLUPDIE	MRLUPDF	CMRLUPDF
		A reset pattern is detected	-	X	RSTIE	RSTF	CRSTF
		Bus activity state is updated (direct/broadcast ENTASx CCC is received)	-	X	ASUPDIE	ASUPDF	CASUPDF
		Broadcast/direct ENEC/DISEC CCC is received	-	X	INTUPDIE	INTUPDF	CINTUPDF
		Broadcast DEFTGTS CCC is received	-	X	DEFIE	DEFF	CDEFF
		Broadcast DEFGRPA CCC is received	-	X	GRPIE	GRPF	CGRPF
	ERR	An error occurred	X	X	ERRIE	ERRF	CERRF

35.16 I3C registers

The I3C registers must be accessed with a 32-bit word aligned address.

Note: I3C_RDR and I3C_TDR registers must be accessed with a single significant LSB data byte for, respectively, reading RX-FIFO and writing TX-FIFO.

35.16.1 I3C message control register (I3C_CR)

Address offset: 0x000

Reset value: 0x0000 0000

This register must be used to control the message to emit on the I3C bus:

- when I3C acts as controller (bit[30] = MTYPE[3] = 0): if there is no CCC code to be emitted bits[29:27] = MTYPE[2:0] differ from 110; else the alternate register description *Section 35.16.2* must be considered.
- when I3C acts as target (bit[30] = MTYPE[3] = 1).

When I3C acts as controller:

- If the control FIFO (C-FIFO) is not full (CFNFF = 1 in the I3C_EVR register), writing into this register means pushing a new control word into the C-FIFO; either by software, or automatically by DMA, as defined by CDMAEN in the I3C_CFGR register.
- If C-FIFO is empty and a restart must be emitted with a new control word, the I3C hardware asserts the control FIFO error underrun flag (COVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.
- After the last message of the frame is completed (a message with MEND = 1 in the I3C_CR register), the I3C hardware asserts the frame completed flag (FCF = 1 in the I3C_EVR register) and the corresponding interrupt, if enabled.

When I3C acts as target, this register is used in register mode:

- Software writes into this register to initiate a command (IBI, controller-role or hot-join request) on the I3C bus.
- C-FIFO is disabled, and there is no DMA mode neither for control words.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEND	MTYPE[3:0]				Res.	Res.	Res.	ADD[6:0]							RNW
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
DCNT[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **MEND**: Message end type/last message of a frame (when the I3C acts as controller)

0: this message from controller is followed by a repeated start (Sr), before another message must be emitted

1: this message from controller ends with a stop (P), being the last message of a frame

Bits 30:27 **MTYPE[3:0]**: Message type (whatever I3C acts as controller/target)

Condition: when I3C acts as I3C controller

0000: **SCL clock is forced to stop** until a next control word is executed

Bits[26:0] are ignored. On a CE1 error detection (ERRF = 1 in the I3C_EVR register and CODERR[3:0] = 0001 in the I3C_SER register) where a start/restart/stop is prevented from being generated, the software must use this message type for SCL “stuck at” recovery. Refer to *Table 282*.

0001: **header message**

Bits[26:0] are ignored. If the addressed target is not responding with an ACK to a private/direct message, as an escalation stage after a failed GETSTATUS tentative, the software must program this with EXITPTRN = 1 in the I3C_CFGR register, so that an HDR exit pattern is emitted on the bus, whatever the header is ACK-ed or NACK-ed (to avoid the target to consider that the I3C bus is in HDR mode). Refer to *Table 282* and MIPI specification about escalation handling.

0010: **private message** (refer to *Figure 415*)

Bits[23:17] (ADD[6:0]) are the emitted 7-bit dynamic address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred private message is:

– {S / S + 0b111_1110 + RnW = 0 + Sr/Sr+*} + 7-bit DynAddr + RnW + (8-bit Data + T)* + Sr/P.
– After an S (start), depending upon bit NOARBH in the I3C_CFGR register, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + RnW = 0) if needed, if it follows a previous message without ending by a P (stop).

0011: **direct message (second part of an I3C SDR direct CCC command)** (refer to *Figure 408*)

Bits[23:17] (ADD[6:0]) are the emitted 7-bit dynamic address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred direct message is: Sr + 7-bit DynAddr + RnW + (8-bit Data + T)* + Sr/P

0100: **legacy I^2C message** (refer to *Figure 417*)

Bits[23:17] (ADD[6:0]) are the emitted 7-bit static address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred legacy I^2C message is:

– {S / S + 0b111_1110 + RnW = 0 + Sr/Sr+*} + 7-bit StaAddr + RnW + (8-bit data + T)* + Sr/P.
– After an S, depending on NOARBH, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + RnW = 0) if needed (if it follows a previous message without ending by a P (stop)).

Others: reserved

Condition: when I3C acts as I3C target

1000: **hot-join request** (W) (refer to *Figure 419*)

The transferred hot-join request is {S +} 0b000_0010 addr + RnW = 0.
Writing the control word initiates the hot-join request if target is allowed to do so (HJEN = 1 in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.

1001: **controller-role request** (W) (refer to *Figure 420*)

The transferred controller-role request is {S +} DA[6:0] + RnW = 0 (DA in the I3C_DEVR0 register)
Writing the control word initiates the controller-role request if target is allowed to do so (CREN = 1 and DAVAL = 1 in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.

1010: **IBI (in-band interrupt) request** (R) (refer to *Figure 418*)

Bits[15:0] (DCNT[15:0]) are the number of the IBI data payload (including the first MDB), if any.
The transferred IBI request is {S +} DA[6:0] + RnW = 1 + optional IBI data payload. Writing the control word initiates the IBI request if target is allowed to do so (IBIEN = 1 and DAVAL = 1 in the I3C_DEVR0.register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.
When acknowledged from controller, the transmitted IBI payload data (optional, depending upon BCR2 in the I3C_BCR register) is defined by DCNT[15:0] in the I3C_CR register and I3C_IBIDR, and must be consistently programmed vs. the IBI payload data size defined by IBIP[2:0] in the I3C_IBIDR register.

Others: reserved

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:17 **ADD[6:0]**: 7-bit I3C dynamic / I^2C static target address (when I3C acts as controller)
When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message)

Bit 16 **RNW**: Read / non-write message (when I3C acts as controller)
When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message), to emit the RnW bit on the I3C bus.
0: write message
1: read message

Bits 15:0 **DCNT[15:0]**: Count of data to transfer during a read or write message, in bytes (whatever I3C acts as controller/target)
When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message), to set the number of exchanged data bytes on the bus. In case of a private or legacy I^2C read/write message, this field must be non-null.
When I3C acts as target, this field is used if MTYPE[3:0] = 1010 (IBI request) and if any IBI data payload (data to be transmitted if BCR2 = 1 in the I3C_BCR register), to set the number of bytes of the IBI data payload (1, 2, 3, or 4).
Linear encoding up to 64 Kbytes - 1
0x0000: no data to transfer
0x0001: 1 byte
0x0002: 2 bytes
...
0xFFFF: 64 Kbytes - 1 byte

35.16.2 I3C message control register [alternate] (I3C_CR)

Address offset: 0x000

Reset value: 0x0000 0000

This write register description must be used to control the message for when the controller has to emit a CCC (whatever is the type of the CCC: for a CCC broadcast, a CCC direct, or a CCC Enter HDR).

This is the alternate description of register I3C_CR, for when MTYPE[3:0] = 0110. Else refer to *I3C message control register (I3C_CR)*.

If the control FIFO (also known as C-FIFO) is not full (CFNFF = 1 in the I3C_EVR register), writing into this register means pushing a new control word into the C-FIFO; either by the software or automatically by DMA, as defined by the CDMAEN bit in the I3C_CFGR register.

When the last message of the frame is completed (a message with MEND = 1 in the I3C_CR register), the I3C hardware asserts the frame completed flag (FCF = 1 in the I3C_EVR register) and the corresponding interrupt, if enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEND	MTYPE[3:0]				Res.	Res.	Res.	CCC[7:0]							
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
DCNT[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **MEND**: Message end type/last message of a frame (when I3C acts as controller)

0: this message from controller is followed by a repeated start (Sr), before another message must be emitted

1: the message from the controller ends with a stop (P), being the last message of a frame

Bits 30:27 **MTYPE[3:0]**: Message type (when I3C acts as controller)

Condition: when I3C acts as I3C controller

0110: broadcast/direct CCC command (refer to *Table 281*, *Figure 408*, *Figure 409*, *Figure 410*)

Bits[23:16] (CCC[7:0]) are the emitted 8-bit CCC code

Bits[15:0] (DCNT[15:0]) are the number of the CCC defining bytes, or CCC sub-command bytes, or CCC data bytes.

If Bit[23] = CCC[7] = 1: this is the first part of an I3C SDR direct CCC command

The transferred direct CCC command (first part) message is:

– {S / S + 0b111_1110 +RnW = 0 / Sr+*} + (direct CCC + T) + (8-bit Data + T)* + Sr

– After an S (start), depending upon NOARBH in the I3C_CFGR register, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.

– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + R/W).

If Bit[23] = CCC[7] = 0: this is an I3C SDR broadcast CCC command (including specific ENTDAA, refer to *Figure 409*)

The transferred broadcast CCC command message is:

– {S / S + 0b111_1110 + RnW = 0 / Sr+*} + (broadcast CCC + T) + (8-bit Data + T)* + Sr/P

– After an S (start), depending on NOARBH, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.

– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + R/W).

Others: reserved

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:16 **CCC[7:0]**: 8-bit CCC code (when I3C acts as controller)

If bit[23] = CCC[7] = 1, this is the first part of an I3C SDR direct CCC command.

If bit[23] = CCC[7] = 0, this is an I3C SDR broadcast CCC command (including ENTDAA).

Bits 15:0 **DCNT[15:0]**: Count of related data to the CCC command to transfer as CCC defining bytes, or CCC sub-command bytes, or CCC data bytes, in bytes

Linear encoding up to 64 Kbytes - 1.

0x0000: no data to transfer.

Note: Value mandatory when emitting ENTDAA broadcast CCC (refer to Figure 409).

0x0001: 1 byte

Note: Value mandatory when emitting RSTACT direct/broadcast CCC (refer to Figure 410).

0x0002: 2 bytes

...

0xFFFF: 64 Kbytes - 1 byte

35.16.3 I3C configuration register (I3C_CFGR)

Address offset: 0x004

Reset value: 0x0000 0000

This register is used to configure:

- features that apply when the I3C acts as controller or target: RX-FIFO and TX-FIFO management (RXDMAEN, RXTHRES, RXFLUSH, TXDMAEN, TXTHRES, TXFLUSH), I3C peripheral role (CRINIT)
- dedicated features when the I3C acts as a controller: frame-based control-word triggering (TSFSET), FIFOs management (TMODE, SMODE, SFLUSH, SDMAEN, CDMAEN), and miscellaneous ones (HJACK, HKSDAEN, EXITPTRN, RSTPATRN, NOARBH)

The configuration fields CRINIT, HKSDAEN can be modified only when EN = 0. This condition is respected if they are modified at the same time when EN is set to 1 (it is not necessary to set EN later on, with another write operation).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TSFSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFLUSH	CDMA EN	TMODE	SMODE	SFLUSH	SDMA EN
	w									w	rw	rw	rw	w	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TX THRES	TX FLUSH	TX DMAEN	Res.	RX THRES	RX FLUSH	RX DMAEN	HJACK	Res.	HK SDAEN	EXIT PTRN	RST PTRN	NO ARBH	CRINIT	EN
	rw	w	rw		rw	w	rw	rw		rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TSFSET**: Frame transfer set (software trigger) (when I3C acts as controller)

This bit can only be written. When I3C acts as I3C controller:

0: no action

1: setting this bit initiates a frame transfer by causing the hardware to assert the flag CFNFF in the I3C_EVR register (C-FIFO not full and a control word is needed)

Note: If this bit is not set, the other alternative for the software to initiate a frame transfer is to directly write the first control word register (I3C_CR) while C-FIFO is empty (CFEF = 1 in the I3C_EVR register). Then, if the first written control word is not tagged as a message end (MEND = 0 in the I3C_CR register), it causes the hardware to assert CFNFF.

Bits 29:24 Reserved, must be kept at reset value.

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **CFLUSH**: C-FIFO flush (when I3C acts as controller)

This bit can only be written.

0: no action

1: flush C-FIFO

Bit 20 **CDMAEN**: C-FIFO DMA request enable (when I3C acts as controller)

When I3C acts as controller:

0: DMA mode is disabled for C-FIFO

- Software writes and pushes control word(s) into C-FIFO (writes I3C_CR register), as needed for a given frame
- A next control word transfer can be written by software either via polling on the flag CFNFF = 1 in the I3C_EVR register, or via interrupt notification (enabled by CFNFIE = 1 in the I3C_IER register).

1: DMA mode is enabled for C-FIFO

- DMA writes and pushes control word(s) into C-FIFO (writes I3C_CR register), as needed for a given frame.
- A next control word transfer is automatically written by the programmed hardware (via the asserted C-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 19 **TMODE**: Transmit mode (when I3C acts as controller)

When I3C acts as controller, this bit is used for the C-FIFO and TX-FIFO management vs. the emitted frame on the I3C bus.

0: C-FIFO and TX-FIFO are not preloaded before starting to emit a frame transfer.

A frame transfer starts as soon as the first control word is present in C-FIFO.

1: C-FIFO and TX-FIFO are first preloaded (also TX-FIFO if needed, depending on the frame format) before starting to emit a frame transfer. Refer to *Section 35.10.2* for more details.

Bit 18 **SMODE**: S-FIFO enable / status receive mode (when I3C acts as controller)

When I3C acts as controller, this bit is used to enable the FIFO for the status (S-FIFO) of the exchanged message on the I3C bus.

When I3C acts as target, this bit must be cleared.

0: S-FIFO is disabled

- Status register (I3C_SR) is used without FIFO mechanism.
- There is no SCL stalling if a new status register content is not read.
- Status register must be read before being overwritten by the hardware.
- Must have SDMAEN = 0 in the I3C_CFGR register.

1: S-FIFO is enabled.

- Each message status must be read.
- There is SCL stalling when the S-FIFO is full and a next message status must be read.
- S-FIFO overrun error is reported after the maximum SCL clock stalling time.

Bit 17 **SFLUSH**: S-FIFO flush (when I3C acts as controller)

This bit can be written and used only when I3C acts as controller.

0: no action

1: flush S-FIFO

Bit 16 **SDMAEN**: S-FIFO DMA request enable (when I3C acts as controller)

This bit must be cleared if SMODE = 0 in the I3C_CFGR register (S-FIFO is disabled). In other words, DMA mode cannot be used if S-FIFO is disabled. Then the status register I3C_SR can be read or not.

This bit can be set or cleared if SMODE = 1 (S-FIFO is enabled). In other words, status register I3C_SR must be read for each message, either by software, or via an allocated DMA channel.

0: DMA mode is disabled for reading status register I3C_SR

- SMODE = 0: software can read the I3C_SR register after a completed frame (FCF = 1 in the I3C_EVR register) or an error (ERRF = 1 in the I3C_EVR register). Via polling on these register flags or via interrupt notification (enabled by FCIE = 1 and ERRIE = 1 in the I3C_IER register).
- SMODE = 1: software must read and pop a status word from S-FIFO (read I3C_SR register) after each asserted flag SFNEF = 1. Via polling on this register flag or via interrupt notification (enabled by SFNEIE = 1 in the I3C_IER register).

1: DMA mode is enabled for reading status register I3C_SR

- Must have SMODE = 1 in the I3C_CFGR register (S-FIFO enabled)
- DMA reads and pops status word(s) from S-FIFO (it reads I3C_SR register)
- Status word(s) are automatically read by the programmed hardware (via the asserted S-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TXTHRES**: TX-FIFO threshold (whatever I3C acts as controller/target)

This threshold defines, compared to the TX-FIFO level, when the TXFNFF flag is set in the I3C_EVR register (and consequently if TXDMAEN = 1 when is asserted a DMA TX request).

0: 1-byte threshold

TXFNFF is set when 1 byte must be written in TX-FIFO (in I3C_TDR).

1: 1-word / 4-byte threshold

TXFNFF is set when 1 word / 4 bytes must be written in TX-FIFO (in the I3C_TDWR register). If the a number of the last transmitted data is not a multiple of 4 bytes (XDCNT[1:0] = 00 in the I3C_SR register), only the relevant 1, 2, or 3 valid LSB bytes of the last word are taken into account by the hardware, and sent on the I3C bus.

Bit 13 **TXFLUSH**: TX-FIFO flush (whatever I3C acts as controller/target)

This bit can only be written.

When the I3C acts as target, this bit can be used to flush the TX-FIFO on a private read if the controller has aborted the data read (driven low the T bit), and there is/are remaining data in the TX-FIFO (ABT = 1, and XDCNT[15:0] in the I3C_SR register < TGTTDCNT[15:0] in the I3C_TGTTDR register).

0: no action

1: flush TX-FIFO

Bit 12 **TXDMAEN**: TX-FIFO DMA request enable (whatever I3C acts as controller/target)

0: DMA mode is disabled for TX-FIFO

- Software writes and pushes a data byte/word into TX-FIFO (writes I3C_TDR or I3C_TDWR register), to be transmitted over the I3C bus.
- A next data byte/word must be written by the software either via polling on the flag TXFNFF = 1 or via interrupt notification (enabled by TXFNFIE = 1).

1: DMA mode is enabled for TX-FIFO

- DMA writes and pushes data byte(s)/word(s) into TX-FIFO (writes I3C_TDR or I3C_TDWR register).
- A next data byte/word transfer is automatically pushed by the programmed hardware (via the asserted TX-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 11 Reserved, must be kept at reset value.

Bit 10 **RXTHRES**: RX-FIFO threshold (whatever I3C acts as controller/target)

This threshold defines, compared to the RX-FIFO level, when the RXFNEF flag in the I3C_EVR register is set (and consequently if RXDMAEN = 1 when is asserted a DMA RX request).

0: 1-byte threshold

RXFNEF is set when 1 byte must be read in RX-FIFO (in the I3C_RDR register).

1: 1-word/4-bytes threshold

RXFNEF is set when 1 word / 4 bytes is/are to be read in RX-FIFO (in I3C_RDWR). In the case of a number of last received data being not a multiple of 4 bytes, only the relevant 1, 2 or 3 valid LSB bytes of the last word are to be considered by the software. The number of effective received data bytes is reported by XDCNT[15:0] in the I3C_SR register.

Bit 9 **RXFLUSH**: RX-FIFO flush (whatever I3C acts as controller/target)

This bit can only be written.

0: no action

1: flush RX-FIFO

Bit 8 **RXDMAEN**: RX-FIFO DMA request enable (whatever I3C acts as controller/target)

0: DMA mode is disabled for RX-FIFO

- Software reads and pops a data byte/word from RX-FIFO (it reads I3C_RDR or I3C_RDWR register).
- A next data byte/word must be read by the software either via polling flag RXFNEF = 1 in the I3C_EVR register, or via interrupt notification (enabled by RXFNEIE = 1 in the I3C_IER register).

1: DMA mode is enabled for RX-FIFO

- DMA reads and pops data byte(s)/word(s) from RX-FIFO (reads I3C_RDR or I3C_RDWR register).
- A next data byte/word is automatically read by the programmed hardware (via the asserted RX-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 7 **HJACK**: Hot-join request acknowledge (when I3C acts as a controller)

0: hot-join request is not acknowledged

After the NACK, the controller continues as initially programmed (the hot-joining target is aware of the NACK and must emit another hot-join request later on).

1: hot-join request is acknowledged

After the ACK, the controller continues as initially programmed. The software is notified by the HJ interrupt (flag HJF is set in the I3C_EVR register), and must initiate the ENTDAA sequence later on, potentially preventing other hot-join requests with a disable target events command (DISEC, with DISHJ = 1).

Bit 6 Reserved, must be kept at reset value.

Bit 5 **HKSDAEN:** High-keeper enable on SDA line (when I3C acts as a controller)

0: High-keeper is disabled

1: High-keeper is enabled, and the weak pull-up is effective on the T bit, instead of the open-drain class pull-up.

Note: This bit can be modified only when EN = 0 in the I3C_CFGR register.

Bit 4 **EXITPTRN:** HDR exit pattern enable (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: HDR exit pattern is not sent after the issued message header (MTYPE[3:0] = 0001 in the I3C_CR register). This is used to send the header, to test ownership of the bus when there is a suspicion of a problem after controller-role hand-off (new controller did not assert its controller-role by accessing the previous one in less than the delay defined by the activity state).

1: HDR exit pattern is sent after the issued message header (MTYPE[3:0] = 0001). This is used on a controller error detection and escalation handling, in case of a not responding target to a private message or a direct read CCC.

The HDR exit pattern is sent whatever the message header {S/Sr + 0x7E addr + W} is ACK-ed or NACK-ed..

Bit 3 **RSTPTRN:** HDR reset pattern enable (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: standard stop emitted at the end of a frame

1: HDR reset pattern is inserted before the stop of any emitted frame that includes a RSTACT CCC command

Bit 2 **NOARBH**: No arbitrable header after a start (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: An arbitrable header (0b111_1110 + RnW = 0) is emitted after a start and before a legacy I^2C message or an I3C SDR private read/write message (default).

1: No arbitrable header

- The target address is emitted directly after a start in case of a legacy I^2C message or an I3C SDR private read/write message.

- This is a more performing option (when the emission of the 0x7E arbitrable header is useless), but must be used only when the controller is sure that the addressed target device cannot emit concurrently an IBI or a controller-role request (to ensure no misinterpretation and no potential conflict between the address emitted by the controller in open-drain mode and the same address a target device can emit after a start, for IBI or MR).

Bit 1 **CRINIT**: Initial controller/target role

This bit can be modified only when EN = 0 in the I3C_CFGR register.

0: target role

Once enabled by setting EN = 1, the I3C peripheral initially acts as a target. I3C does not drive SCL line and does not enable SDA pull-up, until it eventually acquires the controller role.

1: controller role

Once enabled by setting EN = 1, the I3C peripheral initially acts as a controller. It has the I3C controller role, so drives SCL line and enables SDA pull-up, until it eventually offers the controller role to an I3C secondary controller.

Bit 0 **EN**: I3C enable (whatever I3C acts as controller/target)

0: I3C is disabled

- Except registers, the I3C peripheral is under reset (partial reset).

- Before clearing EN, when I3C acts as a controller, all the possible target requests must be disabled using DISEC CCC.

- When I3C acts as a target, software must not disable the I3C, unless a partial reset is needed.

1: I3C is enabled

In this state, some register fields cannot be modified (like CRINIT, HKSDAEN for the I3C_CFGR)

35.16.4 I3C receive data byte register (I3C_RDR)

Address offset: 0x010

Reset value: 0x0000 0000

This register is used to read received data bytes from the I3C bus if the RX-FIFO is configured with a byte-based read access (RXTHRES = 0 in the I3C_CFGR register). Then:

- On read, I3C_RDR returns a 32-bit data word, with the received data byte in LSB position, and other reserved bits read as 0.
- When RXDMAEN = 1 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
- When RXDMAEN = 0: before a read, the software must wait to be notified that a next received data byte must be read via the RX-FIFO non empty flag (RXFNEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last received byte of a given message to be read is also marked with RXLASTF = 1 in the I3C_EVR register when acting as controller.
- If the RX-FIFO is full, a new byte is received and cannot be pushed into the RX-FIFO without waiting anymore, the software is notified by error flag ERRF = 1 in the I3C_EVR register and by a data overrun flag DOVRF = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	RDB0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RDB0[7:0]**: 8-bit received data on I3C bus.

35.16.5 I3C receive data word register (I3C_RDWR)

Address offset: 0x014

Reset value: 0x0000 0000

This register is used to read received data bytes from the I3C bus if the RX-FIFO is configured with a 32-bit word-based read access (RXTHRES = 1 in the I3C_CFGR register). Then:

- When RXDMAEN = 1 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
- When RXDMAEN = 0: before a read, the software must first wait to be notified that must be read a next received data word via the RX-FIFO non empty flag (RXFNEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last received

word/byte(s) of a given message to be read is also marked with RXLASTF = 1 in the I3C_EVR register when acting as controller.

- If the RX-FIFO holds less than four bytes, a read on I3C_RDWR returns a data word padded with null byte(s): the available byte(s) in LSB position(s) is (are) padded with zero byte(s) in MSB position(s).
- If the RX-FIFO is full and a new byte is received and cannot be pushed into the RX-FIFO without waiting anymore, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data overrun DOVR = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDB3[7:0]								RDB2[7:0]							
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
RDB1[7:0]								RDB0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **RDB3[7:0]**: 8-bit received data (latest byte on I3C bus).

Bits 23:16 **RDB2[7:0]**: 8-bit received data (next byte after RDB1 on I3C bus).

Bits 15:8 **RDB1[7:0]**: 8-bit received data (next byte after RDB0 on I3C bus).

Bits 7:0 **RDB0[7:0]**: 8-bit received data (earliest byte on I3C bus).

35.16.6 I3C transmit data byte register (I3C_TDR)

Address offset: 0x018

Reset value: 0x0000 0000

This register is used to write data bytes to be transmitted over the I3C bus.

This register implements a byte-based write access to the transmit FIFO (TX-FIFO), and is used when TXTHRES = 0 in the I3C_CFGR register.

When the I3C acts as controller:

- When TXDMAEN = 1 and if TXTHRES = 0 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive writes.
- When TXDMAEN = 0 and if TXTHRES = 0: before a write, the software must wait to be notified that must be written a next data byte via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last transmitted byte of a given message to be written is also marked with TXLASTF = 1 in the I3C_EVR register.

When the I3C acts as target:

- When TXDMAEN = 1 and if TXTHRES = 0 and if PRELOAD = 1 in the I3C_TGTTDR register: programmed I3C and DMA automatically manage by hardware the relevant and successive number of writes to I3C_TDR register, according to TGTTDCNT[15:0] in the I3C_TGTDR register.
- When TXDMAEN = 0 and if TXTHRES = 0:
  - when PRELOAD = 1 in the I3C_TGTTDR register: before a write, the software must wait to be notified that must be written a next data byte via the TX-FIFO non

full flag (TXFNFF = 1) or via the corresponding interrupt, if enabled. The last transmitted byte of the initially programmed TGTTDCNT[15:0] is also marked with TXLASTF = 1 in the I3C_EVR register.

– when PRELOAD = 0: before a write, the software must wait to be notified that can be written up to 8 next data bytes (TX-FIFO size) via the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register), or via the corresponding interrupt if enabled. If the software needs to write another data byte, it must wait for the TX-FIFO be empty (TXFEF = 1), and then write this data byte to be transmitted before nine SCL clock periods elapse, to avoid a data underrun error flag (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register).

If the TX-FIFO is empty and the controller cannot wait longer a data byte to be transmitted, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data underrun flag DOF = 1 (and corresponding interrupt if enabled) in the I3C_SER 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.	TDB0[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TDB0[7:0]**: 8-bit data to transmit on I3C bus.

35.16.7 I3C transmit data word register (I3C_TDWR)

Address offset: 0x01C

Reset value: 0x0000 0000

This register is used to write 32-bit data words to be transmitted over the I3C bus.

This register implements a word-based write access to the transmit FIFO (TX-FIFO), and is used when TXTHRES = 1 in the I3C_CFGR register.

When the I3C acts as controller:

- When TXDMAEN = 1 and if TXTHRES = 1 in the I3C_CFGR register, programmed I3C and DMA automatically manage by hardware the relevant and successive writes.
- When TXDMAEN = 0 and if TXTHRES = 1: before a write, the software must wait to be notified that next data word/byte(s) must be written via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register), or via the corresponding interrupt if enabled. Last transmitted word/byte(s) of a given message to be written in TX-FIFO is also marked with TXLASTF = 1 in the I3C_EVR register.

When the I3C acts as target:

- When TXDMAEN = 1 and if TXTHRES = 1 and if PRELOAD = 1 in the I3C_TGTTDR register: programmed I3C and DMA automatically manage by hardware the relevant and successive number of writes to the I3C_TDWR register, as per TGTTDCNT[15:0]

in the I3C_TGTDR register.

- When TXDMAEN = 0 and if TXTHRES = 1:
  - when PRELOAD = 1: before a write, the software must wait to be notified that must be written a next data word via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register) or via the corresponding interrupt, if enabled. Last transmitted word of the initially programmed TGTTDCNT[15:0] in the I3C_TGTTDR register is also marked with TXLASTF = 1 in the I3C_EVR register.
  - when PRELOAD = 0: before a write, the software must wait to be notified that can be written up to two next data words (TX-FIFO size) via the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. If the software needs to write another data word, it must wait for TX-FIFO to be empty (TXFEF = 1), and then write the next data word to be transmitted before nine SCL clock periods elapse, to avoid a data underrun error flag (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register).

If the TX-FIFO is empty and the controller/target cannot wait longer a data byte to be transmitted, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data underrun flag DOF = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDB3[7:0]								TDB2[7:0]							
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
TDB1[7:0]								TDB0[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:24 **TDB3[7:0]**: 8-bit transmit data (latest byte on I3C bus).

Bits 23:16 **TDB2[7:0]**: 8-bit transmit data (next byte after TDB1[7:0] on I3C bus).

Bits 15:8 **TDB1[7:0]**: 8-bit transmit data (next byte after TDB0[7:0] on I3C bus).

Bits 7:0 **TDB0[7:0]**: 8-bit transmit data (earliest byte on I3C bus)

35.16.8 I3C IBI payload data register (I3C_IBIDR)

Address offset: 0x020

Reset value: 0x0000 0000

This register is used for the IBI payload data.

When I3C acts as target:

- if BCR2 = 0 in the I3C_BCR register, this register is useless.
- if BCR2 = 1, this register must be written by software
  - to be emitted on the I3C bus as the IBI payload data, after that the IBI request (MTYPE[3:0] = 1010 in the I3C_CR register) is acknowledged by the controller; with the IBI data payload size defined by DCNT[15:0] in the I3C_CR register.
  - Maximum (static) payload data size is given by IBIP[2:0] in the I3C_MAXRLR register. it can be 1, 2, 3 or 4 bytes.
  - DCNT[15:0] must be set between 1 (for the mandatory data byte MDB[7:0]) and the maximum IBIP[2:0].

When I3C acts as controller: if IBIACK= 1 in the I3C_DEVRx register, once it acknowledges on the I3C bus the IBI request from the target x:

- if IBIDEN = 0 (BCR[2] = 0 received from target x) in the I3C_DEVRx register, this register is useless.
- if IBIDEN = 1 (BCR[2] = 1 received from target x):
  - This register is internally written by hardware from the IBI payload data received on the I3C bus (including the first mandatory data byte MDB[7:0]).
  - When the last byte of the payload is received in I3C_IBIDR (when target drives T-bit = 0), IBIF flag is set (and corresponding interrupt if enabled) in the I3C_EVR register.
  - Then, the software can identify the target x via the received and logged 7-bit address in RADD[6:0] in the I3C_RMR register.
  - The software can read this register and interpret byte(s) according to the number of bytes effectively received and logged in IBIRDCNT[2:0] in the I3C_RMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IBIDB3[7:0]								IBIDB2[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
IBIDB1[7:0]								IBIDB0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **IBIDB3[7:0]**: 8-bit IBI payload data (latest byte on I3C bus).

Bits 23:16 **IBIDB2[7:0]**: 8-bit IBI payload data (next byte on I3C bus after IBIDB1[7:0]).

Bits 15:8 **IBIDB1[7:0]**: 8-bit IBI payload data (next byte on I3C bus after IBIDB0[7:0]).

Bits 7:0 **IBIDB0[7:0]**: 8-bit IBI payload data (earliest byte on I3C bus, MDB[7:0] mandatory data byte).

35.16.9 I3C target transmit configuration register (I3C_TGTTDR)

Address offset: 0x024

Reset value: 0x0000 0000

When I3C acts as target, this register must be used to preload a number of data bytes in the TX-FIFO, so that they are ready to be transmitted on the I3C bus, when they are accepted/acknowledged by the controller, whatever the DMA mode is used or not (independently from TXDMAEN in the I3C_CFGR register).

When I3C acts as target, alternatively, if the number of data bytes to be transmitted is less than or equal to the TX-FIFO size (8 bytes), the software can directly use and write byte(s) into the I3C_TDR register (or I3C_TDWR register, depending upon TXTHRES in the I3C_CFGR register), via polling on the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register), or via the corresponding enabled interrupt.

In any case, since an I3C target is unable to stretch/stall the SCL line, the software can identify a TX-FIFO underrun via DOR = 1 in the I3C_EVR 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.	PRELOAD
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TGTTDCNT[15:0]															
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.

Bit 16 **PRELOAD**: Preload of the TX-FIFO (when I3C is configured as target)

This bit must be written and asserted by software in the same access when is written and defined the number of bytes to preload into the TX-FIFO and to transmit.

This bit is cleared by hardware when all the data bytes to transmit are loaded into the TX-FIFO.

0: no TX-FIFO preload

1: TX-FIFO preload

Bits 15:0 **TGTTDCNT[15:0]**: Transmit data counter, in bytes (when I3C is configured as target)

This bitfield must be written by software in the same access when is asserted PRELOAD, in order to define the number of bytes to preload and to transmit.

This bitfield is updated by hardware and reports, when read, the remaining number of bytes to be loaded into the TX-FIFO.

35.16.10 I3C status register (I3C_SR)

Address offset: 0x030

Reset value: 0x0000 0000

This register is used to read the status about the exchanged message on the I3C bus:

- in FIFO mode: when the I3C acts as controller and if S-FIFO is enabled via SMODE = 1 in the I3C_CFGR register:
  - Software is notified via SFNEF = 1 in the I3C_EVR register if there is a status register to be read (and corresponding interrupt if enabled), when not in DMA mode (SDMAEN = 0)
  - In DMA mode (SDMAEN = 1), programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
  - Software is notified via COVR = 1 in the I3C_SER register on an S-FIFO overflow (and ERRF = 1 in the I3C_EVR register and corresponding interrupt if enabled)
- in register mode: if SMODE = 0 in the I3C_CFGR register
  - Software can use the flags FCF in the I3C_SER register and ERRF = 1 in the I3C_EVR register (and corresponding interrupt if enabled) to read this register
  - This register can be overwritten by hardware on a new message completion, without any notification.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MID[7:0]								Res.	Res.	Res.	Res.	Res.	DIR	ABT	Res.
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
XDCNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **MID[7:0]**: Message identifier/counter of a given frame (when the I3C acts as controller)

When I3C acts as controller, this fbitield identifies the control word message (I3C_CR) to whom the I3C_SR status register refers.

First message of a frame is identified with MID[7:0] = 0.

This bitfield is incremented (by hardware) on the completion of a new message control word (I3C_CR) over I3C bus. This field is reset for every new frame start.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **DIR**: Message direction

Whatever the I3C acts as controller or target, this bit indicates the direction of the related message on the I3C bus

0: write

1: read

Note: ENTDAA CCC is considered as a write command.

Bit 17 **ABT**: A private read message is ended prematurely by the target (when the I3C acts as controller)

When the I3C acts as controller, this bit indicates if the private read data transmitted by the target early terminates (the target drives T bit low earlier vs. what the controller expects in terms of programmed number of read data bytes DCNT[15:0] in the I3C_CR register).

0: no early completion from the target

1: early completion from the target

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **XDCNT[15:0]**: Data counter

Condition: during the dynamic address assignment process (ENTDAA CCC)

- When the I3C acts as controller: number of targets detected on the bus
- When the I3C acts as target: number of transmitted bytes

Condition: for other transfers, during the message

- Whatever the I3C acts as controller or target: number of data bytes read from or transmitted on the I3C bus during the message

35.16.11 I3C status error register (I3C_SER)

Address offset: 0x034

Reset value: 0x0000 0000

This read register is used to get more information about the error when an error is raised by hardware and notified to the software via the error flag ERRF = 1 in the I3C_EVR register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	DERR	DNACK	ANACK	COVR	DOVR	STALL	PERR	CODERR[3:0]			
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DERR**: Data error (when the I3C acts as controller)

0: no detected error

1: controller detected a data error during the controller-role hand-off procedure (GETACCCR CCC, formerly known as GETACCMST) when the received target address or/and the parity bit do no match. Active controller keeps controller-role.

Bit 9 **DNACK**: Data not acknowledged (when the I3C acts as controller)

0: no detected error

1: controller detected that a data byte is not acknowledged by a target, either during:

i) a legacy I2C write transfer

ii) the second trial when sending dynamic address during ENTDAA procedure

Bit 8 **ANACK**: Address not acknowledged (when the I3C is configured as controller)

0: no detected error

1: controller detected that the static/dynamic address was not acknowledged by a target, either during:

i) a legacy I2C read/write transfer

ii) a direct CCC write transfer

iii) the second trial of a direct CCC read transfer

iv) a private read/write transfer

Bit 7 **COVR**: C-FIFO underrun or S-FIFO overrun (when the I3C acts as controller)

0: no detected error

1: controller detected either:

i) a C-FIFO underrun: control FIFO is empty and a restart must be emitted

ii) an S-FIFO overrun: S-FIFO is full and a new message ends

Bit 6 **DOVR**: RX-FIFO overrun or TX-FIFO underrun

0: no detected error

1: whatever controller or target, hardware detected either:

i) a TX-FIFO underrun: TX-FIFO is empty and a write data byte must be transmitted

ii) a RX-FIFO overrun: RX-FIFO is full and a new data byte is received

Bit 5 **STALL**: SCL stall error (when the I3C acts as target)

0: no detected error

1: target detected that SCL was stable for more than 125 µs during an I3C SDR data read (during a direct CCC read, a private read, or an IB)

Bit 4 **PERR**: Protocol error

0: no detected error

1: whatever controller or target, hardware detected a protocol error, as detailed in CODERR[3:0]

Bits 3:0 **CODERR[3:0]**: Protocol error code/type
0000: CE0 error (transaction after sending CCC):
controller detected an illegally formatted CCC
0001: CE1 error (monitoring error):
controller detected that transmitted data on the bus is different from expected
0010: CE2 error (no response to broadcast address):
controller detected a not acknowledged broadcast address (0b111_1110)
0011: CE3 error (failed controller-role hand-off):
controller detected the new controller did not drive bus after controller-role hand-off
1000: TE0 error (invalid broadcast address 0b111_1110 + W):
target detected an invalid broadcast address 0b111_1110 + W
1001: TE1 error (CCC code):
target detected a parity error on a CCC code via a parity check (vs. T bit)
1010: TE2 error (write data):
target detected a parity error on a write data via a parity check (vs. T bit)
1011: TE3 error (assigned address during dynamic address arbitration):
target detected a parity error on the assigned address during dynamic address arbitration via a parity check (vs. PAR bit)
1100: TE4 error (0b111_1110 + R missing after Sr during dynamic address arbitration):
target detected a 0b111_1110 + R missing after Sr during dynamic address arbitration
1101: TE5 error (transaction after detecting CCC):
target detected an illegally formatted CCC
1110: TE6 error (monitoring error):
target detected that transmitted data on the bus is different from expected
others: reserved

35.16.12 I3C received message register (I3C_RMR)

Address offset: 0x040

Reset value: 0x0000 0000

When the I3C acts as controller, this read register is used to log the received target address, and the IBI received payload data size.

When the I3C acts as target, this read register is used to log the received CCC code

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **RADD[6:0]**: Received target address (when the I3C is configured as controller)
When the I3C is configured as controller, this field logs the received dynamic address from the target during acknowledged IBI or controller-role request.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 **RCODE[7:0]**: Received CCC code (when the I3C is configured as target)
When the I3C is configured as target, this field logs the received CCC code.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **IBIRDCNT[2:0]**: IBI received payload data count (when the I3C is configured as controller)
When the I3C is configured as controller, this field logs the number of data bytes effectively received in the I3C_IBIDR register.

35.16.13 I3C event register (I3C_EVR)

Address offset: 0x050

Reset value: 0x0000 0003

This is a read register, used for reporting event flags.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPF	DEFF	INT UPDF	AS UPDF	RSTF	MRL UPDF	MWL UPDF	DA UPDF	STAF	GETF	WKPF	Res.	HJF	CR UPDF	CRF	IBI ENDF
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
IBIF	Res.	Res.	Res.	ERRF	RXTGT ENDF	FCF	Res.	RX LASTF	TX LASTF	RX FNEF	TX FNFF	SFNEF	CFNFF	TXFEF	CFEF
r				r	r	r		r	r	r	r	r	r	r	r

Bit 31 **GRPF**: Group addressing flag (when the I3C acts as target)
When the I3C acts as target (and is typically controller-capable), this flag is asserted by hardware to indicate that the broadcast DEFGRPA CCC (define list of group addresses) has been received. Then, software can store the received data for when getting controller role.
The flag is cleared when software writes 1 into the corresponding CGRPF bit in the I3C_CR register.

Bit 30 **DEFF**: DEFTGTS flag (when the I3C acts as target)
When the I3C acts as target (and is typically controller capable), this flag is asserted by hardware to indicate that the broadcast DEFTGTS CCC (define list of targets) has been received. Then, software can store the received data for when getting the controller role.
The flag is cleared when software writes 1 into the corresponding CDEFF bit in the I3C_CEVR register.

Bit 29 **INTUPDF**: Interrupt/controller-role/hot-join update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that the direct or broadcast ENEC/DISEC CCC (enable/disable target events) has been received, where a target event is either an interrupt/IBI request, a controller-role request, or an hot-join request. Then, software must read respectively IBIEN, CREN, or HJEN in the I3C_DEVR0 register.
The flag is cleared when software writes 1 into the corresponding CINTUPDF bit in the I3C_CEVR register.

Bit 28 **ASUPDF**: Activity state update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that the direct or broadcast ENTASx CCC (with x = 0...3) has been received. Then, software must read AS[1:0] in the I3C_DEVR0 register.
The flag is cleared when software writes 1 into the corresponding CASUPDF bit in the I3C_CEVR register.

Bit 27 **RSTF**: Reset pattern flag (when the I3C acts as target)

When I3C acts as target, this flag is asserted by hardware to indicate that a reset pattern has been detected (14 SDA transitions while SCL is low, followed by repeated start, then stop).

Then, when not in Stop mode, software must read RSTACT[1:0] and RSTVAL in the I3C_DEVR0 register, to know the required reset level.

- If RSTVAL = 1: when the RSTF is asserted (and/or the corresponding interrupt if enabled), RSTACT[1:0] in the I3C_DEVR0 register dictates the reset action to be performed by the software, if any.
- If RSTVAL = 0: when the RSTF is asserted (and/or the corresponding interrupt if enabled), the software must issue an I3C reset after a first detected reset pattern, and a system reset on the second one.

When in Stop mode, the corresponding interrupt can be used to wake up the device.

The flag is cleared when software writes 1 into the corresponding CRSTF bit in the I3C_CEVR register.

Bit 26 **MRLUPDF**: Maximum read length update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct SETMRL CCC (set max read length) has been received. Then, software must read MRL[15:0] in the I3C_MAXRLR register to get the maximum read length value.

The flag is cleared when software writes 1 into the corresponding CMRLUPDF bit in the I3C_CEVR register.

Bit 25 **MWLUPDF**: Maximum write length update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct SETMWL CCC (set max write length) has been received. Then, software must read MWL[15:0] in the I3C_MAXRLR register to get the maximum write length value.

The flag is cleared when software writes 1 into the corresponding CMWLUPDF bit in the I3C_CEVR register.

Bit 24 **DAUPDF**: Dynamic address update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a dynamic address update has been received via any of the broadcast ENTDAA, RSTDAA and direct SETNEWDA CCC. Then, software must read DA[6:0] and DAVAL in the I3C_DEVR0 register to get the dynamic address update.

The flag is cleared when software writes 1 into the corresponding CDAUPDF bit in the I3C_CEVR register.

Bit 23 **STAF**: Get status flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct GETSTATUS CCC of format 1 (without defining byte or with defining byte TGTSTAT) has been received.

The flag is cleared when software writes 1 into the corresponding CSTAF bit in the I3C_CEVR register.

Bit 22 **GETF**: Get flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that any direct CCC of get type (GET*** CCC) except the GETSTATUS of format 1 (but including GETSTATUS of format 2) has been received.

The flag is cleared when software writes 1 into the corresponding CGETF bit in the I3C_CEVR register.

Bit 21 **WKPF**: Wake-up/missed start flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a start has been detected (an SDA falling edge followed by an SCL falling edge) but on the next SCL falling edge, the I3C kernel clock is (still) gated. Thus an I3C bus transaction may have been lost by the target.

The corresponding interrupt can be used to wake up the device from a low power (Sleep or Stop) mode.

The flag is cleared when software writes 1 into the corresponding CWKPF bit in the I3C_CEVR register.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJF**: Hot-join flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that an hot join request has been received.

The flag is cleared when software writes 1 into the corresponding CHJF bit in the I3C_CEVR register.

Bit 18 **CRUPDF**: Controller-role update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that it has now gained the controller role after the completed controller-role hand-off procedure.

The flag is cleared when software writes 1 into the corresponding CCRUPDF bit in the I3C_CEVR register.

Bit 17 **CRF**: Controller-role request flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a controller-role request has been acknowledged and completed (by hardware). The software must then issue a GETACCCR CCC (get accept controller role) for the controller-role hand-off procedure.

The flag is cleared when software writes 1 into the corresponding CCRF bit in the I3C_CEVR register.

Bit 16 **IBIENDF**: IBI end flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that an IBI transfer has been received and completed (IBI acknowledged and IBI data bytes read by controller if any).

The flag is cleared when software writes 1 into the corresponding CIBIENDF bit in the I3C_CEVR register.

Bit 15 **IBIF**: IBI flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that an IBI request has been received.

The flag is cleared when software writes 1 into the corresponding CIBIF bit in the I3C_CEVR register.

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRF**: Flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that an error occurred.Then, software must read I3C_SER to get the error type.

The flag is cleared when software writes 1 into the corresponding CERRF bit in the I3C_CEVR register.

Bit 10 **RXTGTENDF**: Target-initiated read end flag (when the I3C acts as controller)

When the I3C acts as controller, and only if the S-FIFO is disabled (SMODE = 0 in the I3C_CFGR register), this flag is asserted by hardware to indicate that the target has prematurely ended a read transfer. Then, software must read the status register I3C_SR to check information related to the last message and get the number of received data bytes on the prematurely read transfer (XDCNT in the I3C_SR register).

The flag is cleared when software writes 1 into the corresponding CRXTGTENDF bit in the I3C_CEVR register.

Bit 9 **FCF**: Frame complete flag (whatever the I3C acts as controller/target)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a frame has been (normally) completed on the I3C bus, for example, when a stop is issued.

When the I3C acts as target, this flag is asserted by hardware to indicate that a message addressed to/by this target has been (normally) completed on the I3C bus, for example, when a next stop or repeated start is then issued by the controller.

The flag is cleared when software writes 1 into the corresponding CFCF bit in the I3C_CEVR register.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **RXLASTF**: Last read data byte/word flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that the last data byte/word (depending upon RXTHRES in the I3C_CFGR register) of a message must be read from the RX-FIFO. The flag is de-asserted by hardware when the last data byte/word of a message is read.

Bit 6 **TXLASTF**: Last written data byte/word flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that the last data byte/word (depending upon TXTHRES in the I3C_CFGR register) of a message must be written to the TX-FIFO. The flag is de-asserted by hardware when the last data byte/word of a message is written.

Bit 5 **RXFNEF**: RX-FIFO not empty flag (whatever the I3C acts as controller/target)

This flag is asserted/de-asserted by hardware to indicate that a data byte must/must not be read from the RX-FIFO.

Note: The software must wait for RXFNEF = 1 (by polling or via the enabled interrupt) before reading from RX-FIFO (reading from I3C_RDR or I3C_RDWR, depending upon RXTHRES).

Bit 4 **TXFNFF**: TX-FIFO not full flag (whatever the I3C acts as controller/target)

This flag is asserted/de-asserted by hardware to indicate that a data byte/word must/must not be written to the TX-FIFO.

Note: The software must wait for TXFNFF = 1 (by polling or via the enabled interrupt) before writing to TX-FIFO (writing to I3C_TDR or I3C_TDWR, depending upon TXTHRES).

Note: When the I3C acts as target, if the software intends to use the TXFNFF flag for writing into I3C_TDR/I3C_TDWR, it must have configured and set the TX-FIFO preload (write PRELOAD in the I3C_TGTTDR register).

Bit 3 **SFNEF**: S-FIFO not empty flag (when the I3C acts as controller)

When the I3C acts as controller, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), this flag is asserted by hardware to indicate that a status word must be read from the S-FIFO. The flag is de-asserted by hardware to indicate that a status word is not to be read from the S-FIFO.

Bit 2 **CFNFF**: C-FIFO not full flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a control word must be written to the C-FIFO. The flag is de-asserted by hardware to indicate that a control word is not to be written to the C-FIFO.

Note: The software must wait for CFNFF = 1 (by polling or via the enabled interrupt) before writing to C-FIFO (writing to I3C_CR).

Bit 1 **TXFEF**: TX-FIFO empty flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that the TX-FIFO is empty.

This flag is de-asserted by hardware to indicate that the TX-FIFO is not empty.

Bit 0 **CFEF**: C-FIFO empty flag (whatever the I3C acts as controller)

This flag is asserted by hardware to indicate that the C-FIFO is empty when controller, and that the I3C_CR register contains no control word (none IBI/CR/HJ request) when target.

This flag is de-asserted by hardware to indicate that the C-FIFO is not empty when controller, and that the I3C_CR register contains one control word (a pending IBI/CR/HJ request) when target.

Note: When the I3C acts as controller, if the C-FIFO and TX-FIFO preload is configured (TMODE = 1 in the I3C_CFGR register), the software must wait for TXFEF = 1 and CFEF = 1 before starting a new frame transfer.

35.16.14 I3C interrupt enable register (I3C_IER)

Address offset: 0x054

Reset value: 0x0000 0000

This register is used to enable/disable, at bit level, an interrupt, for each of the following event/flag.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPIE	DEFIE	INTUPDIE	ASUPDIE	RSTIE	MRLUPDIE	MWLUPDIE	DAUPDIE	STAIE	GETIE	WKPIE	Res.	HJIE	CRUPDIE	CRIE	IBIENDIE
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
IBIIE	Res.	Res.	Res.	ERRIE	RXTGTENDIE	FCIE	Res.	Res.	Res.	RXFNEIE	TXFNFIE	SFNEIE	CFNFIE	Res.	Res.
r				r	r	r				r	r	r	r		

Bit 31 **GRPIE**: DEFGRPA CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 30 **DEFIE**: DEFTGTS CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 29 **INTUPDIE**: ENEC/DISEC CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 28 **ASUPDIE**: ENTASx CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 27 **RSTIE**: reset pattern interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 26 **MRLUPDIE**: SETMRL CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 25 **MWLUPDIE**: SETMWL CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 24 **DAUPDIE**: ENTDAA/RSTDAA/SETNEWDA CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 23 **STAIE**: format 1 GETSTATUS CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 22 **GETIE**: GETxxx CCC interrupt enable (except GETSTATUS of format 1) (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 21 **WKPIE**: Wake-up interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJIE**: Hot-join interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 18 **CRUPDIE**: Controller-role update interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 17 **CRIE**: Controller-role request interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 16 **IBIENDIE**: IBI end interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 15 **IBIIE**: IBI request interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRIE**: error interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 10 **RXTGTENDIE**: target-initiated read end interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 9 **FCIE**: frame complete interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **RXFNEIE**: RX-FIFO not empty interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 4 **TXFNFIE**: TX-FIFO not full interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 3 **SFNEIE**: S-FIFO not empty interrupt enable when the I3C acts as controller
0: interrupt disabled
1: interrupt enabled

Bit 2 **CFNFIE**: C-FIFO not full interrupt enable when the I3C acts as controller
0: interrupt disabled
1: interrupt enabled

Bits 1:0 Reserved, must be kept at reset value.

35.16.15 I3C clear event register (I3C_CEVR)

Address offset: 0x058

Reset value: 0x0000 0000

This write register is used to clear individually, at bit level, the corresponding event flag of the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CGRPF	CDEFF	CINTUPDF	CASUPDF	CRSTF	CMRLUPDF	CMWLUPDF	CDAUPDF	CSTAF	CGETF	CWKPF	Res.	CHJF	CCRUPDF	CCRF	CIBIENDF
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
CIBIF	Res.	Res.	Res.	CERRF	CRXTGTENDF	CFCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w				w	w	w									

Bit 31 **CGRPF**: Clear DEFGRPA CCC flag (when the I3C acts as target)
0: no effect
1: clear GRPF

Bit 30 **CDEFF**: Clear DEFTGTS CCC flag (when the I3C acts as target)
0: no effect
1: clear DEFF

Bit 29 **CINTUPDF**: Clear ENEC/DISEC CCC flag (when the I3C acts as target)
0: no effect
1: clear CINTUPDF

Bit 28 **CASUPDF**: Clear ENTASx CCC flag (when the I3C acts as target)
0: no effect
1: clear ASUPDF

Bit 27 **CRSTF**: Clear reset pattern flag (when the I3C acts as target)
0: no effect
1: clear RSTF

Bit 26 **CMRLUPDF**: Clear SETMRL CCC flag (when the I3C acts as target)
0: no effect
1: clear MRLUPDF

Bit 25 **CMWLUPDF**: Clear SETMWL CCC flag (when the I3C acts as target)
0: no effect
1: clear MWLUPDF

Bit 24 **CDAUPDF**: Clear ENTDAA/RSTDAA/SETNEWDA CCC flag (when the I3C acts as target)
0: no effect
1: clear DAUPDF

Bit 23 **CSTAF**: Clear format 1 GETSTATUS CCC flag (when the I3C acts as target)
0: no effect
1: clear STAF

Bit 22 **CGETF**: Clear GETxxx CCC flag (except GETSTATUS of format 1) (when the I3C acts as target)
0: no effect
1: clear GETF

Bit 21 **CWKPF**: Clear wake-up flag (when the I3C acts as target)
0: no effect
1: clear WKPF

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CHJF**: Clear hot-join flag (when the I3C acts as controller)
0: no effect
1: clear HJF

Bit 18 **CCRUPDF**: Clear controller-role update flag (when the I3C acts as target)
0: no effect
1: clear CRUPDF

Bit 17 **CCRF**: Clear controller-role request flag (when the I3C acts as controller)
0: no effect
1: clear CRF

Bit 16 **CIBIENDF**: Clear IBI end flag (when the I3C acts as target)
0: no effect
1: clear IBIENDF

Bit 15 **CIBIF**: Clear IBI request flag (when the I3C acts as controller)
0: no effect
1: clear IBIF

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **CERRF**: Clear error flag (whatever the I3C acts as controller/target)
0: no effect
1: clear ERRF

Bit 10 **CRXTGTENDF**: Clear target-initiated read end flag (when the I3C acts as controller)
0: no effect
1: clear RXTGTENDF

Bit 9 **CFCF**: Clear frame complete flag (whatever the I3C acts as controller/target)
0: no effect
1: clear FCF

Bits 8:0 Reserved, must be kept at reset value.

35.16.16 I3C own device characteristics register (I3C_DEVR0)

Address offset: 0x060

Reset value: 0x0000 0000

When the I3C peripheral acts as target, this register is used to write or read its own device characteristics.

When the I3C peripheral acts as controller, the field DA[6:0] is used to write and store its own dynamic address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTVAL	RSTACT[1:0]		AS[1:0]		HJEN	Res.	CREN	IBIEN
							r	r	r	r	r	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.	DA[6:0]							DAVAL
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 24 **RSTVAL**: Reset action is valid (when the I3C acts as target)
This bit is asserted by hardware to indicate that the RSTACT[1:0] field has been updated on the reception of a broadcast or direct write RSTACT CCC (target reset action) and is valid.
This bit is cleared by hardware when the target receives a frame start.
When the device is not in Stop mode:
– If RSTVAL = 1: when RSTF in the I3C_EVR register is asserted (and/or the corresponding interrupt if enabled), RSTACT[1:0] in the I3C_DEVR0 register dictates the reset action to be performed by the software, if any.
– If RSTVAL = 0: when RSTF is asserted (and/or the corresponding interrupt if enabled), the software must issue an I3C reset after a first detected reset pattern, and a system reset on the second one.
When in Stop mode, the corresponding interrupt can be used to wake up the device.

Bits 23:22 **RSTACT[1:0]**: Reset action/level on received reset pattern (when the I3C acts as target)

This read field is used by hardware on the reception of a direct read RSTACT CCC in order to return the corresponding data byte on the I3C bus.

This read field is updated by hardware on the reception of a broadcast or direct write RSTACT CCC (target reset action).

Only the defining bytes 0x00, 0x01 and 0x02 are mapped, and RSTACT[1:0] = Defining Byte[1:0].

00: no reset action

01: first level of reset: the application software must either:

a) partially reset the I3C peripheral, by a write and clear of the enable bit of the I3C configuration register (write EN = 0). This resets the I3C bus interface and the I3C kernel sub-parts, without modifying the content of the I3C APB registers (except the EN bit).

b) fully reset the I3C peripheral, including all its registers, via a write and set of the I3C reset control bit of the RCC (reset and clock controller) register.

10: second level of reset: the application software must issue a warm reset, also known as a system reset. This (see *Section 10: Reset and clock control (RCC)*) has the same impact as a pin reset (NRST = 0):

– the software writes and sets the SYSRESETREQ control bit of the AITR register, when the device is controlled by a Cortex®-M.

– the software writes and sets SYSRST = 1 in the RCC_GRSTCSETR register, when the device is controlled by a Cortex®-A.

11: no reset action

Bits 21:20 **AS[1:0]**: Activity state (when the I3C acts as target)

This read field is updated by hardware on the reception of a ENTASx CCC (enter activity state, with x = 0-3):

00: activity state 0

01: activity state 1

10: activity state 2

11: activity state 3

Bit 19 **HJEN**: Hot-join request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISHJ= 1 (cleared) and the reception of ENEC CCC with ENHJ= 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: hot-join request disabled

1: hot-join request enabled

Bit 18 Reserved, must be kept at reset value.

Bit 17 **CREN**: Controller-role request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISCR = 1 (cleared) and the reception of ENEC CCC with ENCR = 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: controller-role request disabled

1: controller-role request enabled

Bit 16 **IBIEN**: IBI request enable (when the I3C acts as target)
This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISINT = 1 (cleared) and the reception of ENEC CCC with ENINT = 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.
0: IBI request disabled
1: IBI request enabled

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:1 **DA[6:0]**: 7-bit dynamic address
When the I3C acts as controller, this field can be written by software, for defining its own dynamic address.
When the I3C acts as target, this field is updated by hardware on the reception of either the broadcast ENTDAA CCC or the direct SETNEWDA CCC.

Bit 0 **DAVAL**: Dynamic address is valid (when the I3C acts as target)
When the I3C acts as controller, this bit can be written by software, for validating its own dynamic address, for example before a controller-role hand-off.
When the I3C acts as target, this bit is asserted by hardware on the acknowledge of the broadcast ENTDAA CCC or the direct SETNEWDA CCC, and this field is cleared by hardware on the acknowledge of the broadcast RSTDAA CCC.

35.16.17 I3C device x characteristics register (I3C_DEVRx)

Address offset: 0x060 + 0x4 * x, (x = 1 to 4)

Reset value: 0x0000 0000

When the I3C peripheral acts as controller, this register is used to define and store some characteristics of a device target x with their related management from the controller, to communicate accordingly with any of this target x over the I3C bus. Then, the hardware can autonomously identify and acknowledge an allowed IBI or/and controller-role request from a target x, receive the expected IBI payload data if any, and notify the software via the corresponding flag IBIF/CRF (and the corresponding interrupt if enabled) in the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	IBIDEN	CRACK	IBIACK
r												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.	DA[6:0]							Res.
								rw	rw	rw	rw	rw	rw	rw	

Bit 31 **DIS**: Disables writes to DA[6:0] (when the I3C acts as controller)
When the I3C acts as controller, once the software sets IBIACK= 1 or CRACK= 1, this read bit is set by hardware (DIS= 1) to lock the configured DA[6:0] and IBIDEN values.
Then, to be able to modify DA[6:0], IBIDEN or SUSP, the software must wait for DIS to be de-asserted by hardware (polling on DIS= 0) before modifying these three assigned values to the target x. Indeed, the target can request an IBI or a controller-role while the controller intends to modify DA[6:0], IBIDEN or SUSP.
0: write to DA[7:0] and to IBIDEN in the I3C_DEVRx register is allowed
1: write to DA[7:0] and to IBIDEN is disabled/locked

Bits 30:20 Reserved, must be kept at reset value.

Bit 19 **SUSP**: Suspend/stop I3C transfer on received IBI (when the I3C acts as controller)

When the I3C acts as controller, this bit can be used to receive an IBI from target x with pending read notification feature (received MDB[7:5] = 3'b101).

If this bit is set, when an IBI is received and completed (IBIF = 1 in the I3C_EVR register), a stop is emitted on the I3C bus and both C-FIFO and TX-FIFO are automatically flushed by hardware. The controller execution flow is stopped, even if a next control message is programmed. When the IBI is completed, the controller software can issue a new control word, such as a private read, to the target device that initiated the IBI request.

0: C-FIFO and TX-FIFO are not flushed after an IBI request from target x is acknowledged and completed, and depending on the presence or absence of a next control word, a repeated start or a stop is emitted

1: I3C transfer is stopped and both C-FIFO and TX-FIFO are flushed after receiving an IBI request from target x

Bit 18 **IBIDEN**: IBI data enable (when the I3C acts as controller)

When the I3C acts as controller, this bit must be written by software to store the BCR[2] bit as received from the target x during broadcast ENTDAA or direct GETBCR CCC via the received I3C_RDR.

Writing to this field has no impact when the DIS = 1 in the I3C_DEVRx register.

0: no data byte follows the acknowledged IBI from target x

1: the mandatory data byte MDB[7:0] follows the acknowledged IBI from target x

Bit 17 **CRACK**: Controller-role request acknowledge (when the I3C acts as controller)

When the I3C acts as controller, this bit is written by software to define the acknowledge policy to be applied on the I3C bus on the reception of a controller-role request from target x:

0: a controller-role request from target x must be NACK-ed

After the NACK, the message continues as initially programmed (the target is aware of the NACK and can emit another controller-role request later on)

1: a controller-role request (with 7-bit dynamic address DA[6:0]) from target x must be ACKed

- The field DIS is asserted by hardware to protect DA[6:0] from being modified by software meanwhile the hardware can store internally the current DA[6:0] into the kernel clock domain.

- After the ACK, the message continues as initially programmed. The software is notified by the controller-role request flag (CRF = 1 in the I3C_EVR register) and/or the corresponding interrupt if enabled; For effectively granting the controller-role to the requesting secondary controller, software must issue a GETACCCR (formerly known as GETACCMST), followed by a stop.

- Independently of CRACK configuration for this or other devices, further controller-role request(s) are NACK-ed until controller-role request flag (CRF) and IBI flag (IBIF) in the I3C_EVR register are both cleared.

Bit 16 **IBIACK**: IBI request acknowledge (when the I3C acts as controller)

When the I3C acts as controller, this bit is written by software to define the acknowledge policy to be applied on the I3C bus on the reception of an IBI request from target x:

0: an IBI request from target x must be NACK-ed

- After the NACK, the message continues as initially programmed (the target is aware of the NACK and can emit another IBI request later on)

1: an IBI request (with 7-bit dynamic address DA[6:0]) from target x must be ACKed

- The field DIS is asserted by hardware to protect DA[6:0] from being modified by software meanwhile the hardware can store internally the current DA[6:0] into the kernel clock domain.
- After the ACK, the controller logs the IBI payload data, if any, depending on I3C_DEVRx.IBIDEN.
- The software is notified by the IBI flag (IBIF = 1) and/or the corresponding interrupt if enabled;
- Independently from IBIACK configuration for this or other devices, further IBI request(s) are NACK-ed until IBI request flag (IBIF) and controller-role request flag (CRF) are both cleared.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:1 **DA[6:0]**: Assigned I3C dynamic address to target x (when the I3C acts as controller)

When the I3C acts as controller, this field must be written by software to store the 7-bit dynamic address that the controller sends via a broadcast ENTDAA or a direct SETNEWDA CCC acknowledged by the target x.

Writing to this field has no impact when the read field DIS = 1 in the I3C_DEVRx register.

Bit 0 Reserved, must be kept at reset value.

35.16.18 I3C maximum read length register (I3C_MAXRLR)

Address offset: 0x090

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set or get the maximum read length value exchanged with the controller during respectively GETMRL or SETMRL CCC. This register is also used to set the IBI data payload size.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

When receiving a private read message, the target ends the data transmission (by driving T-bit = 0) when the count of transmitted data reaches MRL[15:0] in the I3C_MAXRLR 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.	IBIP[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MRL[15:0]															
rw	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.

Bits 18:16 **IBIP[2:0]**: IBI payload data maximum size, in bytes (when I3C acts as target)
This field is initially written by software when EN = 0 to set the maximum number of data bytes to be sent to the controller after an IBI request has been acknowledged.This field can be updated by hardware on the reception of SETMRL command (which potentially also updated IBIP[2:0]).
Software is notified of an MRL update by MRLUPF in the I3C_EVR register and the corresponding interrupt, if enabled.
000: null payload data size (only allowed when BCR2 = 0 in the I3C_BCR register)
001: 1 byte (mandatory data byte MDB[7:0]
010: 2 bytes (including first MDB[7:0])
011: 3 bytes (including first MDB[7:0])
100: 4 bytes (including first MDB[7:0])
others: same as 100

Bits 15:0 **MRL[15:0]**: Maximum data read length (when I3C acts as target)
This field is initially written by software when EN = 0 and updated by hardware on the reception of SETMRL command (with potentially also updated IBIP[2:0]).
Software is notified of an MRL update by MRLUPF and the corresponding interrupt, if enabled.
This field is used by hardware to return the value on the I3C bus when the target receives a GETMRL CCC.

35.16.19 I3C maximum write length register (I3C_MAXWLR)

Address offset: 0x094

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set or get the maximum write length value exchanged with the controller during respectively GETMWL or SETMWL CCC.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

On receiving a private write message, the target stops the data reception (extra received data are not written into RX-FIFO) when the count of received data reaches MWL[15:0] in the I3C_MAXWLR 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
MWL[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 **MWL[15:0]**: Maximum data write length (when I3C acts as target)
This field is initially written by software when EN = 0 and updated by hardware on the reception of SETMWL command.
Software is notified of an MWL update by MWLUPF in the I3C_EVR register and the corresponding interrupt, if enabled.
This field is used by hardware to return the value on the I3C bus when the target receives a GETMWL CCC.

35.16.20 I3C timing register 0 (I3C_TIMINGR0)

Address offset: 0x0A0

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure the SCL clock signal waveform.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCLH_I2C[7:0]								SCLL_OD[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
SCLH_I3C[7:0]								SCLL_PP[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **SCLH_I2C[7:0]**: SCL high duration, used for legacy I^2C messages, in number of kernel clocks cycles:

t_{SCLH_I2C} = (SCLH_I2C + 1) x t_{I3CCLK}

Note: SCLH_I2C is used to generate t_{DIG_H} (I^2C) timing when communicating with I^2C devices.

Note: With I^2C fm+ device t_{DIG_Hmin} = 260 ns, with I^2C fm device t_{DIG_Hmin} = 600 ns.

Bits 23:16 **SCLL_OD[7:0]**: SCL low duration in open-drain phases, used for legacy I^2C messages and for I3C open-drain phases (address phase following a start, ACK phase during controller-initiated messages, and T bit phase during direct/private/IBI payload), in number of kernel clocks cycles:

t_{SCLL_OD} = (SCLL_OD + 1) x t_{I3CCLK}

Note: SCLL_OD is used to generate both t_{DIG_L} (I^2C) and $t_{DIG\_OD\_L}$(I3C) timings.

Note: With I^2C fm+ device t_{DIG_Lmin} = 500 ns, with I^2C fm device t_{DIG_Lmin} = 1320 ns.

Note: I3C messages: $t_{DIG\_OD\_Lmin}$ = 200 ns.

Note: If a single I3C frame is gathering I^2C and I3C messages, the SCL low duration during I3C open-drain phases is increased to fit I^2C timings.

Bits 15:8 **SCLH_I3C[7:0]**: SCL high duration, used for I3C messages (both in push-pull and open-drain phases), in number of kernel clocks cycles:

t_{SCLH_I3C} = (SCLH_I3C + 1) x t_{I3CCLK}

Note: SCLH_I3C is used to generate both t_{DIG_H} (I3C) and $t_{DIG\_H\_MIXED}$ timings.

Note: For mixed bus (with at least one I^2C target): $t_{DIG\_H\_MIXEDmin}$ = 32 ns and $t_{DIG\_H\_MIXEDmax}$ = 45 ns (due to I^2C 50 ns spike filter).

Note: For pure I3C bus (with no I^2C targets): t_{DIG_Hmin} = 32 ns.

Bits 7:0 **SCLL_PP[7:0]**: SCL low duration in I3C push-pull phases, in number of kernel clocks cycles:

t_{SCLL_PP} = (SCLL_PP + 1) x t_{I3CCLK}

Note: SCLL_PP is used to generate t_{DIG_L} (I3C in PP) timing.

Note: t_{DIG_Lmin} = 32 ns (max 40/60 duty cycle at 12.5 MHz).

35.16.21 I3C timing register 1 (I3C_TIMINGR1)

Address offset: 0x0A4

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure some I3C timing settings.

When the I3C acts as target and is controller-capable, this register is used to configure a timing for the controller-role hand-off procedure.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SDA_HD	Res.	Res.	Res.	Res.	Res.	FREE[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.	ASNCR[1:0]		AVAL[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **SDA_HD**: SDA hold time (when the I3C acts as controller), in number of kernel clocks cycles (refer to MIPI timing SDA hold time in push-pull t_{HD_PP}):

SDA hold time = (SDA_HD + 0.5) x t_{I3CCLK}

Note: when controller: t_{HD_PPmiN} = min (t_{CR} , t_{CF}) + 3 ns.

Bits 27:23 Reserved, must be kept at reset value.

Bits 22:16 **FREE[6:0]**: Number of kernel clocks cycles that is used to set some MIPI timings like bus free condition time (when the I3C acts as controller)

When the I3C acts as controller:

1. for I3C start timing: it must wait for (bus free condition) time to be elapsed after a stop and before a start, refer to MIPI timings (I3C) t_{CAS} and (I^2C) t_{BUF}. These timings are defined by: t_{BUF} = t_{CAS} = [(FREE[6:0] + 1) x 2 - (0.5 + SDA_HD)] x t_{I3CCLK}

Note: For pure I3C bus: t_{CASmin} = 38.4 ns, and t_{CASmax} = 1 µs, 100 µs, 2 ms, 50 ms for, respectively, ENTAS0, 1, 2, and 3.

Note: For mixed bus with I^2C fm+ device t_{BUFmin} = 0.5 µs, for mixed bus with I^2C fm device t_{BUFmin} = 1.3 µs.

2. for I3C repeated start timing: must wait for time to be elapsed after a repeated start (SDA is de-asserted) and before driving SCL low, refer to. MIPI timing t_{CASr}. This timing is defined by: t_{CASr} = [(FREE[6:0] + 1) x 2 - (0.5 + SDA_HD)] x t_{I3CCLK}.

Note: $t_{CASr,\ miN}$ = 19.2 ns.

3. for I3C stop timing: must wait for time to be elapsed after that the SCL clock is driven high, and before the stop condition (SDA is asserted). This timing is defined by: t_{CBP} = (FREE[6:0] + 1) x t_{I3CCLK}.

Note: t_{CBPmiN} = 19.2 ns.

4. for I3C repeated start timing (T-bit when controller ends read with repeated start followed by stop): must wait for time to be elapsed after that the SCL clock is driven high, and before the repeated start condition (SDA is de-asserted). This timing is defined by: t_{CBSr} = (FREE[6:0] + 1) x t_{I3CCLK}.

Note: $t_{CBSr,\ miN}$ = 19.2 ns.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **ASNCR[1:0]**: Activity state of the new controller (when I3C acts as active controller)

This field indicates the time to wait before being accessed as new target, refer AVAL[7:0].

This field can be modified only when the I3C acts as controller.

Bits 7:0 **AVAL[7:0]**: Number of kernel clock cycles to set a time unit of 1 µs, whatever I3C acts as controller or target.

This time unit is then used by the hardware to build some internal timers, corresponding to the following MIPI I3C timings:

When the I3C acts as target:

1. for bus available condition time: it must wait for (bus available condition) time to be elapsed after a stop and before issuing a start request for an IBI or a controller-role request (bus free condition is sustained for at least t_{AVAL}). Refer to MIPI timing t_{AVAL} = 1 µs. This timing is defined by: t_{AVAL} = (AVAL[7:0] + 2) x t_{I3CCLK}
2. for bus idle condition time: it must wait for (bus idle condition) time to be elapsed after that both SDA and SCL are continuously high and stable before issuing a hot-join event. Refer to MIPI v1.1 timing t_{IDLE} = 200 µs . This timing is defined by: t_{IDLE} = (AVAL[7:0] + 2) x 200 x t_{I3CCLK}

When the I3C acts as controller, it cannot stall the clock beyond a maximum stall time (stall the SCL clock low), as follows:

1. on first bit of assigned address during dynamic address assignment: it cannot stall the clock beyond the MIPI timing $t_{STALLDAA}$ = 15 ms. This timing is defined by: $t_{STALLDAAmax}$ = (AVAL[7:0] + 1) x 15000 x t_{I3CCLK}
2. on ACK/NACK phase of I3C/I^2C transfer, on parity bit of write data transfer, on transition bit of I3C read transfer: it cannot stall the clock beyond the MIPI timing t_{STALL} = 100 µs. This timing is defined by: $t_{STALLmax}$ = (AVAL[7:0] + 1) x 100 x t_{I3CCLK}

Whatever the I3C acts as controller or as (controller-capable) target, during a controller-role hand-off procedure:

1. The new controller must wait for $t_{NEWCRLock}$ before pulling SDA low (issuing a start) after a completed GETACCR CCC. Then the new controller, within t_{CAS}, can pull SCL low to activate SCL clock. The active controller must wait for the same $t_{NEWCRLock}$ time, or at least 100 µs, before testing if the new controller has gained control of the bus by pulling SDA low. The time to wait depends upon the value of ANSCR[1:0] in the I3C_TIMINGR1 register:
   - ASNCR[1:0] = 00: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x t_{I3CCLK}
   - ASNCR[1:0] = 01: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 100 x t_{I3CCLK}
   - ASNCR[1:0] = 10: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 2000 x t_{I3CCLK}
   - ASNCR[1:0] = 11: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 50000 x t_{I3CCLK}

35.16.22 I3C timing register 2 (I3C_TIMINGR2)

Address offset: 0x0A8

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure and enable SCL clock stalling, to enable SCL clock low stalling, if needed by the addressed I3C or legacy I^2C target(s).

This register can be written only when the I3C acts as controller.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STALL[7:0]								Res.	Res.	Res.	Res.	STALLA	STALLC	STALLD	STALLT
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 **STALL[7:0]**: Controller clock stall time, in number of kernel clock cycles

t_{SCLL_STALL} = STALL x t_{I3CCLK}

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **STALLA:** Controller clock stall enable on ACK phase

The SCL is stalled (during t_{SCLL_STALL}as defined by STALL) in the address ACK/NACK phase (before the ninth bit). This allows the target to prepare data. The ACK driven by the controller itself on a target-initiated request (IBI/HJ/CR) is not impacted by this control bit.

0: no stall

1: stall enabled

Bit 2 **STALLC:** Controller clock stall enable on PAR phase of CCC

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase of common command code (before the ninth bit). This allows the target to decode the command.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

Bit 1 **STALLD:** Controller clock stall enable on PAR phase of Data

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase (before the ninth bit). This allows the target to read received data.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

Bit 0 **STALLT:** Controller clock stall enable on T-bit phase of data (and on the ACK/NACK phase of data byte of a legacy I^2C read)

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase (before the ninth bit). This allows the target to prepare the data to be sent.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

35.16.23 I3C bus characteristics register (I3C_BCR)

Address offset: 0x0C0

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to configure three bits used by hardware to return the data byte BCR[7:0] on reception of GETBCR or ENTDAA CCC. The returned byte BCR[7:0] on the I3C bus is then as follows:

- BCR[7] = 0 (reserved)
- BCR[6] = I3C_BCR6 (controller capable)
- BCR[5] = 1 (advanced capabilities, use GETCAPS CCC to determine which ones)
- BCR[4] = 0 (not a virtual target)
- BCR[3] = 1 (offline capable)
- BCR[2] = I3C_BCR2 (IBI payload)
- BCR[1] = 1 (IBI request capable)
- BCR[0] = I3C_BCR0 (max data speed limitation)

This register can be written by software only when EN = 0 in the I3C_CFGR 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.	BCR6	Res.	Res.	Res.	BCR2	Res.	BCR0
									rw				rw		rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **BCR6**: Controller capable
0: I3C target (no controller capable)
1: I3C controller capable

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **BCR2**: in-band interrupt (IBI) payload
0: no data byte follows the accepted IBI
1: at least one mandatory data byte follows the accepted IBI (and at most 4 data bytes)

Bit 1 Reserved, must be kept at reset value.

Bit 0 **BCR0**: max data speed limitation
0: no limitation
1: limitation, as described by I3C_GETMXDSR.

35.16.24 I3C device characteristics register (I3C_DCR)

Address offset: 0x0C4

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to configure the device characteristics ID, which is used by hardware to return the data byte DCR[7:0] on reception of GETDCR, ENTDAA, or DEFTGTS CCC.

This register can be written by software only when EN = 0 in the I3C_CFGR 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.	DCR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DCR[7:0]**: device characteristics ID
0x00: generic device (for v1.0 devices)
others: ID to describe the type of the I3C sensor/device

Note: The latest MIPI DCR ID assignments are available on https://www.mipi.org.

35.16.25 I3C get capability register (I3C_GETCAPR)

Address offset: 0x0C8

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set the IBI MDB support for pending read notification support, and is used by hardware to return the GETCAP3 byte on reception of GETCAPS CCC, format 1.

The returned byte GETCAP1[7:0] on the I3C bus is then as follows:

- GETCAP1[7:0] = 0 (no HDR)

The returned byte GETCAP2[7:0] on the I3C bus is then as follows:

- GETCAP2[7:6] = 00 (no HDR)
- GETCAP2[5:4] = 00 (no group addressing)
- GETCAP2[3:0] = 0001 (compliant with MIPI specification v1.1)

The returned byte GETCAP3[7:0] on the I3C bus is then as follows:

- GETCAP3[7] = 0 (reserved)
- GETCAP3[6] = CAPPEND in the I3C_GETCAPR register (IBI MDB support for pending read notification)
- GETCAP3[5] = 0 (no HDR)
- GETCAP3[4] = 1 (defining byte support in GETSTATUS)
- GETCAP3[3] = 1 (defining byte support in GETCAPS)
- GETCAP3[2] = 0 (no device-to-device transfer)
- GETCAP3[1] = 0 (no device-to-device transfer)
- GETCAP3[0] = 0 (no multi-lane data transfer)

This register can be written by software only when EN = 0 in the I3C_CFGR 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.	CAPPEND	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw														

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **CAPPEND**: IBI MDB support for pending read notification

This bit is written by software during bus initialization (EN = 0), and indicates the support (or not) of the pending read notification via the IBI MDB[7:0] value.

This bit is used to return the GETCAP3 byte in response to the GETCAPS CCC format 1.

0: this I3C when acting as target sends an IBI request without a mandatory data byte value indicating a pending read notification

1: this I3C when acting as target sends an IBI request with a mandatory data byte value (MDB[7:5] = 101), indicating a pending read notification

Bits 13:0 Reserved, must be kept at reset value.

35.16.26 I3C controller-role capability register (I3C_CRCAPR)

Address offset: 0x0CC

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set features the I3C supports as a secondary controller after controller-role hand-off, and is used by hardware to return the CRCAP1 byte and the CRCAP2 byte on reception of GETCAPS CCC, format 2 with the defining byte CRCAPS (0x91).

The returned CRCAP1[7:0] on the I3C bus is then as follows:

- CRCAP1[7:3] = 00000 (reserved)
- CRCAP1[2] = 0 (no multi-lane)
- CRCAP1[1] = CAPGRP in the I3C_CRCAPR register (group management)
- CRCAP1[0] = 1 (hot-join)

The returned CRLCAP2[7:0] on the I3C bus is then as follows:

- CRCAP2[7:4] = 0000 (reserved)
- CRCAP2[3] = CAPDHOFF in the I3C_CRCAPR register (delayed controller-role handoff)
- CRCAP2[2] = 1 (deep sleep capable)
- CRCAP2[1] = 0 (no automatic controller-role pass-back)
- CRCAP2[0] = 1 (IBI ack capable)

This register can be written by software only when EN = 0 in the I3C_CFGR 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.	CAP GRP	Res.	Res.	Res.	Res.	Res.	CAP DHOFF	Res.	Res.	Res.
						rw						rw			

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

Bit 9 **CAPGRP**: group management support (when acting as controller)

This bit is written by software during bus initialization (EN = 0), and indicates if the I3C is able to support group management when it acts as a controller (after controller-role hand-off) via emitted DEFGRPA, RSTGRPA, and SETGRPA CCC.

This bit is used to return the CRCAP1 byte in response to the GETCAPS CCC format 2.

0: this I3C does not support group address capabilities

1: this I3C supports group address capabilities (when becoming controller)

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **CAPDHOFF**: delayed controller-role hand-off

This bit is written by software during bus initialization (EN = 0), and indicates if this target I3C needs additional time to process a controller-role hand-off requested by the current controller.

This bit is used to return the CRCAP2 byte in response to the GETCAPS CCC format 2.

0: this I3C does not needs additional time to process a controller-role hand-off

1: this I3C needs additional time to process a controller-role hand-off

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

35.16.27 I3C get max data speed register (I3C_GETMXDSR)

Address offset: 0x0D0

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set its capabilities and limitations if any. This register is used by hardware to return data byte(s) on reception of GETMXDS CCC, with format 1 (2 bytes, without maxRdTurn), format 2 (5 bytes, with MaxRdTurn), or format 3 (with defining byte WRRDTURN = 0x00: same 5 bytes as format 2, or with defining byte CRHDLY = 0x91: single byte CRHDLY1).

The returned byte MaxWr[7:0] on the I3C bus is then as follows:

- MaxWr[7:4] = 0000 (reserved)
- MaxWr[3] = 1 (defining byte WRRDTURN and CRHDLY support)
- MaxWr[2:0] = 000 (max sustained data rate for non-CCC messages sent by the controller to the target is designed to operate at 12.5 MHz)

The returned byte MaxRd[7:0] on the I3C bus is then as follows:

- MaxRd[7] = 0 (reserved)
- MaxRd[6] = 1 (stop is allowed between write and read)
- MaxRd[5:3] = 100 if TSCO = 0 (clock to data turnaround time t_{SCO} ≤ 12 ns) in the I3C_GETMXDSR register, else 111 (t_{SCO} > 12 ns, refer to the datasheet for more details)
- MaxRd[2:0] = 000 (max sustained data rate for non-CCC messages sent by the target to the controller is designed to operate at 12.5 MHz)

The returned 3-byte MaxRdTurn[23:0], if FMT[1:0] = 00 in the I3C_GETMXDSR register, on the I3C bus is then as follows:

- MaxRdTurn[23:0], with either the MSB (between 65 ms and 16 s), the middle byte (between 256 µs and 65 ms), or the LSB (less than 256 µs) from RDTURN[7:0] in the I3C_GETMXDSR register (others bits are 0), and depending upon FMT[1:0] in the same register.

The returned byte CRHDLY1[7:0] on the I3C bus is then as follows:

- CRHDLY1[7:3] = 00000 (reserved)
- CRHDLY1[2] = 0 if HOFFAS[1:0] = 00 in the I3C_GETMXDSR register, else 1 (set bus activity state)
- CRHDLY1[1:0] = HOFFAS[1:0] (controller-role hand-off activity state)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCO	RDTURN[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
Res.	Res.	Res.	Res.	Res.	Res.	FMT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	HOFFAS[1:0]	
						rw	rw							rw	rw

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

Bit 24 **TSCO**: clock-to-data turnaround time (t_{SCO})

This bit is written by software during bus initialization (EN = 0 in the I3C_CFGR register) and is used to specify the clock-to-data turnaround time t_{SCO} (vs. the value of 12 ns). This bit is used by the hardware in response to the GETMXDS CCC to return the encoded clock-to-data turnaround time via the returned MaxRd[5:3] bits.

0: $t_{SCO} \leq 12$ ns

1: $t_{SCO} > 12$ ns (refer to the datasheet for more details)

Bits 23:16 **RDTURN[7:0]**: programmed byte of the 3-byte MaxRdTurn (maximum read turnaround byte)

This bit is written by software during bus initialization (EN = 0) and writes the value of the selected byte (via the FMT[1:0] field) of the 3-byte MaxRdTurn, which is returned in response to the GETMXDS CCC format 2 to encode the maximum read turnaround time.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **FMT[1:0]**: GETMXDS CCC format

This field is written by software during bus initialization (EN = 0) and indicates how is returned the GETMXDS format 1 (without MaxRdTurn) and format 2 (with MaxRdTurn).

This field is used to return the 2-byte format 1 (MaxWr, MaxRd) or 5-byte format 2 (MaxWr, MaxRd, 3-byte MaxRdTurn) byte in response to the GETCAPS CCC.

00: format 1 (2 bytes with MaxWr with no defining byte, MaxRd)

01: format 2: (5 bytes w.ith MaxWr with no defining byte, MaxRd, MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = 0, middle byte = 0 and LSB = RDTURN[7:0].
- Max read turnaround time is less than 256 µs.

10: format 2 (5 bytes with MaxWr with no defining byte, MaxRd, and middle byte of MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = 0, middle byte = RDTURN[7:0] and LSB = 0.
- Max read turnaround time is between 256 and 65535 µs

11: format 2 (5 bytes with MaxWr with no defining byte, MaxRd, MSB of MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = RDTURN[7:0], middle byte = 0 and LSB = 0.
- Max read turnaround time is between 65535 µs and 16 s.

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

Bits 1:0 **HOFFAS[1:0]**: Controller hand-off activity state

This field is written by software during bus initialization (EN = 0), and indicates in which initial activity state the (other) current controller must expect the I3C bus after a controller-role hand-off to this controller-capable I3C, when returning the defining byte CRHDLY (0x91) to a GETMXDS CCC.

This 2-bit field is used to return the CRHDLY1 byte in response to the GETCAPS CCC format 3, to state the activity state of this I3C when becoming controller after a controller-role hand-off, and consequently the time the former controller must wait before testing this I3C to be confirmed its ownership.

00: activity state 0 is the initial activity state of this I3C before and when becoming controller

01: activity state 1 is the initial activity state of this I3C when becoming controller

10: activity state 2 is the initial activity state of this I3C when becoming controller

11: activity state 3 is the initial activity state of this I3C when becoming controller

35.16.28 I3C extended provisioned ID register (I3C_EPIDR)

Address offset: 0xD4

Reset value: 0x0208 0000

When the I3C acts as target, this register is used to set the 4-bit MIPI instance ID by software, and some other constant bits used for the 48-bit provisioned ID. It is also used by hardware to return the six bytes for the 48-bit provisioned ID on reception of GETPID and ENTDAA CCC.

The 48-bit provisioned ID on the I3C bus is then returned as:

- provisioned ID [47:32] = 0x0208
- provisioned ID [31:16] = 0x1381
- provisioned ID [15:0] = {MIPIID[3:0], 0x000}

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MIPIMID[14:0]															IDTSEL
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
MIPIID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw												

Bits 31:17 **MIPIMID[14:0]**: 15-bit MIPI manufacturer ID
This read field is the 15-bit STMicroelectronics MIPI ID (0x0104).
This field represents bits[47:33] of the 48-bit provisioned ID.

Bit 16 **IDTSEL**: provisioned ID type selector
This field is set as 0 (vendor fixed value).
This field represents bit[32] of the 48-bit provisioned ID.
Note: Bits[31:16] of the provisioned ID can be 0.

Bits 15:12 **MIPIID[3:0]**: 4-bit MIPI Instance ID
This field is written by software to set and identify individually each instance of this I3C IP with a specific number on a single I3C bus.
This field represents bits[15:12] of the 48-bit provisioned ID.
Note: Bits[11:0] of the provisioned ID can be 0.

Bits 11:0 Reserved, must be kept at reset value.

35.16.29 I3C register map

Table 286. I3C 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	I3C_CR	MEND	MTYPE[3:0]				Res.	Res.	Res.	ADD[6:0] / CCC[7:0]							RNW	DCNT[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
0x004	I3C_CFGR	Res.	TSFSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFLUSH	CDMAEN	TMODE	SMODE	SFLUSH	SDMAEN	Res.	TXTHRES	TXFLUSH	TXDMAEN	Res.	RXTHRES	RXFLUSH	RXDMAEN	HJACK	Res.	HKSDAEN	EXITPTRN	RSTPTRN	NOARBH	CRINIT	EN
	Reset value		0									0	0	0	0	0	0		0	0	0		0	0	0	0		0	0	0	0	0	0
0x010	I3C_RDR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RDB0[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x014	I3C_RDWR	RDB3[7:0]								RDB2[7:0]								RDB1[7:0]								RDB0[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
0x018	I3C_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDB0[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x01C	I3C_TDWR	TDB3[7:0]								TDB2[7:0]								TDB1[7:0]								TDB0[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
0x020	I3C_IBIDR	IBIDB3[7:0]								IBIDB2[7:0]								IBIDB1[7:0]								IBIDB0[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
0x024	I3C_TGTTDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRELOAD	TGTTDCNT[15:0]																
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	I3C_SR	MID[7:0]								Res.	Res.	Res.	Res.	Res.	DIR	ABT	Res.	XDCNT[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
0x034	I3C_SER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERR	DNACK	ANACK	COVR	DOVR	STALL	PERR	CODERR[3:0]			
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x040	I3C_RMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RADD[6:0]							s_e_TP	RCODE[7:0]								Res.	Res.	Res.	Res.	Res.	IBIRDCNT[2:0]		
	Reset value									0	0	0	0	0	0	0		0	0	0	0	0	0	0	0						0	0	0

Table 286. I3C 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
0x050	I3C_EVR	GRPF	DEFF	INTUPDF	ASUPDF	RSTF	MRLUPDF	MWLUPDF	DAUPDF	STAF	GETF	WKPF	Res.	HJF	CRUPDF	CRF	IBIENDF	IBIF	Res.	Res.	Res.	ERRF	RXTGTENDF	FCF	Res.	RXLASTF	TXLASTF	RXFNEF	TXFNFF	SFNEF	CFNFF	TXFEF	CFEF
	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	1	1
0x054	I3C_IER	GRPIE	DEFIE	INTUPDIE	ASUPDIE	RSTIE	MRLUPDIE	MWLUPDIE	DAUPDIE	STAIE	GETIE	WKPIE	Res.	HJIE	CRUPDIE	CRIE	IBIENDIE	IBIIE	Res.	Res.	Res.	ERRIE	RXTGTENDIE	FCIE	Res.	Res.	Res.	RXFNEIE	TXFNFIE	SFNEIE	CFNFIE	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0				0	0	0	0		
0x058	I3C_CEVR	CGRPF	CDEFF	CINTUPDF	CASUPDF	CRSTF	CMRLUPDF	CMWLUPDF	CDAUPDF	CSTAF	CGETF	CWKPF	Res.	CHJF	CRRUPDF	CCRF	CIBIENDF	CIBIF	Res.	Res.	Res.	CERRF	CRXTGTENDF	CFCF	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	0	0									
0x05C	Reserved																																
0x060	I3C_DEVR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTVAL	RSTACT [1:0]		AS [1:0]		HJEN	Res.	CREN	IBIEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							DAVAL
	Reset value								0	0	0	0	0	0		0	0									0	0	0	0	0	0	0	0
0x060 + 4 * x, (x = 1 to 4)	I3C_DEVRx	DIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	IBIDEN	CRACK	IBIACK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							Res.
	Reset value	0												0	0	0	0									0	0	0	0	0	0	0	
0x090	I3C_MAXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IBIP [2:0]			MRL15:0]															
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094	I3C_MAXWLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MWL15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A0	I3C_TIMINGR0	SCLH_I2C[7:0]								SCLL_OD[7:0]								SCLH_I3C[7:0]								SCLL_PP[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
0x0A4	I3C_TIMINGR1	Res.	Res.	Res.	SDA_HD	Res.	Res.	Res.	Res.	Res.	FREE[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	ASNCR[1:0]		AVAL[7:0]							
	Reset value				0						0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x0A8	I3C_TIMINGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AVAL[7:0]								Res.	Res.	Res.	Res.	STALLA	STALLC	STALLD	STALLT
	Reset value																	0	0	0	0	0	0	0	0					0	0	0	0
0x0C0	I3C_BCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCR6	Res.	Res.	Res.	BCR2	Res.	BCR0
	Reset value																										0				0		0
0x0C4	I3C_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 286. I3C 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
0x0C8	I3C_GETCAPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAPPEND	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0														
0x0CC	I3C_CRCAPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAPGR	Res.	Res.	Res.	Res.	Res.	CAPDHOFF	Res.	Res.	Res.
	Reset value																							0						0			
0x0D0	I3C_GETMXDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCO	RDTURN[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	FMT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	HOFFAS [1:0]	
	Reset value								0	0	0	0	0	0	0	0	0							0	0							0	0
0x0D4	I3C_EPIDR	MIPIMID[14:0]															IDTSEL	MIPIID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0												

Refer to *Section 2.2* for the register boundary addresses.

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

36.1 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

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 tables below describe the USART implementation. LPUART is included for comparison.

Table 287. Instance implementation on STM32H503xx

Instance	STM32H503xx
USART1	Full
USART2	Full
USART3	Full
LPUART1	Low-power

Table 288. 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 288. USART/LPUART features (continued)

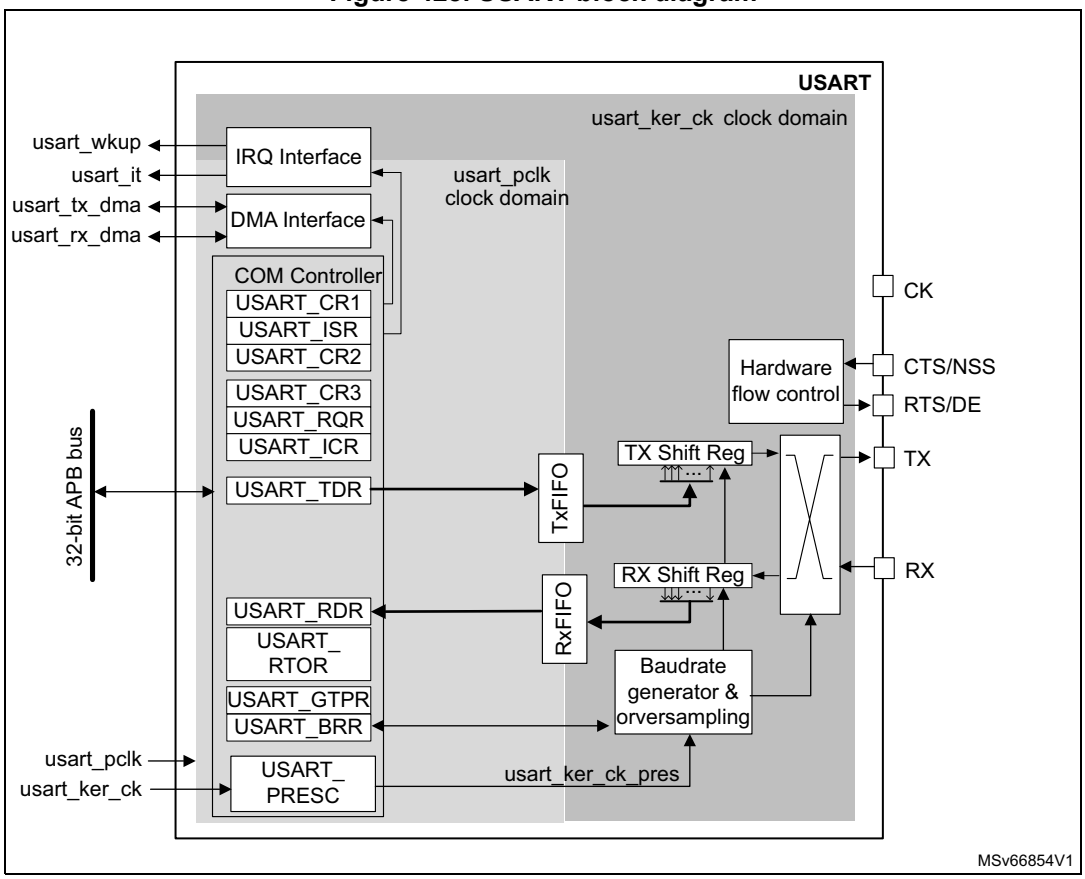
Modes/features[1]	Full feature set	Basic feature set	Low-power feature set
LIN mode	X	X	-
Dual clock domain	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size (bytes)	8		
Wake-up from low-power mode	X[2]	X[2]	X[2]

1. “X” = supported, “-” = not supported.
2. Wake-up supported from Stop mode.

36.5 USART functional description

36.5.1 USART block diagram

Figure 428. USART block diagram



36.5.2 USART pins and internal signals

Description 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 289* and *Table 290* for the list of USART input/output pins and internal signals.

Table 289. 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 290. 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

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 429*):

- 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.

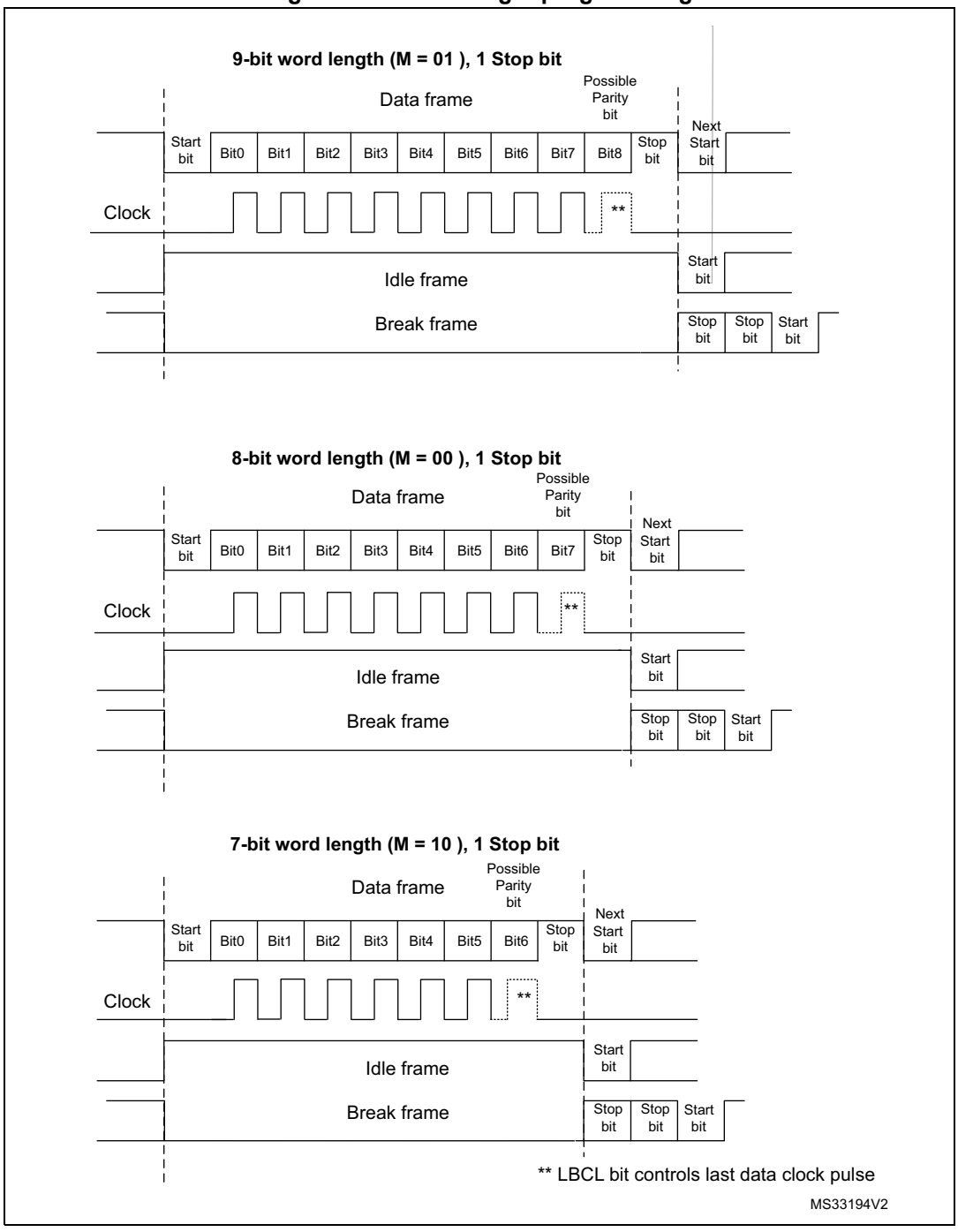
An ***Idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

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 429. 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

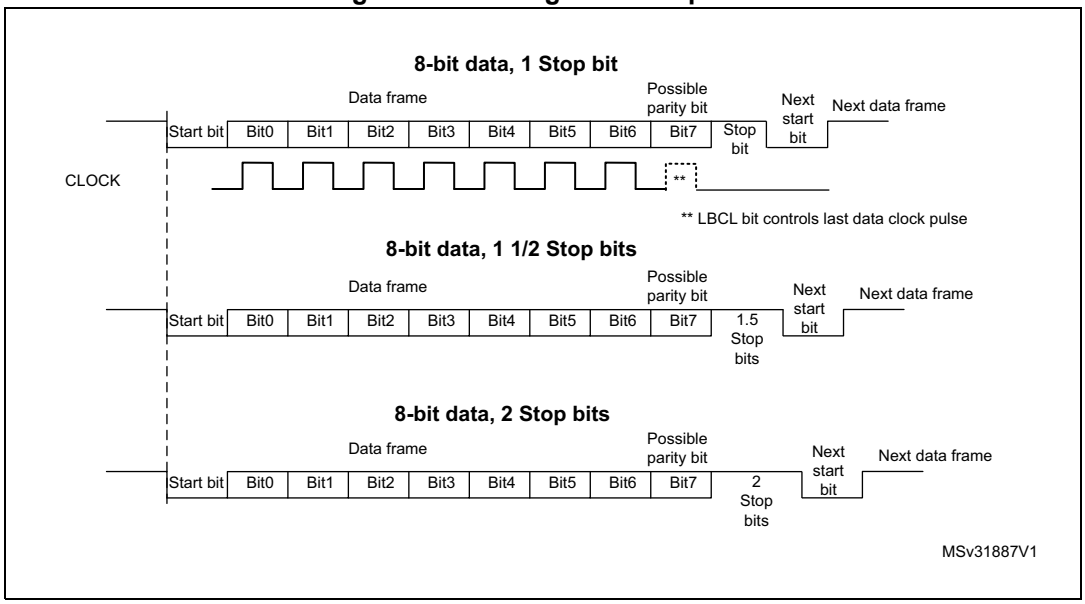
The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13,12.

- ***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 430. 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.

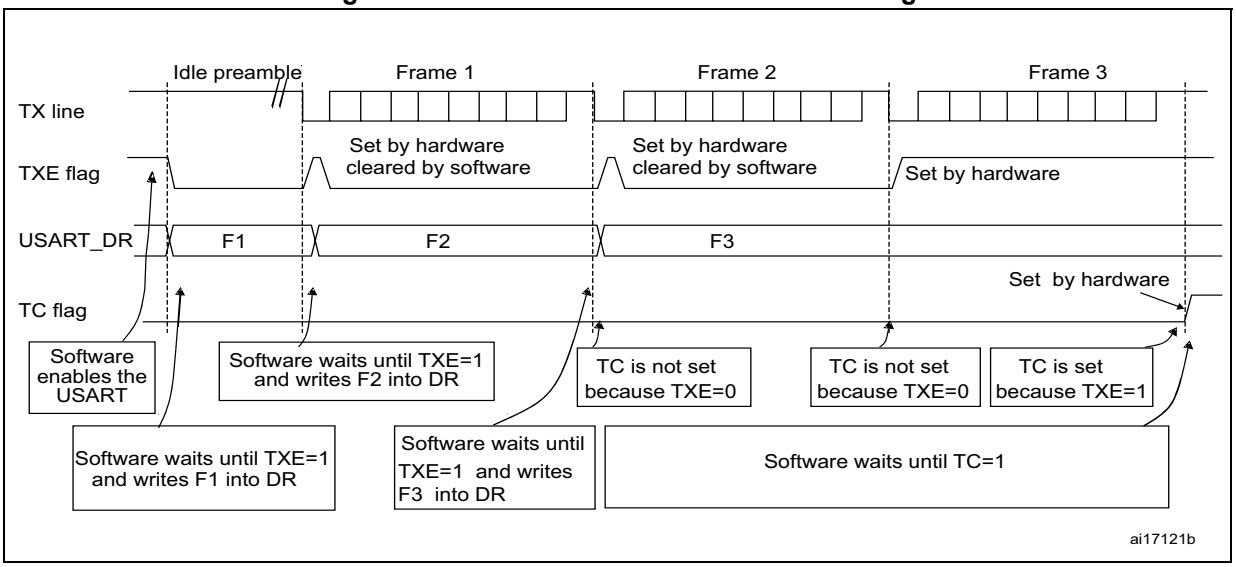
  When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

  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 431*).

Figure 431. 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 429*).

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

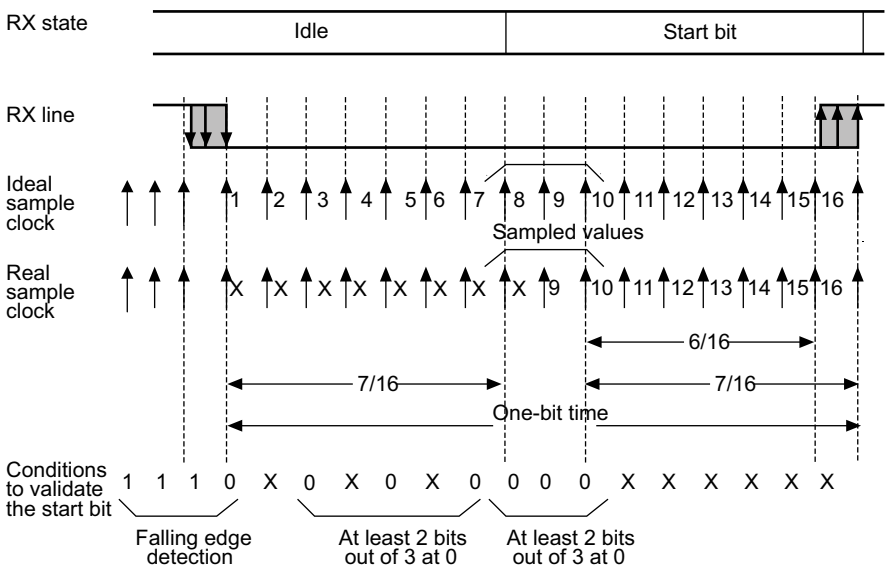
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

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 432. 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:

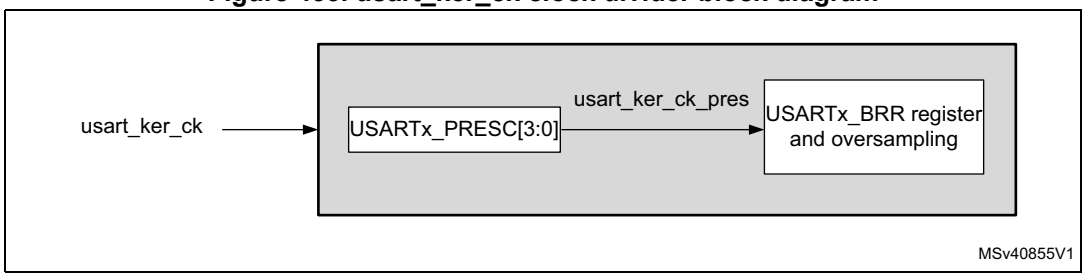
- Possible use of the USART in low-power mode
- Communication speed.

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 433. 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 434* and *Figure 435*).

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 291*) 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 434. Data sampling when oversampling by 16

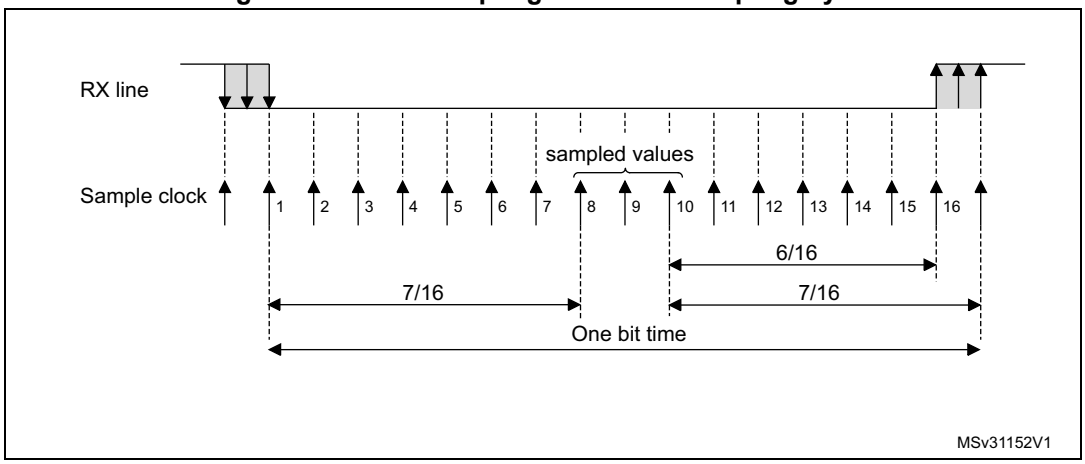


Figure 435. Data sampling when oversampling by 8

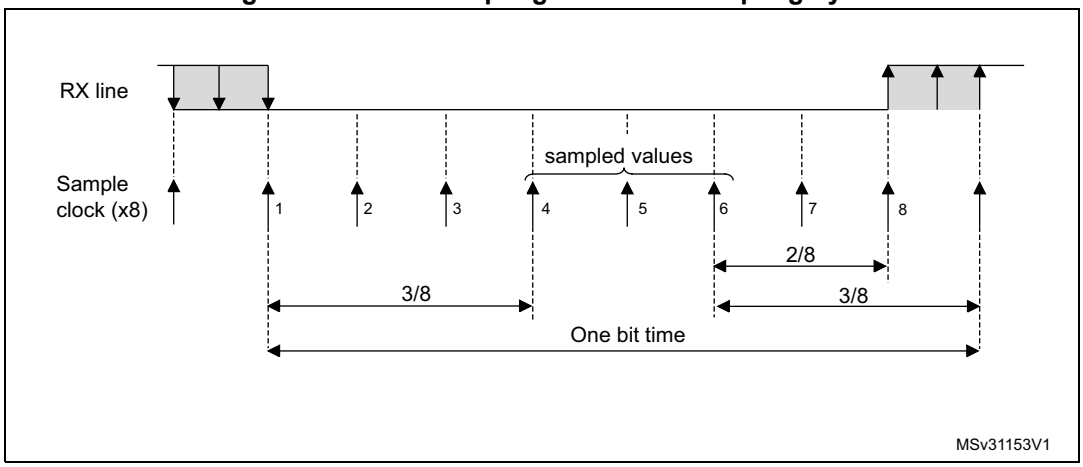


Table 291. Noise detection from sampled data

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 292*, *Table 293*, 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 292. 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 293. 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 292 and Table 293 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:

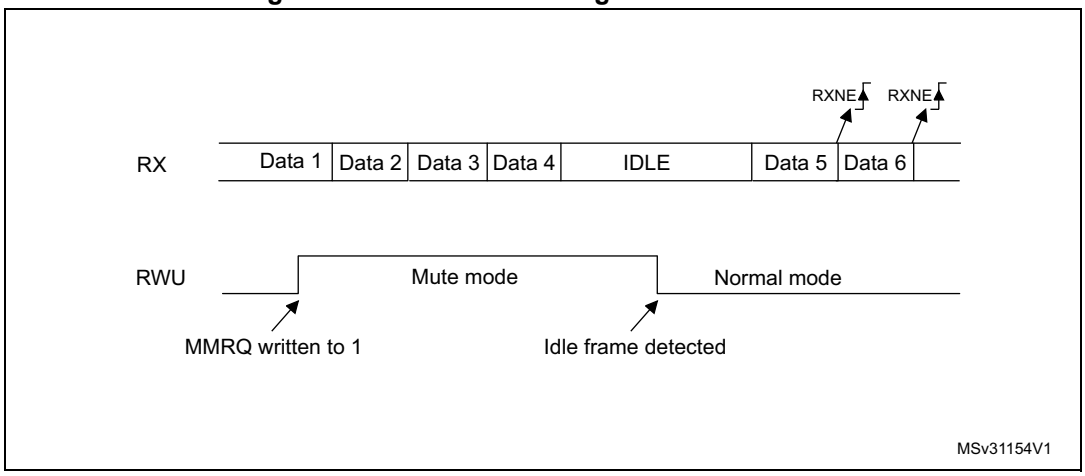
- Idle line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

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 436*.

Figure 436. 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.

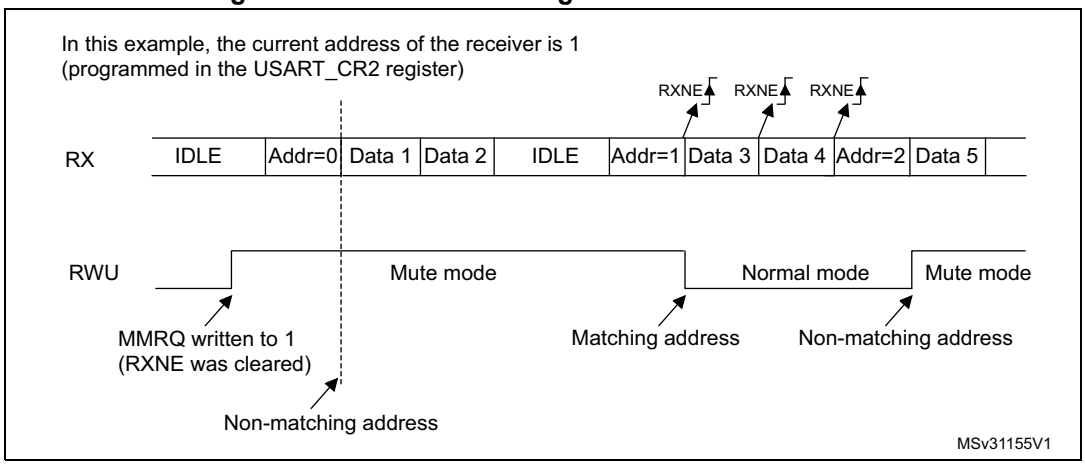
The USART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

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 437*.

Figure 437. 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 294*.

Table 294. 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 on the *Figure 438*.

Examples of break frames are given on *Figure 439*.

Figure 438. Break detection in LIN mode (11-bit break length - LBDL bit is set)

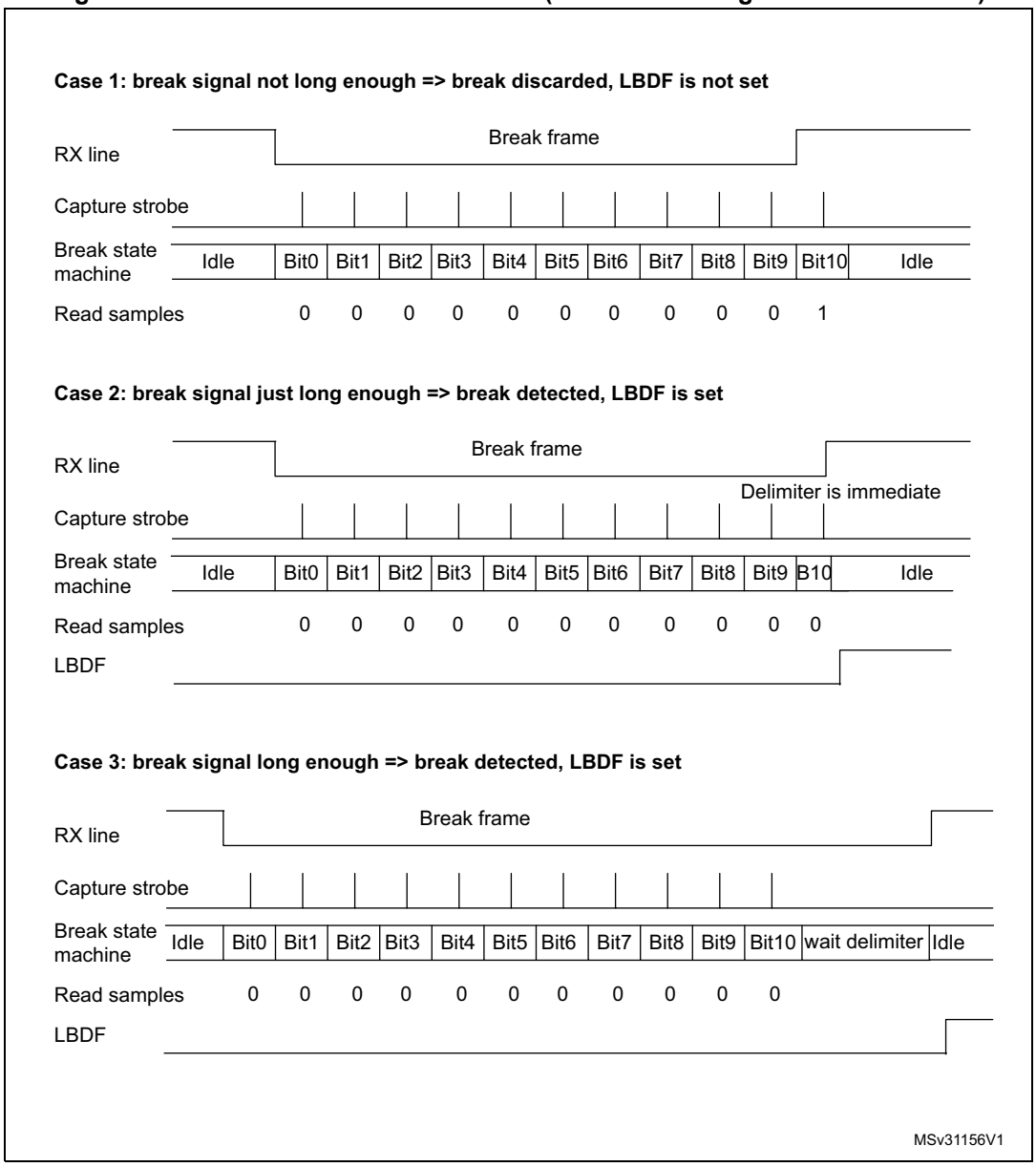
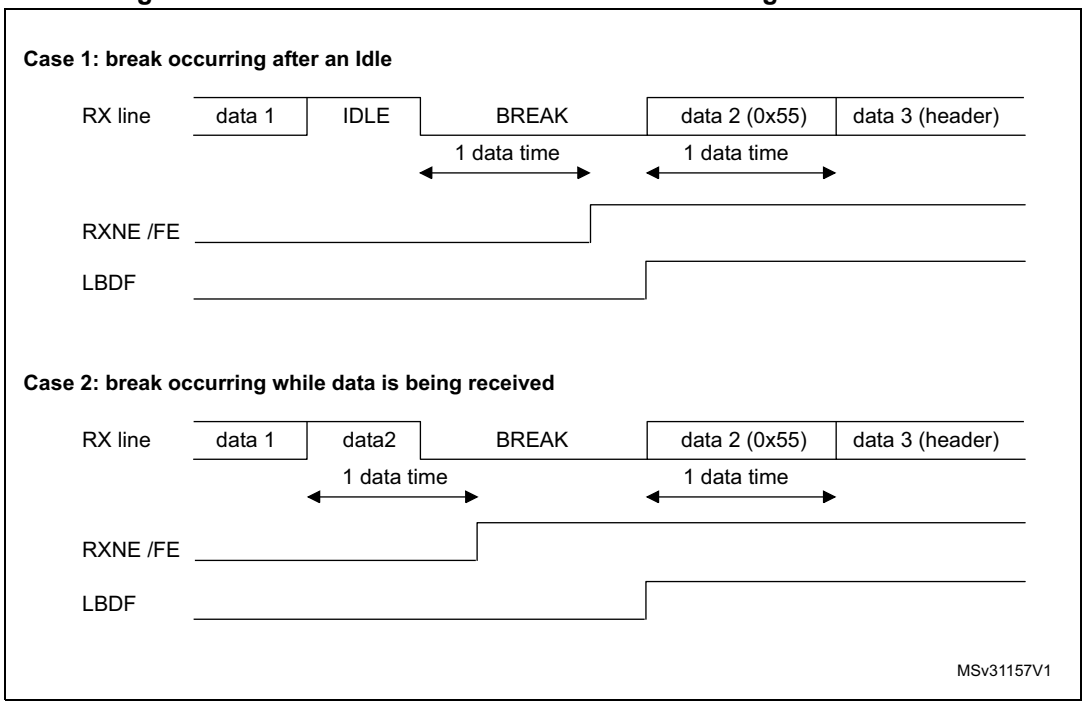


Figure 439. Break detection in LIN mode vs. Framing error detection



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 440*, *Figure 441* and *Figure 442*).

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 440. USART example of synchronous master transmission

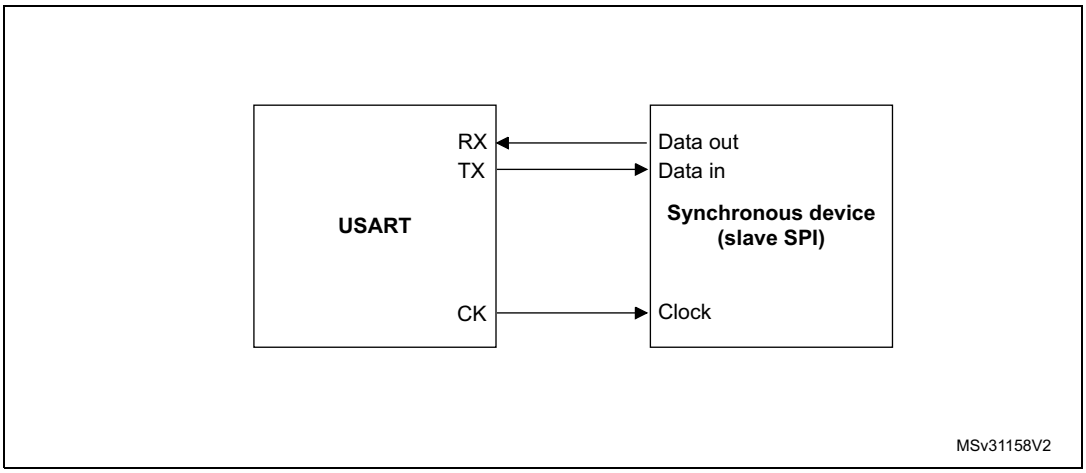


Figure 441. USART data clock timing diagram in synchronous master mode (M bits = 00)

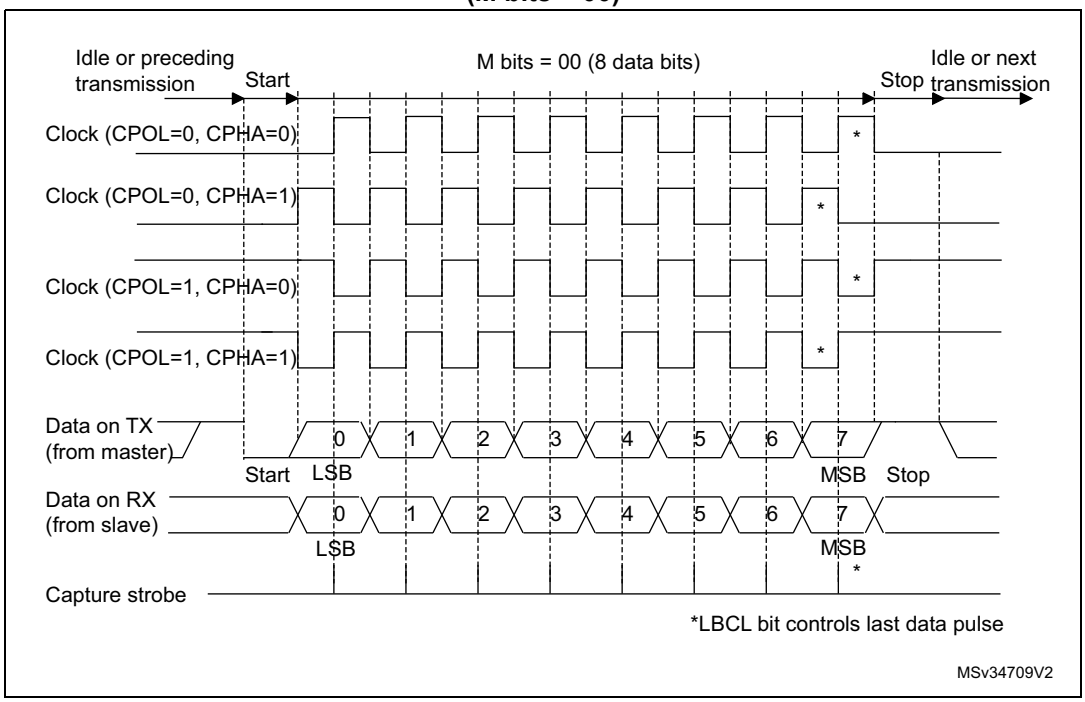
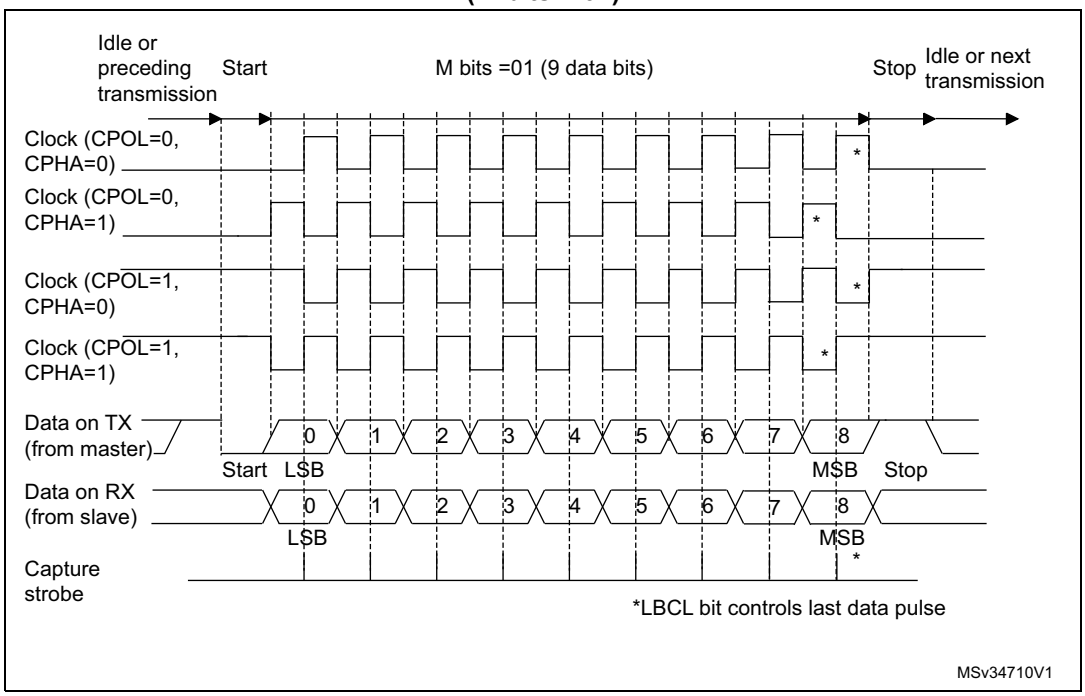


Figure 442. USART data clock timing diagram in synchronous master mode (M bits = 01)



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.

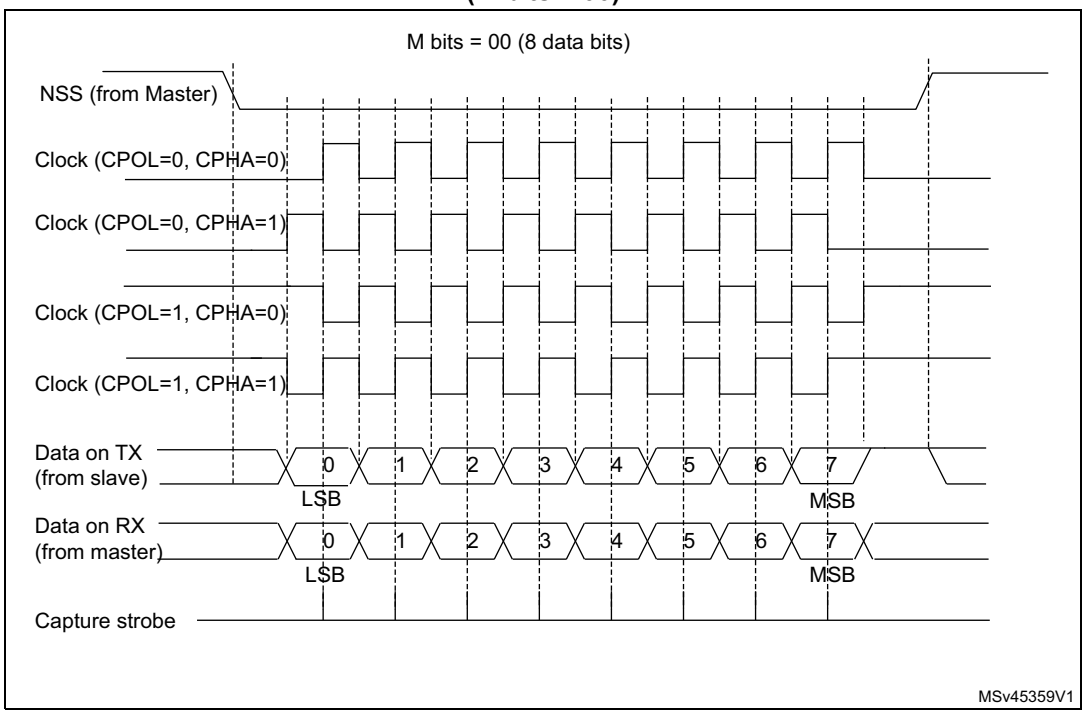
*Note: When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (*usart_ker_ck_pres*) must be greater than 3 times the CK input frequency.*

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 443*).

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 443. 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.

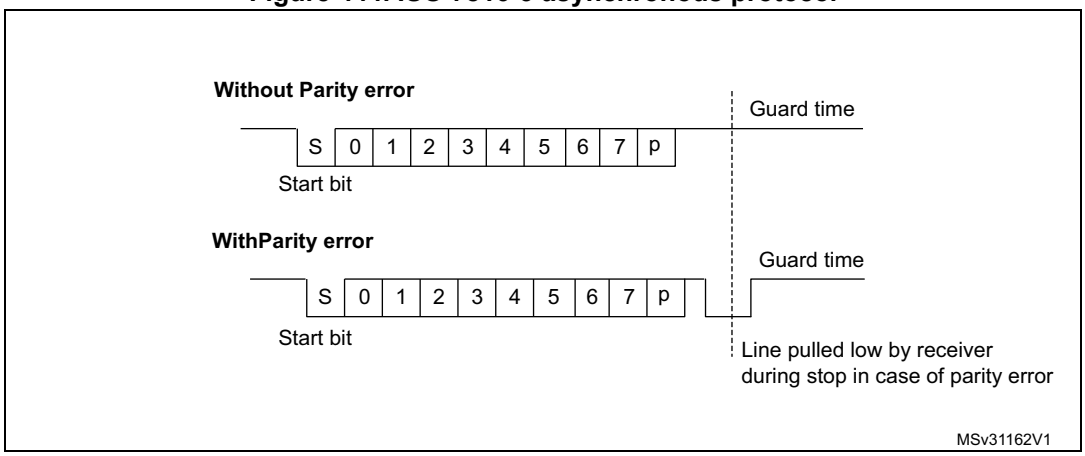
The USART must be configured as:

- 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 444 shows examples of what can be seen on the data line with and without parity error.

Figure 444. 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.

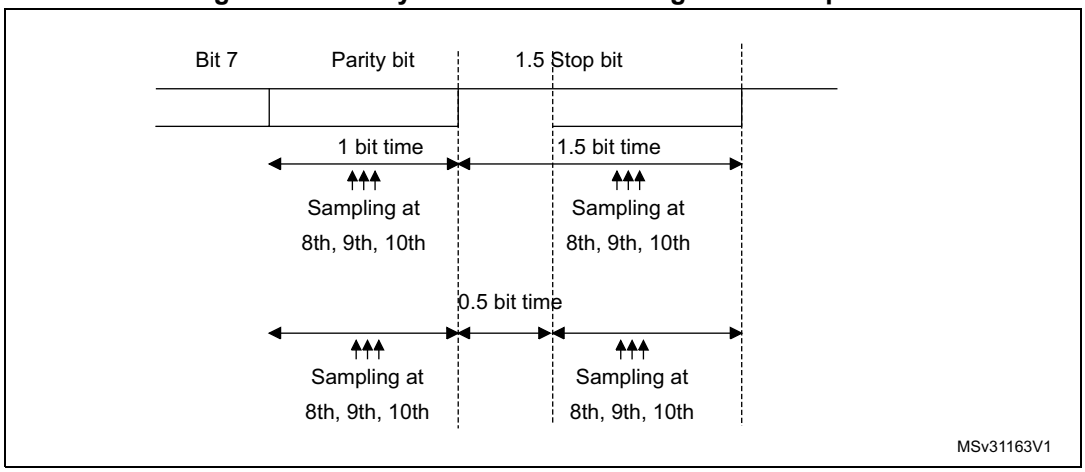
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guard time). If the software wants to repeat it 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 445 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 445. 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 446*).

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 447*).
- 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 446. IrDA SIR ENDEC block diagram

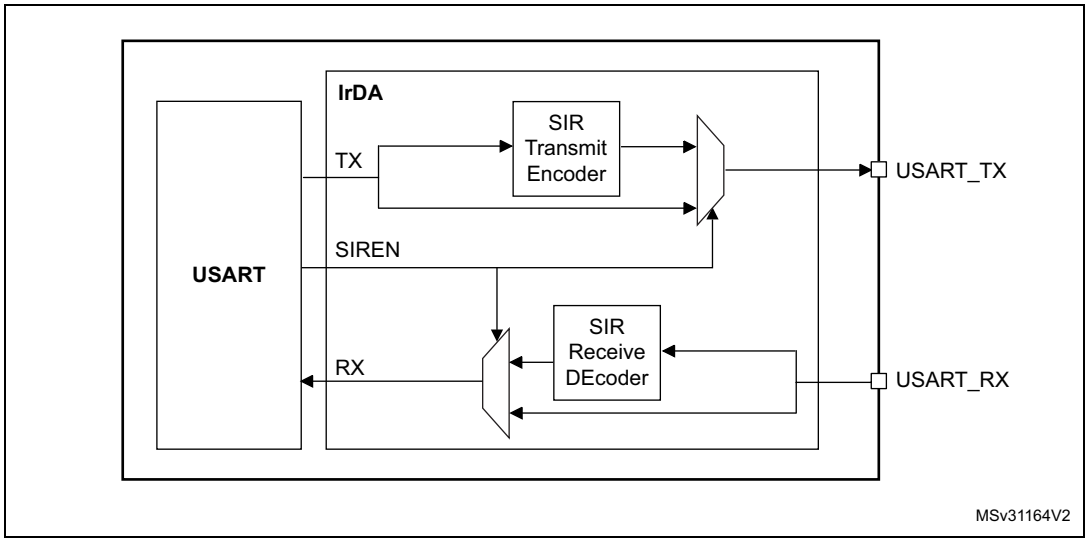
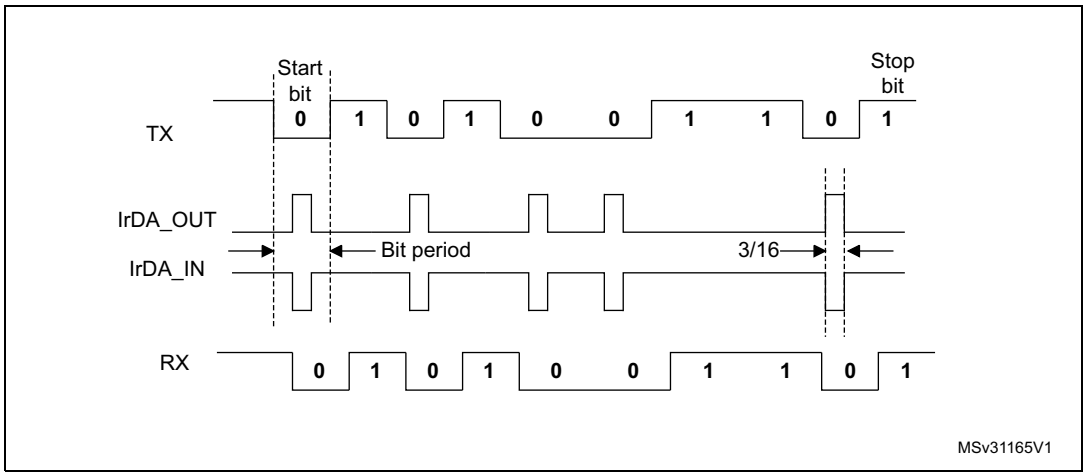


Figure 447. IrDA data modulation (3/16) - normal mode



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):

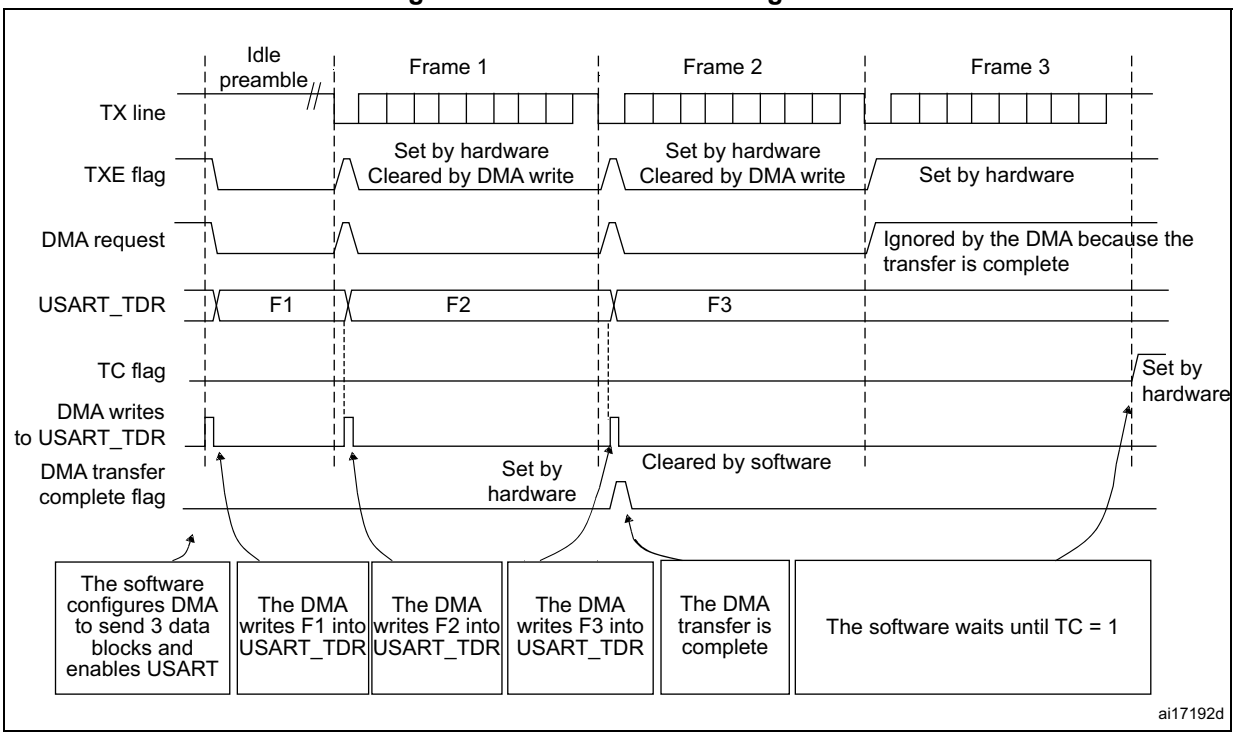
1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the 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 448. 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

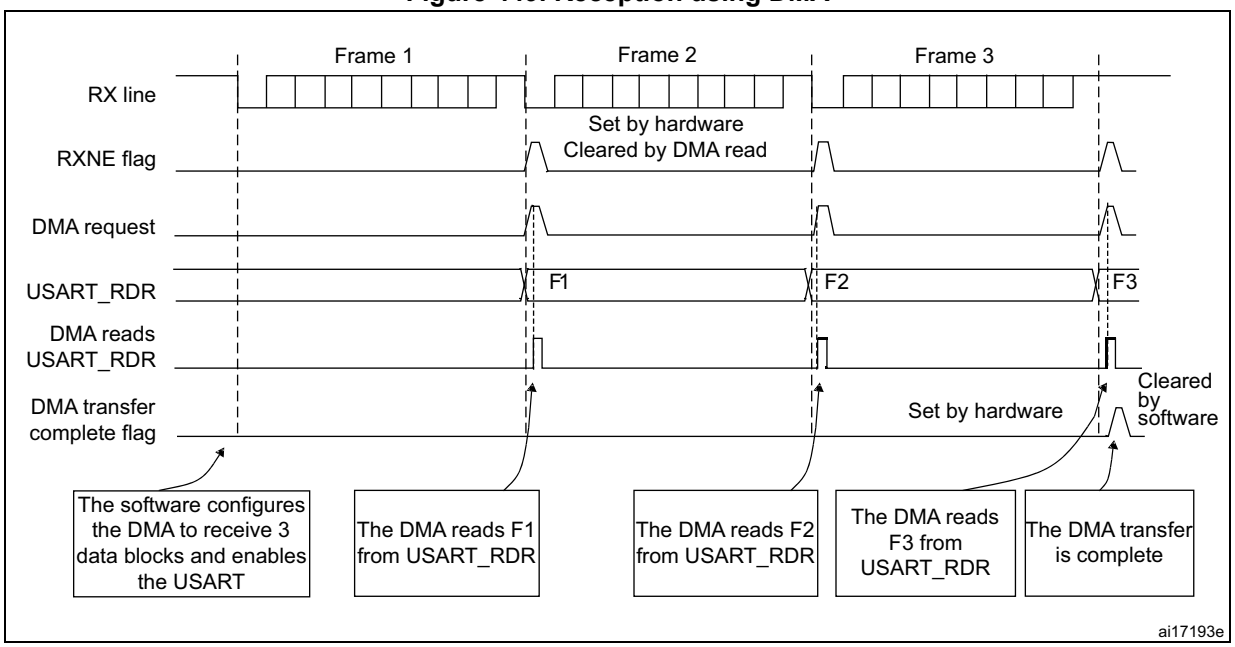
DMA mode can be enabled for reception by setting the DMAR bit in the USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to *section direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for 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 449. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by receive FIFO not empty (that is, RXFNE = 1).

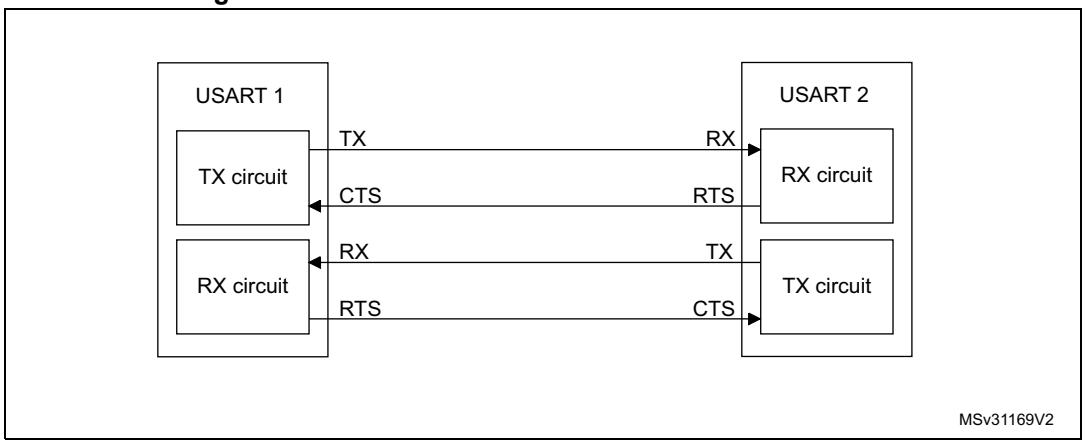
Error flagging and interrupt generation in multibuffer communication

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 450* shows how to connect two devices in this mode:

Figure 450. Hardware flow control between two USARTs

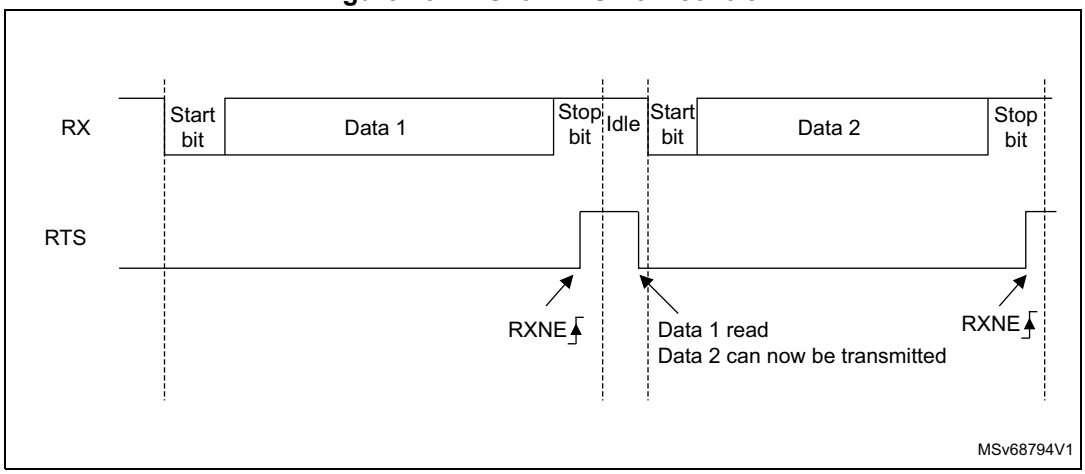


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to 1 in the USART_CR3 register.

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 451* shows an example of communication with RTS flow control enabled.

Figure 451. RS232 RTS flow control



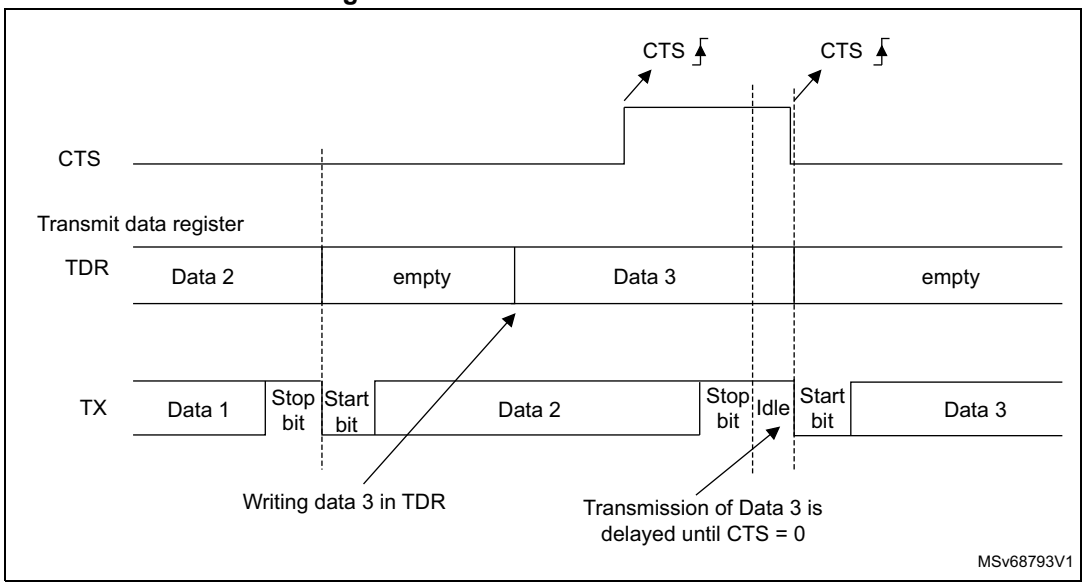
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

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 452* shows an example of communication with CTS flow control enabled.

Figure 452. 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 low-power management

The USART has advanced low-power mode functions, that enables transferring properly data even when the usart_pclk clock is disabled.

The USART is able to wake up the MCU from low-power mode when the UESM bit is set.

When the usart_pclk is gated, the USART provides a wake-up interrupt (**usart_wkup**) if a specific action requiring the activation of the **usart_pclk** clock is needed:

- If FIFO mode is disabled

  usart_pclk clock has to be activated to empty the USART data register.

  In this case, the usart_wkup interrupt source is RXNE set to 1. The RXNEIE bit must be set before entering low-power mode.
- If FIFO mode is enabled

  usart_pclk clock has to be activated to:
  - To fill the TXFIFO, or
  - To empty the RXFIFO

  In this case, the usart_wkup interrupt source can be:
  - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
  - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
  - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

  This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

  To avoid overrun/underrun errors and transmit/receive data in low-power mode, the usart_wkup interrupt source can be one of the following events:
  - TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
  - RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

  For example, the application can set the threshold to the maximum RXFIFO size if the wake-up time is less than the time required to receive a single byte across the line.

  Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wake up the MCU from low-power mode enables doing as many USART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific **usart_wkup** interrupt can be selected through the WUS bitfields.

When the wake-up event is detected, the WUF flag is set by hardware and a **usart_wkup** interrupt is generated if the WUFIE bit is set.

Note: Before entering low-power mode, make sure that no USART transfers are ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wake-up event is detected, independently of whether the MCU is in low-power or active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to make sure the USART is enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and reenabled when exiting from low-power mode.

When the FIFO is enabled, waking up from low-power mode on address match is only possible when mute mode is enabled.

Using mute mode with low-power mode

If the USART is put into mute mode before entering low-power mode:

- Wake-up from mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wake-up from mute mode on address match is used, then the low-power mode wake-up source must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, mute mode can be used with wake-up from low-power mode without any constraints (that is, the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).

Wake-up from low-power mode when USART kernel clock (usart_ker_ck) is OFF in low-power mode

If during low-power mode, the usart_ker_ck clock is switched OFF when a falling edge on the USART receive line is detected, the USART interface requests the usart_ker_ck clock to be switched ON thanks to the usart_ker_ck_req signal. usart_ker_ck is then used for the frame reception.

If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, usart_ker_ck is switched OFF again, the MCU is not woken up and remains in low-power mode, and the kernel clock request is released.

The example below shows the case of a wake-up event programmed to “address match detection” and FIFO management disabled.

Figure 453 shows the USART behavior when the wake-up event is verified.

Figure 453. Wake-up event verified (wake-up event = address match, FIFO disabled)

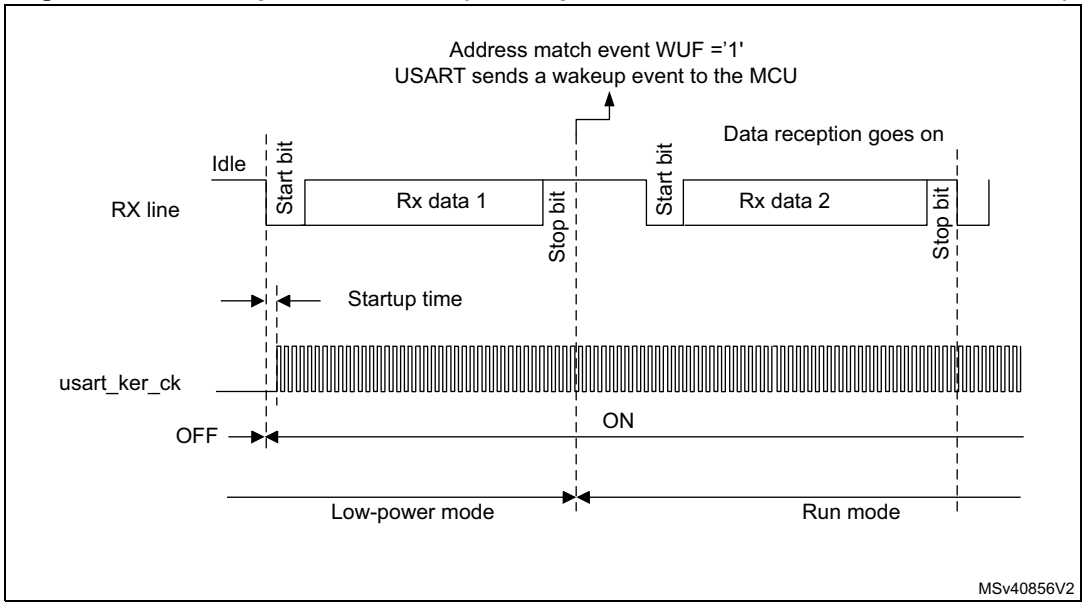
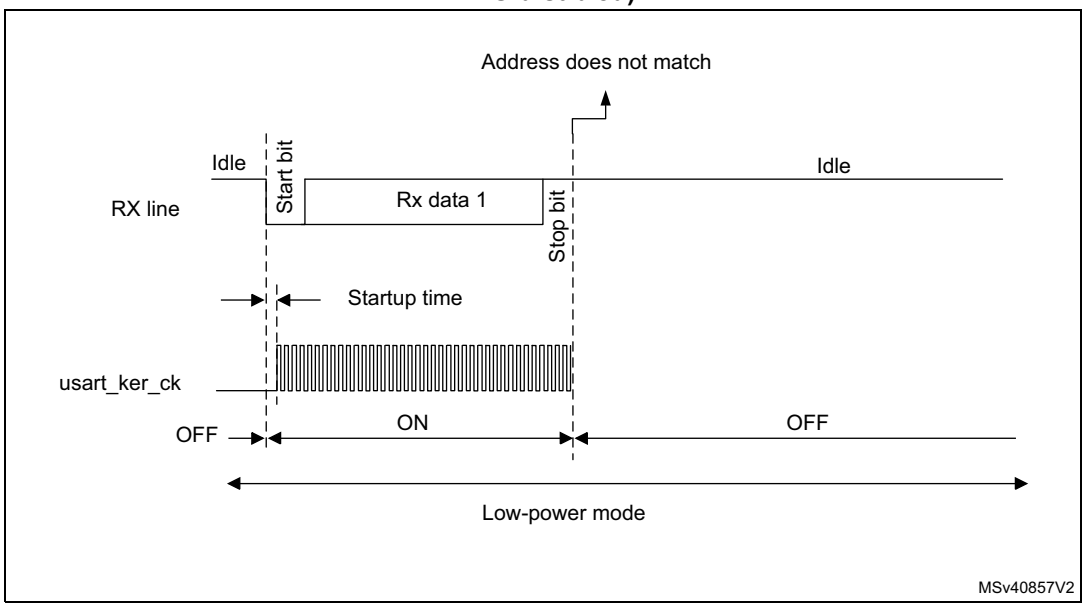


Figure 454 shows the USART behavior when the wake-up event is not verified.

Figure 454. Wake-up event not verified (wake-up event = address match, FIFO disabled)



Note: The figures above are valid when address match or any received frame is used as a wake-up event. If the wake-up event is the start bit detection, the USART sends the wake-up event to the MCU at the end of the start bit.

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 292: 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 295. Effect of low-power modes on the USART

Mode	Description
Sleep	No effect. USART interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the USART registers is kept. The USART is able to wake up the microcontroller from Stop mode when the USART is clocked by an oscillator available in 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 296* for a detailed description of all USART interrupt requests.

Table 296. USART 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
USART or UART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	No	No
	Transmit FIFO not full	TXFNF	TXFNFIE	TXFIFO full		No	
	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		No	
	Transmission complete before guard time	TCBGT	TCBGTIE	Write TDR or write 1 in TCBGT		No	

Table 296. USART interrupt requests (continued)

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
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		No	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		No	
	LIN break	LBDF	LBDIE	Write 1 in LBDCF		No	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		No	
	Overrun error in multibuffer communication.	ORE(4)		Write 1 in ORECF		No	
	Framing error in multibuffer communication.	FE		Write 1 in FECF		No	
	Character match	CMF	CMIE	Write 1 in CMCF		No	
	Receiver timeout	RTOF	RTOFIE	Write 1 in RTOCCF		No	
	End of Block	EOBF	EOBIE	Write 1 in EOBCF		No	
	Wake-up from low-power mode	WUF	WUFIE	Write 1 in WUC		Yes	
	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. When OVRDIS = 0.

36.8 USART registers

Refer to *Section 1.2 on page 65* 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 on page 1335.

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. It is set and cleared by software.

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 wake up the MCU from 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 able to wake up the MCU from low-power mode.

1: USART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

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
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt 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. It is set and cleared by software.
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 wake up the MCU from 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 able to wake up the MCU from low-power mode.
1: USART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

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 should 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 WUS[1:0] is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter should be equal to 1.
- 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 434* and *Figure 435*)

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	WUFIE	WUS1	WUS0	SCARCNT[2:0]			Res.
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
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].

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever WUF = 1 in the USART_ISR register
Note: WUFIE must be set before entering in low-power mode.
If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation on page 1335.

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01: Reserved.
10: WUF active on start bit detection
11: WUF active on RXNE/RXFNE.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation on page 1335.

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 on page 1335.

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 1335.

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 on page 1335.

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 on page 1335.

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 on page 1335.

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 on page 1335.

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 on page 1335.

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 on page 1335.

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 on page 1335.

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.	WUFIE	WUS1	WUS0	SCARCNT[2:0]			Res.
							rw		rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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 on page 1335.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever WUF = 1 in the USART_ISR register

Note: WUFIE must be set before entering in low-power mode.

If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation on page 1335.

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection

This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).

00: WUF active on address match (as defined by ADD[7:0] and ADDM7)

01: Reserved.

10: WUF active on start bit detection

11: WUF active on RXNE/RXFNE.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation on page 1335.

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 on page 1335.

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 on page 1335.

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 1335.

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 on page 1335.

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 on page 1335.

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 on page 1335.

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 on page 1335.

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 on page 1335.

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 on page 1335.

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 on page 1335.

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 on page 1335.

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 on page 1335.

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 on page 1335.

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 on page 1335.

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	WUF	RWU	SBKF	CMF	BUSY
				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
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 27) 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 on page 1335.

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 on page 1335.

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 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE = 1 in the USART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

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 on page 1335.

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 on page 1335.

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 on page 1335.

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 on page 1335.

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 on page 1335.

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.

It is set by hardware when the transmission of a frame containing data has completed, and the TXFE bit is set.

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 a write to the USART_TDR register.

0: Transmission has not completed

1: Transmission has completed

Note: If the TE bit is reset and no transmission is ongoing, the TC bit is immediately set.

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 on page 1354).

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	WUF	RWU	SBKF	CMF	BUSY
						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	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 on page 1335.

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 on page 1335.

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 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE = 1 in the USART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

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 on page 1335.

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 on page 1335.

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 on page 1335.

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 on page 1335.

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 on page 1335.

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 on page 1354).

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.	WUCF	Res.	Res.	CMCF	Res.
											w			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 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wake-up from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the USART_ISR register.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 36.4: USART implementation on page 1335.

Bits 19:18 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 on page 1335

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 on page 1335.

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 on page 1335.

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 on page 1335.

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 on page 1335.

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 on page 1335.

36.8.16 USART register map

Table 297. 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	WUFIE	WUS 1	WUS 0	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	0	0	0
0x08	USART_CR3 FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCBGTIE	Res.	WUFIE	WUS 1	WUS 0	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

Table 297. 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
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
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	WUF	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	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	WUF	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	0	1	0	0	0	0	0	0	0
0x20	USART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	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	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

Refer to *Section 2.2 on page 70* for the register boundary addresses.

37 Low-power universal asynchronous receiver transmitter (LPUART)

37.1 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

37.3 LPUART implementation

The tables below describe the LPUART implementation. USARTs and UARTs are included for comparison.

Table 298. Instance implementation on STM32H503xx

Instance	STM32H503xx
USART1	FULL
USART2	FULL
USART3	FULL
LPUART1	LP

Table 299. 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	-

Table 299. USART/LPUART features (continued)

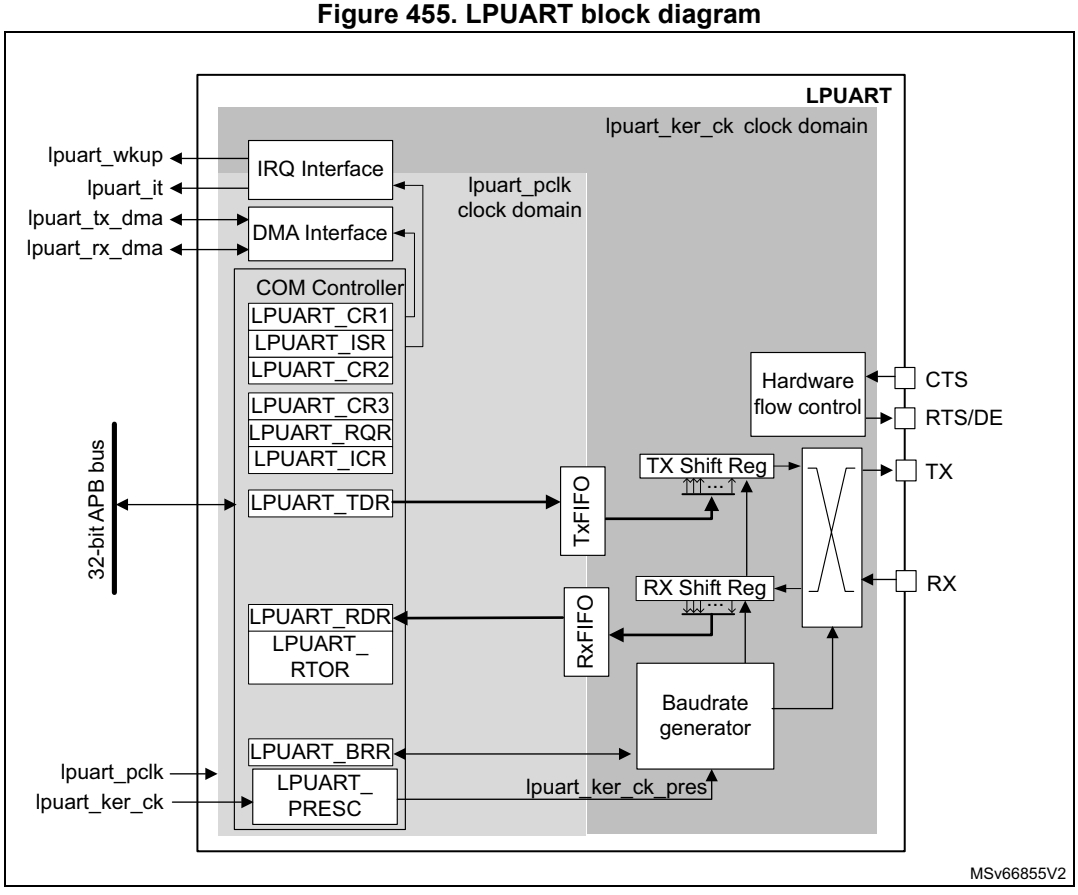
Modes/features(1)	Full feature set	Basic feature set	Low-power feature set
Driver enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size (bytes)	8		
Wake-up from low-power mode	X(2)	X(2)	X(2)

1. X = supported.
2. Wake-up supported from Stop mode.

37.4 LPUART functional description

37.4.1 LPUART block diagram

Figure 455. 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 300* and *Table 301* for the list of LPUART input/output pins and internal signals.

Table 300. 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 301. 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 301. 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

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 429*).

- 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.

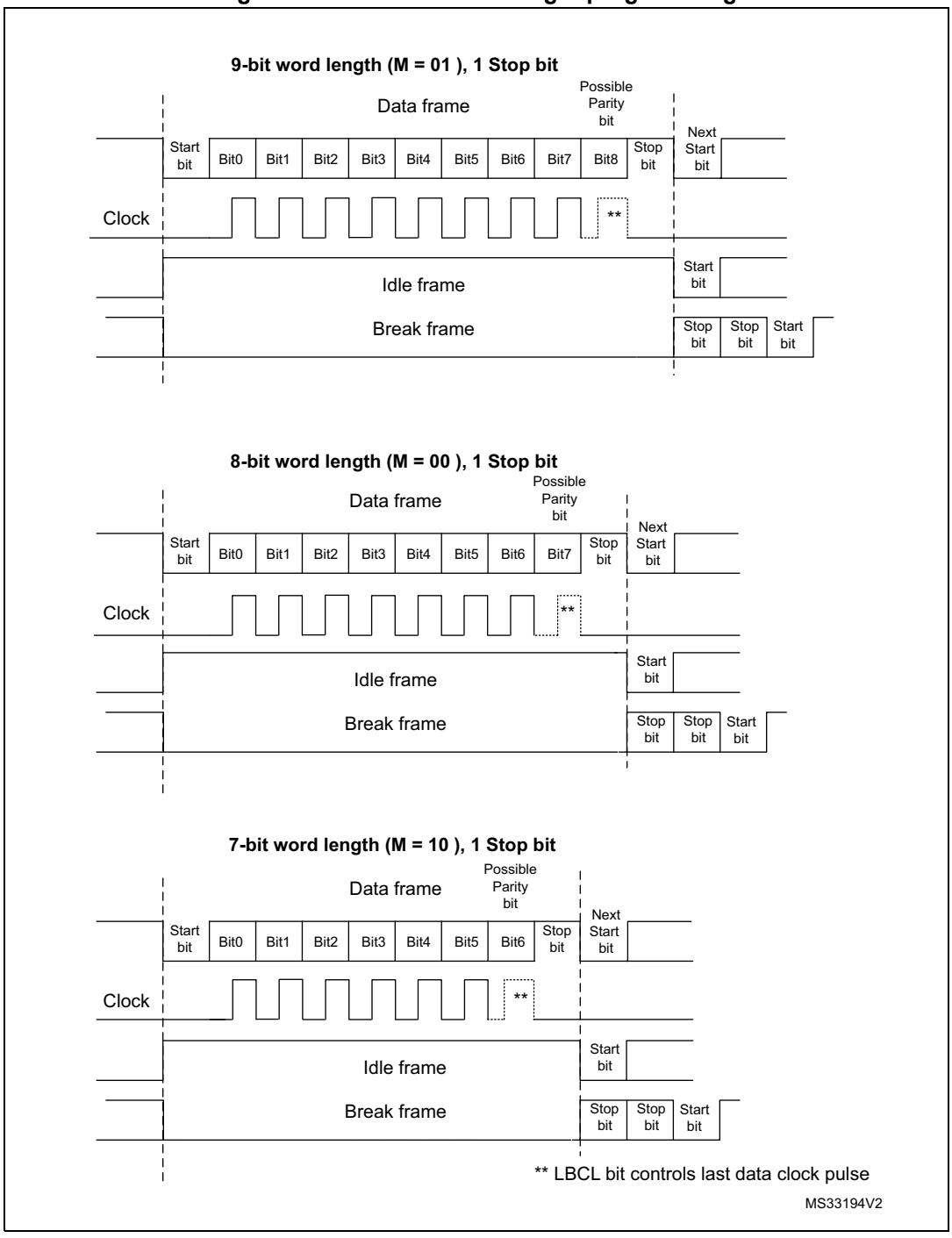
An ***idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

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 456. 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

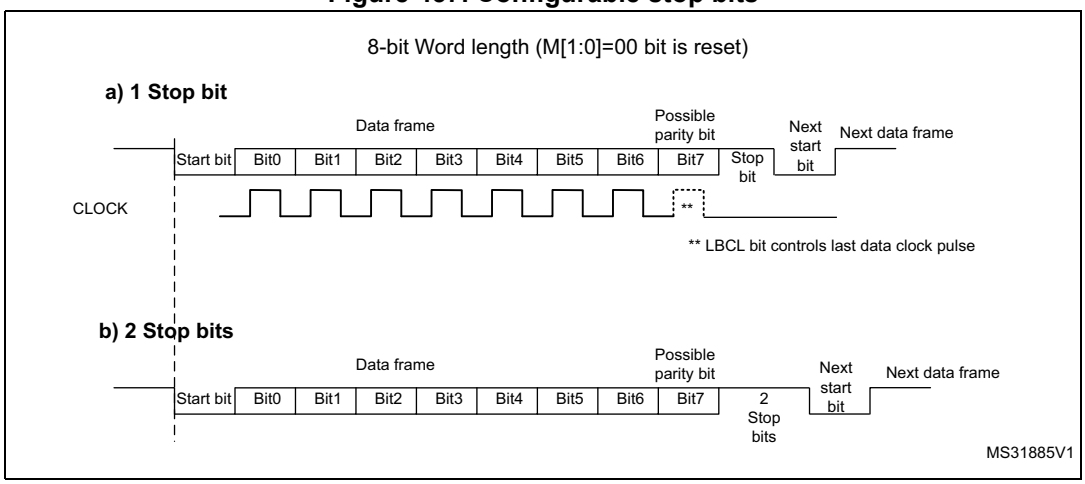
The number of stop bits to be transmitted with every character can be programmed in LPUART_CR2 (bits 13, 12).

- ***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 457. 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.

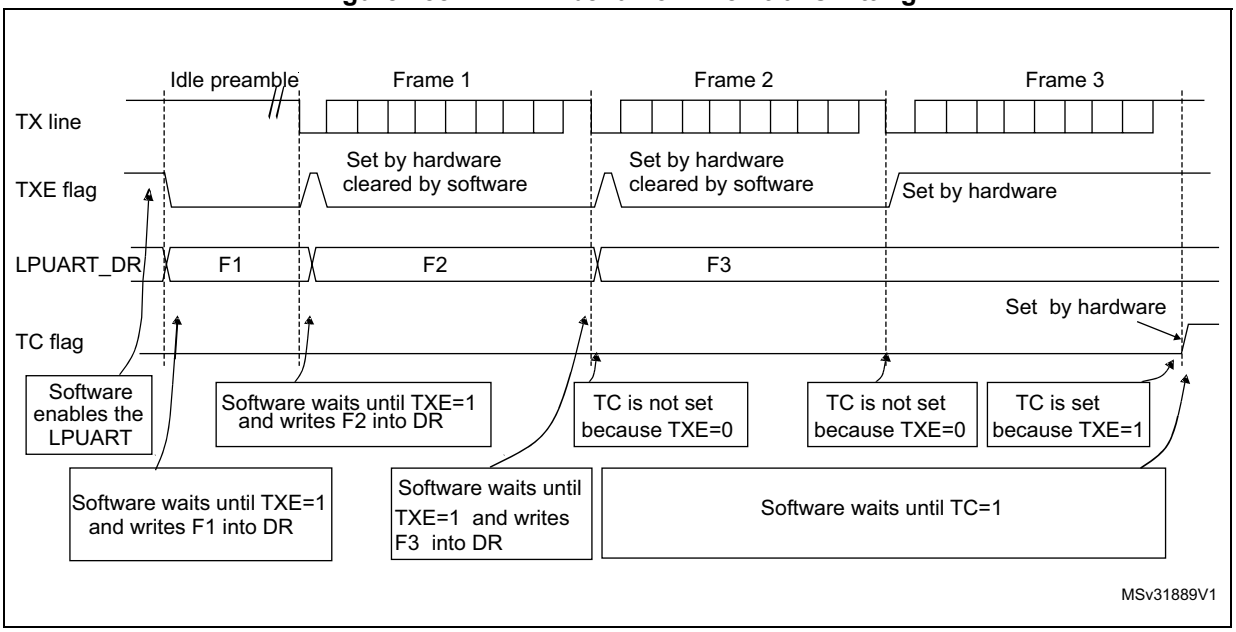
When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write in LPUART_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNEIE bit is set.

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 458: TC/TXE behavior when transmitting*).

Figure 458. 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 456*).

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:

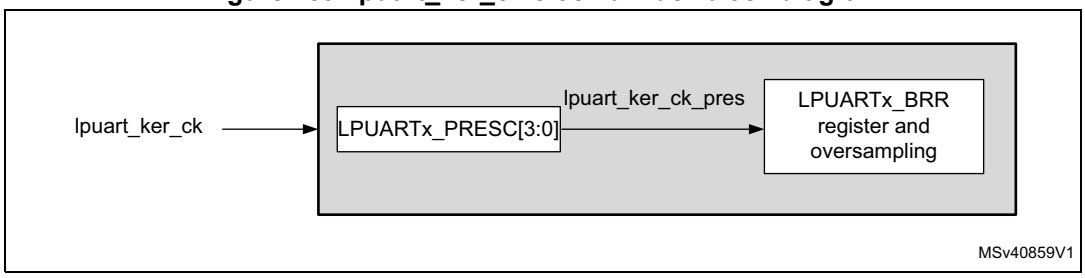
- Possible use of the LPUART in low-power mode
- Communication speed.

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 459. 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 302. 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 303*:

- Number of stop bits defined through STOP[1:0] bits in the LPUART_CR2 register
- LPUART_BRR register value.

Table 303. 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 303 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:

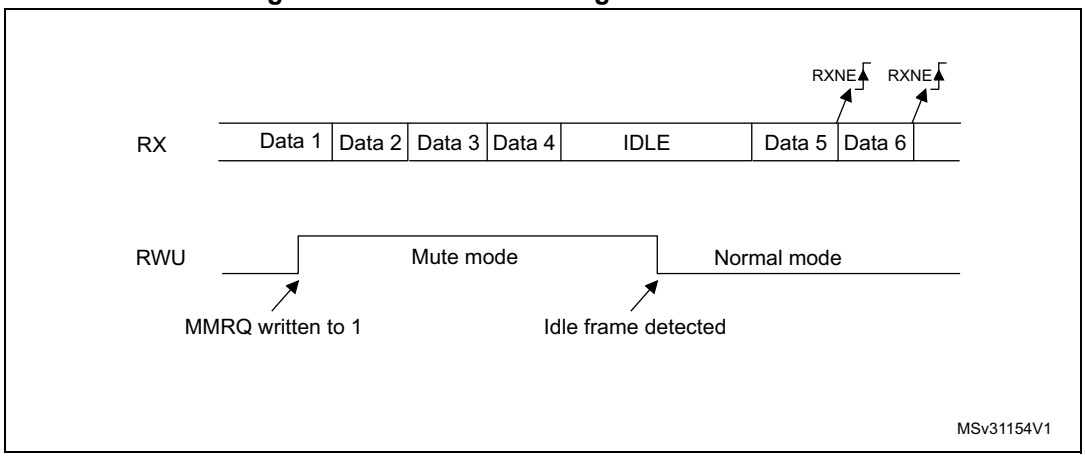
- Idle Line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

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 460*.

Figure 460. 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.

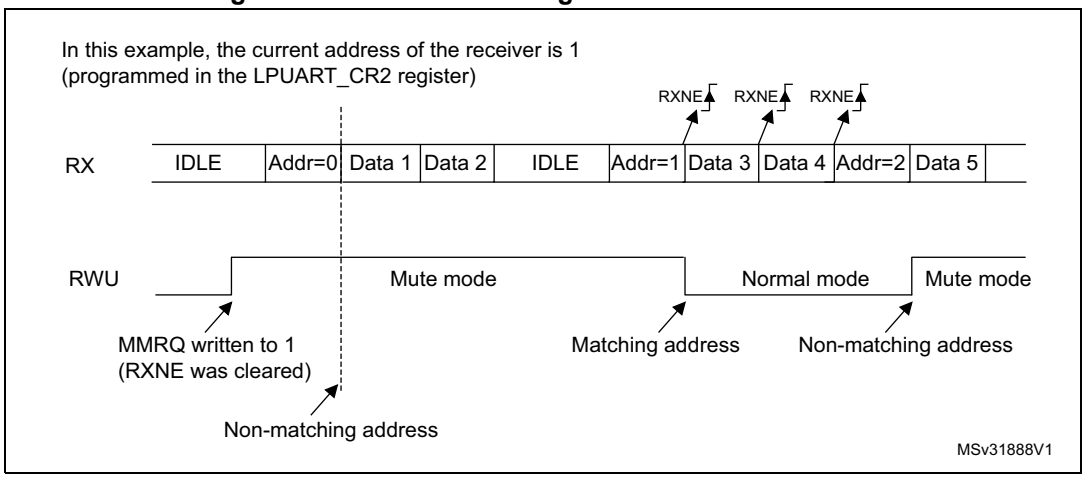
The LPUART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

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 461*.

Figure 461. 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 304*.

Table 304: 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 on page 1425 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):

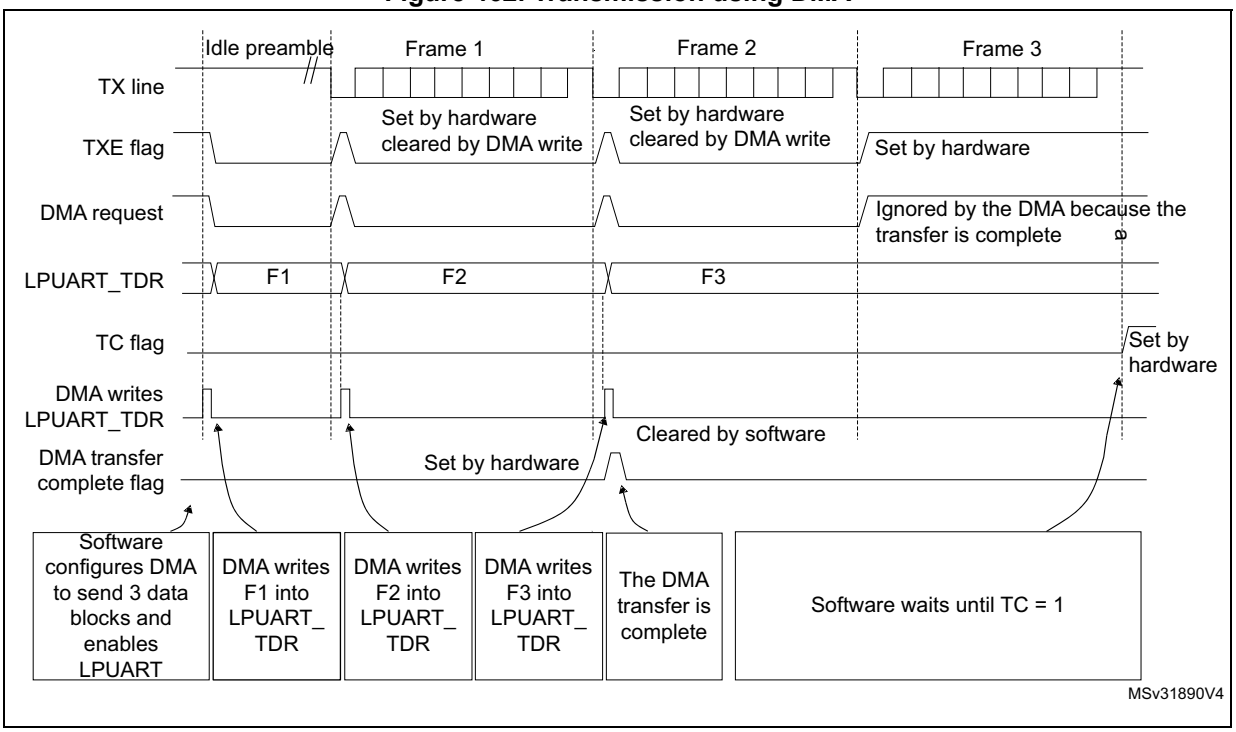
1. Write the LPUART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the 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 462. 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

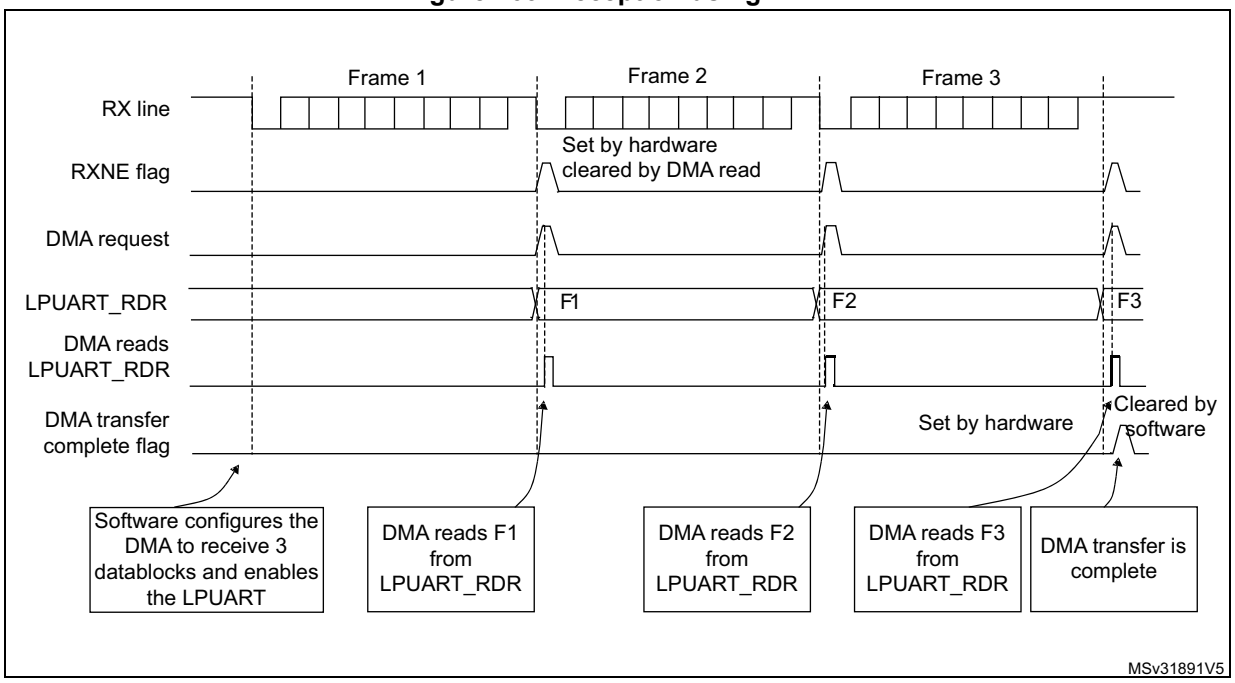
DMA mode can be enabled for reception by setting the DMAR bit in the LPUART_CR3 register. Data are loaded from the LPUART_RDR register to a SRAM area configured using the DMA peripheral (refer to section *direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for 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 463. 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

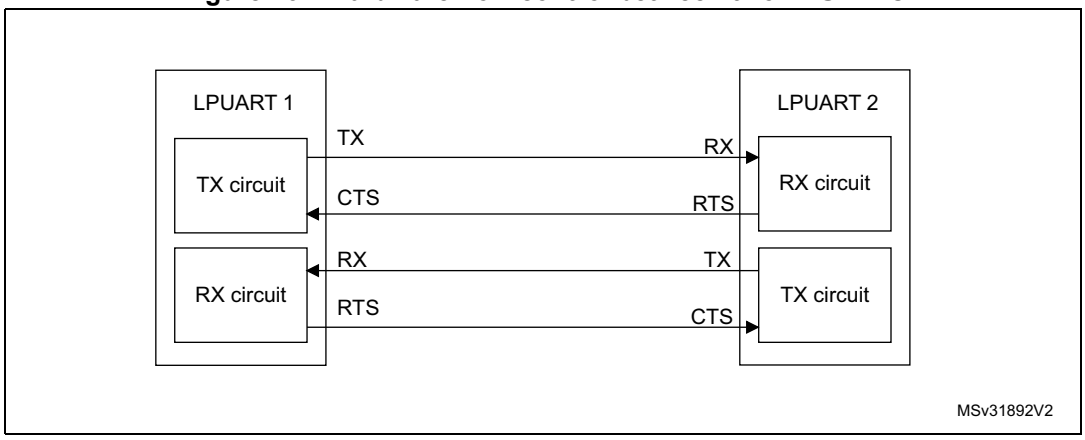
If any error occurs during a transaction In multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set.

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 464* shows how to connect two devices in this mode.

Figure 464. Hardware flow control between two LPUARTs

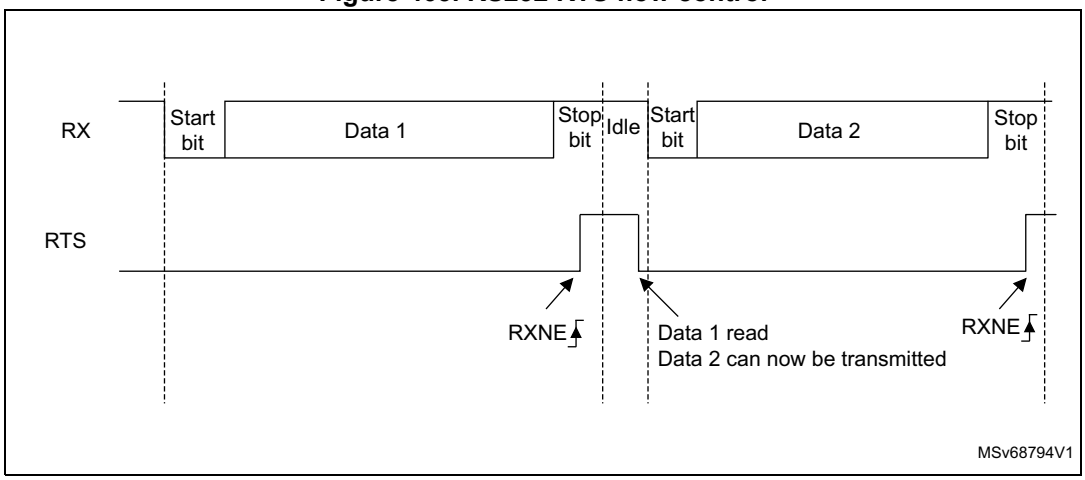


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART_CR3 register).

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 465* shows an example of communication with RTS flow control enabled.

Figure 465. RS232 RTS flow control



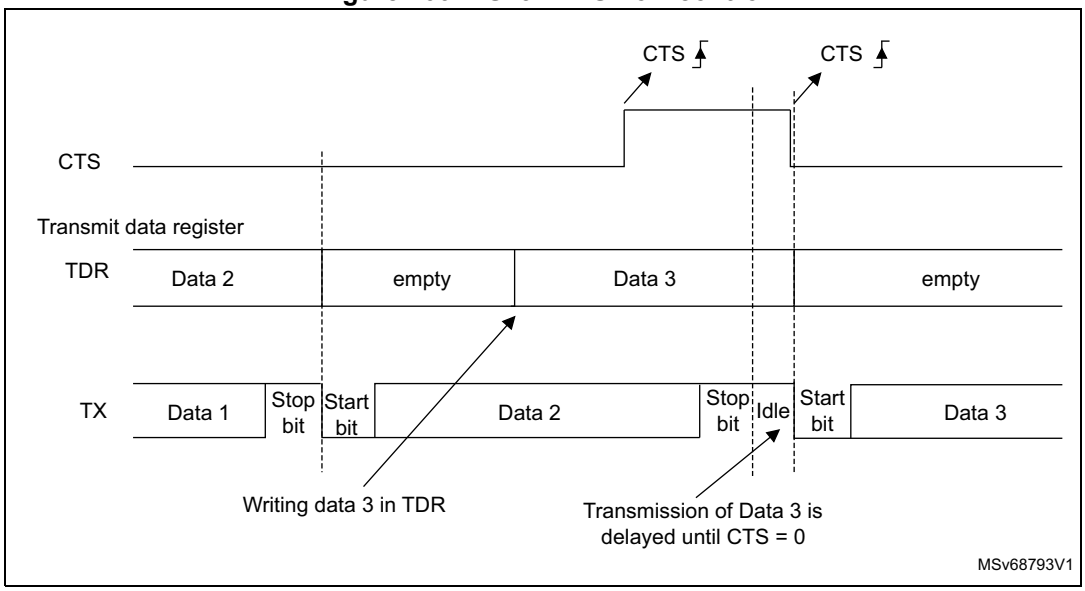
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

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 466* shows an example of communication with CTS flow control enabled.

Figure 466. 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 low-power management

The LPUART has advanced low-power mode functions that enable it to transfer properly data even when the lpuart_pclk clock is disabled.
The LPUART is able to wake up the MCU from low-power mode when the UESM bit is set.
When the lpuart_pclk is gated, the LPUART provides a wake-up interrupt (**lpuart_wkup**) if a specific action requiring the activation of the **lpuart_pclk** clock is needed:

- If FIFO mode is disabled

  luart_pclk clock has to be activated to empty the LPUART data register.

  In this case, the lpuart_wkup interrupt source is the RXNE set to 1. The RXNEIE bit must be set before entering low-power mode.

- If FIFO mode is enabled

  luart_pclk clock has to be activated
  - To fill the TXFIFO, or
  - To empty the RXFIFO

  In this case, the lpuart_wkup interrupt source can be:
  - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
  - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
  - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

  This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

  To avoid overrun/underrun errors and transmit/receive data in low-power mode, the lpuart_wkup interrupt source can be one of the following events:
  - TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
  - RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

  For example, the application can set the threshold to the maximum RXFIFO size if the wake-up time is less than the time to receive a single byte across the line.

  Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wake up the MCU from low-power mode enables doing as many

LPUART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific **lpuart_wkup** interrupt may be selected through the WUS bitfields.

When the wake-up event is detected, the WUF flag is set by hardware and the **lpuart_wkup** interrupt is generated if the WUFIE bit is set.

Note: Before entering low-power mode, make sure that no LPUART transfer is ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wake-up event is detected, independently of whether the MCU is in low-power or in an active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to ensure that the LPUART is actually enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and reenabled upon exit from low-power mode.

When FIFO is enabled, the wake-up from low-power mode on address match is only possible when mute mode is enabled.

Using mute mode with low-power mode

If the LPUART is put into mute mode before entering low-power mode:

- Wake-up from mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wake-up from mute mode on address match is used, then the low-power mode wake-up source from must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, mute mode is used with wake-up from low-power mode without any constraints (that is, the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).

Wake-up from low-power mode when LPUART kernel clock lpuart_ker_ck is OFF in low-power mode

If during low-power mode, the lpuart_ker_ck clock is switched OFF, when a falling edge on the LPUART receive line is detected, the LPUART interface requests the lpuart_ker_ck clock to be switched ON thanks to the lpuart_ker_ck_req signal. The lpuart_ker_ck is then used for the frame reception.

If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, the lpuart_ker_ck is switched OFF again, the MCU is not woken up and stays in low-power mode and the kernel clock request is released.

The example below shows the case of a wake-up event programmed to “address match detection” and FIFO management disabled.

Figure 467 shows the behavior when the wake-up event is verified.

Figure 467. Wake-up event verified (wake-up event = address match, FIFO disabled)

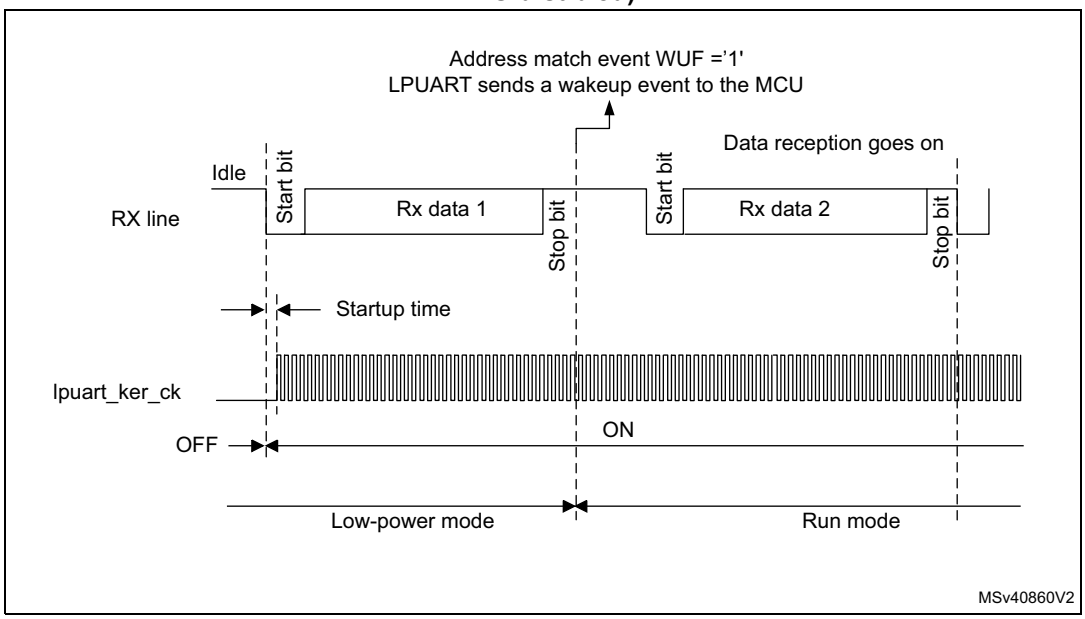
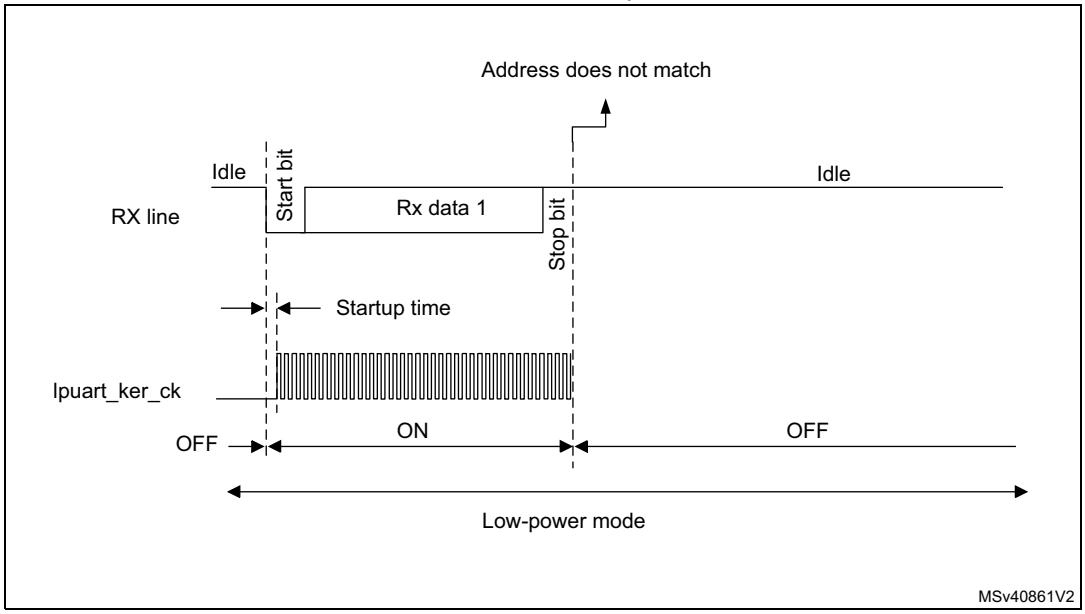


Figure 468 shows the behavior when the wake-up event is not verified.

Figure 468. Wake-up event not verified (wake-up event = address match, FIFO disabled)



Note: The above figures are valid when address match or any received frame is used as wake-up event. In the case the wake-up event is the start bit detection, the LPUART sends the wake-up event to the MCU at the end of the start bit.

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 303: 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 reality, 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 305. Effect of low-power modes on the LPUART

Mode	Description
Sleep	No effect. LPUART interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the LPUART registers is kept. The LPUART is able to wake up the microcontroller from Stop mode when the LPUART is clocked by an oscillator available in Stop mode.
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 306* for a detailed description of all LPUART interrupt requests.

Table 306. 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	No	No
	Transmit FIFO Not Full	TXFNF	TXFNFIE	TXFIFO full		No	
	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		No	
	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		No	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		No	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		No	
	Overrun error in multibuffer communication.	ORE(3)		Write 1 in ORECF		No	
	Framing Error in multibuffer communication.	FE		Write 1 in FECF		No	
	Character match	CMF	CMIE	Write 1 in CMCF		No	
	Wake-up from low-power mode	WUF	WUFIE	Write 1 in WUC		Yes	

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. When OVRDIS = 0.

37.7 LPUART registers

Refer to *Section 1.2 on page 65* 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. It is set and cleared by software.

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 wake up the MCU from 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 able to wake up the MCU from low-power mode.

1: LPUART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

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
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. It is set and cleared by software.

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 wake up the MCU from 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 able to wake up the MCU from low-power mode.

1: LPUART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

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 should 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 WUS[1:0] is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter should be equal to 1.
- 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 reception is disabled (RE = 0) or 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	WUFIE	WUS1	WUS0	Res.	Res.	Res.	Res.
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	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].

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: LPUART interrupt generated whenever WUF = 1 in the LPUART_ISR register
Note: WUFIE must be set before entering in low-power mode.
If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 37.3: LPUART implementation on page 1425.*

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01: Reserved.
10: WUF active on start bit detection
11: WUF active on RXNE/RXFNE.
This bitfield can only be written when the LPUART is disabled (UE = 0).
Note: If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 37.3: LPUART implementation on page 1425.*

Bits 19:16 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.	WUFIE	WUS1	WUS0	Res.	Res.	Res.	Res.
									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 31:23 Reserved, must be kept at reset value.

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: LPUART interrupt generated whenever WUF = 1 in the LPUART_ISR register

Note: WUFIE must be set before entering in low-power mode.

If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to* *Section 37.3: LPUART implementation on page 1425**.*

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection

This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).

00: WUF active on address match (as defined by ADD[7:0] and ADDM7)

01: Reserved.

10: WUF active on start bit detection

11: WUF active on RXNE/RXFNE.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Note: If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to* *Section 37.3: LPUART implementation on page 1425**.*

Bits 19:16 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	WUF	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
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 27) 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 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register.

An interrupt is generated if WUFIE = 1 in the LPUART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 37.3: LPUART implementation on page 1425.*

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.

It is set by hardware when the transmission of a frame containing data has completed and the TXFE bit is set.

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 a write to the LPUART_TDR register.

0: Transmission has not completed

1: Transmission has completed

Note: If the TE bit is reset and no transmission is ongoing, the TC bit is immediately set.

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	WUF	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
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 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register.

An interrupt is generated if WUFIE = 1 in the LPUART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 37.3: LPUART implementation on page 1425.*

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.	WUCF	Res.	Res.	CMCF	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.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
						w			w		w	w	w	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wake-up from low-power mode clear flag
Writing 1 to this bit clears the WUF flag in the LPUART_ISR register.
Note: If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value.* Refer to *Section 37.3: LPUART implementation on page 1425.*

Bits 19:18 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 on page 1425.

37.7.14 LPUART register map

Table 307. 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_CR1 FIFO 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
0x00	LPUART_CR1 FIFO 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	WUFIE	WUS 1	WUS 0	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	0			0			0
0x08	LPUART_CR3 FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUS 1	WUS 0	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
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

Table 307. 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
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_ISR FIFO mode enabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFF	TXFF	REACK	TEACK	WUF	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	0		1	0	0	0	0	0	0	0
0x1C	LPUART_ISR FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	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	0		1	0	0	0	0	0	0	0
0x20	LPUART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	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	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

Refer to *Section 2.2: Memory organization* for the register boundary addresses.

38 Serial peripheral interface (SPI)

38.1 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 (SS) 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)
- From 4-bit up to 32-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
- Hardware or software management of SS for both master and slave
- Adjustable minimum delays between data and between SS and data flow
- Configurable SS 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 SS 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)
- 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. All the SPI instances have a full set of features.

Table 308. SPI features

SPI feature	SPI1, SPI2, SPI3 (full-featured instances)
Data and CRC size	Configurable from 4 to 32 bits
CRC computation	CRC polynomial length configurable from 5 to 33 bits
FIFO size	16x8 bits
FIFO threshold	1 - 16 data
Number of data control (TSIZE)	Up to 65535
I2S feature	Yes
Autonomous in Stop modes with wake-up capability	No
Autonomous in LP-Stop and Standby modes with wake-up capability	No

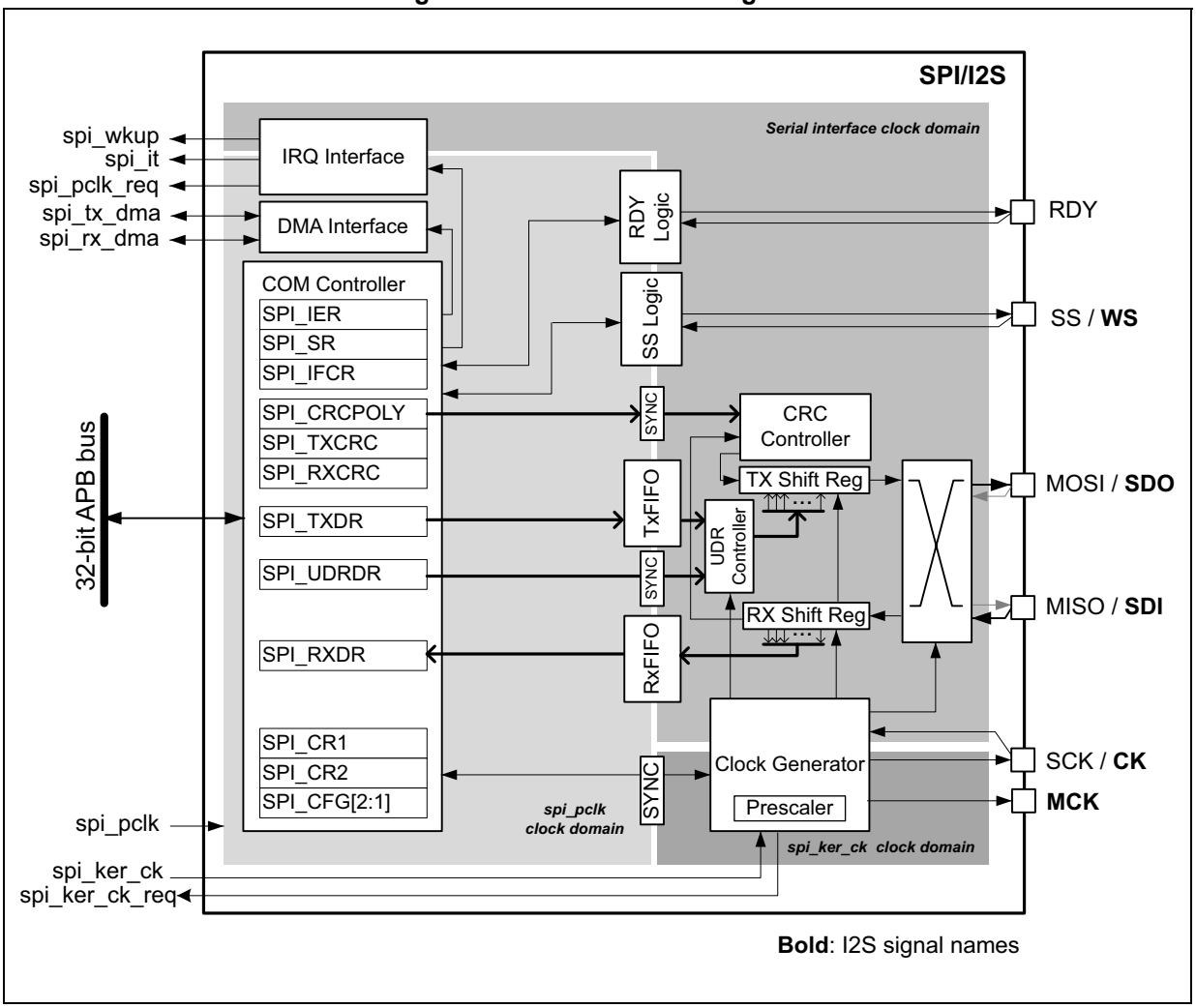
Note: For detailed information about instance capabilities to exit from Stop and Standby modes, refer to Table 312: SPI wake-up and interrupt requests.

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 469*.

Figure 469. SPI/I2S block diagram



The simplified scheme of *Figure 469* 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.
- **SS:** slave select pin. Depending on the SPI and SS 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.

All these pins (except RDY) are shared in the I2S mode. This mode features an additional I2S specific MCK signal. For more details about I2S signals, see *Section 38.9.2*.

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 309* and *Table 310* for the list of SPI input / output pins and internal signals.

Table 309. SPI/I2S input/output pins(1)

Pin name	I/O type	Description
MISO/SDI(2)	Input/output	Master data input / slave data output
MOSI/SDO(2)	Input/output	Master data output / slave data input
SCK/CK	Input/output	Master clock output / slave clock input

Table 309. SPI/I2S input/output pins(1)

Pin name	I/O type	Description
SS/WS	Input/output	Master output / slave selection input
RDY	Input/output	SPI master input / slave FIFOs status occupancy output
MCK	Output	I2S master frequency output

1. Refer to the section *Section 38.9.2: Pin sharing with SPI function* for details.
2. Functionality of MOSI/SDO and MISO/SDI pins can be swapped. Their directions may vary in SPI bidirectional half-duplex mode.

Description of SPI input/output signals

Table 310. 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

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 SS management) or three or four wires (with hardware SS 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 SS 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

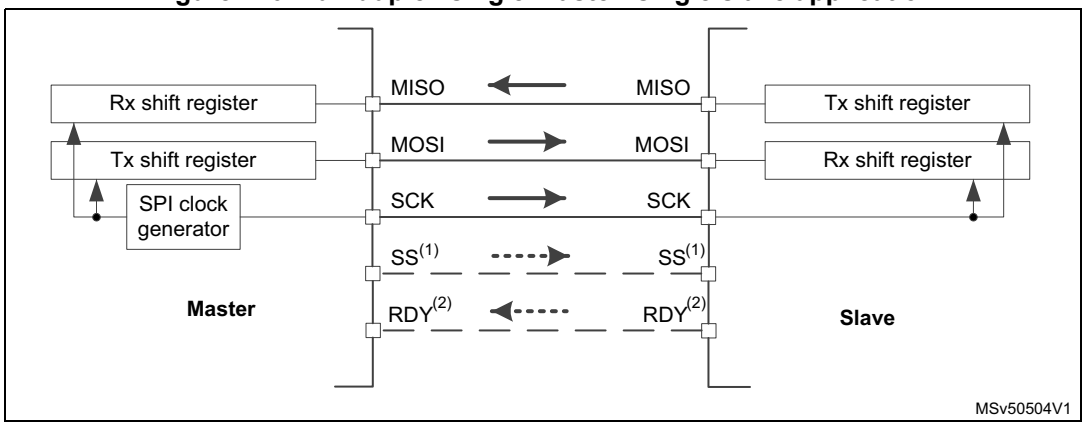
The communication flow can use one of three possible modes: the full-duplex (three wires) mode, half-duplex (two wires) mode, or the simplex (two wires) mode. The SS signal is optional in single master-slave configuration and is often not connected between the two communication nodes. Nevertheless, the SS 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 470. Full-duplex single master/ single slave application

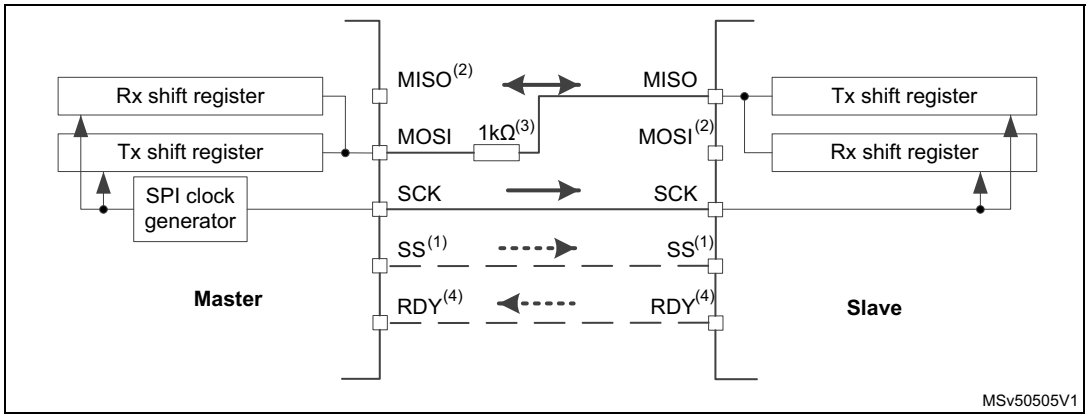


1. The SS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 38.4.7*).
2. The RDY signal provided by the slave can be read by the master optionally.

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 471. Half-duplex single master/ single slave application



1. The SS 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 SS pin (or software managed internal signal) is active (see *Section 38.4.7*).

- **Receive-only mode: COMM[1:0] = 10**

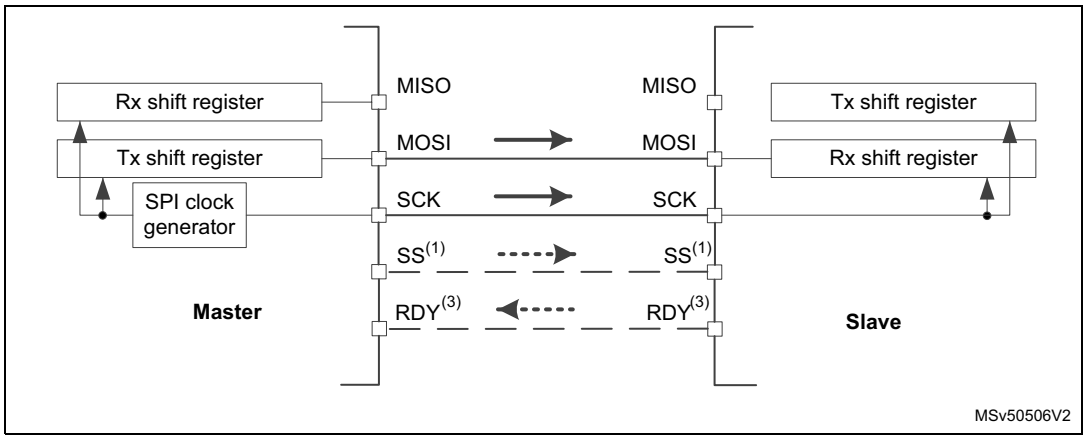
  In master mode, the MOSI output is disabled and can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled and the CSTART bit in the SPI_CR1 register is set. The clock is stopped either by software explicitly requesting this by setting the CSUSP bit in the SPI_CR1 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 472. Simplex single master / single slave application (master in transmit-only / slave in receive-only mode)



1. The SS 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.

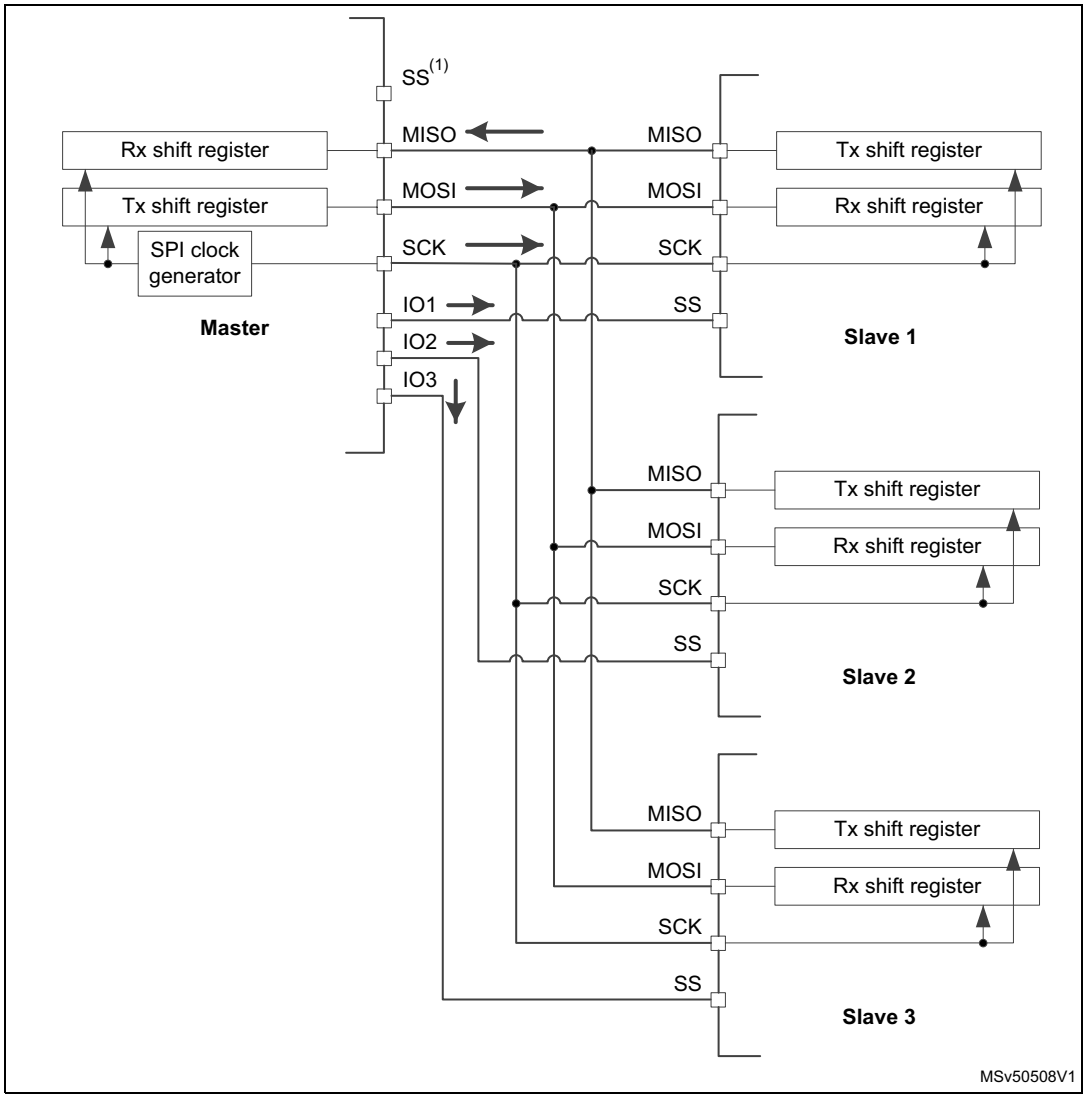
38.4.5 Standard multislave communication

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 473.*).

The master must select one of the slaves individually by pulling low the GPIO connected to the slave SS 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 473. Master and three independent slaves connected in star topology



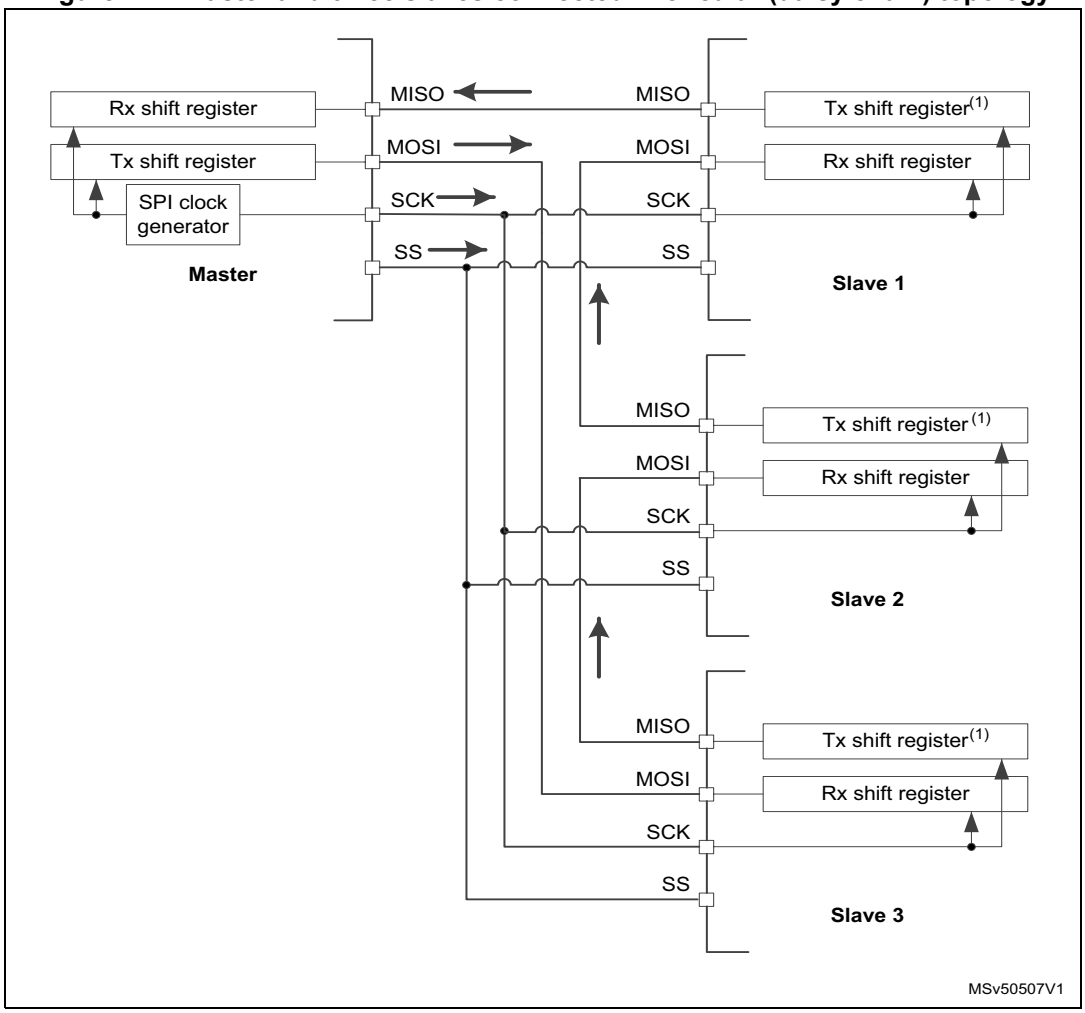
1. The master single SS pin hardware output functionality cannot support this topology (to be replaced by a set of GPIOs under software control). The SS pin is free for other application uses (such as GPIO or other alternate functions). Refer to *Section 38.4.7* for details.
2. If the application cannot ensure that no more than a single SS 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 in time when a circular topology is applied (see *Figure 474*). All the slaves behave like simple shift registers applied in serial chain under control of common slave select (SS) 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 474. 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.

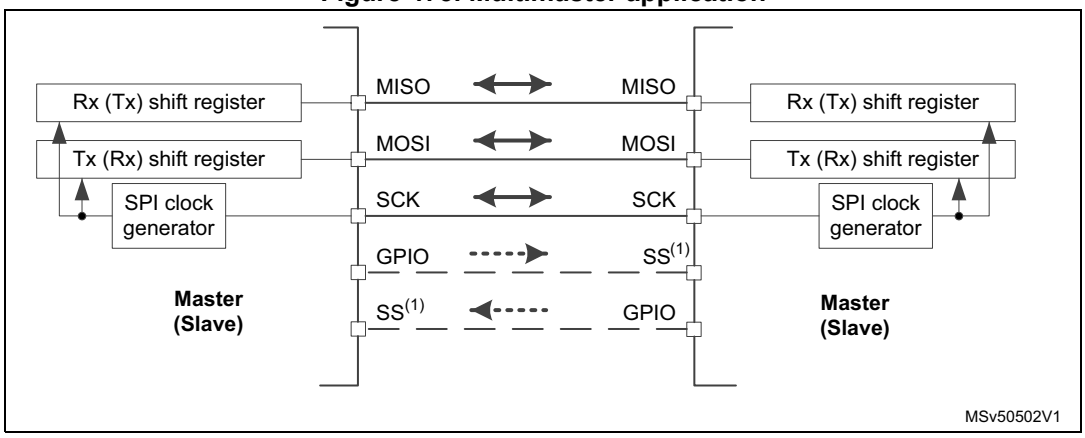
38.4.6 Multimaster communication

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 SS 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 475. Multimaster application



1. The SS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control as the passive node is configured as a slave.
2. The RDY signal is not used in this communication.

38.4.7 Slave select (SS) pin management

In slave mode, the SS works as a standard ‘chip select’ input and lets the slave communicate with the master. In master mode, the SS 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 SS signal can be managed internally (software management of the SS input) or externally when both the SS input and output are associated with the SS pin (hardware SS management). The user can configure which level of this input/output external signal (present on the SS pin) is considered as active by the SSIOP bit setting. SS 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 SS management (SSM = 1)**: in this configuration, slave select information is driven internally by the SSI bit value in the SPI_CR1 register. The external SS pin is free for other application uses (such as GPIO or other alternate functions).
- **Hardware SS management (SSM = 0)**: in this case, there are two possible configurations. The configuration used depends on the SS output configuration (SSOE bit in the SPI_CFG2 register).
  - **SS output enable (SSOE = 1)**: this configuration is only used when the MCU is set as master. The SS 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 SS output correctly. Even if SPI AF is not applied at the SS pin (it can be used as a standard GPIO then), keep anyway SSOE = 1 to ensure the default SS input level and prevent any mode fault evaluation at the input of the master SS internal logic applicable at a multimaster topology exclusively.

  a) When SSOM = 0 and SP = 000, the SS 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 SS 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) SS input is forced to nonactive state internally at master to prevent any mode fault.

  – **SS output disable (SSM = 0, SSOE = 0)**:

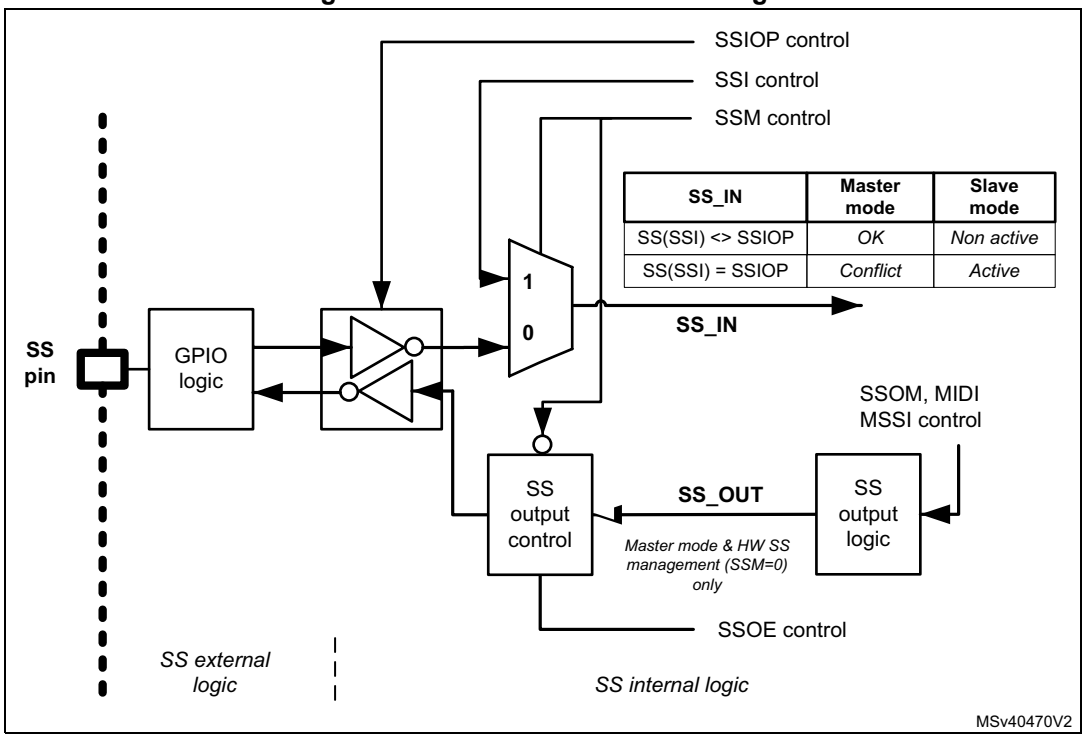
  a) If the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the SS pin is pulled into an active level in this mode, the SPI enters master mode fault state and the SPI device is automatically reconfigured in slave mode (MASTER = 0).

  b) In slave mode, the SS pin works as a standard ‘chip select’ input and the slave is selected while the SS 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.

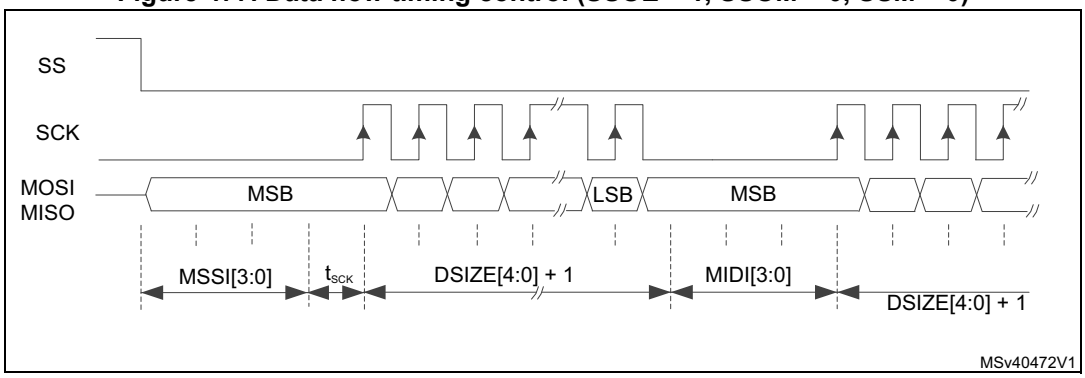
When the SPI slave is enabled in the hardware SS management mode, all the transfers are ignored even in case of the SS is found at active level. They are ignored until the slave detects a start of the SS signal (transition from nonactive to active level) just synchronizing the slave with the master. This is because the hardware management mode cannot be used when the external SS pin is fixed. There is no such protection in the SS 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 476. Scheme of SS control logic



When the hardware output SS 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 SS signal between data frames and can insert an extra delay at the beginning of every transfer (to separate the SS 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 477*).

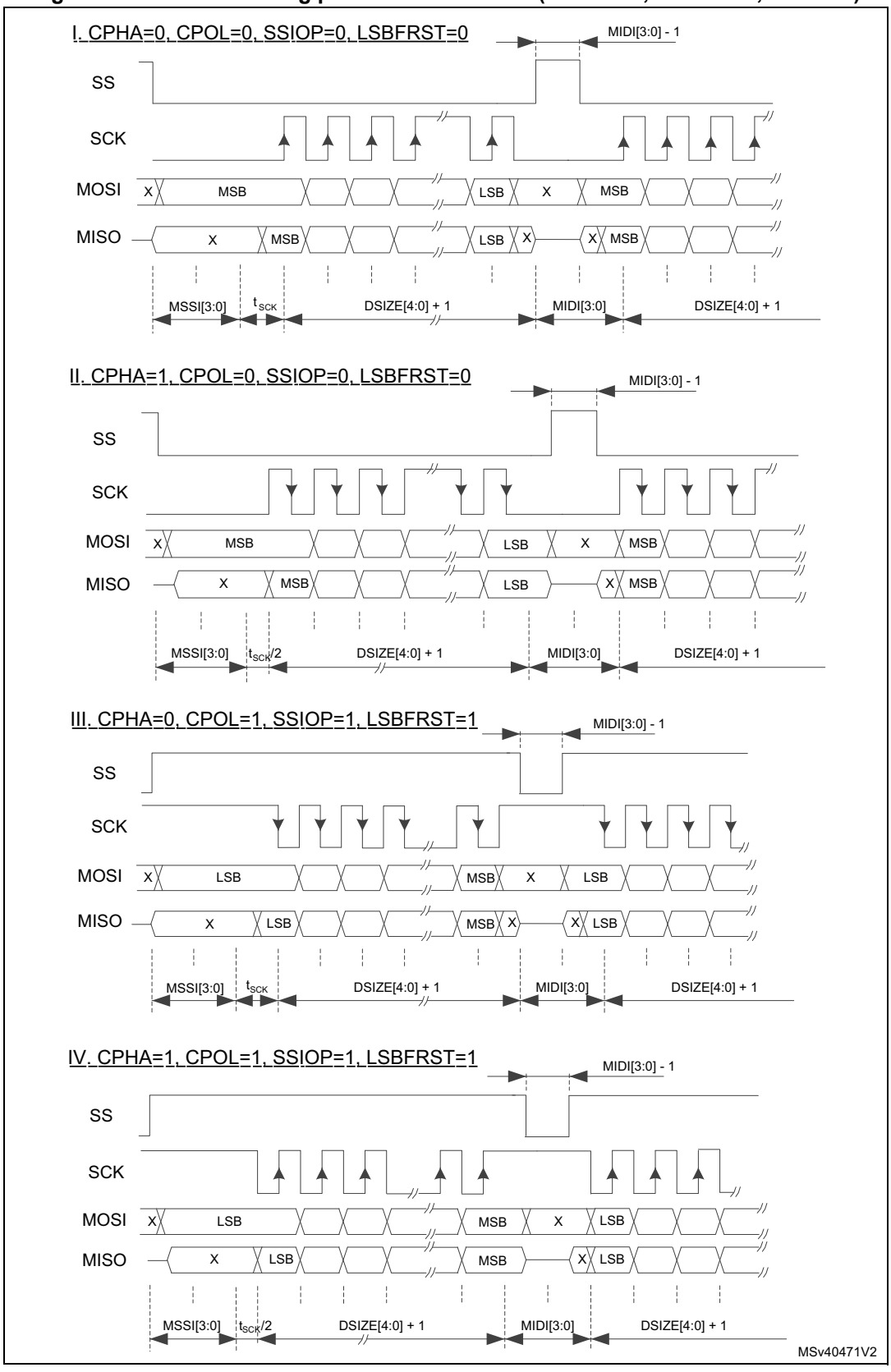
Figure 477. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0)



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 478*.

Figure 478. SS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0)



1. MSSI[3:0] = 0010, MIDI[3:0] = 0010.
2. SS 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 SS.

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-bit.

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.

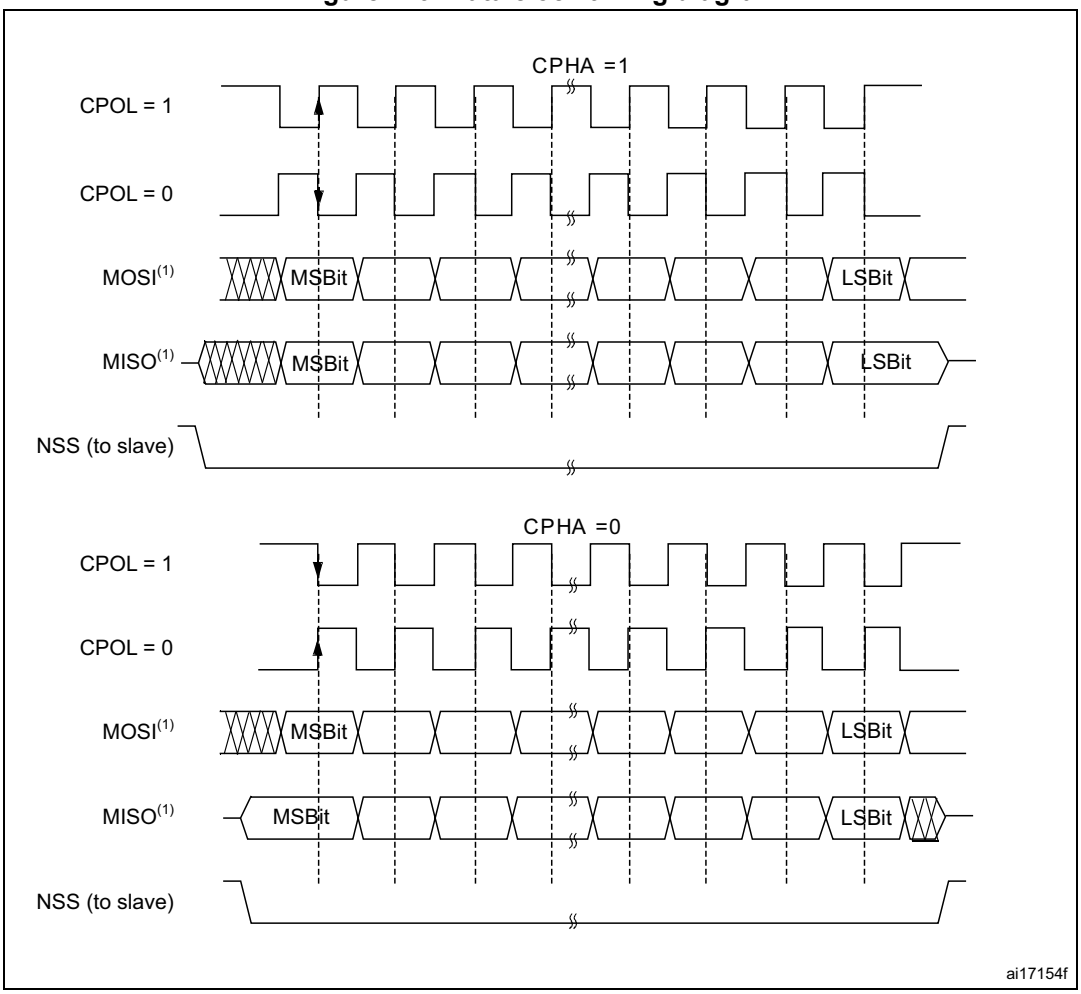
The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edges (dotted lines in *Figure 479*).

Figure 479 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 479. Data clock timing diagram



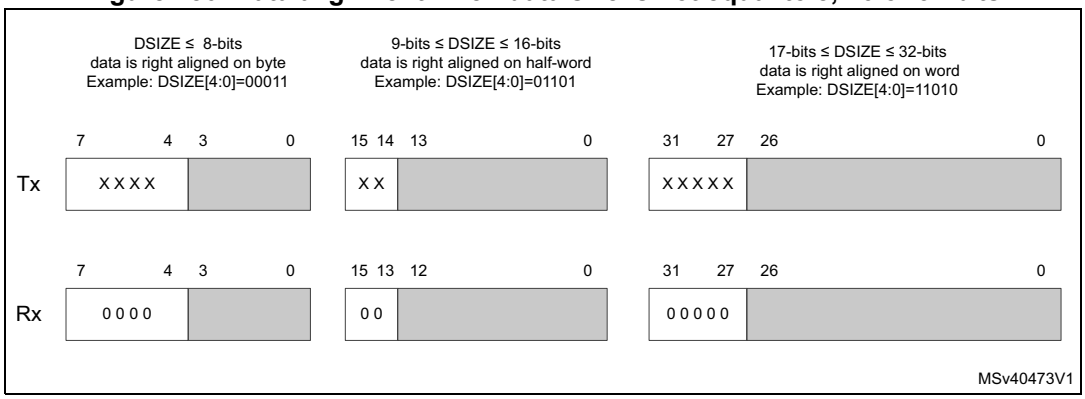
1. The order of data bits depends on the LSBFRST bit setting.

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 (see *Figure 480*). 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 that is not aligned with 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*.

Figure 480. Data alignment when data size is not equal to 8, 16 or 32 bits



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, SS, 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 SS signal is managed by hardware at slave or kept inactive by the slave software when the software management of the SS 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 SS, else the following transfer may be desynchronized.

When the SPI slave is enabled at the hardware SS management mode, all the transfers are ignored even in case of the SS is found at active level. They are ignored until the slave detects a start of the SS 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 SS pin is fixed. There is no such protection at the SS 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.
- Data frames from 17 to 24 bits occupy three bytes of the FIFO.
- Data frames from 25 to 32 bits occupy four 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. If the data pattern is not byte-, half-word, or word-aligned, only the valid data bits are stored right-aligned to the FIFO and transferred on the bus. Unused bits are discarded on the transmitter side and padded with zeros on the receiver side.

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 SS 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 SS 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 SS 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 SS 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 SS 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-SS 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 SS 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.

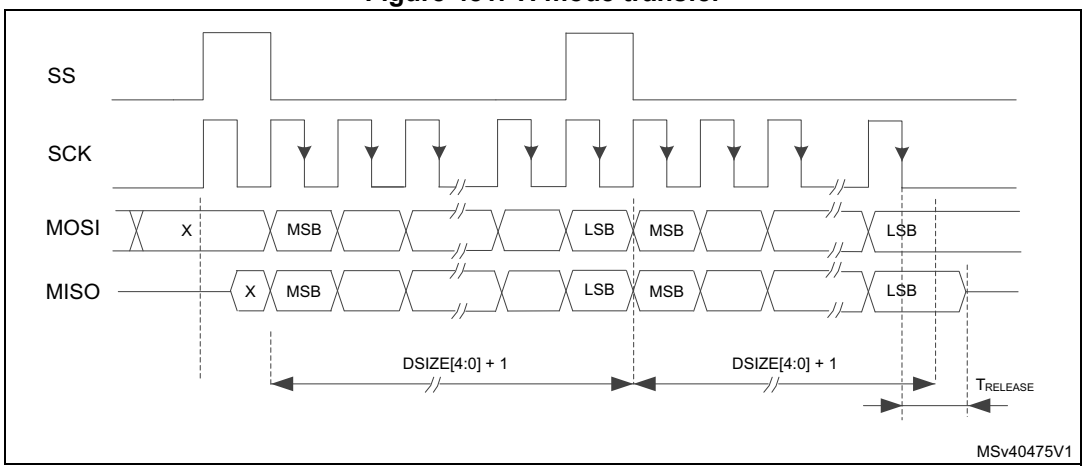
Configuring any DMA data access to less than the configured data size is forbidden. One complete data frame must be always accessed at minimum.

38.5 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 SS 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 SS signal synchronizes the protocol by pulses over the LSB data bit as it is shown in *Figure 481*.

Figure 481. 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 SS 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)

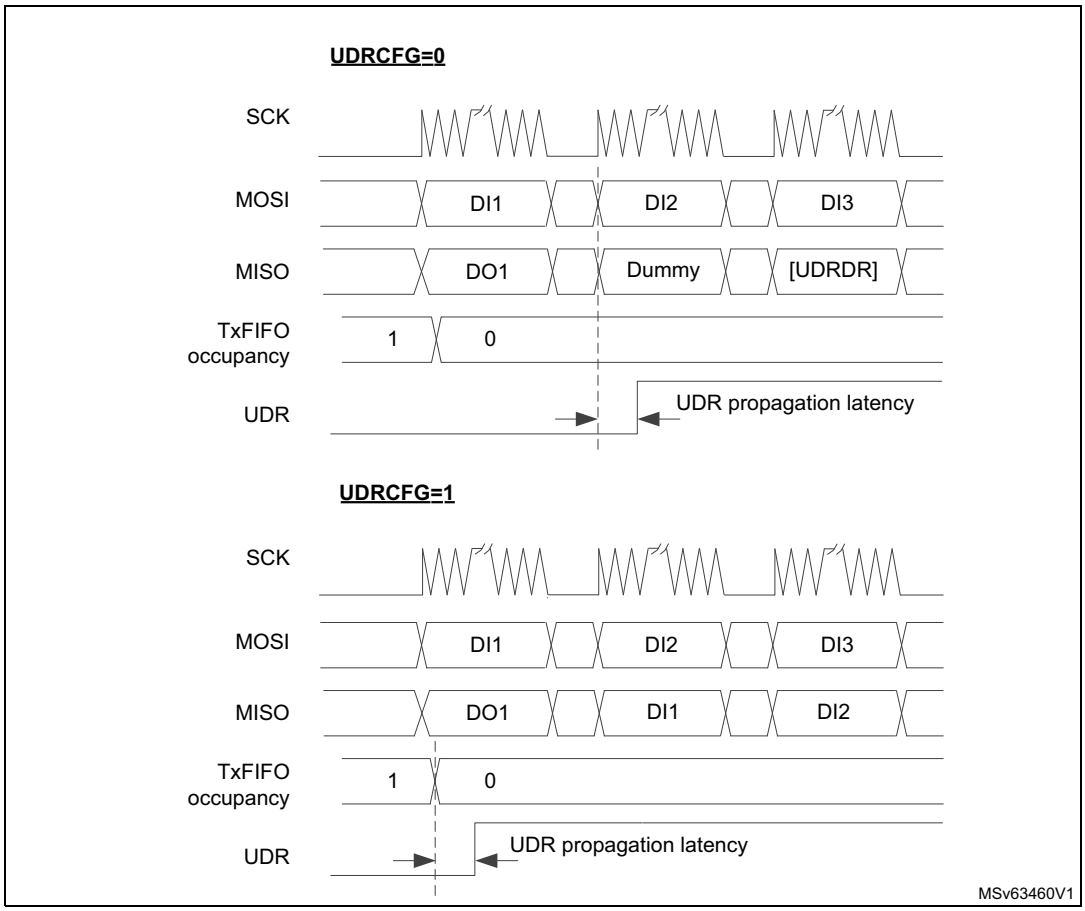
In a slave-transmitting mode, the underrun condition is captured internally by hardware if no data is available for transmission in the slave TxFIFO commonly. The UDR flag setting is then propagated into the status register by hardware (see note below). UDR triggers an interrupt if the UDRIE bit is set.

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 482*). The second configuration (UDRCFG=1) can be used in circular topography structures (see *Figure 474*). 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 lastly 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 482. 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 SS signal (SS pin in SS hardware mode, or SSI bit in SS 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 SS 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 SS control into active level.
3. Perform a data transfer.
4. Put GPIO pin dedicated for another master SS 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 SS 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 SS pulse is ignored, and the SPI waits for the next SS 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 SS 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:

- For a 32-bit instance: write 0x11021 to the SPI_CRCPOLY register and clear the CRC33_17 bit.
- For a 16-bit instance: to obtain the full size, write 0x1021, and set the CRC33_17 bit.

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 0xFFFF for a full-featured instance, and to 0x3FF for a limited-featured instance, 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

Table 311. Effect of low-power modes on the SPI

Mode	Description
Sleep	No effect. SPI interrupts cause the device to exit Sleep mode.
Stop(1)	The SPI registers content is kept.
Standby	The SPI instance is not functional in this mode. It is powered down, and must be reinitialized after exiting Standby mode.

1. Refer to *Section 38.3: SPI implementation* for information about wake-up from Stop mode support per instance as well as Standby mode availability. If an instance is not functional in a Stop mode, it must be disabled before entering this Stop mode.

38.7 SPI interrupts

Table 312 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 312. 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 I2S main features

- Full duplex communication
- Simplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler
- Data length can be 16, 24 or 32 bits[a]
- Channel length can be 16 or 32 in master, any value in slave
- Programmable clock polarity
- Error flags signaling for improved reliability: Underrun, overrun, and frame errors
- Embedded Rx and TxFIFOs
- Supported I^2S protocols:
  - I^2S Philips standard
  - MSB-justified standard (left-justified)
  - LSB-justified standard (right-justified)
  - PCM standard (with short and long frame synchronization)
- Data ordering programmable (LSb or MSb first)
- DMA capability for transmission and reception
- Master clock can be output to drive an external audio component:
  - F_{MCK} = 256 x F_{WS} for all I2S modes
  - F_{MCK} = 128 x F_{WS} for all PCM modes

Note: F_{MCK} is the master clock frequency and F_{WS} is the audio sampling frequency.

38.9 I2S functional description

38.9.1 I2S general description

The block diagram shown on *Figure 469* also applies for I2S mode.

The SPI/I2S block can work in I2S/PCM mode, when the bit I2SMOD is set. A dedicated register (SPI_I2SCFGR) is available for configuring the dedicated I2S parameters, which include the clock generator, and the serial link interface.

The I2S/PCM function uses the clock generator to produce the communication clock when the SPI/I2S is set in master mode. This clock generator is also the source of the master clock output (MCK).

Resources such as RxFIFO, TxFIFO, DMA, and parts of interrupt signaling are shared with the SPI function. The low-power mode function is also available in I2S mode (refer to *Section 38.6: SPI in low-power modes* and *Section 38.10: I2S interrupts*).

a. Not always available, refer to *Section 38.3: SPI implementation* in order to check if 24 and 32-bit data widths are supported.

38.9.2 Pin sharing with SPI function

The I2S shares four common pins with the SPI:

- SDO: serial data output (mapped on the MOSI pin) to transmit the audio samples in master, and receive the audio sample in slave mode. Refer to *Section : Serial data line swapping on page 1522*.
- SDI: serial data input (mapped on the MISO pin) to receive the audio samples in master, and transmit the audio sample in slave mode. Refer to *Section : Serial data line swapping on page 1522*.
- WS: word select (mapped on the SS pin). This is the frame synchronization. It is configured as output in master mode, and as input in slave mode.
- CK: serial clock (mapped on the SCK pin). This is the serial bit clock. It is configured as output in master mode, and as input in slave mode.

An additional pin can be used when a master clock output is needed for some external audio devices:

- MCK: master clock (mapped separately) is used when the I2S is configured in master mode.

38.9.3 Bitfields usable in I2S/PCM mode

When the I2S/PCM mode is selected (I2SMOD = '1'), some bitfields are no longer relevant, and must be forced to a specific value to guarantee the behavior of the I2S/PCM function. *Table 313* shows the list of bits and fields available in the I2S/PCM mode, and indicates which must be forced to a specific value.

Table 313. Bitfields usable in PCM/I2S mode

Register name	Bitfields usable in PCM/I2S Mode	Constraints on other bitfields
SPI/I2S control register 1 (SPI_CR1)	IOLOCK, CSUSP, CSTART, SPE	Other fields set to their reset values
SPI/I2S control register 2 (SPI_CR2)	-	Set to reset value
SPI/I2S configuration register 1 (SPI_CFG1)	TXDMAEN, RXDMAEN, FTHLV	Other fields set to their reset values
SPI/I2S configuration register 2 (SPI_CFG2)	AFCNTR, LSBFRST, IOSWP	Other fields set to their reset values
SPI/I2S interrupt enable register (SPI_IER)	TIFREIE, OVRIE, UDRIE, TXPIE, RXPIE	
SPI/I2S status register (SPI_SR)	SUSP, TIFRE, OVR, UDR, TXP, RXP	Other flags not relevant
SPI/I2S interrupt/status flags clear register (SPI_IFCR)	SUSPC, TIFREC, OVRC, UDRC	Other fields set to their reset values
SPI/I2S receive data register (SPI_RXDR)	The complete register	-

Table 313. Bitfields usable in PCM/I2S mode (continued)

Register name	Bitfields usable in PCM/I2S Mode	Constraints on other bitfields
SPI/I2S polynomial register (SPI_CRCPOLY)	-	Set to reset value
SPI/I2S transmitter CRC register (SPI_TXCRC)	-	
SPI/I2S receiver CRC register (SPI_RXCRC)	-	
SPI/I2S underrun data register (SPI_UDRDR)	-	
SPI/I2S configuration register (SPI_I2SCFGR)	The complete register	-

38.9.4 Slave and master modes

The SPI/I2S block supports master and slave modes for both I2S and PCM protocols. In master mode, both CK, WS and MCK signals are set to output.

In slave mode, both CK and WS signals are set to input. The signal MCK cannot be used in slave mode.

To improve the robustness of the SPI/I2S block in slave mode, the peripheral resynchronizes each reception and transmission on WS signal. This means that:

- In I2S Philips standard, the shift-in or shift-out of each data is triggered one bit clock after each transition of WS.
- In I2S MSB-justified standard, the shift-in or shift-out of each data is triggered as soon as a transition of WS is detected.
- In PCM short standard, the shift-in or shift-out of each data is triggered one bit clock after the active edge of WS.
- In PCM long standard, the shift-in or shift-out of each data is triggered as soon as the active edge of WS is detected

Note: This resynchronization mechanism is not available for the I2S LSB-justified standard.

Note as well that there is no need to provide a kernel clock when the SPI/I2S is configured in slave mode.

38.9.5 Supported audio protocols

The I2S/PCM interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits of the SPI_I2SCFGR register.

In the I2S protocol, the audio data are time-multiplexed on two channels: the left channel and the right channel. The WS signal is used to indicate which channel must be considered as the left, and which one is the right.

In I2S master mode, four frame formats are supported:

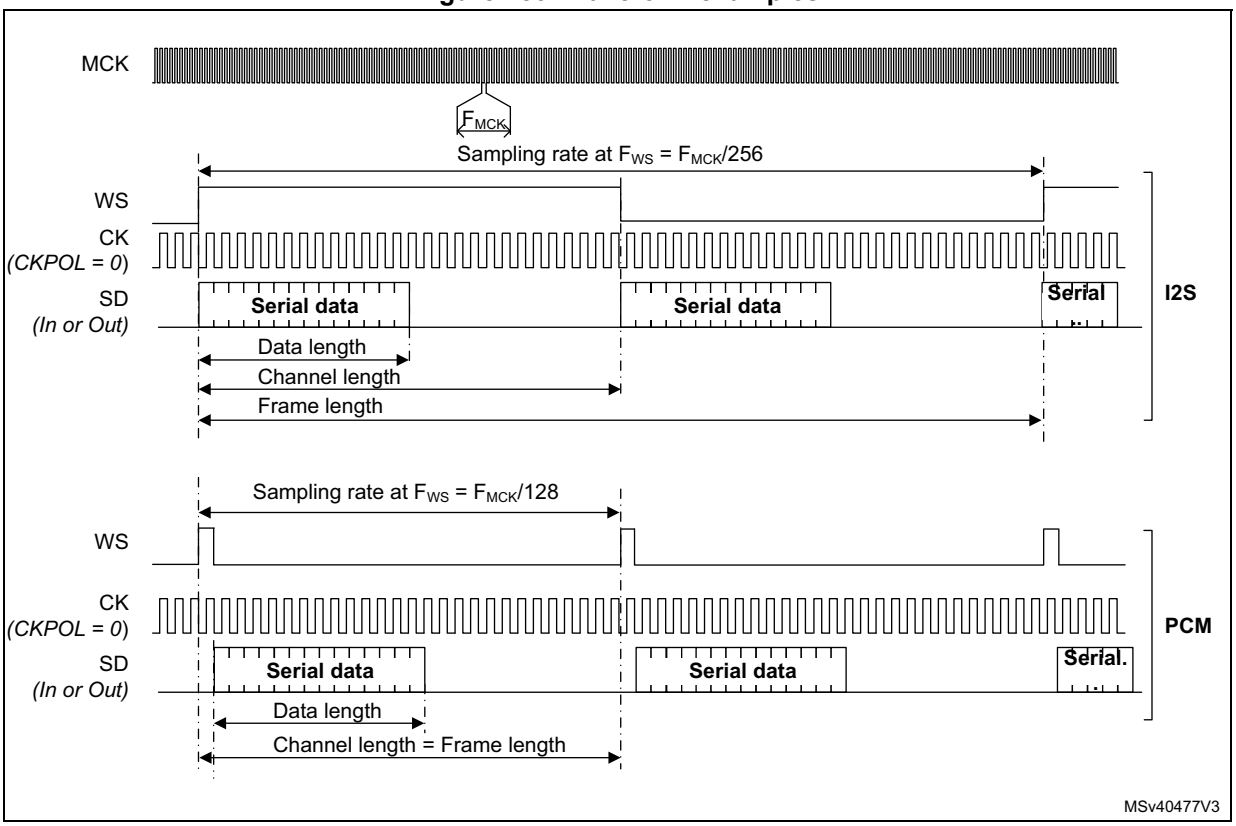
- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel[(a)]
- 32-bit data packed in a 32-bit channel[(a)]

In PCM master mode, three frame formats are supported:

- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel[(a)]

The figure hereafter shows the main definition used in this section: data length, channel length and frame length.

Figure 483. Waveform examples



For simplicity sake, in the next figures, SDI represents the serial data input and SDO the serial data output. Refer to *Section : Serial data line swapping* for details about the direction control of serial data lines.

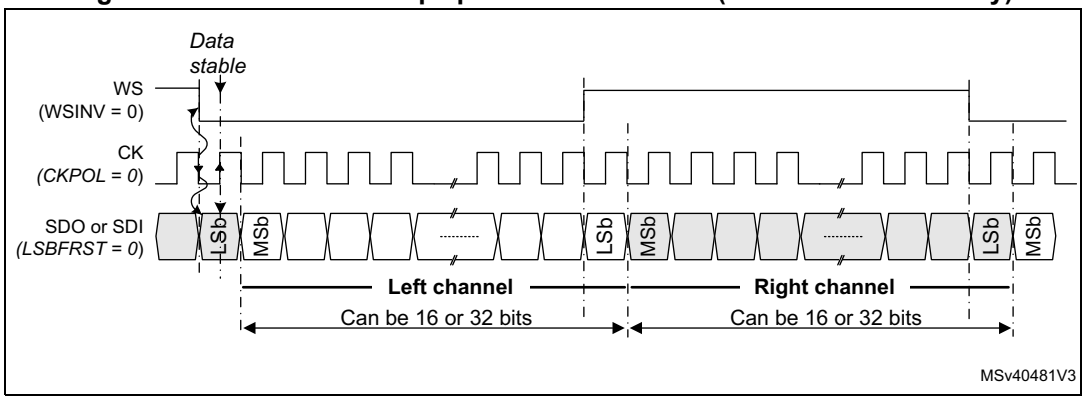
I²S Philips standard

The I2S Philips standard is selected by setting I2SSTD to 0b00. This standard is supported in master and slave mode.

In this standard, the WS signal toggles one CK clock cycle before the first bit (MSb in I2S Philips standard) is available. A falling edge transition of WS indicates that the next data transferred is the left channel, and a rising edge transition indicates that the next data transferred is the right channel.

a. Not always available, refer to *Section 38.3: SPI implementation* to check if 24 and 32-bit data widths are supported.

Figure 484. Master I2S Philips protocol waveforms (16/32-bit full accuracy)



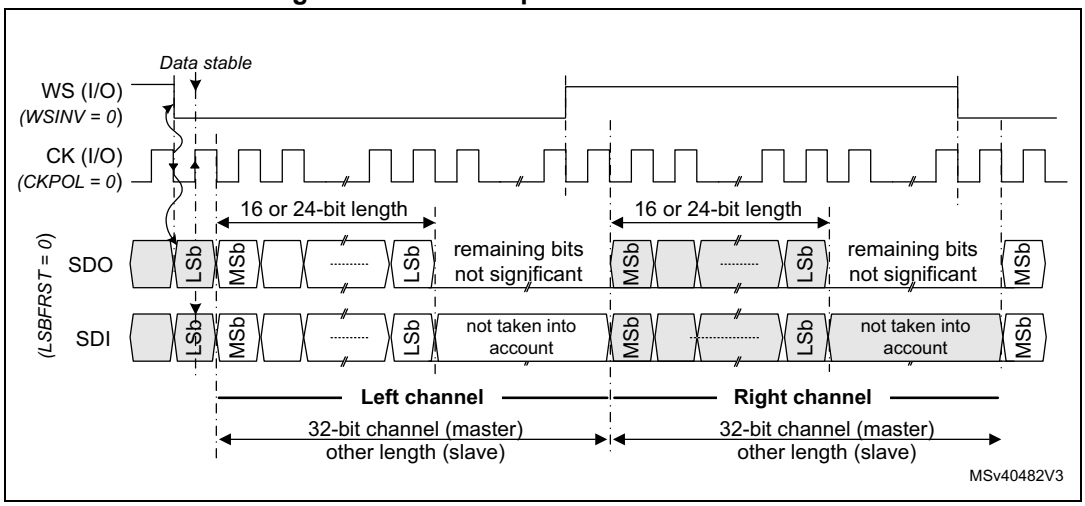
1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 38.3: SPI implementation* for the supported data sizes.

CKPOL is cleared to match the I2S Philips protocol. See *Selection of the CK sampling edge* for information concerning the handling of the WS signal.

Figure 484 shows an example of waveform generated by the SPI/I2S in the case where the channel length is equal to the data length. More precisely, this is true when CHLEN = 0 and DATLEN = 0b00 or when CHLEN = 1 and DATLEN = 0b10.

See *Control of the WS inversion* for information concerning the handling of the WS signal.

Figure 485. I2S Philips standard waveforms



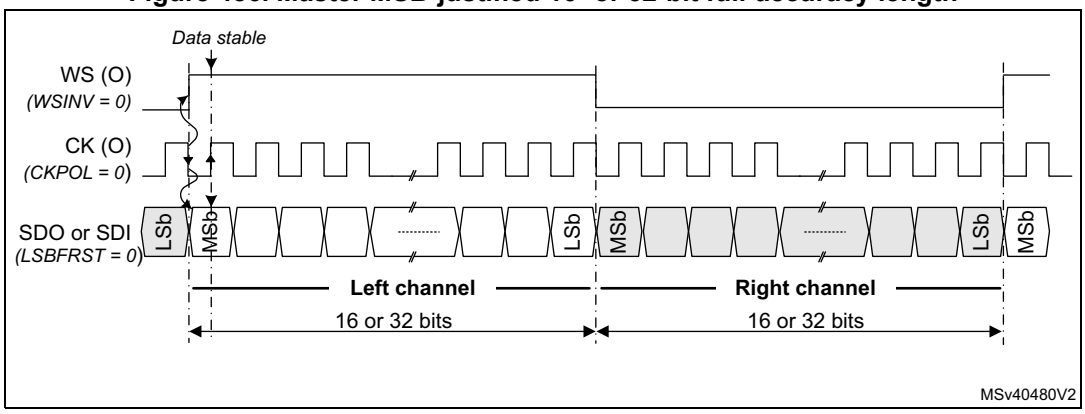
1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 38.3: SPI implementation* for the supported data sizes.

In the case where the channel length is bigger than the data length, the remaining bits are not significant when the SPI/I2S is configured in transmit mode. This is applicable for both master and slave mode.

MSB-justified standard

For this standard, the WS signal toggles when the first data bit is provided. The data transferred represents the left channel if WS is high and the right channel if WS is low.

Figure 486. Master MSB-justified 16- or 32-bit full-accuracy length

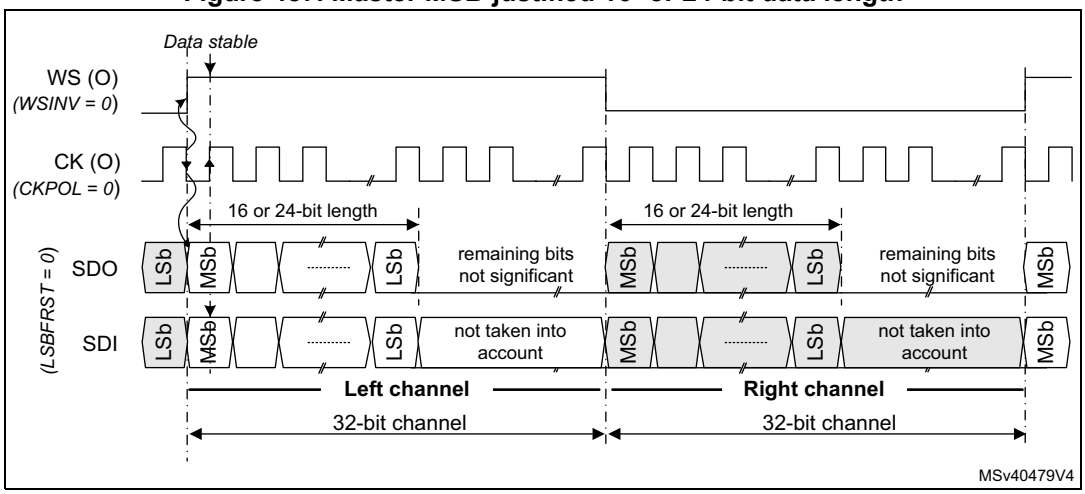


1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 38.3: SPI implementation* to check the supported data size.

CKPOL is cleared to match the I2S MSB-justified protocol. See *Selection of the CK sampling edge* for information concerning the handling of the WS signal.

See *Control of the WS inversion* for information concerning the handling of the WS signal.

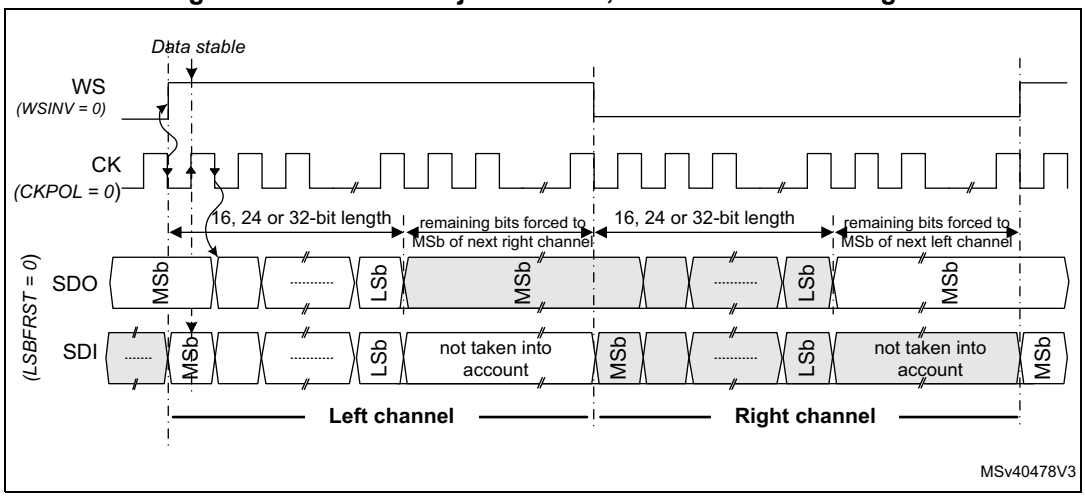
Figure 487. Master MSB-justified 16- or 24-bit data length



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 38.3: SPI implementation* to check the supported data size.

In the case where the channel length is bigger than the data length, the remaining bits are not significant when the SPI/I2S is configured in master transmit mode. In slave transmit mode, the remaining bits are forced to the value of the first bit of the next data to be generated to avoid timing issues (see *Figure 488*).

Figure 488. Slave MSB-justified 16-, 24- or 32-bit data length



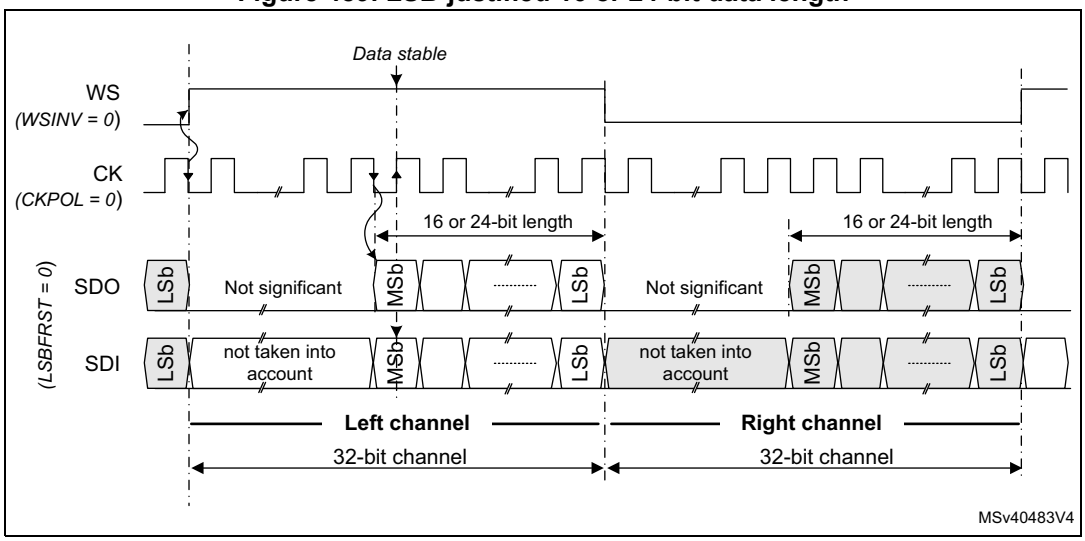
1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 38.3: SPI implementation* to check the supported data size.

LSB-justified standard

This standard is similar to the MSB-justified standard in master mode (no difference for the 16- and 32-bit full-accuracy frame formats). The LSB-justified 16- or 32-bit full-accuracy format gives similar waveforms to MSB-justified mode (see *Figure 486*) because the channel and data have the same length.

Note: In the LSB-justified format, only 16- and 32-bit channel lengths are supported in master and slave mode. This is because it is not possible to transfer properly the data if the channel length is not known by the transmitter and receiver side.

Figure 489. LSB-justified 16 or 24-bit data length



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 38.3: SPI implementation* to check the supported data size.

CKPOL is cleared in order to match the I2S LSB-justified protocol. See *Selection of the CK sampling edge* for information concerning the handling of the WS signal.

See *Control of the WS inversion* for information concerning the handling of the WS signal.

PCM standard

For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and can be selected using the PCMSYNC bit of SPI_I2SCFGR register.

In PCM long frame:

- The assertion time of the WS signal is fixed to 13 cycles of CK in master mode,
- The first data bit is received or transmitted as soon as the WS signal is asserted.

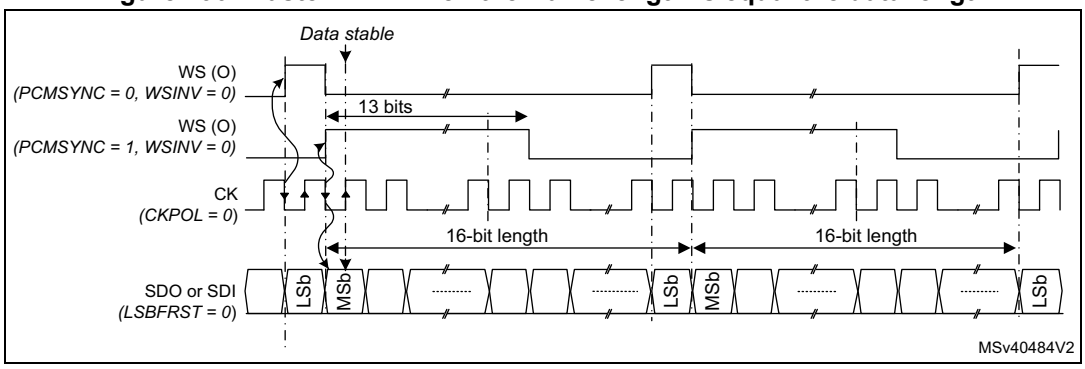
In PCM short frame:

- The assertion time of the WS signal is fixed to one cycle of CK in master mode,
- The first data bit is received or transmitted one cycle of CK after the WS assertion.

For both PCM modes:

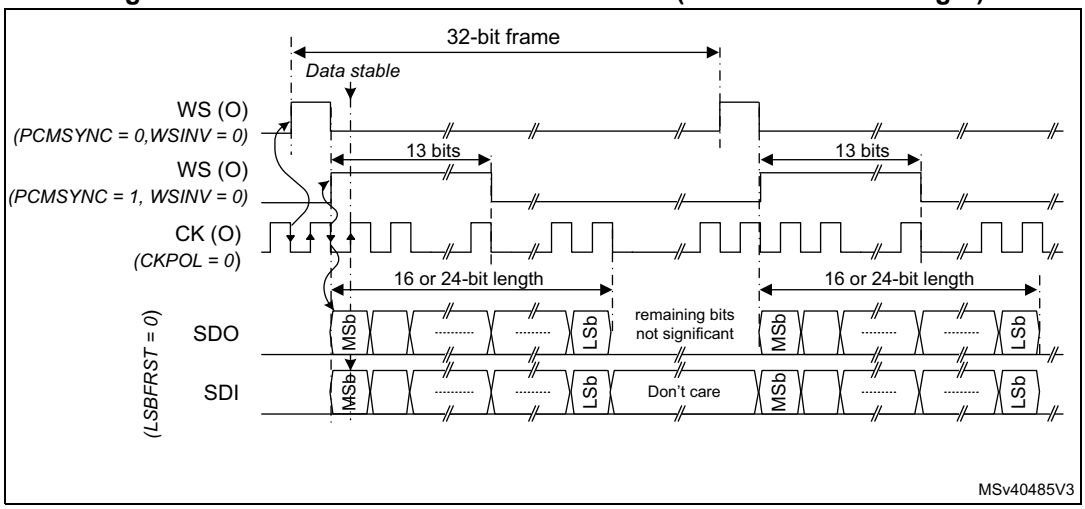
- The first data bit is MSb or LSb depending on the LSBFIRST bit value.
- The CK sampling edge can be selected thanks to CKPOL bit.
- The WS signal can be inverted thanks to the WSINV bit. See *Control of the WS inversion* for information concerning the handling of WS signal.

Figure 490. Master PCM when the frame length is equal the data length



A data size of 16 or 24 bits can be used when the channel length is set to 32 bits.

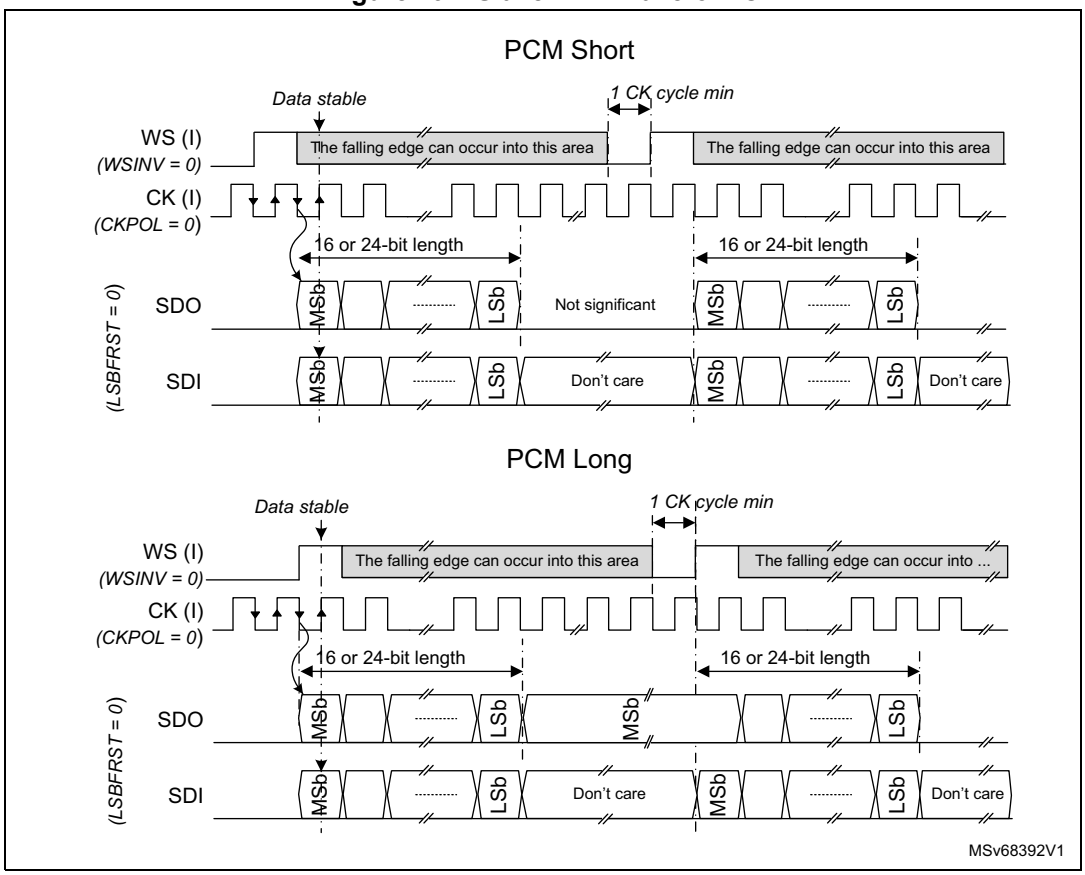
Figure 491. Master PCM standard waveforms (16 or 24-bit data length)



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 38.3: SPI implementation* to check the supported data size.

If the PCM protocol is used in slave mode, frame lengths can be different from 16 or 32 bits. As shown in *Figure 492*, in slave mode various pulse widths of WS can be accepted as the start of frame is detected by a rising edge of WS. The only constraint is that the WS must go back to its inactive state for at least one CK cycle.

Figure 492. Slave PCM waveforms



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 38.3: SPI implementation* to check the supported data size.

Note: In the case where the channel length is bigger than the data length, in slave PCM long, the transmission of the remaining bits is forced to the value of the first bit of the next data to be generated if the TXFIFO contains the next data to be transmitted.

In slave mode, CHLEN must be always programmed properly (whatever FIXCH value). For example, if CHLEN is cleared (16-bit length), the data transfer is truncated to 16 bits even if DATLEN is different from 0. To avoid this situation, CHLEN must be set, if DATLEN = 1 or 2.

38.9.6 Additional serial interface flexibility

Variable frame length in slave

In slave mode, channel lengths different from 16 or 32 bits can be accepted, as long as the channel length is bigger than the data length. This is true for all protocols except for the I2S LSB-justified protocol.

Data ordering

For all data formats and communication standards, it is possible to select the data ordering (MSb or LSb first) thanks to the bit LSBFRST located into *SPI/I2S configuration register 2 (SPI_CFG2)*.

Selection of the CK sampling edge

The CKPOL bit located into *SPI/I2S configuration register (SPI_I2SCFGR)* allows the user to choose the sampling edge polarity of the CK for slave and master modes, for all protocols.

- When CKPOL = 0, serial data SDO and WS (when master) are changed on the falling edge of CK and the serial data SDI and WS (when slave) are read on the rising edge.
- When CKPOL = 1, serial data SDO and WS (when master) are changed on the rising edge of CK and the serial data SDI and WS (when slave) are read on the falling edge.

Control of the WS inversion

It is possible to invert the default WS signal polarity for master and slave modes, for all protocols, by setting WSINV. By default the WS polarity is the following:

- In I2S Philips standard, WS is LOW for left channel, and HIGH for right channel
- In MSB/LSB-justified mode, WS is HIGH for left channel, and LOW for right channel
- In PCM mode, the start of frame is indicated by a rising edge of WS.

When WSINV is set, the WS polarity is inverted, then:

- In I2S Philips standard, WS is HIGH for left channel, and LOW for right channel
- In MSB/LSB-justified mode, WS is LOW for left channel, and HIGH for right channel
- In PCM mode, the start of frame is indicated by a falling edge of WS.

WSINV is located in the *SPI/I2S configuration register (SPI_I2SCFGR)*.

Control of the I/Os

The SPI/I2S block allows the settling of the WS and CK signals to their inactive state before enabling the SPI/I2S thanks to the AFCNTR bit of *SPI/I2S configuration register 2 (SPI_CFG2)*.

This can be done by programming CKPOL and WSINV using the following sequence:

Assuming that AFCNTR is initially cleared:

- Set I2SMOD = 1, (to inform the hardware that the CK and WS polarity is controlled via CKPOL and WSINV).
- Set bits CKPOL and WSINV to the wanted value.
- Set AFCNTR = 1.
  Then the inactive level of CK and WS I/Os is set according to CKPOL and WSINV values, even if the SPI/I2S is not yet enabled.
- Then performs the activation sequence of the I2S/PCM

Table 314 shows the level of WS and CK signals, when the AFCNTR bit is set, and before the SPI/I2S block is enabled (that is inactive level). Note that the level of WS also depends on the protocol selected.

Table 314. WS and CK level before SPI/I2S is enabled when AFCNTR = 1

WSINV	I2SSTD		WS level before SPI/I2S is enabled
0	I2S Std (00)	→	High
	Others	→	Low
1	I2S Std (00)	→	Low
	Others	→	High

CKPOL		CK level before SPI/I2S is enabled
0	→	Low
1	→	High

Note: The bit AFCNTR must not be set when the SPI is in slave mode.

Serial data line swapping

The direction of SDI and SDO depends on the IOSWP bit of *SPI/I2S configuration register 2 (SPI_CFG2)*, and on the slave/master mode. *Table 315* gives details on this feature.

Table 315. Serial data line swapping

Direction	IOSWP	Master mode		Slave mode	
		Input line	Output line	Input line	Output line
RX	0	SDI	-	SDO	-
	1	SDO	-	SDI	-
TX	0	-	SDO	-	SDI
	1	-	SDI	-	SDO
Full-duplex	0	SDI	SDO	SDO	SDI
	1	SDO	SDI	SDI	SDO

38.9.7 Startup sequence

When the bit SPE is cleared, the user is not allowed to read and write from/to the SPI_RXDR and SPI_TXDR registers, but the access to other registers is allowed.

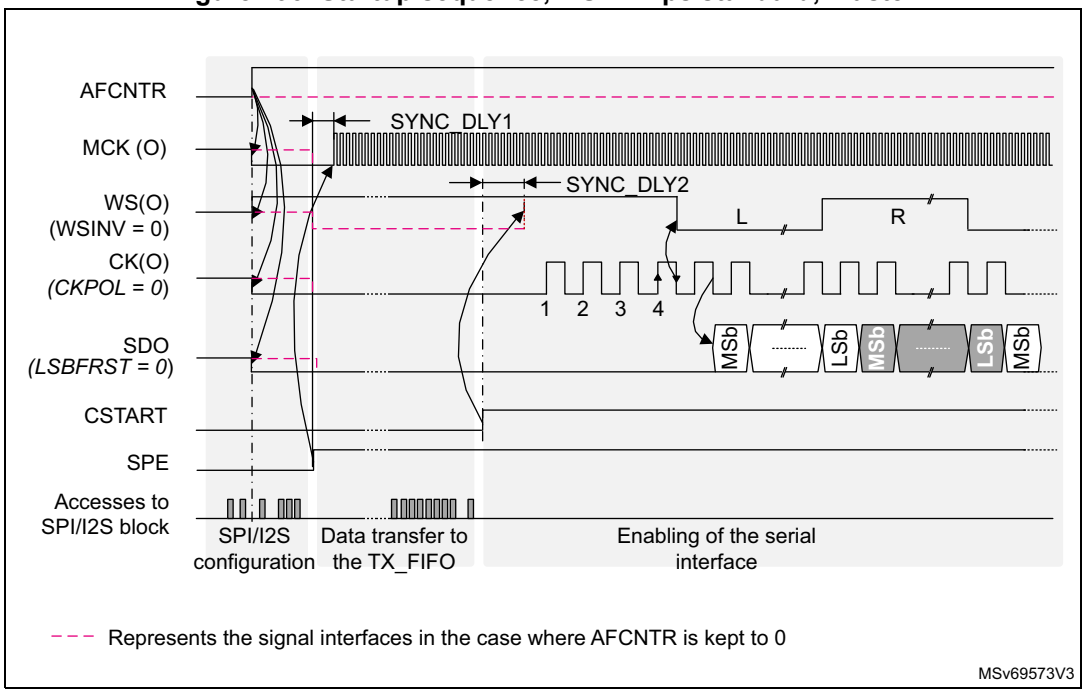
When the application wants to use the SPI/I2S block the user has to proceed as follows:

1. Ensure that the SPE is cleared, otherwise write SPE to 0.
2. Program all the configuration and control registers according to the wanted configuration. Refer to *Section 38.9.16* for detailed programming examples.
3. Clear all the status flags by setting the USPC, TIFREC, OVRC, and UDRC bits of the SPI_IFCR register. Note that if the flag SUSP is not cleared (via the SUSPC bit) the CSTART control bit has no effect.
4. Set the SPE bit to activate the SPI/I2S block. When this bit is set, the serial interface is still disabled, but the DMA and interrupt services are working, allowing for example, the data transfer into the TxFIFO. The generation of MCK can also be started when SPE goes to 1.
5. Set bit CSTART to activate the serial interface.

As shown in *Figure 493*, in I2S Philips standard master TX, the generation of the WS and CK signals starts after a resynchronization delay (SYNC_DLY2) when CSTART goes to 1

and the TxFIFO is not empty. Note that the CK bit clock is activated four rising edges before the falling edge of WS in order to ensure that the external slave device can detect properly WS transition. Other standards behave similarly.

Figure 493. Startup sequence, I2S Philips standard, master



1. As shown in the figure, MCK can be enabled as soon as the bit SPE is set. It is generated after a synchronization delay (SYNC_DLY1). See MCK generation in *Section 38.9.9: Clock generator* for more information.
2. Note that the level of WS and CK signals are controlled by the SPI/I2S block during the configuration phase as soon as the AFCNTR bit is set.

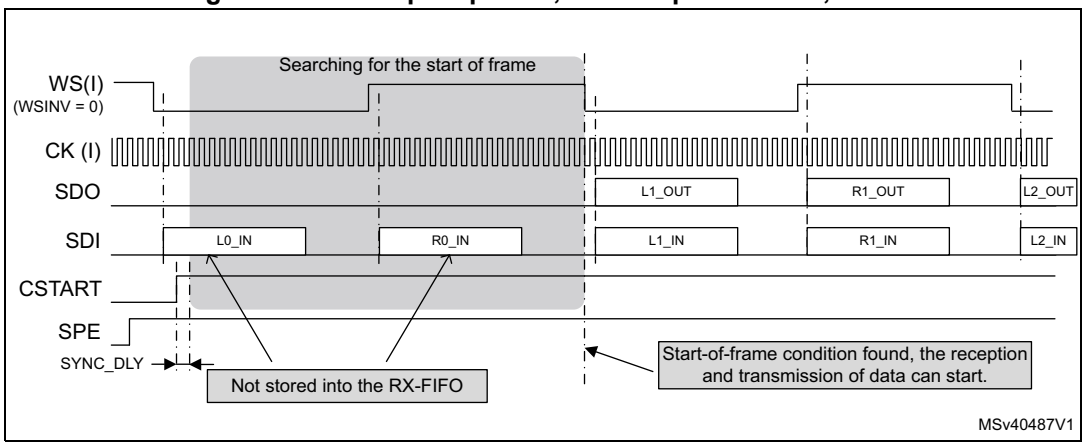
Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after about three periods of CK clock (SYNC_DLY2).

In slave mode, once the bit CSTART is set, the data transfer starts when the start-of-frame condition is met:

- For the I2S Philips standard, the start-of-frame condition is a falling edge of the WS signal. The transmission/reception starts one bit clock later.
  If WSINV = 1, then the start-of-frame condition is a rising edge.
- For other protocols, the start-of-frame condition is a rising edge of the WS signal. The transmission/reception starts at the rising edge of WS for MSB-aligned protocol. The transmission/reception starts one bit clock later for PCM protocol.
  If WSINV = 1, then the start-of-frame condition is a falling edge.

Figure 494 shows an example of startup sequence in I2S Philips standard, slave mode.

Figure 494. Startup sequence, I2S Philips standard, slave



Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after two periods of CK clock (SYNC_DLY).

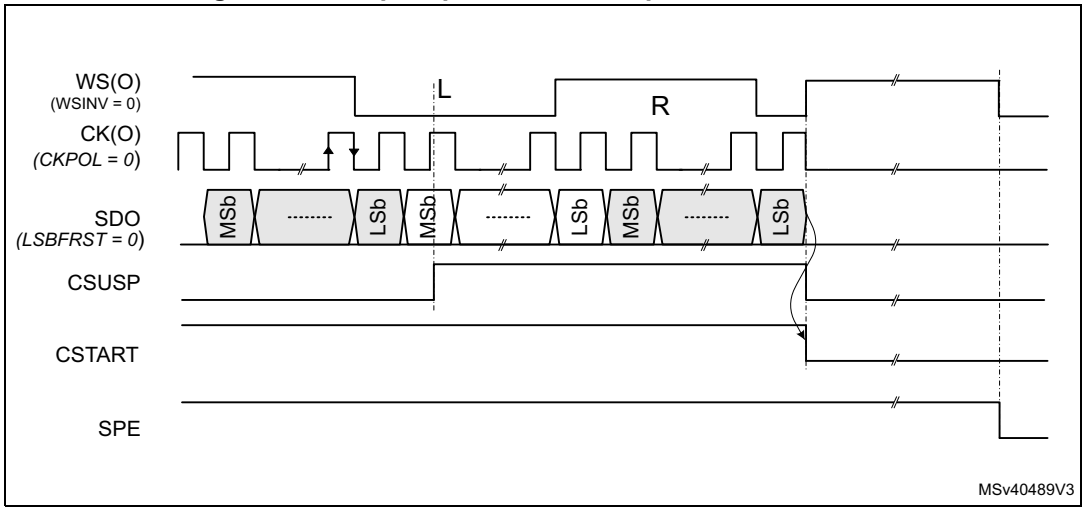
38.9.8 Stop sequence

The application can stop the I2S/PCM transfers by clearing the SPE bit. In that case the communication is stopped immediately, without waiting for the end of the current frame.

In master mode it is also possible to stop the I2S/PCM transfers at the end of the current frame. For that purpose, the user has to set the bit CSUSP, and polls the CSTART bit until it goes to 0. The CSTART bit goes to 0 when the current stereo (if an I2S mode was selected) or mono sample are completely shifted in or out. Then the SPE bit can be cleared.

Figure 495 shows an example of stop sequence in the case of master mode. The CSUSP bit is set during the transmission of the left sample, the transfer continue until the last bit of the right sample is transferred. Then CSTART and CSUSP go back to 0, CK and WS signals go back to their inactive state, and the user can clear the SPE bit.

Figure 495. Stop sequence, I2S Philips standard, master

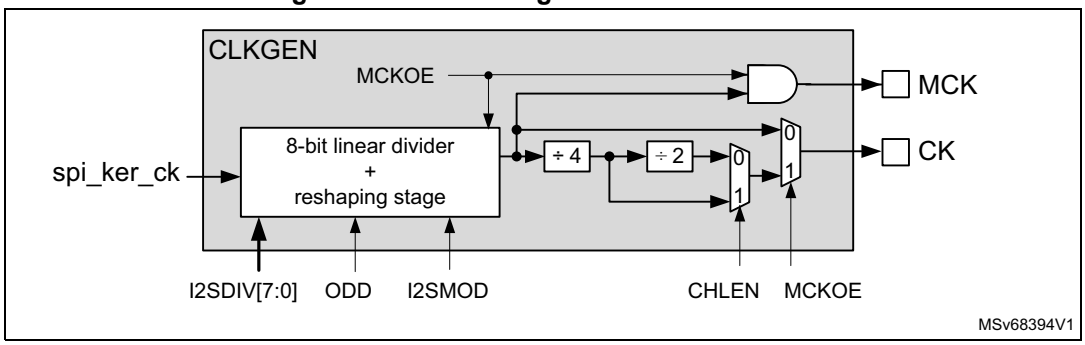


Note: In slave mode, the stop sequence is only controlled by the SPE bit.

38.9.9 Clock generator

When the I2S or PCM is configured in master mode, the user needs to program the clock generator in order to produce the frame synchronization (WS), the bit clock (CK) and the master clock (MCK) at the desired frequency.
If the I2S or PCM is used in slave mode, there is no need to configure the clock generator.

Figure 496. I²S clock generator architecture



The frequency generated on MCK, CK and WS depends mainly on I2SDIV, ODD, CHLEN, and MCKOE. The bit MCKOE indicates if a master clock need to be generated or not. The master clock has a frequency 256 or 128 times higher than the frame synchronization. This master clock is often required to provide a reference clock to external audio codecs.

Note: In master mode, there are no specific constraints on the ratio between the bus clock rate (F_{pclk}) and the bit clock (F_{CK}). The bus clock frequency must be high enough in order to support the data throughput.

When the master clock is generated (MCKOE = 1), the frequency of the frame synchronization is given by the following formula in I2S mode:

$$F_{WS} = \frac{F_{i2s\_clk}}{256 \times \{(2 \times I2SDIV) + ODD\}}$$

And by this formula in PCM mode:

$$F_{WS} = \frac{F_{i2s\_clk}}{128 \times \{(2 \times I2SDIV) + ODD\}}$$

In addition, the frequency of the MCK (F_{MCK}) is given by the formula:

$$F_{MCK} = \frac{F_{i2s\_clk}}{\{(2 \times I2SDIV) + ODD\}}$$

When the master clock is disabled (MCKOE = 0), the frequency of the frame synchronization is given by the following formula in I2S mode:

$$F_{WS} = \frac{F_{i2s\_clk}}{32 \times (CHLEN + 1) \times \{(2 \times I2SDIV) + ODD\}}$$

And by this formula in PCM mode:

$$F_{WS} = \frac{F_{i2s\_clk}}{16 \times (CHLEN + 1) \times \{(2 \times I2SDIV) + ODD\}}$$

Where F_{WS} is the frequency of the frame synchronization, and F_{i2s_clk} is the frequency of the kernel clock provided to the SPI/I2S block.

Note: CHLEN and ODD can be either 0 or 1.

I2SDIV can take any values from 0 to 255 when ODD = 0, but when ODD = 1, the value I2SDIV = 1 is not allowed.

When I2SDIV = 0, then {(2 x I2SDIV) + ODD} is forced to 1.

Note: When {(2 x I2SDIV) + ODD} is odd, the duty cycle of the MCK or the CK signal is not 50%. Care must be taken when an odd ratio is used: it can impact margin on setup and hold time. For example if {(2 x I2SDIV) + ODD} = 5, then the duty cycle can be 40%.

Table 316 provides examples of clock generator programming for I2S modes.

MCK generation

The master clock MCK is generated when the following conditions are met:

- I2SMOD must be equal to 1,
- I2SCFG must select a master mode,
- MCKOE must be set,
- SPE must be set

Table 316. CLKGEN programming examples for usual I2S frequencies

i2s_clk (MHz)	Channel length (bits)	I2SDIV	ODD	MCK	Sampling rate: F_{WS} (kHz)
12.288	16	12	0	No	16
12.288	32	6	0		16
12.288	16	6	0		32
12.288	32	3	0		32
49.152	16	16	0		48
49.152	32	8	0		48
49.152	16	8	0		96
49.152	32	4	0		96
49.152	16	4	0		192
49.152	32	2	0		192

Table 316. CLKGEN programming examples for usual I2S frequencies (continued)

i2s_clk (MHz)	Channel length (bits)	I2SDIV	ODD	MCK	Sampling rate: F_{WS} (kHz)
4.096	16 or 32	0	-	Yes	16
24.576	16 or 32	3	0		
49.152	16 or 32	3	0		32
12.288	16 or 32	0	-		48
49.152	16 or 32	2	0		
61.44	16 or 32	2	1		
98.304	16 or 32	2	0		96
196.608	16 or 32	2	0		192

38.9.10 Internal FIFOs

The I2S interface can use a dedicated FIFO for the RX and the TX path. The samples to transmit can be written into the TxFIFO via the SPI_TXDR register. The reading of RxFIFO is performed via the SPI_RXDR register.

Data alignment and ordering

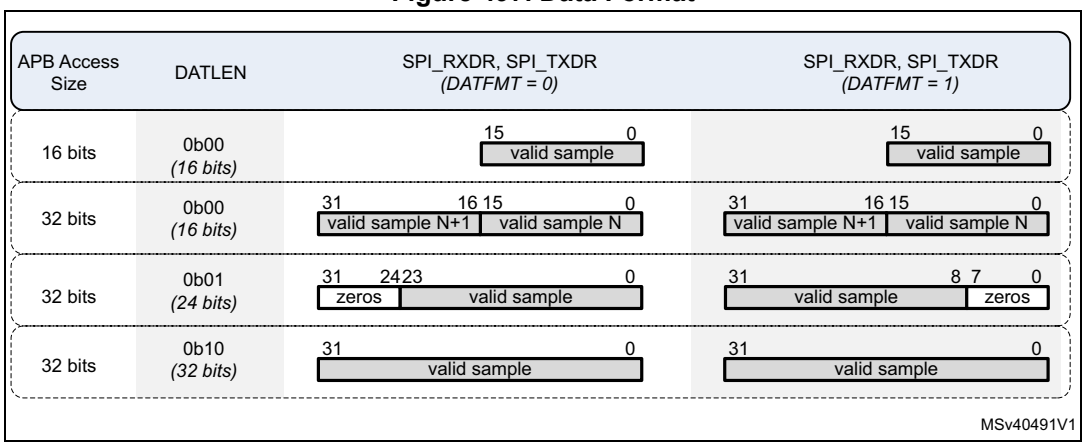
It is possible to select the data alignment into the SPI_RXDR and SPI_TXDR registers thanks to the DATFMT bit.

Note as well that the format of the data located into the SPI_RXDR or SPI_TXDR depends as well on the way those registers are accessed via the APB bus.

Figure 497 shows the allowed settings between APB access sizes, DATFMT, and DATLEN.

Note: Caution must be taken when the APB access size is 32 bits, and DATLEN = 0. For read operation the RxFIFO must contain at least two data, otherwise the read data are invalid. In the same way, for write operation, the TxFIFO must have at least two empty locations, otherwise a data can be lost.

Figure 497. Data Format



1. In I2S mode, the sample N represents the left sample, and the sample N+1 is the right sample.
2. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 38.3: SPI implementation* to check the supported data size.

It is possible to generate an interrupt or a DMA request according to a programmable FIFO threshold levels. The FIFO threshold is common to RX and TxFIFOs can be adjusted via FTHLV.

In I2S mode, the left and right audio samples are interleaved into the FIFOs. It means that for transmit operations, the user has to start to fill-up the TxFIFO with a left sample, followed by a right sample, and so on. For receive mode, the first data read from the RxFIFO is supposed to represent a left channel, the next one is a right channel, and so on.

Note that the read and write pointers of the FIFOs are reset when the bit SPE is cleared.

Refer to *Section 38.9.11* and *Section 38.9.15* for additional information.

FIFO size optimization

The basic element of the FIFO is the byte. This allows an optimization of the FIFO locations. For example when the data size is fixed to 24 bits, each audio sample takes three basic FIFO elements.

For example, a FIFO with 16 basic elements can have a depth of:

- 8 samples, if the DATLEN = 0 (16 bits),
- 5 samples, if the DATLEN = 1 (24 bits)[a],
- 4 samples, if the DATLEN = 2 (32 bits)[a].

38.9.11 FIFO status flags

Two status flags are provided for the application to fully monitor the state of the I2S interface. Both flags can generate an interrupt request. The receive interrupt is generated if RXPIE bit is enabled, the transmit interrupt is generated if TXPIE bit is enabled. Those bits are located into the SPI_IER register.

TxFIFO threshold reached (TXP)

When set, this flag indicates that the TxFIFO contains at least FTHLV empty locations. Thus, FTHLV new data to be transmitted can be written to SPI_TXDR. The TXP flag is reset when the number of empty locations is lower than FTHLV. Note that TXP = 1, when the I2S is disabled (SPE bit is reset).

RxFIFO threshold reached (RXP)

When set, this flag indicates that there is at least FTHLV valid data into the RxFIFO, thus the user can read those data via SPI_RXDR. It is reset when the RxFIFO contains less than FTHLV data.

See *Section 38.10* for additional information on the interrupt function in I2S mode.

38.9.12 Handling of underrun situation

In transmit mode, an underrun situation is detected when a new data needs to be loaded into the shift register while the TxFIFO is already empty. In such a situation the UDR flag is set, and at least one audio frame contains unexpected data.

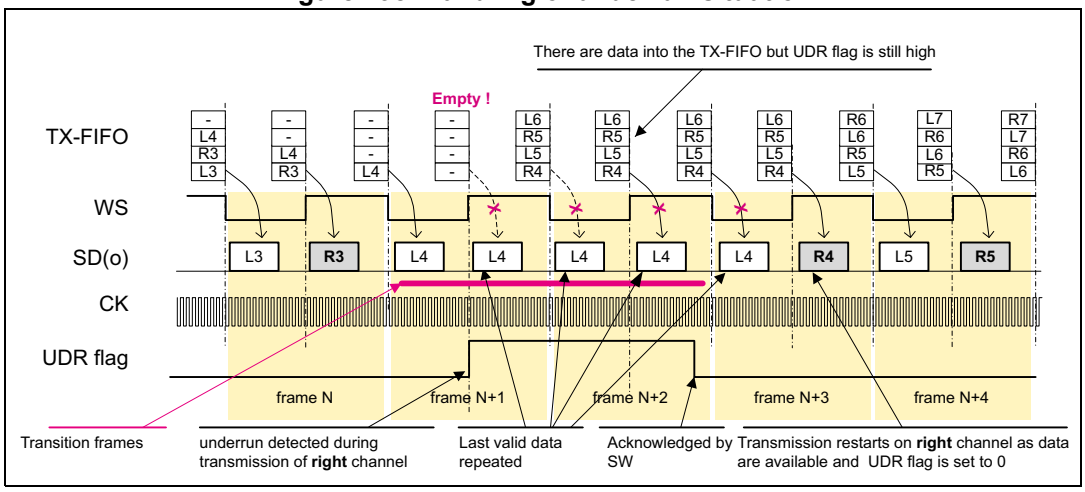
a. Not always available, refer to *Section 38.3: SPI implementation* in order to check if 24 and 32-bit data widths are supported.

For I2S modes, there is a hardware mechanism to prevent misalignment situations (left and right channel swapped). When an underrun situation is detected, the last valid data present in the shift register is repeated if the real channel length matches the length selected by the CHLEN bit. In the other cases, an undefined data is repeated. Typically, if the block is programmed in slave TX mode, and the external master audio device is using channel lengths other than 16 or 32 bits, the repeated data value is undefined in case of underrun.

The following figure shows the case where an underrun occurs and the peripheral replays the last valid data on left and right channels as long as conditions of restart are not met. The transmission restarts:

- When there is enough data into the TxFIFO, and
- When the UDR flag is cleared by the software, and
- When the restart condition is met:
  - If the underrun occurs when a right channel data needs to be transmitted, the transmission restarts when a right channel needs to be transmitted, or
  - If the underrun occurs when a left channel data needs to be transmitted, the transmission restarts when a left channel needs to be transmitted.

Figure 498. Handling of underrun situation



When the block is configured in one of the PCM modes, the transmission restarts at the start of the next frame, when there is enough data in the TxFIFO, and the UDR flag is cleared.

The UDR flag can trigger an interrupt if the UDRIE bit of the SPI_IER register is set. The UDR bit is cleared by setting the UDRC bit of the SPI_IFCR register.

Note: An underrun situation can occur in master or slave mode. In master mode, when an underrun occurs, the WS, CK, and MCK signals are not gated.

Due to resynchronization, any change of the UDR flag is taken into account by the hardware after at least two periods of CK clock.

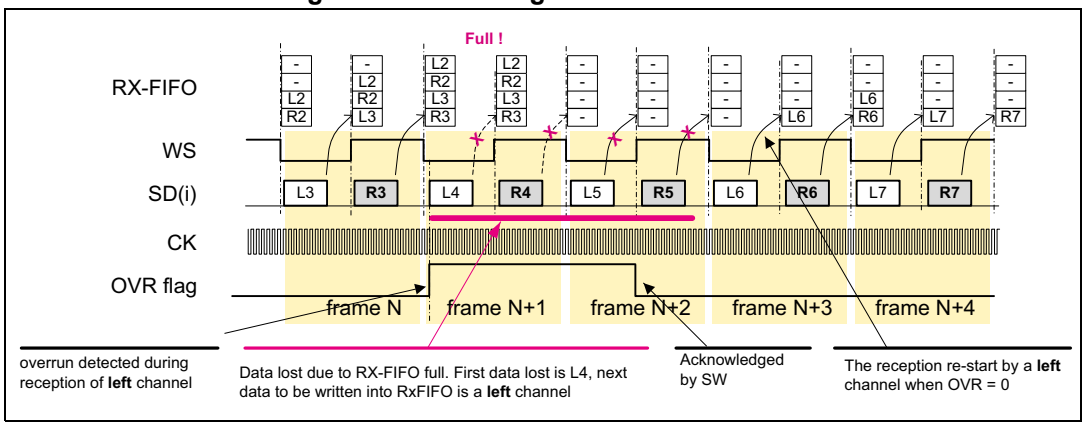
38.9.13 Handling of overrun situation

In receive mode, an overrun situation is detected when the shift register needs to store a new data into the RxFIFO, while the RxFIFO is full. In such a situation the OVR flag is set, and the incoming data is lost.

In I2S mode, there is a hardware mechanism in to prevent misalignment situations (left and right channel swapped). As shown in the following figure, when an overrun occurs, the peripheral stops writing data into the RxFIFO as long as restart conditions are not met.

The reception restarts when there is enough room into the RxFIFO, and the OVR flag is cleared. The block starts by writing next the right channel into the RxFIFO if the overrun occurs when a right channel data is received or by writing the next left channel if the overrun occurs when a left channel data is received.

Figure 499. Handling of overrun situation



When the block is configured in PCM mode, after an overrun error, the block stops writing data in the RxFIFO as long as conditions of restart are not met. When there is enough room in the RxFIFO, and the OVR flag is cleared, the next received data are written in the RxFIFO.

An interrupt can be generated if the OVRIE bit of the SPI_IER register is set. The OVR bit is cleared by setting the OVRC bit of the SPI_IFCR register.

Note: An overrun situation can occur in master or slave mode. In master mode when an overrun occurs, the WS, CK, and MCK signals are not gated.

38.9.14 Frame error detection

When configured in slave mode, the SPI/I2S block detects two kinds of frame errors:

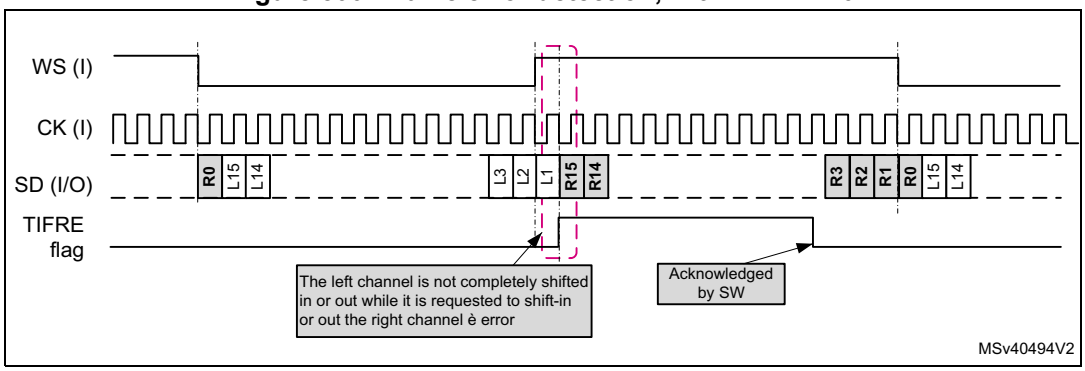
- A frame synchronization is received while the shift-in or shift-out of the previous data is not complete (early frame error). This mode is selected with FIXCH = 0.
- A frame synchronization occurs at an unexpected position. This mode is selected with FIXCH = 1.

In slave mode, if the frame length provided by the external master device is different from 32 or 64 bits, the user has to clear FIXCH. As the SPI/I2S synchronize each transfer with the WS there is no misalignment risk, but in a noisy environment, if a glitch occurs in the CK signal, a sample may be affected and the application is not aware of this.

If the frame length provided by the external master device is equal to 32 or 64 bits, then the user can set FIXCH and adjust accordingly CHLEN. As the SPI/I2S synchronize each transfer with the WS there is still no misalignment risk, and if the amount of bit clock between each channel boundary is different from CHLEN, the frame error flag (TIFRE) is set.

Figure 500 shows an example of frame error detection. The SPI/I2S block is in slave mode and the number of bit clock periods for the left channel are not enough to shift-in or shift-out the data. The figure shows that the ongoing transfer is interrupted and the next one is started in order to remain aligned to the WS signal.

Figure 500. Frame error detection, with FIXCH = 0

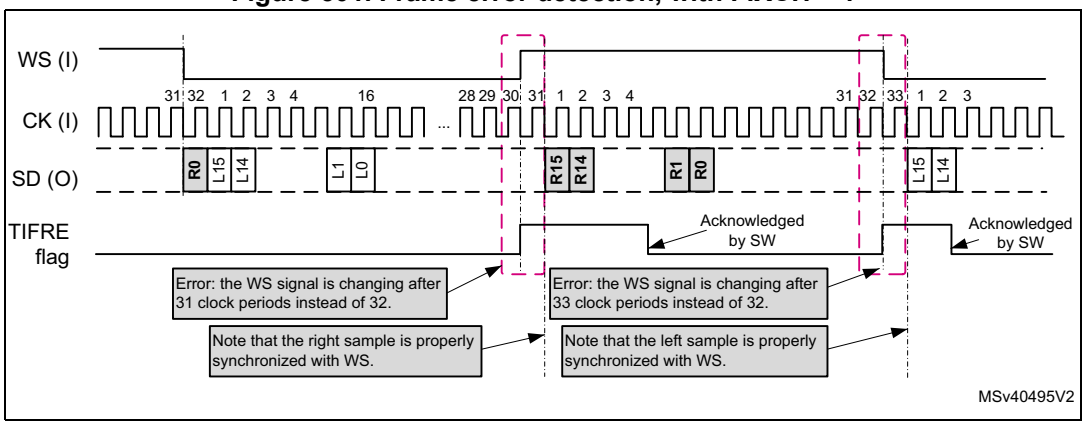


An interrupt can be generated if the TIFREIE bit is set. The frame error flag (TIFRE) is cleared by writing the TIFREC bit of the SPI_IFCR register to 1.

It is possible to extend the coverage of the frame error flag by setting the bit FIXCH. When this bit is set, then the SPI/I2S is expecting fixed channel lengths in slave mode. This means that the expected channel length can be 16 or 32 bits, according to CHLEN. As shown in *Figure 501*, in this mode, the SPI/I2S block is able to detect if the WS signal is changing at the expected moment (too early or too late).

Note: *Figure 500* and *Figure 501* show the mechanism for the slave transmit mode, but this is also true for slave receive and slave full-duplex.

Figure 501. Frame error detection, with FIXCH = 1



The frame error detection can be generally due to noisy environment disturbing the good reception of WS or CK signals.

Note: *The SPI/I2S is not able to recover properly if an overrun and an early frame occur within the same frame. In this case the user has to disable and re-enable the SPI/I2S.*

38.9.15 DMA interface

The I2S/PCM mode shares the same DMA requests lines than the SPI function. There is a separated DMA channel for TX and RX paths. Each DMA channel can be enabled via RXDMAEN and TXDMAEN bits of the SPI_CFG1 register.

In receive mode, the DMA interface is working as follows:

1. The hardware evaluates the RxFIFO level,
2. If the RxFIFO contains at least FTHLV samples, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the RxFIFO contains less than FTHLV samples, no DMA request is generated, and the hardware loop to step 1

In transmit mode, the DMA interface is working as follows:

1. The hardware evaluates the TxFIFO level,
2. If the TxFIFO contains at least FTHLV empty locations, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the TxFIFO contains less than FTHLV empty locations, no DMA request is generated, and the hardware loop to step 1

38.9.16 Programing examples

Master I2S Philips standard, full-duplex

This example shows how to program the interface for supporting the I2S Philips standard protocol in master full-duplex mode, with a sampling rate of 48 kHz, using the master clock. The assumption has been taken that the SPI/I2S is receiving a kernel clock (i2s_clk) of 61.44 MHz from the clock controller of the circuit. In the example above we took the assumption that the external audio codec needs to be programmed, for example via an I2C interface before starting the transfer. In addition, it is supposed that this external audio codec needs the MCK to accept I2C commands.

Procedure

1. Via the RCC block, enable the bus interface and the kernel clocks, assert and release the reset signal if needed.
2. Program the AFMUX in order to select the wanted I/Os. In the current example MCK, CK, WS, SDO, SDI are needed.
3. Program the clock generator to provide the MCK clock, and to have a frame synchronization rate at exactly 48 kHz. Set I2SDIV to 2, ODD to 1, and MCKOE to 1.
4. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, CHLEN = 1 (32 bits per channel) DATLEN = 1 (24 bits), I2SSTD = 0 (I2S Philips standard), I2SCFG = 5 (master full-duplex), I2SMOD = 1, for I2S/PCM mode.
5. Adjust the FIFO threshold by setting the wanted value into FTHLV. For example, if a threshold of two audio samples is required, FTHLV = 1.
6. If the application wishes to perform data transfer via DMA, set the bits TXDMAEN and RXDMAEN.
7. Clear all interrupt enable fields located in the SPI_IER register.
8. Clear all status flags, by setting SUSPC, TIFREC, OVRC, UDRC of the SPI_IFCR register.
9. Set the SPE bit, then the MCK is generated. In the example presented here, the TxFIFO is not filled-up as soon as SPE = 1. The application can then configure the external audio codec via an I2C interface even if audio samples to transmit are not yet available. An *Alternative sequence* is proposed hereafter.
10. When the application wants to start the data transfer:
    - If the data transfer uses DMA:
    a) Program the DMA peripheral: two channels, one for RX and one for TX.
    b) Initialize the memory buffer with valid audio samples for TX path.
    c) Enable the DMA channels.
    d) Enable interrupt events such as UDRIE and OVRIE if needed in order to detect transfer errors.
    e) Enable the DMA channel. If TXDMAEN was set, the TxFIFO is filled up.
    - If the data transfer is done via interrupt:

      Enable the interrupt events UDRIE, OVRIE, TXPIE, and RXPIE (by writing 1). An interrupt request is immediately activated allowing the interrupt handler to fill-up the TxFIFO.
11. Finally, the SPI/I2S serial interface can be enabled by setting the bit CSTART. CSTART bit is located into the SPI_CR1 register.

Alternative sequence

- Steps 1 to 8 are similar to the previous sequence.
- If the data transfer uses DMA:
  - Program the DMA peripheral: two channels, one for RX and one for TX
  - Initialize the memory buffer with valid audio samples for the TX path
  - Enable interrupt events such as UDRIE and OVRIE if needed in order to detect transfer errors.
  - Enable the DMA channels,

- If the data transfer is done via interrupt:
  - Enable the interrupt events UDRIE, OVRIE, TXPIE, and RXPIE (by writing 1). An interrupt request is immediately activated allowing the interrupt handler to fill-up the TxFIFO.
- Set SPE, as soon as this bit is set to one the following actions may happen:
  - If the interrupt generation is enabled, the SPI/I2S generates an interrupt request allowing the interrupt handler to fill-up the TxFIFO.
  - If the DMA transfer is enabled, the SPI/I2S generates DMA requests in order to fill-up the TxFIFO
  - The MCK is generated. The application can then configure the external audio codec via an I2C interface.
- Finally, the SPI/I2S serial interface can be enabled by setting the bit CSTART. CSTART bit is located into the SPI_CR1 register.

Stop procedure in master mode

1. Set the bit CSUSP, in order to stop ongoing transfers
2. Check the value of the CSTART bit until it goes to 0
3. Clear the SUSP flag by setting SUSPC
4. Stop DMA peripheral, bus clock...
5. Clear bit SPE in order to disable the SPI/I2S block

Slave I2S Philips standard, receive

This example shows how to program the interface for supporting the I2S Philips standard protocol in slave receiver mode, with a sampling rate of 48 kHz. Note that in slave mode the SPI/I2S block cannot control the sample rate of the received samples. In this example we took the assumption that the external master device is delivering an I2S frame structure with a channel length of 24 bits. So, we cannot use the capability offered for frame error detection when FIXCH is set.

Procedure

1. Via the RCC block, enable the bus interface and the kernel clocks, assert and release the reset signal if needed,
2. Program the AFMUX in order to select the wanted I/Os. In the current example CK, WS, SDI,
3. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, FIXCH = 0 (because channel length is different from 16 and 32 bits), DATLEN = 0 (16 bits), I2SSTD = 0 (Philips protocol), I2SCFG = 1 (slave RX), I2SMOD = 1, for I2S mode.
4. Adjust the FIFO threshold by setting the wanted value into FTHLV. For example if a threshold of two audio samples is required, FTHLV = 1.
5. Clear all status flag registers.
6. Enable the flags that generate an interrupt such as OVRIE and TIFRE.
7. If the data transfer uses DMA:
   - Program the DMA peripheral: one RX channel
   - Enable the DMA channel,
   - In the SPI/I2S block, enable the DMA by setting the RXDMAEN bit.

8. If the data transfer is done via interrupt, then the user has to enable the interrupt by setting the RXPIE bit.
9. Set SPE.
10. Finally, the user can set the bit CSTART in order to enable the serial interface. The SPI/I2S starts to store data into the RxFIFO on the next occurrence of left data transmitted by the external master device.

Stop procedure in slave mode

1. Clear bit SPE in order to disable the SPI/I2S block
2. Stop DMA peripheral, bus clock...

38.10 I2S interrupts

In PCM/I2S mode an interrupt (**spi_it**) or a wake-up event signal (**spi_wkup**) can be generated according to the events described in the *Table 317*.

Interrupt events can be enabled and disabled separately.

Table 317. I2S interrupt requests

Interrupt vector	Interrupt event	Event flag	Event/Interrupt clearing method	Exit Sleep mode	Exit Stop modes	Exit Standby mode
SPI	TxFIFO threshold reached	TXP	When the TxFIFO contains less than FTHLV empty locations	Yes	Yes	No
	RxFIFO threshold reached	RXP	When the RxFIFO contains less than FTHLV samples			
	Overrun error	OVR	Write OVRC to 1			
	Underrun error	UDR	Write UDRC to 1			
	Frame error flag	TIFRE	Write TIFREC to 1		No	

38.11 SPI/I2S registers

38.11.1 SPI/I2S 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 (33-bit or 17-bit) CRC polynomial configuration
0: Full size (33-bit or 17-bit) CRC polynomial is not used
1: Full size (33-bit or 17-bit) CRC polynomial is used

Bit 12 **SSI:** internal SS 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 SS input internally and the I/O value of the SS 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 or I2S/PCM communication. In SPI mode, it is cleared by hardware when end of transfer (EOT) flag is set or when a transfer suspend request is accepted. In I2S/PCM mode, it is also cleared by hardware as described in the *Section 38.9.8: Stop sequence*.
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, 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 SS 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

38.11.2 SPI/I2S 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 at TSIZE. TSIZE cannot be set to 0xFFFF respective 0x3FF value when CRC is enabled.

Note: TSIZE[15:10] bits are reserved for limited-featured instances, and must be kept at reset value.

38.11.3 SPI/I2S 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.

00000: reserved
00001: reserved
00010: reserved
00011: 4-bits
00100: 5-bits
00101: 6-bits
00110: 7-bits
00111: 8-bits
.....
11101: 30-bits
11110: 31-bits
11111: 32-bits

Note: The most significant bit at CRCSIZE bit field is reserved at the peripheral instances where data size is limited to 16-bit.

The CRCSIZE[2:0] bits are fixed to 1 for the peripheral instances with a limited set of features.

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 482: 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 lastly 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 ≤ 8bit, better to select FTHLV = 4, 8, 12.

0000: 1-data
0001: 2-data
0010: 3-data
0011: 4-data
0100: 5-data
0101: 6-data
0110: 7-data
0111: 8-data
1000: 9-data
1001: 10-data
1010: 11-data
1011: 12-data
1100: 13-data
1101: 14-data
1110: 15-data
1111: 16-data

Note: FTHLV[3:2] bits are reserved for instances with a limited set of features

Bits 4:0 **DSIZE[4:0]:** number of bits in a single SPI data frame

00000: not used
00001: not used
00010: not used
00011: 4 bits
00100: 5 bits
00101: 6 bits
00110: 7 bits
00111: 8 bits
.....
11101: 30 bits
11110: 31 bits
11111: 32 bits

Note: The most significant bit DSIZE[4] is reserved for instances which data size is limited to 16 bits.

DSIZE[2:0] bits are fixed to 1 for instances with a limited set of features.

38.11.4 SPI/I2S 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

Note: This bit can be also used in PCM and I2S modes.

The bit AFCNTR must not be set, when the block is in slave mode.

Bit 30 **SSOM**: SS output management in master mode

This bit is taken into account in master mode when SSOE is enabled. It allows the SS output to be configured between two consecutive data transfers.

0: SS is kept at active level until data transfer is complete, it becomes inactive with EOT flag

1: SPI data frames are interleaved with SS nonactive pulses when MIDI[3:0] > 1

Bit 29 **SSOE:** SS output enable

This bit is taken into account in master mode only

0: SS output is disabled and the SPI can work in multimaster configuration

1: SS output is enabled. The SPI cannot work in a multimaster environment. It forces the SS 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**: SS input/output polarity

0: low level is active for SS signal

1: high level is active for SS signal

Bit 27 Reserved, must be kept at reset value.

Bit 26 **SSM:** software management of SS signal input

0: SS input value is determined by the SS PAD

1: SS input value is determined by the SSI bit

Note: When master uses hardware SS output (SSM = 0 and SSOE = 1) the SS 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
Note: This bit can be also used in PCM and I2S modes.

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
Note: This bit can be also used in PCM and I2S modes to swap SDO and SDI pins.

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-bit, 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 SS Idleness

Specifies an extra delay, expressed in number of SPI clock cycle periods, inserted additionally between active edge of SS 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.

38.11.5 SPI/I2S 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

38.11.6 SPI/I2S 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
CTSIZE[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
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 **CTSIZE[15:0]**: number of data frames remaining in current TSIZE session
The value is not quite reliable when traffic is ongoing on bus .
Note: CTSIZE[15:0] bits are not available in instances with limited set of features.

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 and data size is set up to 16-bit, the value gives number of remaining data frames persisting at RxFIFO.

When data size is greater than 16-bit, these bits are always read as 00. In that consequence, the single data frame received at the FIFO cannot be detected neither by RWNE nor by RXPLVL bits if data size is set from 17 to 24 bits. The user must then apply other methods to detect the number of data received, such as monitor the EOT event when TSIZE > 0 or RXP events when FTHLV = 0.

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-bit 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. In I2S/PCM mode, (when available) this flag applies to master and slave mode.

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 SS 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 respective DATLEN[1:0] in I2S/PCM 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 DATLEN[1:0] in I2S/PCM 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

38.11.7 SPI/I2S 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.

38.11.8 SPI/I2S 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: In SPI mode, data is always right-aligned. Alignment of data at I2S mode depends on DATLEN and DATFMT setting. 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.

38.11.9 SPI/I2S 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: In SPI mode, data is always right-aligned. Alignment of data at I2S mode depends on DATLEN and DATFMT setting. 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.

38.11.10 SPI/I2S 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 for instances with data size limited to 16-bit. 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.

38.11.11 SPI/I2S 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: This bitfield is not used in I2S mode.

Note: TXCRC[31-16] bits are reserved for instances with data size limited to 16-bit. 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.

38.11.12 SPI/I2S 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.

This bitfield is not used in I2S mode.

RXCRC[31-16] bits are reserved at the peripheral instances with data size limited to 16-bit. 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.

38.11.13 SPI/I2S 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 at the peripheral instances with data size limited to 16-bit. 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.

38.11.14 SPI/I2S configuration register (SPI_I2SCFGR)

Address offset: 0x050

Reset value: 0x0000 0000

This register must be configured when the I2S is disabled (SPE = 0). The content of this register is not taken into account in SPI mode except for the I2SMOD bit, which needs to be kept at 0.

This register is reserved for instances not supporting I2S mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV[7:0]							
						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
Res.	DATFMT	WSINV	FIXCH	CKPOL	CHLEN	DATLEN[1:0]		PCMSYNC	Res.	I2SSTD[1:0]		I2SCFG[2:0]			I2SMOD
	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 **MCKOE**: master clock output enable

0: Master clock output is disabled

1: Master clock output is enabled

Bit 24 **ODD**: odd factor for the prescaler

Refer to *Section 38.9.9: Clock generator* for details

0: Real divider value is = I2SDIV *2

1: Real divider value is = (I2SDIV * 2) + 1

Bits 23:16 **I2SDIV[7:0]**: I^2S linear prescaler
I2SDIV can take any values except the value 1, when ODD is also equal to 1.
Refer to *Section 38.9.9: Clock generator* for details

Bit 15 Reserved, must be kept at reset value.

Bit 14 **DATFMT**: data format
0: The data inside the SPI_RXDR or SPI_TXDR register are right aligned
1: The data inside the SPI_RXDR or SPI_TXDR register are left aligned.

Bit 13 **WSINV**: word select inversion
This bit is used to invert the default polarity of WS signal.
0: In I2S Philips standard, the left channel transfer starts one CK cycle after the WS falling edge, and the right channel one CK cycle after the WS rising edge.
In MSB- or LSB-justified mode, the left channel is transferred when WS is HIGH, and the right channel when WS is LOW.
In PCM short mode the data transfer starts at the falling edge of WS, while it starts at the rising edge of WS in PCM long mode.
1: In I2S Philips standard, the left channel transfer starts one CK cycle after the WS rising edge, and the right channel one CK cycle after the WS falling edge.
In MSB- or LSB-justified mode, the left channel is transfered when WS is LOW, and right channel when WS is HIGH.
In PCM short mode the data transfer starts at the rising edge of WS, while it starts at the falling edge of WS in PCM long mode.

Bit 12 **FIXCH**: fixed channel length in slave
0: the channel length in slave mode is different from 16 or 32 bits (CHLEN must be set)
1: the channel length in slave mode is supposed to be 16 or 32 bits (according to CHLEN)

Bit 11 **CKPOL**: serial audio clock polarity
0: the signals generated by the SPI/I2S (that is SDO and WS) are changed on the falling edge of CK and the signals received by the SPI/I2S (that is SDI and WS) are read of the rising edge of CK.
1: the signals generated by the SPI/I2S (that is SDO and WS) are changed on the rising edge of CK and the signals received by the SPI/I2S (that is SDI and WS) are read of the falling edge of CK.

Bit 10 **CHLEN**: channel length (number of bits per audio channel)
0: 16-bit wide
1: 32-bit wide

Bits 9:8 **DATLEN[1:0]**: data length to be transferred.
00: 16-bit data length
01: 24-bit data length
10: 32-bit data length
11: Not allowed
Note: Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10), refer to Section 38.3: SPI implementation to check the supported data size.

Bit 7 **PCMSYNC**: PCM frame synchronization
0: short frame synchronization
1: long frame synchronization

Bit 6 Reserved, must be kept at reset value.

Bits 5:4 **I2SSTD[1:0]**: I²S standard selection
For more details on I²S standards, refer to *Section 38.9.5: Supported audio protocols*
00: I²S Philips standard.
01: MSB-justified standard (left justified)
10: LSB-justified standard (right justified)
11: PCM standard

Bits 3:1 **I2SCFG[2:0]**: I2S configuration mode
000: slave - transmit
001: slave - receive
010: master - transmit
011: master - receive
100: slave - Full Duplex
101: master - Full Duplex
others: not used

Bit 0 **I2SMOD**: I2S mode selection
0: SPI mode is selected
1: I2S/PCM mode is selected

38.11.15 SPI/I2S register map

Table 318. 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**	CTSIZE[15:0](1)																RXWNE	RXPLVL[1:0]		TXC	SUSP	Res.	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DPXP	TXP	RXP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0		0	0	0	0	0	0	0	0	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	Reserved	Res.																															
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

Table 318. SPI 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
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
0x050	**SPI_I2SCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV[7:0]								Res.	Res.	WSINV	FIXCH	CKPOL	CHLEN	DATLEN[1:0]		PCMSYNC	Res.	I2SSTD[1:0]		I2SCFG[2:0]		I2SMOD	
	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. 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 for the peripheral instances with data size limited to 16-bit. 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 2.2* for the register boundary addresses.

39 FD controller area network (FDCAN)

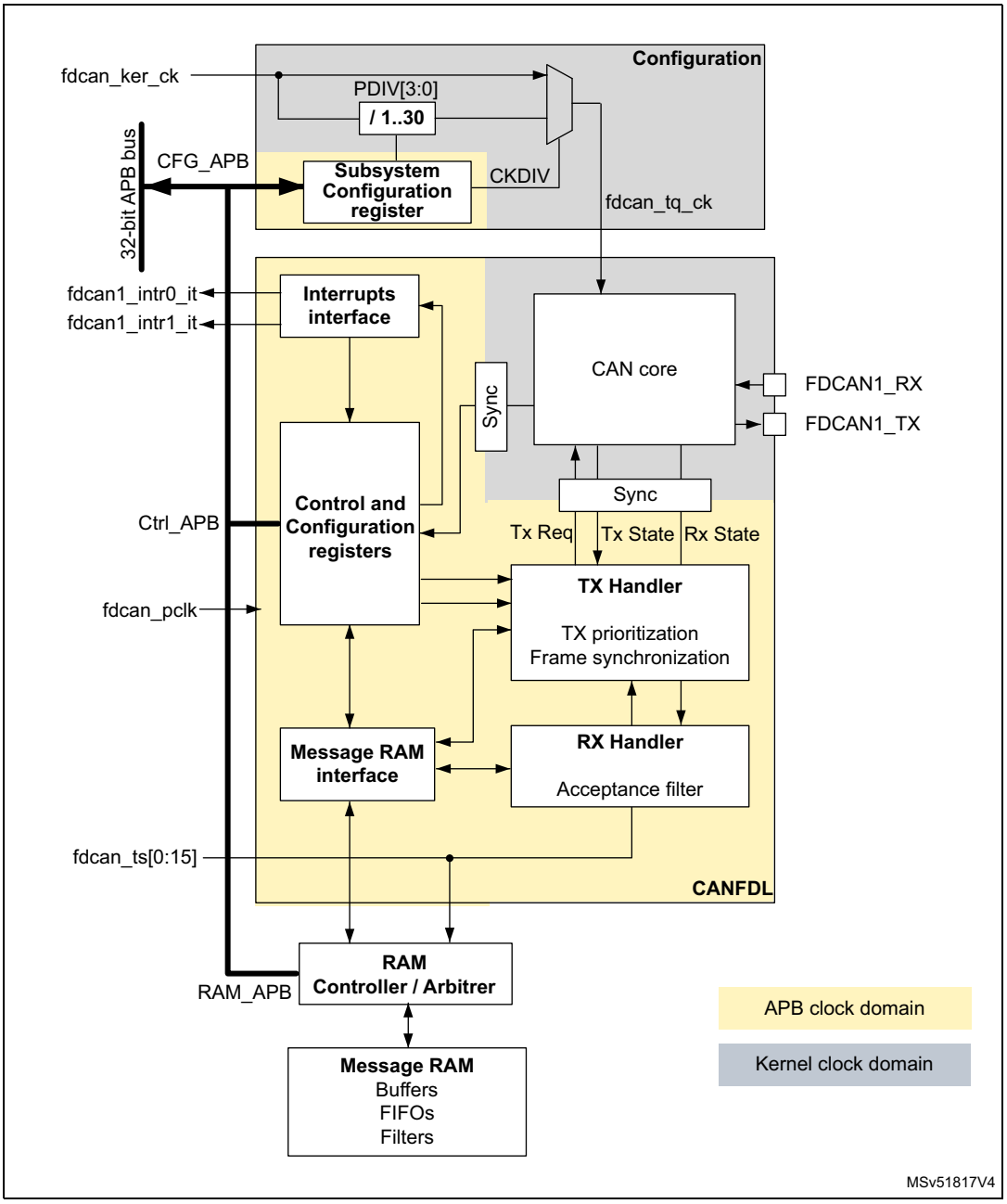
39.1 Introduction

The controller area network (CAN) subsystem (see *Figure 502*) consists of one CAN module, a shared message RAM, and a configuration block. Refer to the memory map for the base address of each of these parts.

The modules (FDCAN) are compliant with ISO 11898-1: 2015 (CAN protocol specification version 2.0 part A, B) and CAN FD protocol specification version 1.0.

A 0.8-Kbyte message RAM per FDCAN instance is used for filtering, transmitting event FIFOs, and receiving and transmitting FIFOs.

Figure 502. CAN subsystem.



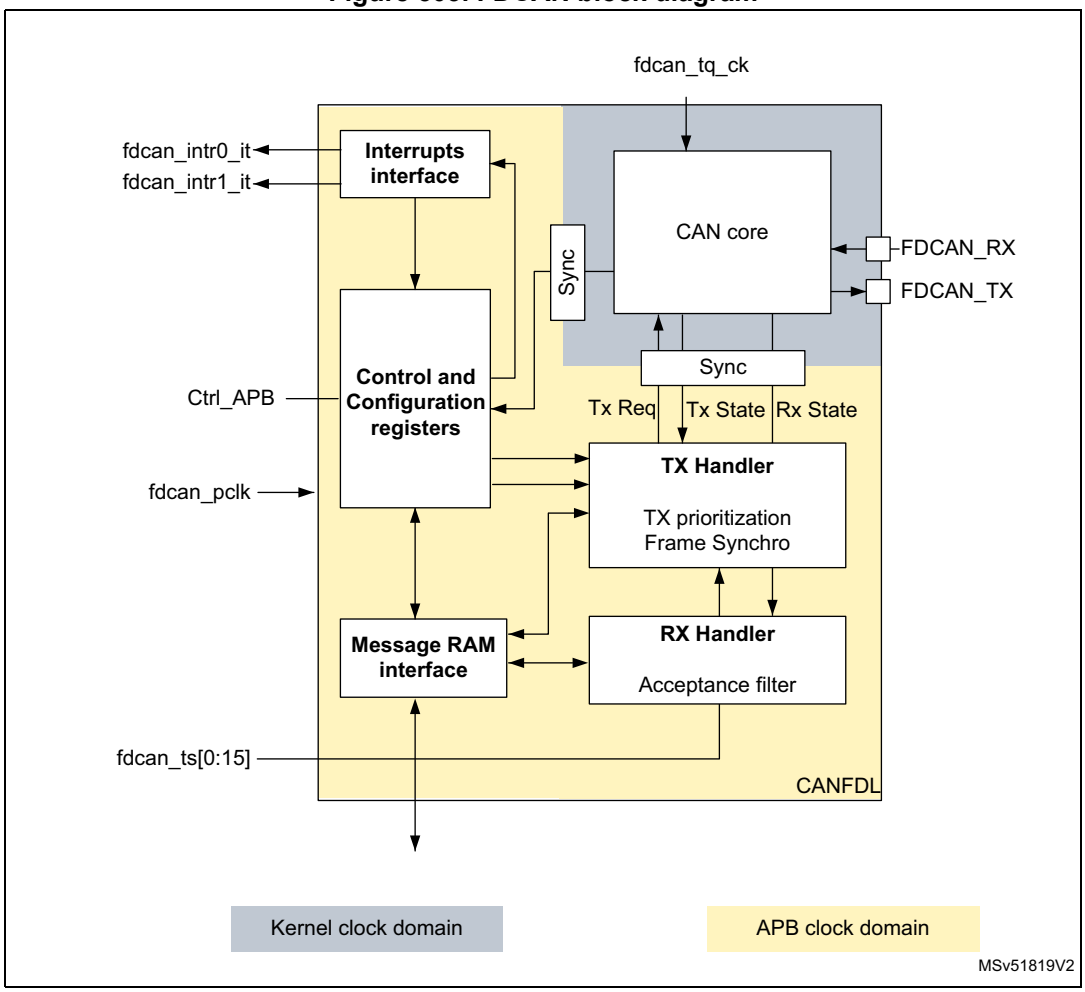
39.2 FDCAN main features

- Conform with CAN protocol version 2.0 part A, B, and ISO 11898-1: 2015
- CAN FD with maximum 64 data bytes supported
- CAN error logging
- AUTOSAR and J1939 support
- Improved acceptance filtering
- Two receive FIFOs of three payloads each (up to 64 bytes per payload)
- Separate signaling on reception of high priority messages
- Configurable transmit FIFO/queue of three payloads (up to 64 bytes per payload)
- Transmit event FIFO
- Programmable loop-back test mode
- Maskable module interrupts
- Two clock domains: APB bus interface and CAN core kernel clock
- Power-down support

39.3 FDCAN functional description

39.3.1 FDCAN block diagram

Figure 503. FDCAN block diagram



Dual interrupt lines

The FDCAN peripheral provides two interrupt lines, fdcan_intr0_it and fdcan_intr1_it.

By programming the EINT0 and EINT1 bits of the FDCAN_ILE register, the interrupt lines can be independently enabled or disabled.

CAN core

The CAN core contains the protocol controller and receive/transmit shift registers. It handles all ISO 11898-1: 2015 protocol functions and supports both 11-bit and 29-bit identifiers.

Sync

This block synchronizes signals from the APB clock domain to the CAN kernel clock domain and vice versa.

Tx handler

The Tx handler controls the message transfer from the message RAM to the CAN core. A maximum of three Tx buffers is available for transmission. The Tx buffer can be used as Tx FIFO or as a Tx queue. Tx event FIFO stores Tx timestamps together with the corresponding message ID. Transmit cancellation is also supported.

Rx handler

The Rx handler controls the transfer of received messages from the CAN core to the external message RAM. The Rx handler supports two receive FIFOs, for storage of all messages that have passed acceptance filtering. An Rx timestamp is stored together with each message. Up to 28 filters can be defined for 11-bit IDs; up to eight filters for 29-bit IDs.

APB interface

The APB interface connects the FDCAN to the APB bus for configuration registers, controller configuration, and RAM access.

Message RAM interface

The message RAM interface connects the FDCAN access to an external 1-Kbyte message RAM through a RAM controller/arbiter.

39.3.2 FDCAN pins and internal signals

The CAN subsystem I/O signals and pins are detailed, respectively, in *Table 319*, *Table 320*, and *Figure 502*.

Table 319. CAN subsystem I/O signals

Name	Type	Description
fdcan_ker_ck	Digital input	CAN subsystem kernel clock input
fdcan_pclk		CAN subsystem APB interface clock input
fdcan_intr0_it	Digital output	FDCAN interrupt0
fdcan_intr1_it		FDCAN interrupt1
fdcan_ts[0:15]	-	External timestamp vector
APB interface	Digital input/output	Single APP with multiple psel for configuration, control and RAM access

Table 320. CAN subsystem I/O pins

Name	Type	Description
FDCAN_RX	Digital input	FDCAN receive pin
FDCAN_TX	Digital output	FDCAN transmit pin

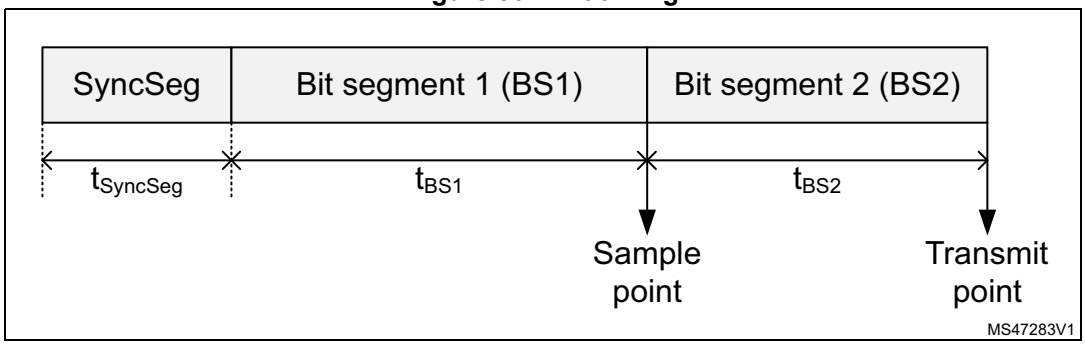
39.3.3 Bit timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.

As shown in *Figure 504*, this operation can be explained simply by splitting the bit time in three segments, as follows:

- Synchronization segment (SYNC_SEG): a bit change is expected to occur within this time segment, having a fixed length of one time quantum ($1 \times t_q$).
- Bit segment 1 (BS1): defines the location of the sample point. It includes the PROP_SEG and PHASE_SEG1 of the CAN standard. Its duration is programmable from 1 to 16 time quanta, but can be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of various nodes of the network.
- Bit segment 2 (BS2): defines the location of the transmit point. It represents the PHASE_SEG2 of the CAN standard, its duration is programmable between one and eight time quanta, but can also be automatically shortened to compensate for negative phase drifts.

Figure 504. Bit timing



The baud rate is the inverse of the bit time (baud rate = 1 / bit time), which, in turn, is the sum of three components (see *Figure 504*):

bit time = $t_{SyncSeg} + t_{BS1} + t_{BS2}$

Where:

– For the nominal bit time

$t_q = (NBRP[8:0] + 1) \times t_{fdcan\_tq\_clk}$

$t_{SyncSeg} = 1 \times t_q$

$t_{BS1} = t_q \times (NTSEG1[7:0] + 1)$

$t_{BS2} = t_q \times (NTSEG2[6:0] + 1)$

Where NBRP[8:0], NTSEG1[7:0], and NTSEG2[6:0] bitfields belong to the FDCAN_NBTP register.

– For the data bit time

$t_q = (DBRP[4:0] + 1) \times t_{fdcan\_tq\_clk}$

$t_{SyncSeg} = 1 \times t_q$

$t_{BS1} = t_q \times (DTSEG1[4:0] + 1)$

$t_{BS2} = t_q \times (DTSEG2[3:0] + 1)$

Where DBRP[4:0], DTSEG1[4:0], and DTSEG2[3:0] belong to the FDCAN_DBTP register.

The (re)synchronization jump width (SJW) defines an upper bound for the amount of lengthening or shortening of the bit segments. It is programmable between one and four time quanta.

A valid edge is defined as the first transition in a bit time from dominant to recessive bus level, provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC_SEG, BS1 is extended by up to SJW, so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC_SEG, BS2 is shortened by up to SJW, so that the transmit point is moved earlier.

As a safeguard against programming errors, the configuration of the bit timing register is only possible while the device is in Standby mode. The FDCAN_DBTP and FDCAN_NBTP registers (dedicated, respectively, to data and nominal bit timing) are only accessible when the CCE and INIT of the FDCA_CCCR register are set.

The FDCAN requires that the CAN time quanta clock is always below or equal to the APB clock ($f_{dcan\_tq\_ck} < f_{dcan\_pclk}$).

Note: For a detailed description of the CAN bit timing and resynchronization mechanism, refer to the ISO 11898-1 standard.

39.3.4 Operating modes

Configuration

Access to peripheral version, hardware, and input clock divider configuration. When the clock divider is set to 0, the primary input clock is used as it is.

Software initialization

Software initialization is started by setting the INIT bit of the FDCAN_CCCR register, by software, by a hardware reset, or by entering bus-off state. While the INIT bit is set, message transfers from and to the CAN bus are stopped, and the status of the CAN bus output FDCAN_TX is recessive (high). The EML (error management logic) counters are unchanged. Setting the INIT bit does not change any configuration register. Clearing INIT bit of FDCAN_CCCR finishes the software initialization. Afterwards the bit stream processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (bus-idle) before it can take part in bus activities and start the message transfer.

Access to the FDCAN configuration registers is only enabled when the INIT bit and the CCE bit of the FDCAN_CCCR register are both set.

The CCE bit of the FDCAN_CCCR register can only be set/cleared while the INIT bit of FDCAN_CCCR is set. The CCE bit is automatically cleared when the INIT bit is cleared.

The following registers are reset when the CCE bit of the FDCAN_CCCR register is set:

- FDCAN_HPMS: High priority message status
- FDCAN_RXF0S: Rx FIFO 0 status
- FDCAN_RXF1S: Rx FIFO 1 status
- FDCAN_TXFQS: Tx FIFO/queue status
- FDCAN_TXBRP: Tx buffer request pending
- FDCAN_TXBTO: Tx buffer transmission occurred
- FDCAN_TXBCF: Tx buffer cancellation finished
- FDCAN_TXEFS: Tx event FIFO status

The timeout counter value (TOC[15:0] bit of the FDCAN_TOCV register) is preset to the value configured by the TOP[15:0] of the FDCAN_TOCC register when the CCE bit of the FDCAN_CCCR is set.

In addition, the state machines of the Tx handler and Rx handler are held in idle state while the CCE bit is set.

The following registers can be written only when the CCE bit is cleared:

- FDCAN_TXBAR: Tx buffer add request
- FDCAN_TXBCR: Tx buffer cancellation request

The TEST and the MON bits of the FDCAN_CCCR register can be set only by software while the INIT and the CCE bits of the FDCAN_CCCR register are both set. Both bits can be reset at any time. The DAR bit of FDCAN_CCCR can only be set/cleared while the INIT and CCE bits are both set.

Normal operation

The FDCAN default operating mode after hardware reset is event-driven CAN communication. TT operation mode is not supported.

Once the FDCAN is initialized and the INIT bit of the FDCAN_CCCR register is cleared, the FDCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including message ID and DLC are stored into the Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted, the Tx FIFO or the Tx queue can be initialized or updated. Automated transmission on reception of remote frames is not supported.

CAN FD operation

There are two variants in the FDCAN protocol:

- Long frame mode (LFM), where the data field of a CAN frame may be longer than eight bytes.
- Fast frame mode (FFM), where the control field, data field, and CRC field of a CAN frame are transmitted with a higher bit rate compared to the beginning and to the end of the frame.

The fast frame mode can be used in combination with the long frame mode.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers are decoded as FDF bit: FDF recessive signifies a CAN FD frame, while FDF dominant signifies a classic CAN frame.

In a CAN FD frame, the two bits following FDF (res and BRS) decide whether the bit rate inside this CAN FD frame is switched. A CAN FD bit rate switch is signified by res dominant and BRS recessive. The coding of res recessive is reserved for future expansion of the protocol. In case the FDCAN receives a frame with FDF recessive and res recessive, it signals a protocol exception event by setting the PXE bit of the FDCAN_PSR register. When protocol exception handling is enabled (PXHD = 0 in FDCAN_CCCR), this causes the operation state to change from receiver (ACT[1:0] = 10 in FDCAN_PSR) to integrating (ACT[1:0] = 00 in FDCAN_PSR) at the next sample point. If protocol exception handling is disabled (PXHD = 1 in FDCAN_CCCR), the FDCAN treats a recessive res bit as a form error and responds with an error frame.

CAN FD operation is enabled by programming the FDOE bit of the FDCAN_CCCR register. In case FDOE = 1, transmission and reception of CAN FD frames are enabled. Transmission and reception of classic CAN frames are always possible. Whether a CAN FD frame or a classic CAN frame is transmitted can be configured via the FDF bit in the respective Tx buffer element. With FDOE = 0, received frames are interpreted as classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if the FDF bit of a Tx buffer element is set. The FDOE and BRSE bits of the FDCAN_CCCR register can only be changed while the INIT and CCE bits are both set.

With FDOE = 0, the setting of the FDF and BRS bits is ignored, and frames are transmitted in classic CAN format. With FDOE = 1 and BRSE = 0, only the FDF bit of a Tx buffer element is evaluated. With FDOE = 1 and BRSE = 1, transmission of CAN FD frames with bit rate switching is enabled. All Tx buffer elements with FDF and BRS bits set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is recommended only under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case, disable the CAN FD bit rate switching option for transmissions.
- During system startup, all nodes transmit classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wake-up messages in CAN partial networking have to be transmitted in classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non-CAN FD nodes are held in silent mode until programming is complete. Then all nodes switch back to classic CAN communication.

In the FDCAN format, the coding of the DLC differs from that of the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15 (that in standard CAN all code a data field of 8 bytes) are coded according to *Table 321*.

Table 321. DLC coding in FDCAN

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD fast frames, the bit timing is switched inside the frame, after the BRS (bit rate switch) bit, if this bit is recessive. Before the BRS bit, in the FDCAN arbitration phase, the standard CAN bit timing is used as defined by the FDCAN_DBTP register. In the following

FDCAN data phase, the fast CAN bit timing is used as defined by the FDCAN_DBTP register. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the FDCAN kernel clock frequency. For example, with an FDCAN kernel clock frequency of 20 MHz and the shortest configurable bit time of four time quanta (t_q), the bit rate in the data phase is 5 Mbit/s.

In both data frame formats (CAN FD long frames and CAN FD fast frames), the value of bit ESI (error status indicator) is determined by the transmitter error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant. In CAN FD remote frames, the ESI bit is always transmitted dominant, independent of the transmitter error state. The data length code of CAN FD remote frames is transmitted as 0.

In case an FDCAN Tx buffer is configured for FDCAN transmission with DLC > 8, the first eight bytes are transmitted as configured in the Tx buffer while the remaining part of the data field is padded with 0xCC. When the FDCAN receives a FDCAN frame with DLC > 8, the first eight bytes of that frame are stored into the matching Rx FIFO. The remaining bytes are discarded.

Transceiver delay compensation

During the data phase of an FDCAN transmission, only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin FDCAN_TX, the protocol controller receives the transmitted data from its local CAN transceiver via pin FDCAN_RX. The received data is delayed by the CAN transceiver loop delay. If this delay is greater than TSEG1 (time segment before sample point), and a bit error is detected. Without transceiver delay compensation, the bit rate in the data phase of an FDCAN frame is limited by the transceiver loop delay.

The FDCAN implements a delay compensation mechanism to compensate the CAN transceiver loop delay, thereby enabling transmission with higher bit rates during the FDCAN data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the secondary sample point (SSP). If a bit error is detected, the transmitter reacts on this bit error at the next following regular sample point. During the arbitration phase, the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay. This is enabled by setting the TDC bit of the FDCAN_DBTP register, and described in detail in the ISO11898-1 specification.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the FDCAN transmit output pin FDCAN_TX through the transceiver to the receive input pin FDCAN_RX plus the transmitter delay compensation offset as configured by TDCO[6:0] of FDCAN_TDCR. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (for example, half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mt_q (minimum time quantum, one period of fdcan_tq_ck clock).

The TDCV[6:0] bitfield of the FDCAN_PSR register shows the actual transmitter delay compensation value. TDCV[6:0] is cleared when the INIT is set in the FDCAN_CCCR. It is

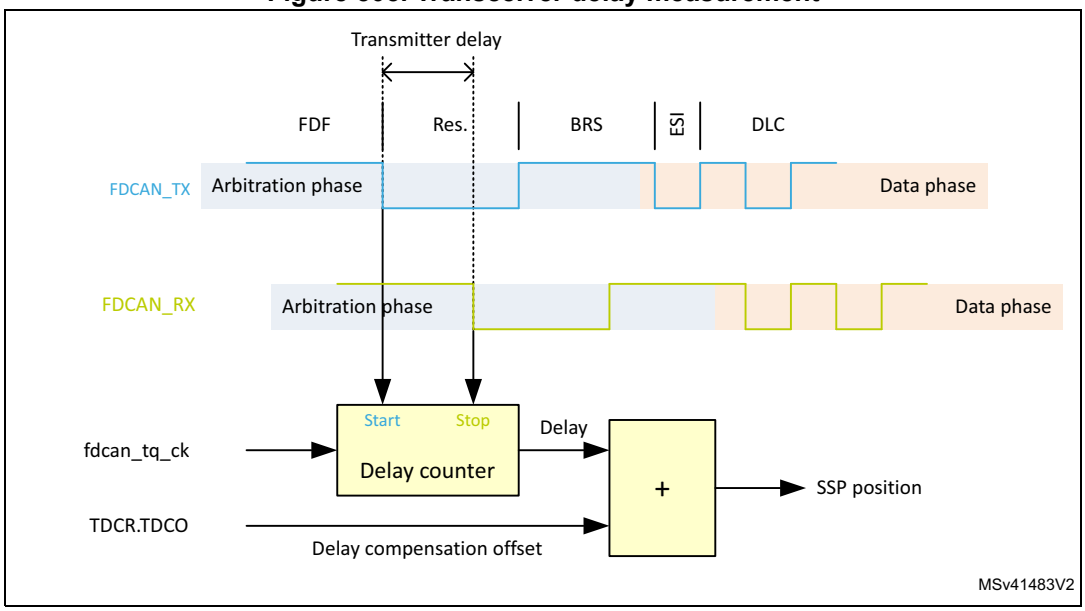
updated at each transmission of an FD frame while the TDC bit of the FDCAN_DBTP register is set.

The following boundary conditions have to be considered for the transmitter delay compensation implemented in the FDCAN:

- The sum of the measured delay from FDCAN_Tx to FDCAN_Rx and the configured transmitter delay compensation offset TDCO[6:0] has to be lower than 6-bit times in the data phase.
- The sum of the measured delay from FDCAN_TX to FDCAN_RX and the configured transmitter delay compensation offset TDCO[6:0] has to be lower than or equal to 127 × mt_q. If the sum exceeds this value, the maximum value (127 × mt_q) is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, which stops checking received bits at the SSPs.

If transmitter delay compensation is enabled by setting the TDC bit of the FDCAN_DBTP; the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input pin FDCAN_TX of the transmitter. The resolution of this measurement is one mt_q.

Figure 505. Transceiver delay measurement



To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit (resulting in a too early SSP position), the use of a transmitter delay compensation filter window can be enabled by programming the TDCF[6:0] bitfield of the FDCAN_TDCR register. This defines a minimum value for the SSP position. Dominant edges on FDCAN_RX that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least TDCF[6:0] and FDCAN_RX is low.

Restricted operation mode

In restricted operation mode, the node is able to receive data and remote frames, and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits. Instead, it waits for the occurrence of a bus-idle condition to resynchronize itself to the CAN communication. The error counters (REC[6:0] and TEC[7:0] in FDCAN_ECR) are frozen while the error logging (CEL[7:0]) is active. The software can set the FDCAN into restricted operation mode by setting the ASM bit of FDCAN_CCCR. The bit can only be set by software when both CCE and INIT bits are set in FDCAN_CCCR. The bit can be cleared by software at any time.

Restricted operation mode is automatically entered when the Tx handler is not able to read data from the message RAM in time. To leave restricted operation mode, the software has to clear the ASM bit of FDCAN_CCCR.

The restricted operation mode can be used in applications that adapt themselves to different CAN bit rates. In this case, the application tests different bit rates and leaves the restricted operation mode after it has received a valid frame.

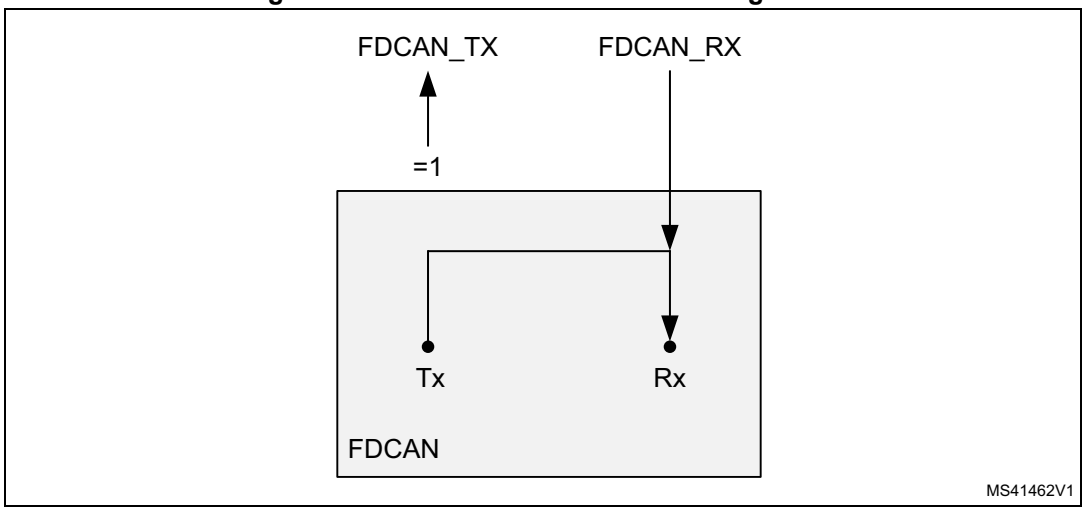
Note: The restricted operation mode must not be combined with the loop-back mode (internal or external).

Bus monitoring mode

The FDCAN is set in bus monitoring mode by setting the MON bit of the FDCAN_CCCR register. In bus monitoring mode (for more details refer to ISO11898-1, 10.12 bus monitoring), the FDCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the FDCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the FDCAN can monitor it, even if the CAN bus remains in recessive state. In bus monitoring mode, the FDCAN_TXBRP register is held in reset state.

The bus monitoring mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. *Figure 506* shows the connection of FDCAN_TX and FDCAN_RX signals to the FDCAN in bus monitoring mode.

Figure 506. Pin control in bus monitoring mode



Disabled automatic retransmission mode (DAR)

According to the CAN specification (see ISO11898-1, 6.3.3 Recovery Management), the FDCAN provides means for automatic retransmission of frames that have lost arbitration or have been disturbed by errors during transmission. By default, automatic retransmission is enabled. The DAR mode can be disabled through the DAR bit of the FDCAN_CCCR register.

Frame transmission in DAR mode

In DAR mode, all transmissions are automatically canceled after they have been started on the CAN bus. A Tx buffer Tx request pending bit (TRPx in FDCAN_TXBRP) is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, when it has been aborted due to lost arbitration, or when an error has occurred during frame transmission.

- Successful transmission
  - The corresponding Tx buffer transmission occurred bit TOx is set in FDCAN_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is cleared in the FDCAN_TXBCF register.
- Successful transmission in spite of cancellation
  - The corresponding Tx buffer transmission occurred bit TOx is set in the FDCAN_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is set in the FDCAN_TXBCF register.
- Arbitration loss or frame transmission disturbed
  - The corresponding Tx buffer transmission occurred bit TOx is cleared in the FDCAN_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is set in the FDCAN_TXBCF register.

In case of a successful frame transmission, and if the storage of Tx events is enabled, a Tx event FIFO element is written with event type ET = 10 (transmission in spite of cancellation).

Power-down (Sleep mode)

Power-down entry

The FDCAN can be set into power-down mode controlled by setting the CSR bit of the FDCAN_CCCR register. As long as the clock stop request is active, CSR is read as 1.

When all pending transmission requests have completed, the FDCAN waits until the bus-idle state is detected. The FDCAN then sets the INIT bit of the FDCAN_CCCR register to prevent any further CAN transfers. Now, the FDCAN acknowledges that it is ready for power-down by setting the CSA bit of the FDCAN_CCCR register. In this state, before the clocks are switched off, further register accesses can be made. A write access to the INIT bit has no effect. Now, the module clock inputs can be switched off.

Power-down exit

To leave power-down mode, the application has to turn on the module clocks before clearing the CSR bit. The FDCAN acknowledges this by clearing the CSA bit. Afterwards, the application can restart CAN communication by clearing the INIT bit.

Test modes

To enable write access to *FDCAN test register (FDCAN_TEST)*, the TEST bit of the FDCAN_CCCR register must be set, thus enabling the configuration of test modes and functions.

Four output functions are available for the CAN transmit pin FDCAN_TX by programming the TX[1:0] bitfield of the FDCAN_TEST register. In addition to its default function (the serial data output), it can drive the CAN sample point signal to monitor the FDCAN bit timing as well as drive constant dominant or recessive values. The actual value at pin FDCAN_RX can be read from the RX bit of FDCAN_TEST. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN kernel clock and APB clock domain, there can be a delay of several APB clock periods between writing to TX[1:0] until the new configuration is visible at FDCAN_TX output pin. This applies also when reading FDCAN_RX input pin via RX.

Note: Test modes must be used for production tests or self-test only. The software control for FDCAN_TX pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

External loop-back mode

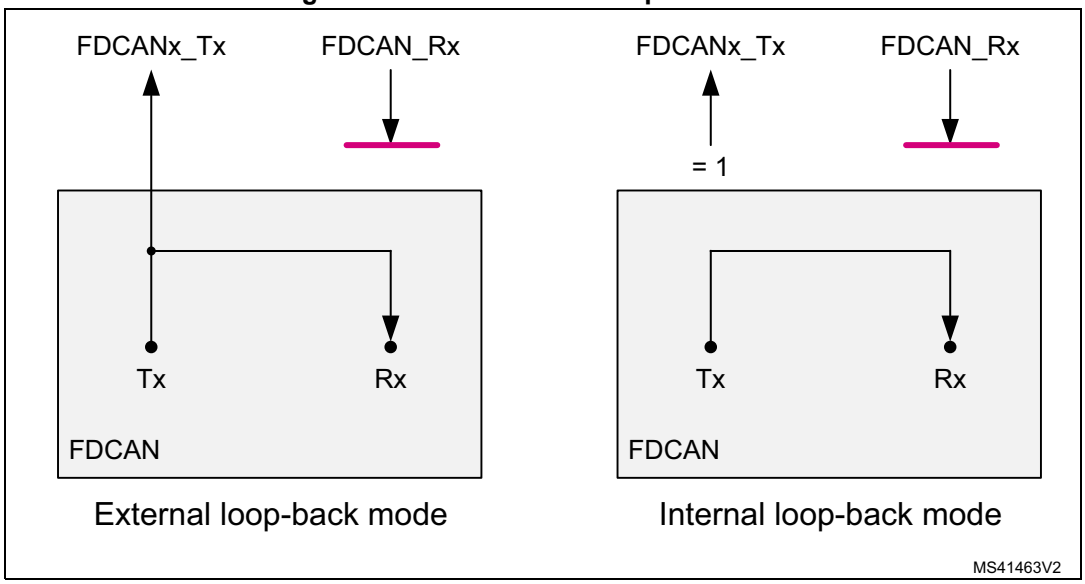
The FDCAN can be set in external loop-back mode by setting the LBCK bit of the FDCAN_TEST register. In loop-back mode, the FDCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. *Figure 507* shows the connection of transmit and receive signals FDCAN_TX and FDCAN_RX to the FDCAN in external loop-back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the FDCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in loop-back mode. In this mode, the FDCAN performs an internal feedback from its transmit output to its receive input. The actual value of the FDCAN_RX input pin is disregarded by the FDCAN. The transmitted messages can be monitored at the FDCAN_TX transmit pin.

Internal loop-back mode

Internal loop-back mode is entered by setting both the LBCK bit of FDCAN_TEST and the MON bit of FDCAN_CCR. This mode can be used for a “hot self-test”, meaning the FDCAN can be tested without affecting a running CAN system connected to the FDCAN_TX and FDCAN_RX pins. In this mode, FDCAN_RX pin is disconnected from the FDCAN and FDCAN_TX pin is held recessive. *Figure 507* shows the connection of FDCAN_TX and FDCAN_RX pins to the FDCAN in case of internal loop-back mode.

Figure 507. Pin control in loop-back mode



Timestamp generation

For timestamp generation, the FDCAN supplies a 16-bit wrap-around counter. A prescaler (TCP[3:0] of FDCAN_TSCC) can be configured to clock the counter in multiples of CAN bit times (1 to 16). The counter is readable via the TCV[15:0] bitfield of the FDCAN_TSCV register. A write access to TSCV15:0] resets the counter to 0. When the timestamp counter wraps around, the interrupt flag (TSW bit of FDCAN_ISR) is set.

On start of frame reception/transmission, the counter value is captured and stored into the timestamp section of an Rx FIFO (RXTS[15:0]) or Tx event FIFO (TXTS[15:0]) element.

By programming TSS[1:0] of FDCAN_TSCC, a 16-bit timestamp can be used.

Debug mode behavior

In debug mode, the set/reset on read feature is automatically disabled during the debugger register access, and enabled during normal MCU operation

Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO the FDCAN supplies a 16-bit timeout counter. It operates as a down-counter and uses the same prescaler controlled by TCP[3:0] of FDCAN_TSCC as the timestamp counter. The timeout counter is configured via the FDCAN_TOCC register. The actual counter value can be read from the TOC[15:0] bitfield of FDCAN_TOCV. The timeout counter can only be started while the INIT bit of FDCAN_CCCR is cleared. It is stopped when INIT is set, for example, when the FDCAN enters bus-off state.

The operation mode is selected by TOS[1:0] of FDCAN_TOCC. When operating in continuous mode, the counter starts when INIT is cleared. A write to FDCAN_TOCV presets the counter to the value configured by TOP[15:0] in FDCAN_TOCC and continues down-counting.

When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOP[15:0]. Down-counting is started when the first FIFO element is stored. Writing to FDCAN_TOCV has no effect.

When the counter reaches 0, the TOO interrupt flag is set in the FDCAN_IR register. In continuous mode, the counter is immediately restarted at TOP[15:0].

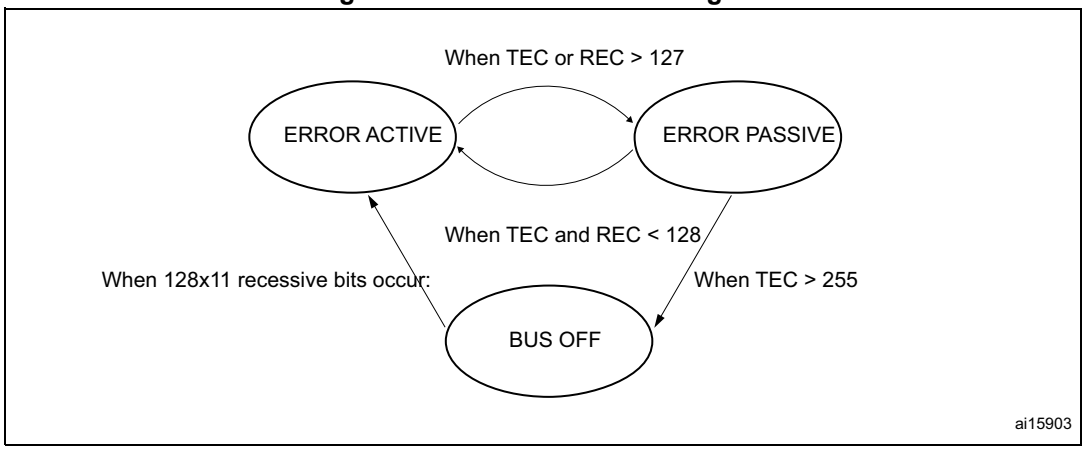
Note: The clock signal for the timeout counter is derived from the CAN core sample point signal. Therefore, the point in time where the timeout counter is decremented may vary due to the synchronization/resynchronization mechanism of the CAN core. If the baud rate switch feature in FDCAN is used, the timeout counter is clocked differently in the arbitration and data fields.

39.3.5 Error management

As described in the CAN protocol, the error management is handled entirely by hardware using the transmit error counter (the TEC[7:0] bitfield of the *FDCAN error counter register (FDCAN_ECR)*) and the receive error counter (the REC[6:0] bitfield of the *FDCAN error counter register (FDCAN_ECR)*). These values are incremented or decremented according to the error condition. For detailed information on TEC[7:0] and REC[6:0] management, refer to the CAN standard. Both values can be read by software to determine the stability of the network.

The bus-off state is reached when TEC[7:0] is greater than 255. This state is also indicated by the BO flag of the *FDCAN protocol status register (FDCAN_PSR)*. In bus-off state, the CAN is no longer able to transmit and receive messages. It has to wait at least for the duration of the recovery sequence specified in the CAN standard (128 occurrences of 11 consecutive recessive bits monitored on FDCAN_RX input).

Figure 508. CAN error state diagram

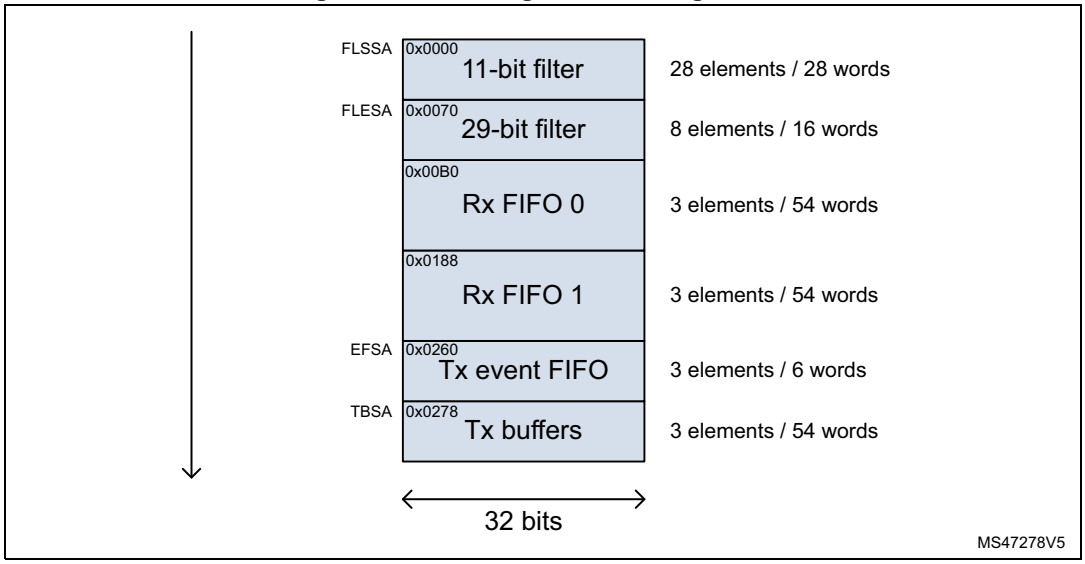


Note: In initialization mode, the CAN does not monitor the FDCAN_RX *signal, and therefore cannot complete the recovery sequence. To recover from an error state, the CAN must operate in normal mode.*

39.3.6 Message RAM

The message RAM is 32-bit wide, and the FDCAN module is configured to allocate up to 212 words in it. It is not necessary to configure each of the sections shown in *Figure 509*.

Figure 509. Message RAM configuration



When the FDCAN addresses the message RAM, it addresses 32-bit words (aligned), not a single byte. The RAM addresses are 32-bit words, that is, only bits 15 to 2 are evaluated, the two least significant bits are ignored.

In case of multiple instances, the RAM start address for the FDCANn is computed by end address + 4 of FDCANn - 1, and the FDCANn end address is computed by FDCANn start address + 0x0350 - 4.

As an example, for two instances:

- FDCAN1:
  - Start address 0x0000
  - End address 0x034C (as in *Figure 509*)
- FDCAN2:
  - Start address = 0x034C (FDCAN1 end address) + 4 = 0x0350
  - End address = 0x0350 (FDCAN2 start address) + 0x0350 - 4 = 0x069C.

Rx handling

The Rx handler controls the acceptance filtering, the transfer of received messages to one of the two Rx FIFOs, as well as the Rx FIFO put and get indices.

Acceptance filter

The FDCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and another for extended identifiers. These filters can be assigned to Rx FIFO 0 or Rx FIFO 1. For acceptance filtering, each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element, and the following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - Range filter (from - to)
  - Filter for one or two dedicated IDs
  - Classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering.
- Each filter element can be enabled/disabled individually.
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global filter configuration (RXGFC)
- Extended ID AND mask (XIDAM)

Depending on the configuration of the filter element (SFEC[2:0]/EFEC[2:0]), a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Reject received frame
- Set the high priority message interrupt flag HPM in FDCAN_IR
- Set the high priority message interrupt flag HPM in FDCAN_IR, and store the received frame in FIFO 0 or FIFO 1.

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx FIFO has been found, the message handler starts writing the received message data in 32-bit portions to the matching Rx FIFO. If the CAN protocol controller has detected an error condition (for example, CRC error), this message is discarded with the following impact:

- **Rx FIFO**

  The put index of the matching Rx FIFO is not updated, but the related Rx FIFO element is partly overwritten with the received data. For error type, see LEC[2:0] and DLEC[2:0] bitfields of the FDCAN_PSR register. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in *Rx FIFO overwrite mode* have to be considered.

Note: When an accepted message is written to one of the two Rx FIFOs, the unmodified received identifier is stored independently from the used filters. The result of the acceptance filter process strongly depends on the sequence of configured filter elements.

Range filter

The filter matches for all received frames with message IDs in the range defined by SF1ID/SF2ID and EF1ID/EF2ID.

There are two possibilities when range filtering is used together with extended frames:

- EFT[1:0] = 00: the message ID of received frames is AND-ed with the extended ID AND mask (XIDAM) before the range filter is applied.
- EFT[1:0] = 11: the extended ID AND mask (XIDAM) is not used for range filtering.

Filter for dedicated IDs

A filter element can be configured to filter for one or two specific message IDs. To filter for one specific message ID, the filter element has to be configured with SF1ID = SF2ID and EF1ID = EF2ID.

Classic bit mask filter

The classic bit mask filtering is intended to filter groups of message IDs by masking single bits of a received message ID. With classic bit mask filtering SF1ID/EF1ID is used as message ID filter, while SF2ID/EF2ID is used as filter mask.

0 bit at the filter mask masks out the corresponding bit position of the configured ID filter. For example, the value of the received message ID at that bit position is not relevant for acceptance filtering. Only the bits of the received message ID where the corresponding mask bits are 1 are relevant for acceptance filtering.

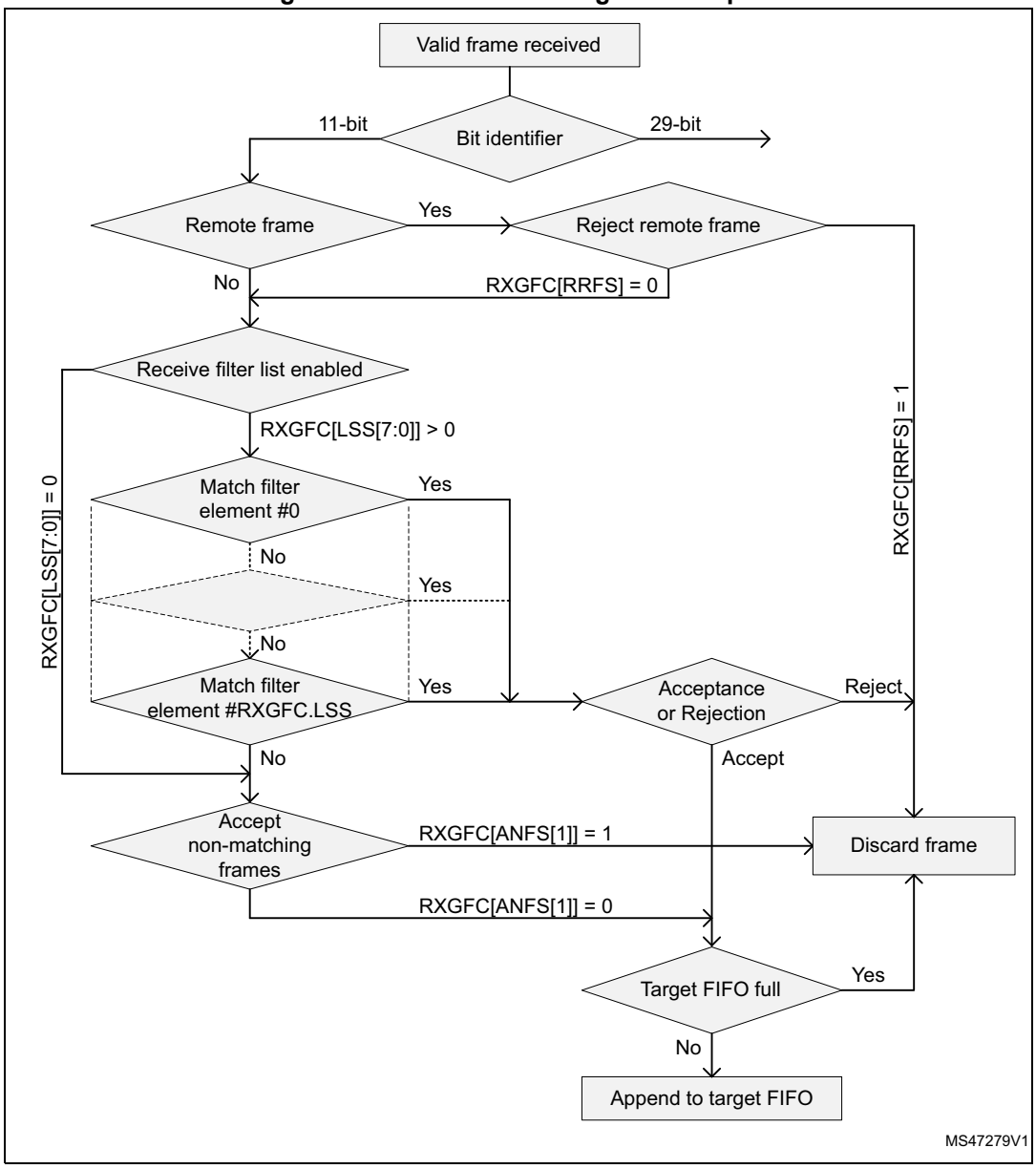
In case all mask bits are 1, a match occurs only when the received message ID and the message ID filter are identical. If all mask bits are 0, all message IDs match.

Standard message ID filtering

Figure 510 shows the flow for standard message ID (11-bit identifier) filtering. The standard message ID filter element is described in *Section 39.3.11*.

The standard message filtering is controlled by the FDCAN_RXGFC register. The standard message ID, the remote transmission request bit (RTR), and the identifier extension bit (IDE) of the received frames are compared against the list of configured filter elements.

Figure 510. Standard message ID filter path

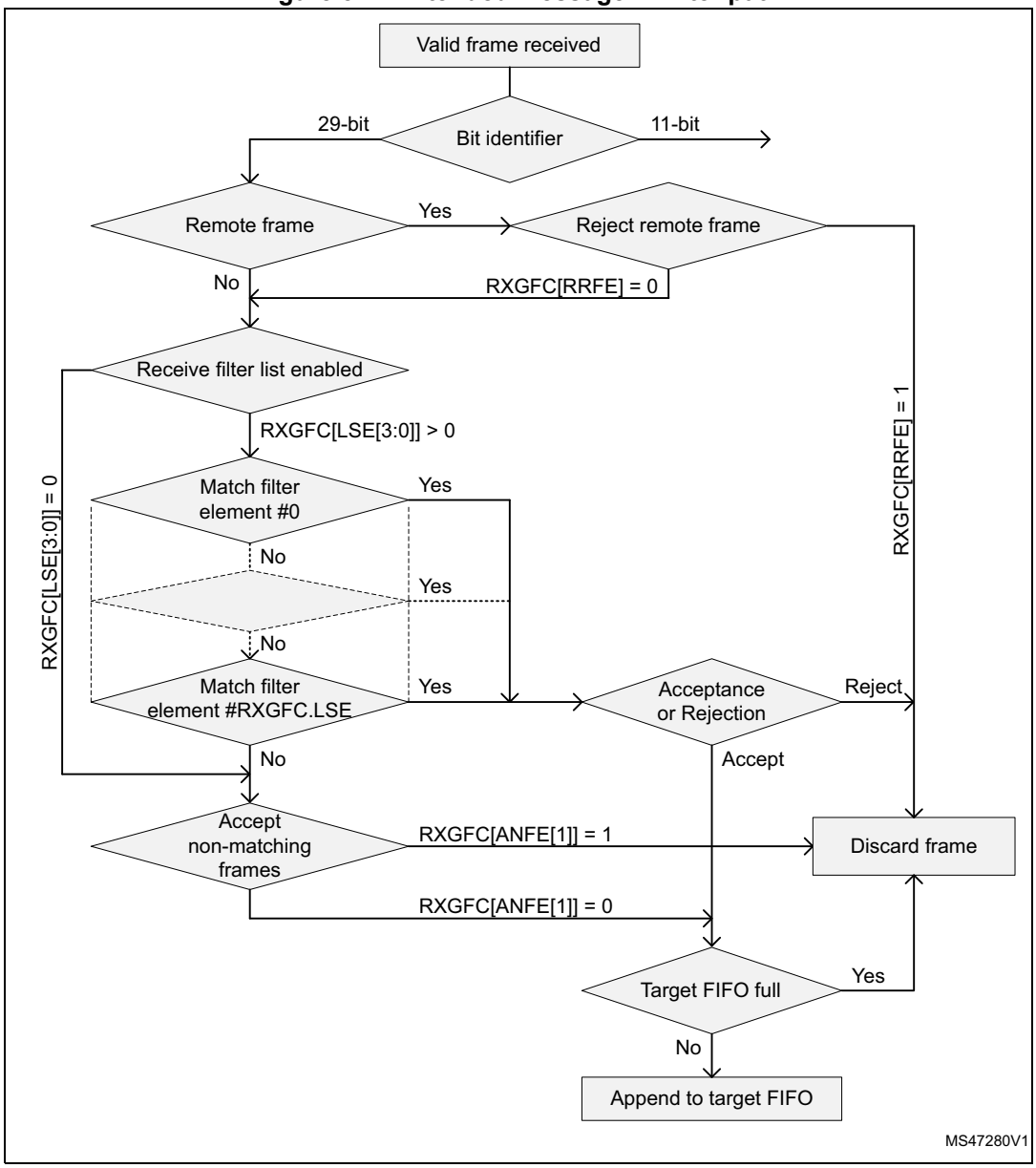


Extended message ID filtering

Figure 511 shows the flow for extended message ID (29-bit identifier) filtering. The extended message ID filter element is described in *Section 39.3.12*.

The extended message filtering is controlled by the FDCAN_RXGFC register. The extended message ID, the remote transmission request bit (RTR), and the identifier extension bit (IDE) of the received frames are compared against the list of configured filter elements.

Figure 511. Extended message ID filter path



The extended ID AND mask (XIDAM) is AND-ed with the received identifier before the filter list is executed.

Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can hold up to three elements each.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1, see *Acceptance filter*. The Rx FIFO element is described in *Section 39.3.8*.

When an Rx FIFO full condition is signaled by RFnF in FDCAN_IR (where n is the FIFO number), no further messages are written to the corresponding Rx FIFO until at least one message has been read out, and the Rx FIFO get index has been incremented. In case a message is received while the corresponding Rx FIFO is full, this message is discarded, and the interrupt flag RFnL is set in the FDCAN_IR register.

When reading from an Rx FIFO, the Rx FIFO get index (FnGI of FDCAN_RXFnS) + FIFO element size has to be added to the corresponding Rx FIFO start address (FnSA).

Rx FIFO blocking mode

The Rx FIFO blocking mode is configured by clearing the FnOM bit in the FDCAN_RXGFC register. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (FnPI = FnGI in FDCAN_RXFnS), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO get index has been incremented. An Rx FIFO full condition is signaled by FnF = 1 in FDCAN_RXFnS. In addition, the RFnF interrupt flag is set in FDCAN_IR.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded, and the message lost condition is signaled by setting RFnL bit in FDCAN_RXFnS. In addition, the RFnL interrupt flag is set in FDCAN_IR.

Rx FIFO overwrite mode

The Rx FIFO overwrite mode is configured by setting the FnOM bit of the FDCAN_RXGFC register.

When an Rx FIFO full condition (FnPI = FnGI of FDCAN_RXFnS) is signaled by FnF = 1 in FDCAN_RXFnS, the next message accepted for the FIFO overwrites the oldest FIFO message. Put and get indices are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signaled, reading from the Rx FIFO elements must start at least at get index + 1. This is because it may happen that a received message is written to the message RAM (put index) while the CPU is reading from the message RAM (get index). In this case, inconsistent data can be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO.

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO acknowledge index (FnA of FDCAN_RXFnA). This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (FnF = 0 in FDCAN_RXFnS).

Tx handling

The Tx handler handles transmission requests for the Tx FIFO and the Tx queue. It controls the transfer of transmit messages to the CAN core, the put and get indices, and the Tx event FIFO. Up to three Tx buffers can be set up for message transmission. The CAN message data field is configured to 64 bytes. the Tx FIFO allocates eighteen 32-bit words for storage of a Tx element.

Table 322. Possible configurations for frame transmission

CCCR		Tx buffer element		Frame transmission
BRSE	FDOE	FDF	BRS	
Ignored	0	Ignored	Ignored	Classic CAN
0	1	0	Ignored	Classic CAN
0	1	1	Ignored	FD without bit rate switching
1	1	0	Ignored	Classic CAN
1	1	1	0	FD without bit rate switching
1	1	1	1	FD with bit rate switching

Note: AUTOSAR requires at least three Tx queue buffers and support of transmit cancellation.

The Tx handler starts a Tx scan to check for the highest priority pending Tx request (Tx buffer with lowest message ID) when the Tx buffer request pending register (FDCAN_TXBRP) is updated, or when a transmission has been started.

Transmit pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are permanently specified to specific values and cannot easily be changed. These message identifiers can have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority must be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

As an example, if CAN ECU-1 has the feature enabled and is requested by its application software to transmit four messages, it waits, after the first successful message transmission, for two CAN bit times of bus-idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, these messages are started in the idle time, and they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The feature is controlled by the TXP bit of the CCCR register. If the bit is set, the FDCAN, each time it has successfully transmitted a message, pauses for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. By default, this feature is disabled (TXP = 0 in FDCAN_CCCR).

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

Tx FIFO

Tx FIFO operation is configured by clearing the TFQM bit of the FDCAN_TXBC register. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the get index (TFGI[1:0] bitfield of FDCAN_TXFQS). After each transmission, the get index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same message ID from different Tx buffers in the order that these messages have been written to the Tx FIFO. The FDCAN calculates the Tx FIFO free level (TFFL[2:0] bitfield of FDCAN_TXFQS) as the difference between the get and put index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx buffer referenced by the put index (TFQPI[1:0] bitfield of FDCAN_TXFQS). An add request increments the put index to the next free Tx FIFO element. When the put index reaches the get index, Tx FIFO full (TFQF = 1 in FDCAN_TXFQS) is signaled. In this case, no further messages must be written to the Tx FIFO until the next message has been transmitted and the get index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by setting the FDCAN_TXBAR bit related to the Tx buffer referenced by the Tx FIFO put index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx buffers starting with the put index. The transmissions are then requested via the FDCA_TXBAR register. The put index is then cyclically incremented by n. The number of requested Tx buffers must not exceed the number of free Tx buffers as indicated by the Tx FIFO free level.

When a transmission request for the Tx buffer referenced by the get index is canceled, the get index is incremented to the next Tx buffer with a transmission request is pending and the Tx FIFO free level is recalculated. When transmission cancellation is applied to any other Tx buffer, the get index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates eighteen 32-bit words in the message RAM. The Therefore, the start address of the next available (free) Tx FIFO buffer, is calculated by adding 18 times the put index TFQPI[1:0] (0 … 2) to the Tx buffer start address TBSA.

Tx queue

Tx queue operation is configured by setting the TFQM of the FDCAN_TXBC register. Messages stored in the Tx queue are transmitted starting with the message with the lowest message ID (highest priority).

In case of mixing of standard and extended message IDs, the standard message IDs are compared to bits [28:18] of extended message IDs.

In case multiple queue buffers are configured with the same message ID, the queue buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx buffer referenced by the put index (TFQPI[1:0] in FDCAN_TXFQS). An add request cyclically increments the put index to the next free Tx buffer. In case the Tx queue is full (TFQF = 1 in FDCAN_TXFQS), the put index is not valid and no further message must be written to the Tx queue until at least one of the requested messages has been sent out or a pending transmission request has been canceled.

The application can use the FDCAN_TXBRP register instead of the put index and can place messages to any Tx buffer without pending transmission request.

A Tx queue buffer allocates eighteen 32-bit words in the message RAM. The start address of Therefore, the next available (free) Tx queue buffer is calculated by adding 18 times the Tx queue put index TFQPI[1:0] (0 ... 2) to the Tx buffer start address TBSA.

Transmit cancellation

The FDCAN supports transmit cancellation. To cancel a requested transmission from a Tx queue buffer, the host has to write 1 to the corresponding bit position (= number of Tx buffer) of the FDCAN_TXBCR register. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of the FDCAN_TXBCF register.

In case a transmit cancellation is requested while a transmission from a Tx buffer is already ongoing, the corresponding FDCAN _TXBRP bit remains set as long as the transmission is in progress. If the transmission is successful, the corresponding FDCAN_TXBTO and FDCAN_TXBCF bits are set. If the transmission is not successful, it is not repeated and only the corresponding FDCAN_TXBCF bit is set.

Note: In case a pending transmission is canceled immediately before it has been started, there is a short time window where no transmission is started even if another message is pending in the node. This can enable another node to transmit a message that can have a priority lower than that of the second message in the node.

Tx event handling

To support Tx event handling the FDCAN has implemented a Tx event FIFO. After the FDCAN has transmitted a message on the CAN bus, message ID and timestamp are stored in a Tx event FIFO element. To link a Tx event to a Tx event FIFO element, the message marker from the transmitted Tx buffer is copied into the Tx event FIFO element.

The Tx event FIFO is configured to three elements. The Tx event FIFO element is described in *Tx FIFO*.

The purpose of the Tx event FIFO is to decouple handling transmit status information from transmit message handling that is, a Tx buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx buffer before overwriting that Tx buffer.

When a Tx event FIFO full condition is signaled by the TEFF bit of the FDCAN_IR, no further elements are written to the Tx event FIFO until at least one element has been read out and the Tx event FIFO get index has been incremented. In case a Tx event occurs while the Tx event FIFO is full, this event is discarded and the TEFL interrupt flag is set in the FDCAN_IR register.

When reading from the Tx event FIFO, the Tx event FIFO get index (EFGI[1:0] of FDCAN_TXEFS) has to be added twice to the Tx event FIFO start address EFSA.

39.3.7 FIFO acknowledge handling

The get indices of Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO are controlled by writing to the corresponding FIFO acknowledge index (see *Section 39.4.23* and *Section 39.4.25*). Writing to the FIFO acknowledge index sets the FIFO get index to the FIFO acknowledge index plus one and thereby updates the FIFO fill level. There are two use cases:

- When only a single element has been read from the FIFO (the one being pointed to by the get index), this get index value is written to the FIFO acknowledge index.
- When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO acknowledge index only once at the end of that read sequence (value = index of the last element read), to update the FIFO get index.

Because the CPU has free access to the FDCAN message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (get index not considered). This might be useful when reading a high priority message from one of the two Rx FIFOs. In this case, the FIFO acknowledge index must not be written because this would set the get index to a wrong position and alter the FIFO fill level. In this case, some of the older FIFO elements would be lost.

Note: The application has to ensure that a valid value is written to the FIFO acknowledge index. The FDCAN does not check for erroneous values.

39.3.8 FDCAN Rx FIFO element

Two Rx FIFOs are configured in the message RAM. Each Rx FIFO section can be configured to store up to three received messages. The structure of an Rx FIFO element is described in *Table 323*. The description is provided in *Table 324*.

Table 323. Rx FIFO element

<table>
<tr><th>Bit</th><th colspan="4">31 24</th><th colspan="4">23 16</th><th>15 8</th><th>7 0</th></tr>
<tr><td>R0</td><td>ESI</td><td>XTD</td><td>RTR</td><td colspan="6">ID[28:0]</td></tr>
<tr><td>R1</td><td>ANMF</td><td colspan="3">FIDX[6:0]</td><td>Res.</td><td>FDF</td><td>BRS</td><td>DLC[3:0]</td><td colspan="2">RXTS[15:0]</td></tr>
<tr><td>R2</td><td colspan="4">DB3[7:0]</td><td colspan="4">DB2[7:0]</td><td>DB1[7:0]</td><td>D[7:0]</td></tr>
<tr><td>R3</td><td colspan="4">DB7[7:0]</td><td colspan="4">DB6[7:0]</td><td>DB5[7:0]</td><td>DB4[7:0]</td></tr>
<tr><td>⋮</td><td colspan="4">⋮</td><td colspan="4">⋮</td><td colspan="2">⋮</td></tr>
<tr><td>Rn</td><td colspan="4">DBm[7:0]</td><td colspan="4">DBm-1[7:0]</td><td>DBm-2[7:0]</td><td>DBm-3[7:0]</td></tr>
</table>

The element size configured for storage of CAN FD messages is set to 64-byte data field.

Table 324. Rx FIFO element description

Field	Description
R0 Bit 31 ESI	Error state indicator – 0: Transmitting node is error active – 1: Transmitting node is error passive
R0 Bit 30 XTD	Extended identifier Signals to the host whether the received frame has a standard or extended identifier. – 0: 11-bit standard identifier – 1: 29-bit extended identifier

Table 324. Rx FIFO element description (continued)

Field	Description
R0 Bit 29 RTR	Remote transmission request Signals to the host whether the received frame is a data frame or a remote frame. – 0: Received frame is a data frame – 1: Received frame is a remote frame
R0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 Bit 31 ANMF	Accepted non-matching frame Acceptance of non-matching frames can be enabled via ANFS[1:0] and ANFE[1:0] bitfield of FDCAN_RXGFC. – 0: Received frame matching filter index FIDX – 1: Received frame did not match any Rx filter element
R1 Bits 30:24 FIDX[6:0]	Filter index 0-27=Index of matching Rx acceptance filter element (invalid if ANMF = 1). Range: 0 to LSS[4:0] - 1 or LSE[3:0] - 1 in FDCAN_RXGFC.
R1 Bit 21 FDF	FD format – 0: Standard frame format – 1: FDCAN frame format (new DLC-coding and CRC)
R1 Bit 20 BRS	Bit rate switch – 0: Frame received without bit rate switching – 1: Frame received with bit rate switching
R1 Bits 19:16 DLC[3:0]	Data length code – 0-8: Classic CAN + CAN FD: received frame has 0-8 data bytes – 9-15: Classic CAN: received frame has 8 data bytes – 9-15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
R1 Bits 15:0 RXTS[15:0]	Rx timestamp Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the timestamp counter prescaler TCP[3:0] of FDCAN_TSCC.
R2 Bits 31:24 DB3[7:0]	Data byte 3
R2 Bits 23:16 DB2[7:0]	Data byte 2
R2 Bits 15:8 DB1[7:0]	Data byte 1
R2 Bits 7:0 D[7:0]	Data byte 0
R3 Bits 31:24 DB7[7:0]	Data byte 7
R3 Bits 23:16 DB6[7:0]	Data byte 6

Table 324. Rx FIFO element description (continued)

Field	Description
R3 Bits 15:8 DB5[7:0]	Data byte 5
R3 Bits 7:0 DB4[7:0]	Data byte 4
⋮	⋮
Rn Bits 31:24 DBm[7:0]	Data byte m
Rn Bits 23:16 DBm-1[7:0]	Data byte m-1
Rn Bits 15:8 DBm-2[7:0]	Data byte m-2
Rn Bits 7:0 DBm-3[7:0]	Data byte m-3

39.3.9 FDCAN Tx buffer element

The Tx buffers section (three elements) can be configured to hold Tx FIFO or Tx queue. The Tx handler distinguishes between Tx FIFO and Tx queue using the Tx buffer configuration TFQM bit of the FDCAN_TXBC register. The element size is configured for storage of CAN FD messages with up to 64-byte data.

Table 325. Tx buffer and FIFO element

Bit	31			24	23					16	15	8	7	0
T0	ESI	XTD	RTR	ID[28:0]										
T1	MM[7:0]				EFC	Res.	FDF	BRS	DLC[3:0]		Res.			
T2	DB3[7:0]				DB2[7:0]						DB1[7:0]		D[7:0]	
T3	DB7[7:0]				DB6[7:0]						DB5[7:0]		DB4[7:0]	
⋮	⋮				⋮						⋮			
Tn	DBm[7:0]				DBm-1[7:0]						DBm-2[7:0]		DBm-3[7:0]	

Table 326. Tx buffer element description

Field	Description
T0 Bit 31 ESI(1)	Error state indicator – 0: ESI bit in CAN FD format depends only on error passive flag – 1: ESI bit in CAN FD format transmitted recessive
T0 Bit 30 XTD	Extended identifier – 0: 11-bit standard identifier – 1: 29-bit extended identifier

Table 326. Tx buffer element description (continued)

Field	Description
T0 Bit 29 RTR(2)	Remote transmission request – 0: Transmit data frame – 1: Transmit remote frame
T0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
T1 Bits 31:24 MM[7:0]	Message marker Written by CPU during Tx buffer configuration. Copied into Tx event FIFO element for identification of Tx message status.
T1 Bit 23 EFC	Event FIFO control – 0: Do not store Tx events – 1: Store Tx events
T1 Bit 21 FDF	FD format – 0: Frame transmitted in classic CAN format – 1: Frame transmitted in CAN FD format
T1 Bit 20 BRS(3)	Bit rate switching – 0: CAN FD frames transmitted without bit rate switching – 1: CAN FD frames transmitted with bit rate switching
T1 Bits 19:16 DLC[3:0]	Data length code – 0 - 8: Classic CAN + CAN FD: received frame has 0-8 data bytes – 9 - 15: Classic CAN: received frame has 8 data bytes – 9 - 15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
T2 Bits 31:24 DB3[7:0]	Data byte 3
T2 Bits 23:16 DB2[7:0]	Data byte 2
T2 Bits 15:8 DB1[7:0]	Data byte 1
T2 Bits 7:0 D[7:0]	Data byte 0
T3 Bits 31:24 DB7[7:0]	Data byte 7
T3 Bits 23:16 DB6[7:0]	Data byte 6
T3 Bits 15:8 DB5[7:0]	Data byte 5
T3 Bits 7:0 DB4[7:0]	Data byte 4
⋮	⋮

Table 326. Tx buffer element description (continued)

Field	Description
Tn Bits 31:24 DBm[7:0]	Data byte m
Tn Bits 23:16 DBm-1[7:0]	Data byte m-1
Tn Bits 15:8 DBm-2[7:0]	Data byte m-2
Tn Bits 7:0 DBm-3[7:0]	Data byte m-3

1. The ESI bit of the transmit buffer is OR-ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node can optionally transmit the ESI bit recessive, but an error passive node always transmits the ESI bit recessive.
2. When RTR = 1, the FDCAN transmits a remote frame according to ISO11898-1, even if the transmission in CAN FD format is enabled by the FDOE bit of the FDCAN_CCCR.
3. Bits ESI, FDF, and BRS are only evaluated when CAN FD operation is enabled by setting the FDOE bit of the FDCAN_CCCR. Bit BRS is only evaluated when in addition BRSE bit is set in FDCAN_CCCR.

39.3.10 FDCAN Tx event FIFO element

Each element stores information about transmitted messages. By reading the Tx event, FIFO the host CPU gets this information in the order that the messages were transmitted. Status information about the Tx event FIFO can be obtained from FDCAN_TXEFS register.

Table 327. Tx event FIFO element

Bit	31 24	23 16	15 8	7 0
E0	ESI, XTD, RTR, ID[28:0]			
E1	MM[7:0]	ET[1:0], EDL, BRS, DLC[3:0]	TXTS[15:0]	

Table 328. Tx event FIFO element description

Field	Description
E0 Bit 31 ESI	Error state indicator – 0: Transmitting node is error active – 1: Transmitting node is error passive
E0 Bit 30 XTD	Extended identifier – 0: 11-bit standard identifier – 1: 29-bit extended identifier
E0 Bit 29 RTR	Remote transmission request – 0: Transmit data frame – 1: Transmit remote frame
E0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].

Table 328. Tx event FIFO element description (continued)

Field	Description
E1 Bits 31:24 MM[7:0]	Message marker Copied from Tx buffer into Tx event FIFO element for identification of Tx message status.
E1 Bits 23:22 EFC	Event type – 00: Reserved – 01: Tx event – 10: Transmission in spite of cancellation (always set for transmissions in DAR mode) – 11: Reserved
E1 Bit 21 EDL	Extended data length – 0: Standard frame format – 1: FDCAN frame format (new DLC-coding and CRC)
E1 Bit 20 BRS	Bit rate switching – 0: Frame transmitted without bit rate switching – 1: Frame transmitted with bit rate switching
T1 Bits 19:16 DLC[3:0]	Data length code 0 - 8: Frame with 0-8 data bytes transmitted 9 - 15: Frame with 8 data bytes transmitted
E1 Bits 15:0 TXTS[15:0]	Tx Timestamp Timestamp counter value captured on start of frame transmission. Resolution depending on configuration of the timestamp counter prescaler TCP[3:0] of FDCAN_TSCC.

39.3.11 FDCAN standard message ID filter element

Up to 28 filter elements can be configured for 11-bit standard IDs. When accessing a standard message ID filter element, its address is the filter list standard start address FLSSA plus the index of the filter element (0 … 27).

Table 329. Standard message ID filter element

Bit	31		24	23	16	15	8	7	0
S0	SFT[1:0]	SFEC[2:0]		SFID1[10:0]		Res.		SFID2[10:0]	

Table 330. Standard message ID filter element field description

Field	Description
Bit 31:30 SFT[1:0](1)	Standard filter type – 00: Range filter from SFID1 to SFID2 – 01: Dual ID filter for SFID1 or SFID2 – 10: Classic filter: SFID1 = filter, SFID2 = mask – 11: Filter element disabled
Bit 29:27 SFEC[2:0]	Standard filter element configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC[2:0] = 100, 101 or 110 a match sets interrupt flag IR.HPM and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Not used
Bits 26:16 SFID1[10:0]	Standard filter ID 1 First ID of standard ID filter element.
Bits 10:0 SFID2[10:0]	Standard filter ID 2 Second ID of standard ID filter element.

1. With SFT[1:0] = 11 the filter element is disabled and the acceptance filtering continues (same behavior as with SFEC[2:0] = 000).

Note: In case a reserved value is configured, the filter element is considered disabled.

39.3.12 FDCAN extended message ID filter element

Up to eight filter elements can be configured for 29-bit extended IDs. When accessing an extended message ID filter element, its address is the filter list extended start address FLESA plus twice the index of the filter element (0 … 7).

Table 331. Extended message ID filter element

Bit	31		24	23 16	15 8	7 0
F0	EFEC[2:0]		EFID1[28:0]			
F1	EFT[1:0]	Res.	EFID2[28:0]			

Table 332. Extended message ID filter element field description

Field	Description
F0 Bits 31:29 EFEC[2:0]	Extended filter element configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC[2:0] = 100, 101 or 110 a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Not used
F0 Bits 28:0 EFID1[28:0]	Extended filter ID 1 First ID of extended ID filter element. When filtering for Rx FIFO, this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only XIDAM masking mechanism.
F1 Bits 31:30 EFT[1:0]	Extended filter type – 00: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID) – 01: Dual ID filter for EF1ID or EF2ID – 10: Classic filter: EF1ID = filter, EF2ID = mask – 11: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID), XIDAM mask not applied
F1 Bit 29	Not used
F1 Bits 28:0 EFID2[28:0]	Extended filter ID 2 Second ID of extended ID filter element.

39.4 FDCAN registers

39.4.1 FDCAN core release register (FDCAN_CREL)

Address offset: 0x0000

Reset value: 0x3214 1218

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REL[3:0]				STEP[3:0]				SUBSTEP[3:0]				YEAR[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
MON[7:0]								DAY[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **REL[3:0]**: 3

Bits 27:24 **STEP[3:0]**: 2

Bits 23:20 **SUBSTEP[3:0]**: 1

Bits 19:16 **YEAR[3:0]**: 4

Bits 15:8 **MON[7:0]**: 12

Bits 7:0 **DAY[7:0]**: 18

39.4.2 FDCAN endian register (FDCAN_ENDN)

Address offset: 0x0004

Reset value: 0x8765 4321

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETV[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
ETV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ETV[31:0]**: Endianness test value

The endianness test value is 0x8765 4321.

Note: The register read must give the reset value to ensure no endianness issue.

39.4.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP)

Address offset: 0x000C

Reset value: 0x0000 0A33

This register is only writable if the CCE and INIT bits of the FDCAN_CCCR are set. The CAN time quantum can be programmed in the range of 1 to 32 FDCAN clock periods: t_q = (DBRP[4:0] + 1) FDCAN clock periods.

DTSEG1[4:0] is the sum of PROP_SEG and PHASE_SEG1. DTSEG2[3:0] is PHASE_SEG2. Therefore, the length of the bit time is
(programmed values) × [DTSEG1[4:0] + DTSEG2[3:0] + 3] × t_q or
(functional values) × [SYNC_SEG + PROP_SEG + PHASE_SEG1 + PHASE_SEG2] × t_q.

The information processing time (IPT) is 0, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]				
								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.	DTSEG1[4:0]					DTSEG2[3:0]				DSJW[3:0]			
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TDC**: Transceiver delay compensation
0: Transceiver delay compensation disabled
1: Transceiver delay compensation enabled

Bits 22:21 Reserved, must be kept at reset value.

Bits 20:16 **DBRP[4:0]**: Data bit rate prescaler
The value by which the oscillator frequency is divided to generate the bit time quanta. The bit time is built up from a multiple of this quantum. Valid values for the baud rate prescaler are 0 to 31. The hardware interpreters this value as the value programmed plus 1.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DTSEG1[4:0]**: Data time segment before sample point
Valid values are 0 to 31. The value used by the hardware is the one programmed, incremented by 1, that is t_{BS1} = (DTSEG1[4:0] + 1) × t_q.

Bits 7:4 **DTSEG2[3:0]**: Data time segment after sample point
Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1, i.e. t_{BS2} = (DTSEG2[3:0] + 1) × t_q.

Bits 3:0 **DSJW[3:0]**: Synchronization jump width
Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1: t_{SJW} = (DSJW[3:0] + 1) × t_q.

Note: With an FDCAN clock of 8 MHz, the reset value 0x0000 0A33 configures the FDCAN for a fast bit rate of 500 kbit/s.

The data phase bit rate must be higher than or equal to the nominal bit rate.

39.4.4 FDCAN test register (FDCAN_TEST)

Write access to this register is enabled by setting the TEST bit of the FDCAN_CCCR register. All register functions are set to their reset values when this bit is cleared.

Loop-back mode and software control of Tx pin FDCANx_TX are hardware test modes. Programming TX[1:0] differently from 00 can disturb the message transfer on the CAN bus.

Address offset: 0x0010

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.	RX	TX[1:0]		LBCK	Res.	Res.	Res.	Res.
								r	rw	rw	rw				

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **RX**: Receive pin

This bit is used to monitor the actual value of FDCANx_RX. It is synchronized with the FDCANx_RX pin, so it is set after reset if the FDCAN is connected to a network.

0: The CAN bus is dominant (FDCANx_RX = 0)

1: The CAN bus is recessive (FDCANx_RX = 1)

Bits 6:5 **TX[1:0]**: Control of transmit pin

00: Reset value, FDCANx_TX TX is controlled by the CAN core, updated at the end of the CAN bit time

01: Sample point can be monitored at pin FDCANx_TX

10: Dominant (0) level at pin FDCANx_TX

11: Recessive (1) at pin FDCANx_TX

Bit 4 **LBCK**: Loop-back mode

0: Reset value, loop-back mode is disabled

1: Loop-back mode is enabled (see *Power-down (Sleep mode)*)

Bits 3:0 Reserved, must be kept at reset value.

39.4.5 FDCAN RAM watchdog register (FDCAN_RWD)

The RAM watchdog monitors the READY output of the message RAM. A message RAM access starts the message RAM watchdog counter with the value configured through the WDC[7:0] bitfield of the FDCAN_RWD register.

The counter is reloaded with WDC[7:0] when the message RAM signals successful completion by activating its READY output. In case there is no response from the message RAM until the counter has counted down to 0, the counter stops, and the interrupt flag WDI is set in the FDCAN_IR register. The RAM watchdog counter is clocked by the fdcan_pclk clock.

Address offset: 0x0014

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
WDV[7:0]								WDC[7:0]							
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **WDV[7:0]**: Watchdog value
Actual message RAM watchdog counter value.

Bits 7:0 **WDC[7:0]**: Watchdog configuration
Start value of the message RAM watchdog counter. With the reset value of 00, the counter is disabled.
This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

39.4.6 FDCAN CC control register (FDCAN_CCCR)

Address offset: 0x0018

Reset value: 0x0000 0001

For details about setting and clearing single bits, see *Software initialization*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	r	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **NISO**: Non-ISO operation
If this bit is set, the FDCAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0.
0: CAN FD frame format according to ISO11898-1
1: CAN FD frame format according to Bosch CAN FD Specification V1.0

Bit 14 **TXP:** Transmit pause enable
If this bit is set, the FDCAN pauses for two CAN bit times before starting the next transmission after successfully transmitting a frame.
0: Disabled
1: Enabled

Bit 13 **EFBI**: Edge filtering during bus integration
0: Edge filtering disabled
1: Two consecutive dominant t_q required to detect an edge for hard synchronization

Bit 12 **PXHD**: Protocol exception handling disable
0: Protocol exception handling enabled
1: Protocol exception handling disabled

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **BRSE**: FDCAN bit rate switching
0: Bit rate switching for transmissions disabled
1: Bit rate switching for transmissions enabled

Bit 8 **FDOE**: FD operation enable
0: FD operation disabled
1: FD operation enabled

Bit 7 **TEST**: Test mode enable
0: Normal operation, FDCAN_TEST holds reset values
1: Test mode, write access to FDCAN_TEST enabled

Bit 6 **DAR**: Disable automatic retransmission
0: Automatic retransmission of messages not transmitted successfully enabled
1: Automatic retransmission disabled

Bit 5 **MON**: Bus monitoring mode
This bit can only be set by software when both CCE and INIT are set. The bit can be cleared by the host at any time.
0: Bus monitoring mode disabled
1: Bus monitoring mode enabled

Bit 4 **CSR**: Clock stop request
0: No clock stop requested
1: Clock stop requested. When clock stop is requested, first INIT and then CSA is set after all pending transfer requests have been completed and the CAN bus is idle.

Bit 3 **CSA**: Clock stop acknowledge
0: No clock stop acknowledged
1: FDCAN can be set in power-down by stopping APB clock and kernel clock.

Bit 2 **ASM**: ASM restricted operation mode
The restricted operation mode is intended for applications that adapt themselves to different CAN bit rates. The application tests different bit rates and leaves the restricted operation mode after it has received a valid frame. In the optional restricted operation mode the node is able to transmit and receive data and remote frames and it gives acknowledge to valid frames, but it does not send active error frames or over.load frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus-idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. Bit ASM can only be set by software when both CCE and INIT are set. The bit can be cleared by the software at any time.
0: Normal CAN operation
1: Restricted operation mode active

Bit 1 **CCE**: Configuration change enable
0: The CPU has no write access to the protected configuration registers.
1: The CPU has write access to the protected configuration registers (while INIT set in FDCAN_CCCR).

Bit 0 **INIT**: Initialization
0: Normal operation
1: Initialization started

Note: Due to the synchronization mechanism between the two clock domains, there can be a delay until the value written to INIT can be read back. Therefore, the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

39.4.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP)

Address offset: 0x001C

Reset value: 0x0600 0A03

This register is only writable if the CCE and INIT bits of the FDCAN_CCCR register are both set. The CAN bit time can be programed in the range of 4 to 81 × t_q. The CAN time quantum can be programmed in the range of 1 to 1024 FDCAN kernel clock periods:
t_q = (BRP + 1) × FDCAN clock period fdcan_ker_ck.

NTSEG1[7:0] is the sum of PROP_SEG and PHASE_SEG1. NTSEG2[6:0] is PHASE_SEG2. Therefore, the length of the bit time is
(programmed values) × [NTSEG1[7:0] + NTSEG2[6:0] + 3] × t_q or
(functional values) × [SYNC_SEG + PROP_SEG + PHASE_SEG1 + PHASE_SEG2] × t_q.

The information processing time (IPT) is 0, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSJW[6:0]							NBRP[8:0]								
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
NTSEG1[7:0]								Res.	NTSEG2[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:25 **NSJW[6:0]**: Nominal (re)synchronization jump width

Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that the used value is the one programmed incremented by one.

This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 24:16 **NBRP[8:0]**: Bit rate prescaler

Value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quantum. Valid values are 0 to 511. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:8 **NTSEG1[7:0]**: Nominal time segment before sample point

Valid values are 0 to 255. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

This bitfield is write-protected write (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **NTSEG2[6:0]**: Nominal time segment after sample point

Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

Note: With a CAN kernel clock of 48 MHz, the reset value of 0x0600 0A03 configures the FDCAN for a bit rate of 3 Mbit/s.

39.4.8 FDCAN timestamp counter configuration register (FDCAN_TSCC)

Address offset: 0x0020

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.	TCP[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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSS[1:0]	
														rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **TCP[3:0]**: Timestamp counter prescaler

Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1…16].

The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

In CAN FD mode, the internal timestamp counter TCP does not provide a constant time base due to the different CAN bit times between arbitration phase and data phase. Thus CAN FD requires an external counter for timestamp generation (TSS[1:0] = 10).

This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:2 Reserved, must be kept at reset value.

Bits 1:0 **TSS[1:0]**: Timestamp select

00: Timestamp counter value always 0x0000

01: Timestamp counter value incremented according to TCP

10: External timestamp counter from TIM3 value (tim3_cnt[0:15])

11: Same as 00.

These bits are write-protected write (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

39.4.9 FDCAN timestamp counter value register (FDCAN_TSCV)

Address offset: 0x0024

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
TSC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TSC[15:0]**: Timestamp counter

The internal/external timestamp counter value is captured on start of frame (both Rx and Tx). When TSS[1:0] = 01 in FDCAN_TSCC, the timestamp counter is incremented in multiples of CAN bit times [1 … 16] depending on the configuration of TCP[3:0] in FDCAN_TSCC. A wrap around sets the TSW interrupt flag in FDCAN_IR. Write access resets the counter to 0.
When TSS[1:0] = 10, TSC[15:0] reflects the external timestamp counter value. A write access has no impact.

Note: A "wrap around" is a change of the timestamp counter value from non-0 to 0 that is not caused by write access to FDCAN_TSCV.

39.4.10 FDCAN timeout counter configuration register (FDCAN_TOCC)

Address offset: 0x0028

Reset value: 0xFFFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOP[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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOS[1:0]		ETOC
													rw	rw	rw

Bits 31:16 **TOP[15:0]**: Timeout period

Start value of the timeout counter (down-counter). Configures the timeout period.
This bitfield is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **TOS[1:0]**: Timeout select

When operating in continuous mode, a write to FDCAN_TOCV presets the counter to the value configured by TOP[15:0] in FDCAN_TOCC and continues down-counting. When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOP[15:0]. Down-counting is started when the first FIFO element is stored.
00: Continuous operation
01: Timeout controlled by Tx event FIFO
10: Timeout controlled by Rx FIFO 0
11: Timeout controlled by Rx FIFO 1
This bitfield is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 0 **ETOC**: Timeout counter enable

0: Timeout counter disabled
1: Timeout counter enabled
This bit is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

For more details, see *Timeout counter*.

39.4.11 FDCAN timeout counter value register (FDCAN_TOCV)

Address offset: 0x002C

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.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TOC[15:0]**: Timeout counter

The timeout counter is decremented in multiples of CAN bit times [1 … 16] depending on the configuration of the TCP[3:0] bitfield of the FDCAN_TSCC register. When decremented to 0, the TOO interrupt flag is set in FDCAN_IR and the timeout counter is stopped. Start and reset/restart conditions are configured via TOS[1:0] in FDCAN_TOCC.

39.4.12 FDCAN error counter register (FDCAN_ECR)

Address offset: 0x0040

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.	CEL[7:0]							
								rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RP	REC[6:0]							TEC[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **CEL[7:0]**: CAN error logging

The counter is incremented each time when a CAN protocol error causes the transmit error counter or the receive error counter to be incremented. It is reset by read access to CEL[7:0]. The counter stops at 0xFF; the next increment of TEC[7:0] or REC[6:0] sets the ELO interrupt flag in FDCAN_IR.

Access type is rc_r: cleared on read.

Bit 15 **RP**: Receive error passive

0: The receive error counter is below the error passive level of 128.

1: The receive error counter has reached the error passive level of 128.

Bits 14:8 **REC[6:0]:** Receive error counter

Actual state of the receive error counter, values between 0 and 127.

Bits 7:0 **TEC[7:0]**: Transmit error counter

Actual state of the transmit error counter, values between 0 and 255.

When the ASM bit of the FDCAN_CCCR is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL[7:0] is still incremented.

39.4.13 FDCAN protocol status register (FDCAN_PSR)

Address offset: 0x0044

Reset value: 0x0000 0707

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]						
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PXE	REDL	RBRS	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	rc_r	rc_r	rc_r	rc_r	rs	rs	rs	r	r	r	r	r	rs	rs	rs

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **TDCV[6:0]**: Transmitter delay compensation value

Position of the secondary sample point, defined by the sum of the measured delay from FDCAN_TX to FDCAN_RX and TDCO[6:0] in FDCAN_TDCR. The SSP position is, in the data phase, the number of minimum time quanta (mt_q) between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 × mt_q.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **PXE**: Protocol exception event

0: No protocol exception event occurred since last read access

1: Protocol exception event occurred

Bit 13 **REDL**: Received FDCAN message

This bit is set independent of acceptance filtering.

0: Since this bit was cleared by the CPU, no FDCAN message has been received.

1: Message in FDCAN format with EDL flag set has been received.

Access type is rc_r: cleared on read.

Bit 12 **RBRS:** BRS flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.

0: Last received FDCAN message did not have its BRS flag set.

1: Last received FDCAN message had its BRS flag set.

Access type is rc_r: cleared on read.

Bit 11 **RESI**: ESI flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.

0: Last received FDCAN message did not have its ESI flag set.

1: Last received FDCAN message had its ESI flag set.

Access type is rc_r: cleared on read.

Bits 10:8 **DLEC[2:0]**: Data last error code

Type of last error that occurred in the data phase of a FDCAN format frame with its BRS flag set. Coding is the same as for LEC[2:0]. This field is cleared when a FDCAN format frame with its BRS flag set has been transferred (reception or transmission) without error.

Access type is rs: set on read.

Bit 7 **BO**: Bus-off status

0: The FDCAN is not in bus-off state.

1: The FDCAN is in bus-off state.

Bit 6 **EW**: Warning status

0: Both error counters are below the error-warning limit of 96.

1: At least one of error counter has reached the error-warning limit of 96.

Bit 5 **EP**: Error passive

0: The FDCAN is in the error-active state. It normally takes part in bus communication and sends an active error flag when an error has been detected.

1: The FDCAN is in the error-passive state.

Bits 4:3 **ACT[1:0]**: Activity

Monitors the module's CAN communication state.

00: Synchronizing: node is synchronizing on CAN communication.

01: Idle: node is neither receiver nor transmitter.

10: Receiver: node is operating as receiver.

11: Transmitter: node is operating as transmitter.

Bits 2:0 **LEC[2:0]:** Last error code

LEC[2:0] indicates the type of the last error to occur on the CAN bus. This bitfield is cleared when a message has been transferred (reception or transmission) without error.

000: No error occurred since LEC[2:0] has been cleared by successful reception or transmission.

001: Stuff error. More than five equal bits in a sequence have occurred in a part of a received message where this is not allowed.

010: Form error. A fixed format part of a received frame has the wrong format.

011: Ack error. The message transmitted by the FDCAN was not acknowledged by another node.

100: Bit1 error. During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.

101: Bit0 error. During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During bus-off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the bus-off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).

110: CRC error. The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.

111: No change. Any read access to the protocol status register reinitializes LEC[2:0] to 7. When the LEC[2:0] shows the value 7, no CAN bus event was detected since the last CPU read access to the protocol status register.

Access type is rs: set on read.

Note: When a frame in FDCAN format has reached the data phase with the BRS flag set, the next CAN event (error or valid frame) is shown in DLEC[2:0] instead of LEC[2:0]. An error in a fixed stuff bit of an FDCAN CRC sequence is shown as a form error, not as a stuff error.

The bus-off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or clearing the INIT bit of the FDCAN_CCCR register. If the device enters bus-off, it sets the INIT bit of its own, stopping all bus activities. Once INIT has been cleared by the CPU, the device waits for 129 occurrences of bus-idle (129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the bus-off recovery sequence, the error management counters are reset. During the waiting time after clearing INIT, each time a sequence of 11 recessive bits has been monitored, a bit0 error code is written to LEC[2:0] of FDCAN_PSR, enabling the CPU to check up whether the CAN bus is

stuck at dominant or continuously disturbed, and to monitor the bus-off recovery sequence. The REC[6:0] bitfield of the FDCAN_ECR register is used to count these sequences.

39.4.14 FDCAN transmitter delay compensation register (FDCAN_TDCR)

Address offset: 0x0048

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.	TDCO[6:0]							Res.	TDCF[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **TDCO[6:0]**: Transmitter delay compensation offset

Offset value defining the distance between the measured delay from FDCAN_TX to FDCAN_RX and the secondary sample point. Valid values are 0 to 127 × mt_q.

This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **TDCF[6:0]**: Transmitter delay compensation filter window length

Defines the minimum value for the SSP position, dominant edges on FDCAN_RX that would result in an earlier SSP position are ignored for transmitter delay measurements.

This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

39.4.15 FDCAN interrupt register (FDCAN_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the host clears them. A flag is cleared by writing 1 to the corresponding bit position.

Writing 0 has no effect. A hard reset clears the register. The configuration of FDCAN_IE controls whether an interrupt is generated. The configuration of FDCAN_ILS controls on which interrupt line an interrupt is signaled.

Address offset: 0x0050

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.	ARA	PED	PEA	WDI	BO	EW	EP	ELO
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOO	MRAF	TSW	TEFL	TEFF	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1N	RF0L	RF0F	RF0N
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 31:24 Reserved, must be kept at reset value.

Bit 23 **ARA**: Access to reserved address
0: No access to reserved address occurred
1: Access to reserved address occurred

Bit 22 **PED**: Protocol error in data phase (data bit time is used)
0: No protocol error in data phase
1: Protocol error in data phase detected (DLEC[2:0] different from 0 and 7 in FDCAN_PSR)

Bit 21 **PEA**: Protocol error in arbitration phase (nominal bit time is used)
0: No protocol error in arbitration phase
1: Protocol error in arbitration phase detected (LEC[2:0] different from 0 and 7 in FDCAN_PSR)

Bit 20 **WDI**: Watchdog interrupt
0: No message RAM watchdog event occurred
1: Message RAM watchdog event due to missing READY

Bit 19 **BO**: Bus-off status
0: Bus-off status unchanged
1: Bus-off status changed

Bit 18 **EW**: Warning status
0: Error-warning status unchanged
1: Error-warning status changed

Bit 17 **EP**: Error passive
0: Error-passive status unchanged
1: Error-passive status changed

Bit 16 **ELO**: Error logging overflow
0: CAN error logging counter did not overflow.
1: Overflow of CAN error logging counter occurred.

Bit 15 **TOO**: Timeout occurred
0: No timeout
1: Timeout reached

Bit 14 **MRAF**: Message RAM access failure
The flag is set when the Rx handler:
- has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx handler starts processing of the following message.
- was unable to write a message to the message RAM. In this case message storage is aborted.

In both cases the FIFO put index is not updated. The partly stored message is overwritten when the next message is stored to this location.
The flag is also set when the Tx handler was not able to read a message from the message RAM in time. In this case message transmission is aborted. In case of a Tx handler access failure, the FDCAN is switched into restricted operation mode (see *Restricted operation mode*). To leave restricted operation mode, the host CPU has to clear the ASM of the FDCAN_CCCR register.
0: No message RAM access failure occurred
1: Message RAM access failure occurred

Bit 13 **TSW**: Timestamp wraparound
0: No timestamp counter wrap-around
1: Timestamp counter wrapped around

Bit 12 **TEFL**: Tx event FIFO element lost
0: No Tx event FIFO element lost
1: Tx event FIFO element lost

Bit 11 **TEFF**: Tx event FIFO full
0: Tx event FIFO Not full
1: Tx event FIFO full

Bit 10 **TEFN**: Tx event FIFO new entry
0: Tx event FIFO unchanged
1: Tx handler wrote Tx event FIFO element.

Bit 9 **TFE**: Tx FIFO empty
0: Tx FIFO non-empty
1: Tx FIFO empty

Bit 8 **TCF**: Transmission cancellation finished
0: No transmission cancellation finished
1: Transmission cancellation finished

Bit 7 **TC**: Transmission completed
0: No transmission completed
1: Transmission completed

Bit 6 **HPM**: High-priority message
0: No high-priority message received
1: High-priority message received

Bit 5 **RF1L**: Rx FIFO 1 message lost
0: No Rx FIFO 1 message lost
1: Rx FIFO 1 message lost

Bit 4 **RF1F**: Rx FIFO 1 full
0: Rx FIFO 1 not full
1: Rx FIFO 1 full

Bit 3 **RF1N**: Rx FIFO 1 new message
0: No new message written to Rx FIFO 1
1: New message written to Rx FIFO 1

Bit 2 **RF0L**: Rx FIFO 0 message lost
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost

Bit 1 **RF0F**: Rx FIFO 0 full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bit 0 **RF0N**: Rx FIFO 0 new message
0: No new message written to Rx FIFO 0
1: New message written to Rx FIFO 0

39.4.16 FDCAN interrupt enable register (FDCAN_IE)

The settings in the interrupt enable register determine which status changes in the interrupt register are signaled on an interrupt line.

Address offset: 0x0054

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.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1NE	RF0LE	RF0FE	RF0NE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ARAE**: Access to reserved address enable

Bit 22 **PEDE**: Protocol error in data phase enable

Bit 21 **PEAE**: Protocol error in arbitration phase enable

Bit 20 **WDIE**: Watchdog interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 19 **BOE**: Bus-off status
0: Interrupt disabled
1: Interrupt enabled

Bit 18 **EWE**: Warning status interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **EPE**: Error passive interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **ELOE**: Error logging overflow interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 15 **TOOE**: Timeout occurred interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 14 **MRAFE:** Message RAM access failure interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 13 **TSWE**: Timestamp wraparound interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 12 **TEFLE**: Tx event FIFO element lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 11 **TEFFE**: Tx event FIFO full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 10 **TEFNE**: Tx event FIFO new entry interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 9 **TFEE**: Tx FIFO empty interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 8 **TCFE**: Transmission cancellation finished interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 7 **TCE**: Transmission completed interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 6 **HPME**: High-priority message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 5 **RF1LE**: Rx FIFO 1 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 4 **RF1FE**: Rx FIFO 1 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 3 **RF1NE**: Rx FIFO 1 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 2 **RF0LE**: Rx FIFO 0 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 1 **RF0FE**: Rx FIFO 0 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **RF0NE**: Rx FIFO 0 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

39.4.17 FDCAN interrupt line select register (FDCAN_ILS)

This register assigns an interrupt generated by a specific group of interrupt flags from the interrupt register to one of the two module interrupt lines. For interrupt generation, the respective interrupt line has to be enabled via the EINT0 and EINT1 bit of the FDCAN_ILE register.

Address offset: 0x0058

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.	PERR	BERR	MISC	TFERR	SMSG	RXFIFO1	RXFIFO0
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **PERR:** Protocol error grouping the following interruption
ARAL: Access to reserved address line
PEDL: Protocol error in data phase line
PEAL: Protocol error in arbitration phase line
WDIL: Watchdog interrupt line
BOL: Bus-off status
EWL: Warning status interrupt line

Bit 5 **BERR:** Bit and line error grouping the following interruption
EPL Error passive interrupt line
ELOL: Error logging overflow interrupt line

Bit 4 **MISC:** Interrupt regrouping the following interruption
TOOL: Timeout occurred interrupt line
MRAFL: Message RAM access failure interrupt line
TSWL: Timestamp wraparound interrupt line

Bit 3 **TFERR:** Tx FIFO ERROR grouping the following interruption
TEFLL: Tx event FIFO element lost interrupt line
TEFFL: Tx event FIFO full interrupt line
TEFNL: Tx event FIFO new entry interrupt line
TFEL: Tx FIFO empty interrupt line

Bit 2 **SMSG:** Status message bit grouping the following interruption
TCFL: Transmission cancellation finished interrupt line
TCL: Transmission completed interrupt line
HPML: High-priority message interrupt line

Bit 1 **RXFIFO1:** RX FIFO bit grouping the following interruption
RF1LL: Rx FIFO 1 message lost interrupt line
RF1FL: Rx FIFO 1 full interrupt line
RF1NL: Rx FIFO 1 new message interrupt line

Bit 0 **RXFIFO0:** RX FIFO bit grouping the following interruption
RF0LL: Rx FIFO 0 message lost interrupt line
RF0FL: Rx FIFO 0 full interrupt line
RF0NL: Rx FIFO 0 new message interrupt line

39.4.18 FDCAN interrupt line enable register (FDCAN_ILE)

Each of the two interrupt lines to the CPU can be enabled/disabled separately by programming the EINT0 and EINT1 bits.

Address offset: 0x005C

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.	EINT1	EINT0
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **EINT1**: Enable interrupt line 1
0: Interrupt line fdcan_intr0_it disabled
1: Interrupt line fdcan_intr0_it enabled

Bit 0 **EINT0**: Enable interrupt line 0
0: Interrupt line fdcan_intr1_it disabled
1: Interrupt line fdcan_intr1_it enabled

39.4.19 FDCAN global filter configuration register (FDCAN_RXGFC)

Global settings for message ID filtering. The global filter configuration controls the filter path for standard and extended messages as described in *Figure 510* and *Figure 511*.

Address offset: 0x0080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	LSE[3:0]				Res.	Res.	Res.	LSS[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
Res.	Res.	Res.	Res.	Res.	Res.	F0OM	F1OM	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
						rw	rw			rw	rw	rw	rw	rw	rw

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

Bits 27:24 **LSE[3:0]**: Number of extended filter elements in the list
0: No extended message ID filter
1 to 8: Number of extended message ID filter elements
> 8: Values greater than 8 are interpreted as 8.
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **LSS[4:0]**: Number of standard filter elements in the list
0: No standard message ID filter
1 to 28: Number of standard message ID filter elements
> 28: Values greater than 28 are interpreted as 28.
This bitfield is write protected (P), which means that write access by the bits is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **F0OM**: FIFO 0 operation mode (overwrite or blocking)
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 8 **F1OM**: FIFO 1 operation mode (overwrite or blocking)
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ANFS[1:0]**: Accept Non-matching frames standard
Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
This bitfield is write-protected (P), which means write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 3:2 **ANFE[1:0]**: Accept non-matching frames extended
Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 1 **RRFS**: Reject remote frames standard
0: Filter remote frames with 11-bit standard IDs
1: Reject all remote frames with 11-bit standard IDs
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 0 **RRFE**: Reject remote frames extended
0: Filter remote frames with 29-bit standard IDs
1: Reject all remote frames with 29-bit standard IDs
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

39.4.20 FDCAN extended ID and mask register (FDCAN_XIDAM)

Address offset: 0x0084

Reset value: 0x1FFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EIDM[28:16]												
			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
EIDM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:0 **EIDM[28:0]**: Extended ID mask
For acceptance filtering of extended frames the extended ID AND mask is AND-ed with the message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set, the mask is not active.
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

39.4.21 FDCAN high-priority message status register (FDCAN_HPMS)

This register is updated every time a message ID filter element configured to generate a priority event match. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Address offset: 0x0088

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
FLST	Res.	Res.	FIDX[4:0]					MSI[1:0]		Res.	Res.	Res.	BIDX[2:0]		
r			r	r	r	r	r	r	r				r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **FLST**: Filter list
Indicates the filter list of the matching filter element:
0: Standard filter list
1: Extended filter list

Bits 14:13 Reserved, must be kept at reset value.

Bits 12:8 **FIDX[4:0]**: Filter index
Index of matching filter element.
Range: 0 to LSS[4:0] - 1 or LSE[3:0] - 1 in FDCAN_RXGFC.

Bits 7:6 **MSI[1:0]**: Message storage indicator
00: No FIFO selected
01: FIFO overrun
10: Message stored in FIFO 0
11: Message stored in FIFO 1

Bits 5:3 Reserved, must be kept at reset value.

Bits 2:0 **BIDX[2:0]**: Buffer index
Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = 1.

39.4.22 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S)

Address offset: 0x0090

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.	RF0L	F0F	Res.	Res.	Res.	Res.	Res.	Res.	F0PI[1:0]	
						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.	F0GI[1:0]		Res.	Res.	Res.	Res.	F0FL[3:0]			
						r	r					r	r	r	r

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

Bit 25 **RF0L**: Rx FIFO 0 message lost
This bit is a copy of the RF0L interrupt flag of the FDCAN_IR register. When RF0L is cleared, this bit is also cleared.
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size 0

Bit 24 **F0F**: Rx FIFO 0 full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **F0PI[1:0]**: Rx FIFO 0 put index
Rx FIFO 0 write index pointer.
Range: 0 to 2.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **F0GI[1:0]**: Rx FIFO 0 get index
Rx FIFO 0 read index pointer.
Range: 0 to 2.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **F0FL[3:0]**: Rx FIFO 0 fill level
Number of elements stored in Rx FIFO 0.
Range: 0 to 3.

39.4.23 CAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A)

Address offset: 0x0094

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.	F0AI[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **F0AI[2:0]**: Rx FIFO 0 acknowledge index

After the host has read a message or a sequence of messages from Rx FIFO 0, it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI[2:0]. This sets the Rx FIFO 0 get index (F0GI[1:0] of FDCAN_RXF0S) to F0AI[2:0] + 1 and updates the FIFO 0 fill level (F0FL[3:0] FDCAN_RXF0S).

39.4.24 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S)

Address offset: 0x0098

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.	RF1L	F1F	Res.	Res.	Res.	Res.	Res.	Res.	F1PI[1:0]	
						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.	F1GI[1:0]		Res.	Res.	Res.	Res.	F1FL[3:0]			
						r	r					r	r	r	r

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

Bit 25 **RF1L**: Rx FIFO 1 message lost

This bit is a copy of the RF1L interrupt flag of the FDCAN_IR register. When RF1L is cleared, this bit is also cleared.

0: No Rx FIFO 1 message lost

1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size 0

Bit 24 **F1F**: Rx FIFO 1 full

0: Rx FIFO 1 not full

1: Rx FIFO 1 full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **F1PI[1:0]**: Rx FIFO 1 put index

Rx FIFO 1 write index pointer.

Range: 0 to 2.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **F1GI[1:0]**: Rx FIFO 1 get index
Rx FIFO 1 read index pointer.
Range: 0 to 2.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **F1FL[3:0]**: Rx FIFO 1 fill level
Number of elements stored in Rx FIFO 1.
Range: 0 to 3.

39.4.25 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A)

Address offset: 0x009C

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.	F1AI[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **F1AI[2:0]**: Rx FIFO 1 acknowledge index
After the host has read a message or a sequence of messages from Rx FIFO 1, it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI[2:0]. This sets the Rx FIFO 1 get index (F1GI[1:0] of FDCAN_RXF1S) to F1AI[2:0] + 1 and updates the FIFO 1 fill level (F1FL[3:0] FDCAN_RXF1S).

39.4.26 FDCAN Tx buffer configuration register (FDCAN_TXBC)

Address offset: 0x00C0

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.	TFQM	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 **TFQM**: Tx FIFO/queue mode
0: Tx FIFO operation
1: Tx queue operation.
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 23:0 Reserved, must be kept at reset value.

39.4.27 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS)

The Tx FIFO/queue status is related to the pending Tx requests listed in the FDCAN_TXBRP register. Therefore, the effect of add/cancellation requests can be delayed due to a running Tx scan (FDCAN_TXBRP not yet updated).

Address offset: 0x00C4

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.	TFQF	Res.	Res.	Res.	TFQPI[1:0]	
										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.	TFGI[1:0]		Res.	Res.	Res.	Res.	Res.	TFFL[2:0]		
						r	r						r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **TFQF**: Tx FIFO/queue full
0: Tx FIFO/queue not full
1: Tx FIFO/queue full

Bits 20:18 Reserved, must be kept at reset value.

Bits 17:16 **TFQPI[1:0]**: Tx FIFO/queue put index
Tx FIFO/queue write index pointer, range 0 to 3

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **TFGI[1:0]**: Tx FIFO get index
Tx FIFO read index pointer, range 0 to 3. Read as 0 when Tx queue operation is configured (TFQM = 1 in FDCAN_TXBC)

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **TFFL[2:0]**: Tx FIFO free level
Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 3. Read as 0 when Tx queue operation is configured (TFQM = 1 in FDCAN_TXBC).

39.4.28 FDCAN Tx buffer request pending register (FDCAN_TXBRP)

Address offset: 0x00C8

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.	TRP[2:0]		
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **TRP[2:0]**: Transmission request pending

Each Tx buffer has its own transmission request pending bit. The bits are set via the FDCAN_TXBAR register. The bits are cleared after a requested transmission has completed or has been canceled via the FDCAN_TXBCR register.

After the FDCAN_TXBRP bit has been set, a Tx scan is started to check for the pending Tx request with the highest priority (Tx buffer with lowest message ID).

A cancellation request resets the corresponding transmission request pending bit of the FDCAN_TXBRP register. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are directly cleared after the corresponding FDCAN_TXBRP bit has been cleared.

After a cancellation has been requested, a finished cancellation is signaled via the FDCAN_TXBCF in the following cases:

- after successful transmission together with the corresponding TXBTO bit
- when the transmission has not yet been started at the point of cancellation
- when the transmission has been aborted due to lost arbitration
- when an error occurred during frame transmission

In DAR mode, all transmissions are automatically canceled if they are not successful. The corresponding FDCAN_TXBCF bit is set for all unsuccessful transmissions.

0: No transmission request pending

1: Transmission request pending

Note: FDCAN_TXBRP bits set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx buffer, this add request is canceled immediately. The corresponding FDCAN_TXBRP bit is cleared.

39.4.29 FDCAN Tx buffer add request register (FDCAN_TXBAR)

Address offset: 0x00CC

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.	AR[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **AR[2:0]**: Add request

Each Tx buffer has its own add request bit. Writing a 1 sets the corresponding add request bit; writing a 0 has no impact. This enables the host to set transmission requests for multiple Tx buffers with one write to FDCAN_TXBAR. When no Tx scan is running, the bits are cleared immediately, else the bits remain set until the Tx scan process has completed.

0: No transmission request added

1: Transmission requested added.

Note: If an add request is applied for a Tx buffer with pending transmission request (corresponding FDCAN_TXBRP bit already set), the request is ignored.

39.4.30 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR)

Address offset: 0x00D0

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.	CR[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **CR[2:0]**: Cancellation request

Each Tx buffer has its own cancellation request bit. Writing a 1 sets the corresponding CR bit; writing a 0 has no impact.

This enables the host to set cancellation requests for multiple Tx buffers with one write to FDCAN_TXBCR. The bits remain set until the corresponding FDCAN_TXBRP bit is cleared.

0: No cancellation pending

1: Cancellation pending

39.4.31 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO)

Address offset: 0x00D4

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.	TO[2:0]		
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **TO[2:0]**: Transmission occurred.

Each Tx buffer has its own TO bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a successful transmission. The bits are cleared when a new transmission is requested by writing a 1 to the corresponding bit of register FDCAN_TXBAR.

0: No transmission occurred

1: Transmission occurred

39.4.32 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF)

Address offset: 0x00D8

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.	CF[2:0]		
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **CF[2:0]**: Cancellation finished

Each Tx buffer has its own CF bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a cancellation was requested via FDCAN_TXBCR. In case the corresponding FDCAN_TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are cleared when a new transmission is requested by writing a 1 to the corresponding bit of the FDCAN_TXBAR register.

0: No transmit buffer cancellation

1: Transmit buffer cancellation finished

39.4.33 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE)

Address offset: 0x00DC

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.	TIE[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **TIE[2:0]**: Transmission interrupt enable

Each Tx buffer has its own TIE bit.

0: Transmission interrupt disabled

1: Transmission interrupt enable

39.4.34 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE)

Address offset: 0x00E0

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.	CFIE[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **CFIE[2:0]**: Cancellation finished interrupt enable.
Each Tx buffer has its own CFIE bit.
0: Cancellation finished interrupt disabled
1: Cancellation finished interrupt enabled

39.4.35 FDCAN Tx event FIFO status register (FDCAN_TXEFS)

Address offset: 0x00E4

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.	TEFL	EFF	Res.	Res.	Res.	Res.	Res.	Res.	EFPI[1:0]	
						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.	EFGI[1:0]		Res.	Res.	Res.	Res.	Res.	EFFL[2:0]		
						r	r						r	r	r

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

Bit 25 **TEFL**: Tx event FIFO element lost
This bit is a copy of the TEFL interrupt flag of the FDCAN_IR. When TEFL is cleared, this bit is also cleared.
0 No Tx event FIFO element lost
1 Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size 0.

Bit 24 **EFF**: Event FIFO full
0: Tx event FIFO not full
1: Tx event FIFO full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **EFPI[1:0]**: Event FIFO put index
Tx event FIFO write index pointer.
Range: 0 to 3.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **EFGI[1:0]**: Event FIFO get index
Tx event FIFO read index pointer.
Range: 0 to 3.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **EFFL[2:0]**: Event FIFO fill level
Number of elements stored in Tx event FIFO.
Range: 0 to 3.

39.4.36 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA)

Address offset: 0x00E8

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.	EFAI[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **EFAI[1:0]**: Event FIFO acknowledge index
After the host has read an element or a sequence of elements from the Tx event FIFO, it has to write the index of the last element read from Tx event FIFO to EFAI[1:0]. This sets the Tx event FIFO get index (EFGI[1:0] of FDCAN_TXEFS) to EFAI[1:0] + 1 and updates the FIFO 0 fill level (EFFL[2:0] of FDCAN_TXEFS).

39.4.37 FDCAN CFG clock divider register (FDCAN_CKDIV)

Address offset: 0x0100

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.	PDIV[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PDIV[3:0]**: input clock divider

The CAN kernel clock can be divided prior to be used by the CAN subsystem. The rate must be computed using the divider output clock.

0000: Divide by 1
0001: Divide by 2
0010: Divide by 4
0011: Divide by 6
0100: Divide by 8
0101: Divide by 10
0110: Divide by 12
0111: Divide by 14
1000: Divide by 16
1001: Divide by 18
1010: Divide by 20
1011: Divide by 22
1100: Divide by 24
1101: Divide by 26
1110: Divide by 28
1111: Divide by 30

This bitfield is write-protected (P): which means that write access is possible only when the CCE bit of the FDCAN_CCCR register is set.

Note: The clock divider is common to all FDCAN instances. Only FDCAN1 instance has FDCAN_CKDIV register, which changes clock divider for all instances.

39.4.38 FDCAN register map

Table 333. FDCAN 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	FDCAN_CREL	REL[3:0]				STEP[3:0]				SUBSTEP [3:0]				YEAR[3:0]				MON[7:0]								DAY[7:0]							
	Reset value	0	0	0	0	0	0	0	1	1	1	1	0	1	0	1	0	0	1	1	0	1	1	1	1	1	0	1	0	0	0	1	0
0x0004	FDCAN_ENDN	ETV[31:0]																															
	Reset value	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1
0x0008	Reserved	Reserved																															
0x000C	FDCAN_DBTP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]					Res.	Res.	Res.	DTSEG1[4:0]					DTSEG2[3:0]				DSJW[3:0]			
	Reset value									0	0	0	0	0	0	0	0				0	1	0	1	0	0	0	1	1	0	0	1	1
0x0010	FDCAN_TEST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX	TX[1:0]		LBCK	Res.	Res.	Res.	Res.
	Reset value																									0	0	0	0				
0x0014	FDCAN_RWD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDV[7:0]								WDC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 333. FDCAN 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
0x0018	FDCAN_CCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INT
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x001C	FDCAN_NBTP	NSJW[6:0]							NBRP[8:0]									NTSEG1[7:0]								Res.	NTSEG2[6:0]						
	Reset value	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1
0x0020	FDCAN_TSCC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCP[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSS[1:0]	
	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
0x0024	FDCAN_TSCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSC[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
0x0028	FDCAN_TOCC	TOP[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOS[1:0]		ETOC
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	FDCAN_TOCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOC[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0030-0x003C	Reserved	Reserved																															
0x0040	FDCAN_ECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEL[7:0]								RP	REC[6:0]							TEC[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
0x0044	FDCAN_PSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]							Res.	PXE	REDL	RBRSRESI1	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x0048	FDCAN_TDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCO[6:0]							Res.	TDCF[6:0]						
	Reset value																		0	0	0	0	0	0		0	0	0	0	0	0	0	0
0x004C	Reserved	Reserved																															
0x0050	FDCAN_IR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARA	PED	PEA	WDI	BO	EW	EP	ELO	TOO	MRAF	TSW	TEFL	TEFF	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1N	RF0L	RF0F	RF0N
	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
0x0054	FDCAN_IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1NE	RF0LE	RF0FE	RF0NE
	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
0x0058	FDCAN_ILS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERR	BERR	MISC	TFERR	SMSG	RXFIFO1	RXFIFO0
	Reset value	0																									0	0	0	0	0	0	0

Table 333. FDCAN 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
0x005C	FDCAN_ILE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
	Reset value																															0	0
0x0060-0x007C	Reserved	Reserved																															
0x0080	FDCAN_RXGFC	Res.	Res.	Res.	Res.	LSE[3:0].				Res.	Res.	Res.	LSS[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	F0OM	F1OM	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
	Reset value					0	0	0	0				0	0	0	0	0							0	0			0	0	0	0	0	0
0x0084	FDCAN_XIDAM	Res.	Res.	Res.	EIDM[28:0]																												
	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
0x0088	FDCAN_HPMS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLST	Res.	Res.	FIDX[4:0]					MSI[1:0]		Res.	Res.	Res.	BIDX[2:0]		
	Reset value																	0			0	0	0	0	0	0	0				0	0	0
0x0090	FDCAN_RXF0S	Res.	Res.	Res.	Res.	Res.	Res.	RF0L	F0F	Res.	Res.	Res.	Res.	Res.	Res.	F0PI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	F0GI[1:0]		Res.	Res.	Res.	Res.	F0FL[3:0]			
	Reset value							0	0							0	0							0	0					0	0	0	0
0x0094	FDCAN_RXF0A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0AI[2:0]		
	Reset value																														0	0	0
0x0098	FDCAN_RXF1S	Res.	Res.	Res.	Res.	Res.	Res.	RF1L	F1F	Res.	Res.	Res.	Res.	Res.	Res.	F1PI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	F1GI[1:0]		Res.	Res.	Res.	Res.	F1FL[3:0]			
	Reset value							0	0							0	0							0	0					0	0	0	0
0x009C	FDCAN_RXF1A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F1AI[2:0]		
	Reset value																														0	0	0
0x00A0-0x00BC	Reserved	Reserved																															
0x00C0	FDCAN_TXBC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQM	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																							
0x00C4	FDCAN_TXFQS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQF	Res.	Res.	Res.	TFQPI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	TFGI[1:0]		Res.	Res.	Res.	Res.	Res.	TFFL[2:0]		
	Reset value											0				0	0							0	0						0	1	1
0x00C8	FDCAN_TXBRP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRP2	TRP1	TRP0
	Reset value																														0	0	0

Table 333. FDCAN 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
0x00CC	FDCAN_TXBAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR2	AR1	AR0
	Reset value																														0	0	0
0x00D0	FDCAN_TXBCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CR2	CR1	CR0
	Reset value																														0	0	0
0x00D4	FDCAN_TXBTO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TO2	TO1	TO0
	Reset value																														0	0	0
0x00D8	FDCAN_TXBCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CF2	CF1	CF0
	Reset value																														0	0	0
0x00DC	FDCAN_TXBTIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE2	TIE1	TIE0
	Reset value																														0	0	0
0x00E0	FDCAN_TXBCIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFIE2	CFIE1	CFIE0
	Reset value																														0	0	0
0x00E4	FDCAN_TXEFS	Res.	Res.	Res.	Res.	Res.	Res.	TEFL	EFF	Res.	Res.	Res.	Res.	Res.	Res.	EFPI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	EFG[1:0]		Res.	Res.	Res.	Res.	Res.	EFFL[2:0]		
	Reset value							0	0							0	0							0	0						0	0	0
0x00E8	FDCAN_TXEFA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EFAI[1:0]	
	Reset value																															0	0
0x0100	FDCAN_CKDIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PDIV[3:0]			
	Reset value																												0	0	0	0	0

Refer to *Section 2.2* for the register boundary addresses.

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

40.1 Introduction

The USB peripheral implements an interface between a full-speed USB 2.0 bus and the APB2 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 334 describes the USB implementation in the devices.

Table 334. STM32H503xx 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
Battery Charging Detection (BCD) support for device	X
Embedded pull-up resistor on USB_DP line	X

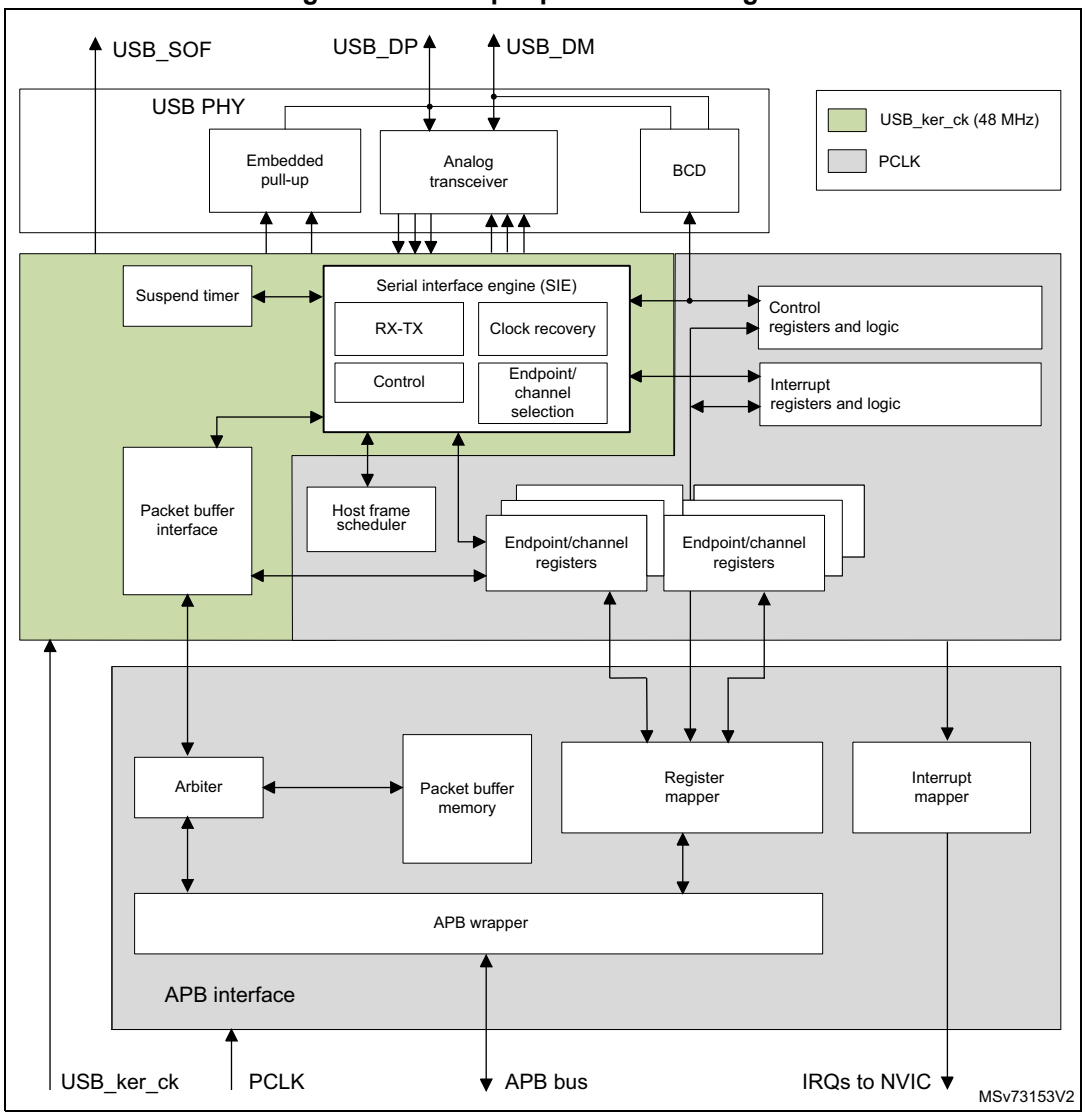
1. X= supported

40.4 USB functional description

40.4.1 USB block diagram

Figure 512 shows the block diagram of the USB peripheral.

Figure 512. USB peripheral block diagram



40.4.2 USB pins and internal signals

Table 335. USB input/output pins

Pin name	Pin type	Description
USB_DP	Digital input/output	D+ line
USB_DM	Digital input/output	D- line
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.
- 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 APB2 bus through an APB2 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 APB2 bus and from the USB interface. It resolves the conflicts by giving priority to APB2 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 APB2 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 APB2.
- APB2 wrapper: this provides an interface to the APB2 for the memory and register. It also maps the whole USB peripheral in the APB2 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 APB2 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 APB2 bus. Different clock configurations are possible where the APB2 clock frequency can be higher or lower than the USB peripheral one.

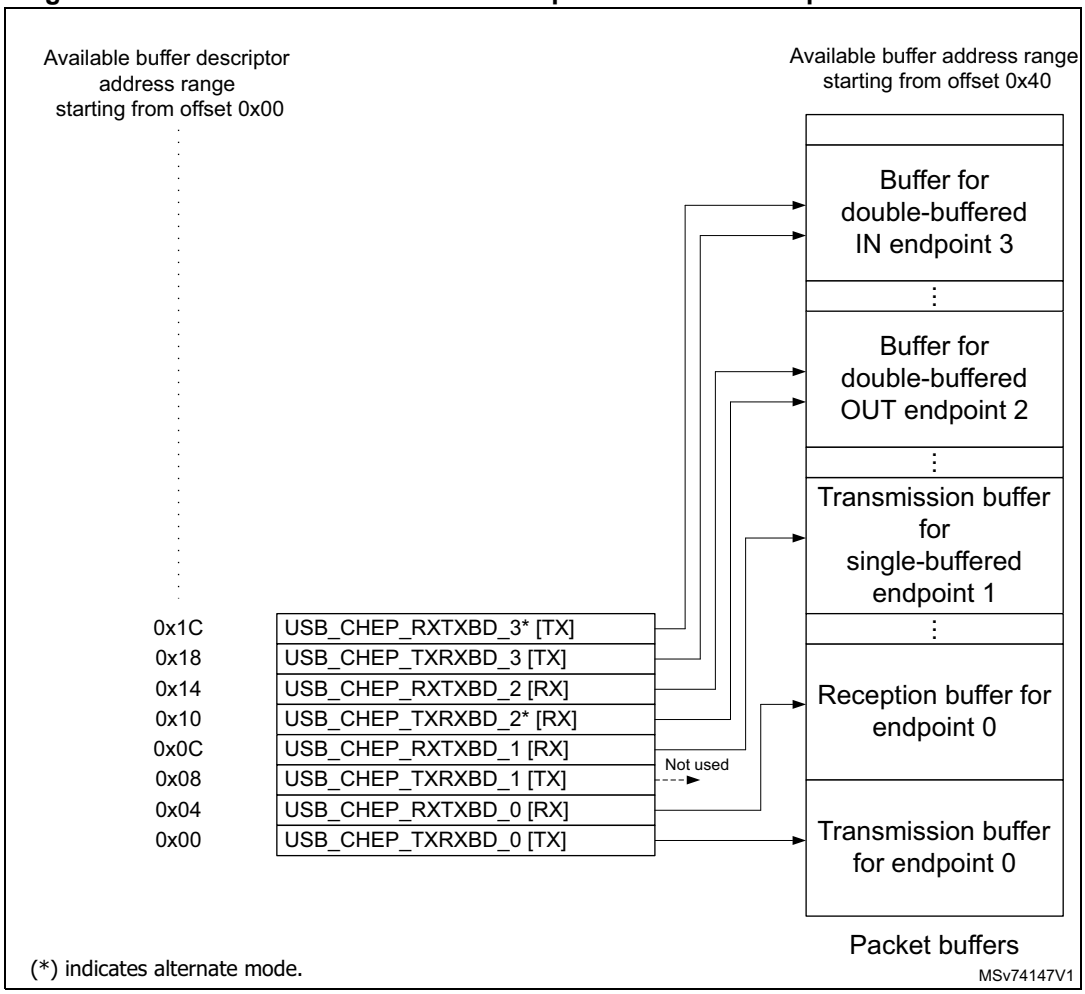
Note: Due to USB data rate and packet memory interface requirements, the APB2 clock must have a minimum frequency of 12 MHz to avoid data overrun/underrun problems.

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 513*.

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 513. 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 1627*) 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 USB_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 336. 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 337. 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	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations
	1	0	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	0	0	None (1)	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	1	1	None (1)	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
Receive (OUT)	0	1	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	0	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	0	0	None (1)	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	1	None (1)	USB_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 337*). 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 338: Bulk double-buffering memory buffers usage (Host mode)* shows the programming settings for OUT and IN tokens.

Table 338. 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	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations
	1	0	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	0	0	None (1)	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	1	1	None (1)	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
Receive (IN)	0	1	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	0	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	0	0	None (1)	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	1	None (1)	USB_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 339*.

Table 339. 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	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations
	1	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
Receive (OUT)	0	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations	USB_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 340* to understand the relationship between the DTOG bit buffers and the buffer usage.

Table 340. 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	USB_CHEP_TXRXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations.	USB_CHEP_RXTXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations
	1	USB_CHEP_RXTXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations	USB_CHEP_TXRXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations.
Receive (IN)	0	USB_CHEP_RXTXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations.	USB_CHEP_TXRXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations.
	1	USB_CHEP_TXRXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations	USB_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 341*, 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 341. 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 342*, 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 342. 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	DCDET	SDEN	PDEN	DCDEN	BCDEN
rw								r	r	r	r	rw	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 **DCDET:** Data contact detection (DCD) status

– Device mode

This bit gives the result of DCD.

0: data lines contact not detected.

1: data lines contact detected.

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 (DCD, 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 (DCD, PD, SD or OFF) must be selected to work correctly.

Bit 1 **DCDEN:** Data contact detection (DCD) mode enable
– Device mode
This bit is set by the software to put the BCD into DCD mode. Only one detection mode (DCD, PD, SD or OFF) must be selected to work correctly.

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	DTOG RX	STATRX[1:0]		SETUP	UTYPE[1:0]		EP KIND	VTTX	DTOG TX	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 343: Reception status encoding on page 1659*. 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 344: 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 345* 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 346*. 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 346* 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 343. 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 344. Endpoint/channel type encoding

UTYPE[1:0]	Meaning
00	BULK
01	CONTROL
10	ISO
11	INTERRUPT

Table 345. 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 346. 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 346. 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 347. USB 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
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
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

Table 347. USB 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
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	DCDET	SDEN	PDEN	DCDEN	BCDEN
	Reset value																	0								0	0	0	0	0	0	0	0

Refer to *Section 2.2* 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 1627*.

40.7.1 Channel/endpoint transmit buffer descriptor n (USB_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] (USB_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 348*.

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 348. 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	
16 (10000)	32 bytes	
...	...	...
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 (USB_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 348*.

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] (USB_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 349. USBSRAM 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
0x00	**USB_ 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	**USB_ 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	**USB_ 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	**USB_ 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	**USB_ 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	**USB_ 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	**USB_ 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	**USB_ 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	**USB_ 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	**USB_ 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	**USB_ 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	**USB_ 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 Introduction

A comprehensive set of debug features is provided to support software development and system integration:

- Breakpoint debugging of the CPU core
- Code execution tracing
- Software instrumentation
- Cross-triggering

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 debug features are based on Arm® CoreSight™ components.

- SWJ-DP: JTAG/Serial-wire debug port
- AHB-AP: AHB access port
- ROM table
- System control space (SCS)
- Breakpoint unit (BPU)
- Data watchpoint and trace unit (DWT)
- Instrumentation trace macrocell (ITM)
- Embedded Trace Macrocell™ (ETM)
- Cross trigger interface (CTI)
- Trace port interface unit (TPU)

The debug features are accessible by the debugger via the AHB-AP.

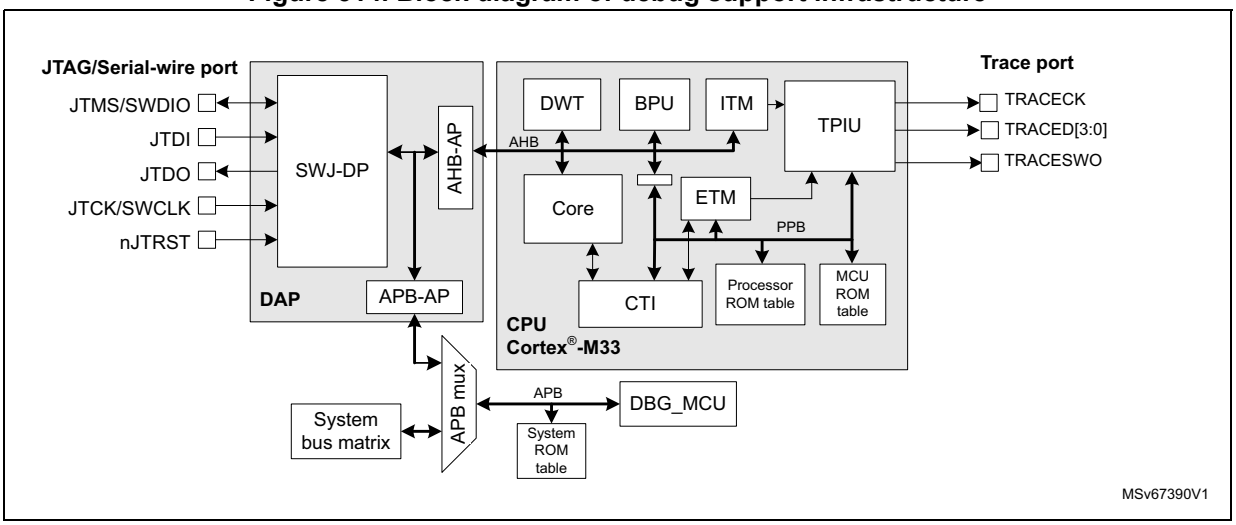
Additional information can be found in the Arm® documents referenced in *Section 41.13*.

Note: Only nonsecure memory access is supported in this product.

41.2 DBG functional description

41.2.1 DBG block diagram

Figure 514. Block diagram of debug support infrastructure



41.2.2 DBG pins and internal signals

Table 351. JTAG/Serial-wire debug port pins

Pin name	JTAG debug port		SW debug port		Pin assignment
	Type	Description	Type	Description	
JTMS/SWDIO	I	JTAG test mode select	IO	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	O	JTAG test data output	-	-	PB3
nJTRST	I	JTAG test reset	-	-	PB4

Table 352. Trace port pins

Pin name	Type	Description	Pin assignment
TRACED0	O	Trace synchronous data out 0	Refer to the datasheet
TRACED1		Trace synchronous data out 1	
TRACED2		Trace synchronous data out 2	
TRACED3		Trace synchronous data out 3	
TRACECK		Trace clock	

Table 353. 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.

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 DCLK domain that covers the rest of the SWJ-DP and the access port.

The DCLK is a gated version of the system clock.

The DCLK domain is enabled by the debugger using the CDBGPWRUPREQ bit in the *DP control and status register (DP_CTRLSTATR)*. 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 DP_CTRL/STATR. The DCLK is disabled at power-up, and must be disabled when the debugger is disconnected, to avoid wasting energy.

The debug and trace components included in the processor are clocked with the processor clock.

41.2.4 DBG power domains

The debug components are located in the core power domain. This means that the debugger connection is not possible in standby low-power mode. To avoid losing the connection when the device enters standby mode, the power can be maintained to the core by setting a bit in the *DBGMCU configuration register (DBGMCU_CR)*. This also keeps the processor clocks active and holds off the reset, so that the debug session is maintained.

41.2.5 Debug and low-power modes

The devices include power saving features that allow the core power domain to be switched off or stopped when not required. If the power is switched off or if the core is not clocked, all debug components are inaccessible to the debugger. To avoid this, power-saving mode emulation is implemented. If the emulation is enabled for a domain, the domain still enters power-saving mode, but its clock and power are maintained. In other words, the domain behaves as if it is in power-saving mode, but the debugger does not lose the connection.

The emulation mode is programmed in the microcontroller debug (DBGMCU) unit. For more information, refer to *Section 41.12: Microcontroller debug unit (DBGMCU)*.

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.

Debugger access is disabled while the processor is booting from system flash memory.

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 are enabled.
- **niden**: enables trace and performance monitoring (noninvasive debug).
  0: Trace generation is disabled.
  1: Trace generation in the nonsecure state is enabled.

For detailed information on the behavior of each component according to the state of the authentication signals, refer to the relevant component chapter or to the relevant Arm® technical documentation.

The state of the signals is set according to the debug state as shown in *Table 354*.

Table 354. Authentication signal states

Debug state	Authentication signal state	Description
OPEN	dbgen = 1 spiden = 1 niden = 1 spniden = 1	Debug and trace is enabled whatever the state of the processor. All memory and resources are accessible to the debugger.
CLOSED	dbgen = 0 spiden = 0 niden = 0 spniden = 0	Debug and trace is disabled.

The state of the authentication signals can be read from the DAUTHSTATUS register in the system control space (SCS) of the Cortex®-M33.

The debug state depends on the product life cycle state (see *Section 3.5: Product life-cycle*).

Table 355. Life cycle state and debug states

Product life cycle state	Debug state
OPEN	OPEN
CLOSED	CLOSED
LOCKED	CLOSED

41.2.7 Debug authentication

Figure 515. Product life cycle states and debug authentication

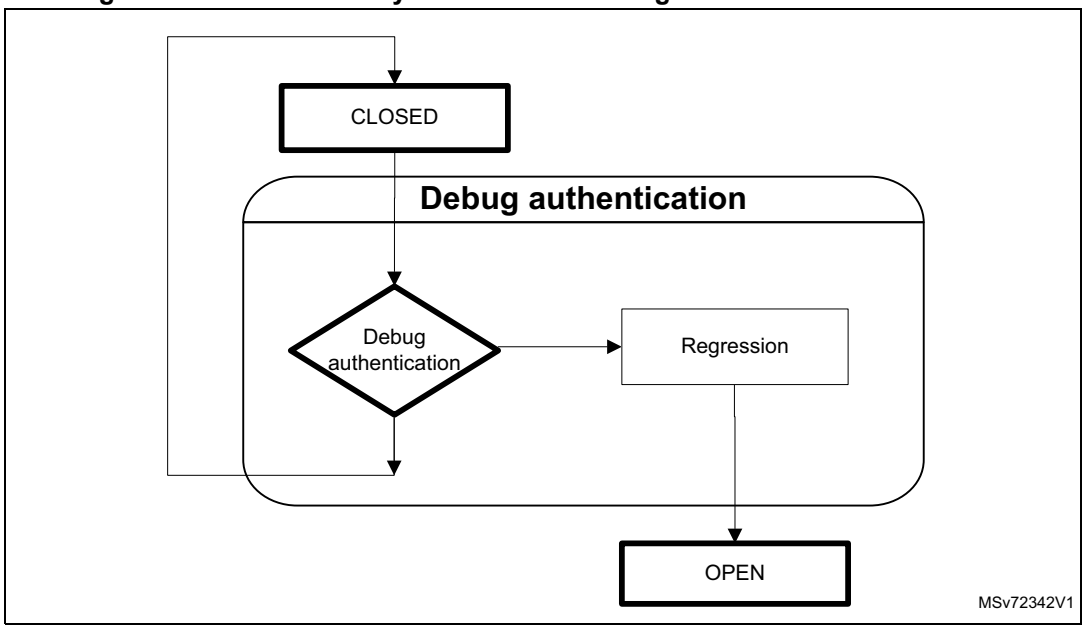


Figure 515 shows the product life cycle and debug authentication states. If the device is in PRODUCT_STATE “CLOSED”, the debug state is CLOSED. The debug authentication procedure allows a trusted debugger to open access by performing a full-regression.

Full regression corresponds to releasing all code and assets. The intermediate state allowing full regression management is called regression.

Debug authentication control principle

The debug authentication is one of the most critical security features of the system considering that with a debugger the user can access a large part of the system.

To control the reopening of the debug, the device imposes a debug authentication protocol.

The protocol is based on a password to trigger the full regression action.

The protocol implements a bidirectional communication between the host and the device through a mailbox interface located in the DBGMCU.

The host can write to the mailbox via the JTAG/SWD interface. It expects to get responses and messages from the device via the same mailbox.

The debug authentication protocol is launched on a power-on reset of the device, when an “open request” message is posted by the host.

The protocol is based on:

- Initial message: posted by the host combined with a reset to launch the debug authentication process on the device.
- Password message: the host sends a message containing the user password.

The implementation is ensured by code embedded in the system flash memory. This code is called automatically after reset if an initial message has been posted by the host in the mailbox.

The STMicroelectronics implementation is based on the Arm® PSA-ADAC solution for debug authentication.

The debug authentication can be implemented using a proprietary or open-source protocol.

As this feature is critical for security, the STM32H5 devices come with debug authentication provisioned in system flash memory. STMicroelectronics provides the host tools integrated with some debug environments (IDEs).

Debug authentication provisioning

Debug authentication is natively supported by STM32H5 platform. It means that the data used by ST debug authentication (ST-DA) must be provisioned at a defined location in OTP flash area (OTP defined in flash memory).

The debug authentication configuration must be done when the PRODUCT_STATE is “Provisioning”. It must not be performed when PRODUCT_STATE is “Open”.

The SHA256 of the password must be provisioned (refer to *Section 7.6.2: RSS user functions*) at the very beginning of the OTP area.

Caution: The user must take care to configure debug authentication as explained above before changing to states different from “Open” or “Provisioning”.

41.3 Serial-wire and JTAG debug port (SWJ-DP)

The SWJ-DP is a CoreSight™ component that implements an external access port for connecting debugging equipment.

Two 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)

These two modes are mutually exclusive, since they share the same IO pins.

By default, the JTAG-DP is selected after a system or a power-on reset. The five IO pins are configured by hardware in debug alternative function mode. The SWJ-DP incorporates the 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.

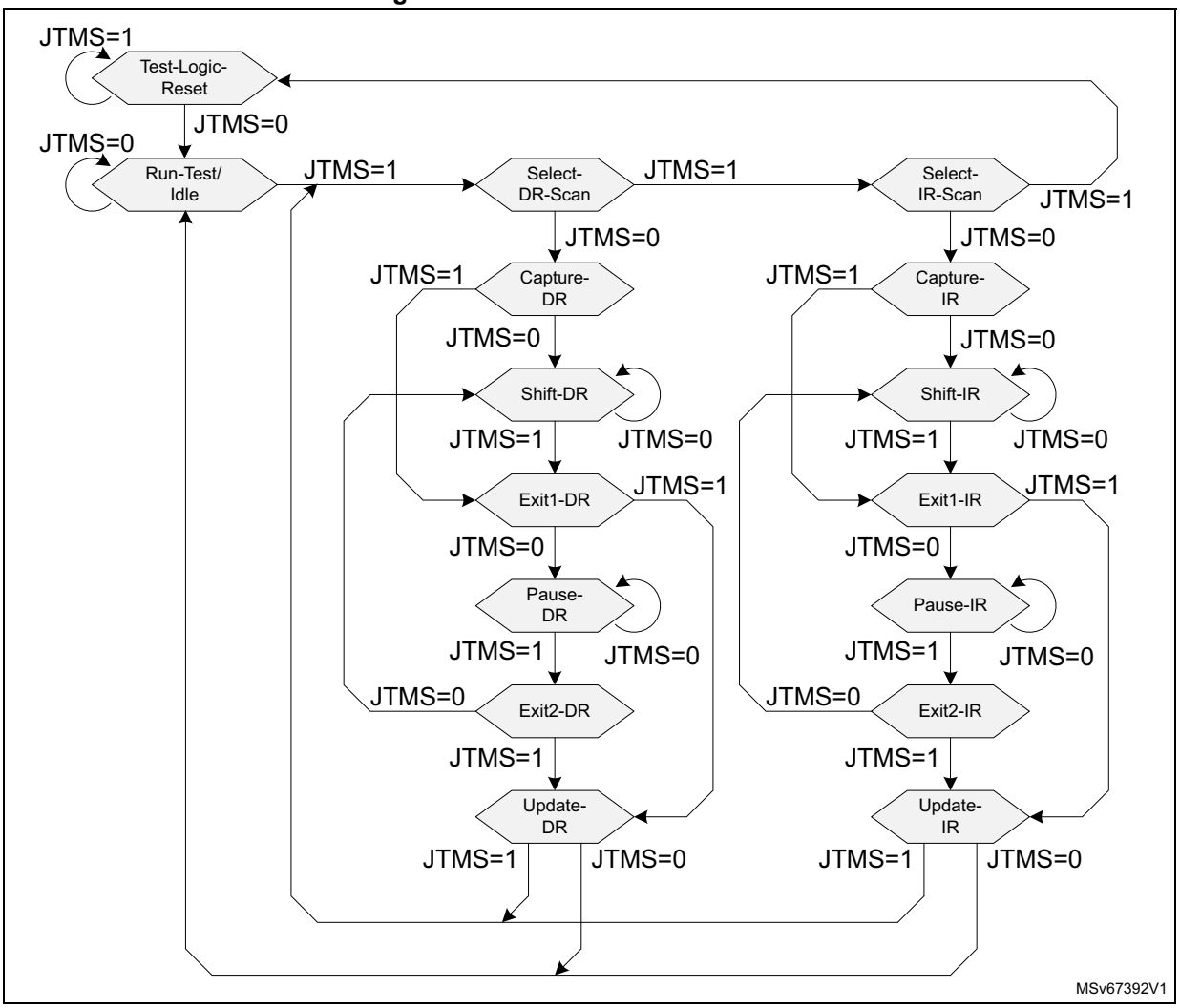
In SW-DP mode, the unused JTAG pins JTDI, JTDO, and nJTRST can be used for other functions.

Note: All SWJ port IOs can be reconfigured to other functions by software, but debugging is no longer possible.

41.3.1 JTAG debug port

The JTAG-DP implements a TAP state machine (TAPSM), shown in the figure below, 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 516. 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 to 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 to the instruction register.
4. When the TAPSM goes through the Capture-DR state, a value is transferred from one of the data registers to 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 shifts 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 to 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 IR.

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 the table below.

Table 356. JTAG-DP data registers

IR instruction	DR register	Scan chain length	Description
0000 to 0111	(BYPASS)	1	Not implemented: BYPASS selected
1000	ABORT	35	ABORT register – bits 31:1 = 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 356. JTAG-DP data registers (continued)

IR instruction	DR 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	Identification code 0x6BA0 0477: Cortex®-M33 JTAG debug port ID code
1111	BYPASS	1	Bypass A single JTCK cycle delay is inserted between JTDI and JTDO.

The DR registers are described in more detail in the Arm® Debug Interface Architecture Specification (see *Section 41.13*).

41.3.2 Serial-wire debug port

The serial-wire debug protocol uses the 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 bits) transmitted by the host (see *Table 357*).
2. Acknowledge response (3 bits) transmitted by the target (see *Table 358*).
3. Data transfer (33 bits) transmitted by the host (in case of a write) or target (in case of a read) (see *Table 359*).

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 357. Packet request

Bit field	Name	Description
0	Start	Must be 1
1	APnDP	– 0: DP register access - see *Section 41.3.3: Debug port registers* – 1: AP register access - see *Section 41.4: Access ports*
2	RnW	– 0: write request – 1: read request
4:3	A(3:2)	Address field of the DP or AP register (refer to *Table 361* or *Table 363*)
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by host, must be read as 1 by target

Table 358. ACK response

Bit field	Name	Description
2:0	ACK	– 000: FAULT – 010: WAIT – 100: OK

Table 359. 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.

41.3.3 Debug port registers

Both the SW-DP and the JTAG-DP access the debug port (DP) registers listed in *Table 361*.

The debugger can access the DP registers as follows:

1. Program the A(3:2) field in the DPACC register, if using JTAG, 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 357*). 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 contents of the DP_SELECTR register.

Table 360. Debug port registers

Address	A(3:2) value	R/W	Description
0x0	00	R	*DP debug port identification register (DP_DPIDR)* contains the IDCODE for the debug port.
		W	*DP abort register (DP_ABORTR)*(1) aborts the current AP transaction. This register is also used to clear the error flags in the DP_CTRLSTATR register.
0x4	01	R/W	If DP_SELECTR.DPBANKSEL[3:0] = 0x0, *DP control and status register (DP_CTRLSTATR)* controls the DP and provides status information.
			If DP_SELECTR.DPBANKSEL[3:0] = 0x1, *DP data link control register (DP_DLCR)*(2) controls the operating mode of the SWD data link.
			If DP_SELECTR.DPBANKSEL[3:0] = 0x2, *DP target identification register (DP_TARGETIDR)* provides target identification information.
			If DP_SELECTR.DPBANKSEL[3:0] = 0x3, *DP data link protocol identification register (DP_DLPIDR)*(2) provides the SWD protocol version.
0x8	10	R	*DP event status register (DP_RESENDR)*(2) returns the value that was returned by the last AP read or DP_RDBUFF read. Used in the event of a corrupted read transfer.
		W	*DP access port select register (DP_SELECTR)* selects the access port, access port register bank, and DP register at address 0x4.
0xC	11	R	*DP read buffer register (DP_RDBUFFR)* – Via JTAG-DP, this register allows the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation) – Via SW-DP, this register contains the result of the preceding AP read access, allowing a new AP access to be avoided.

1. Access to the AP ABORT register from the JTAG-DP is done using the ABORT instruction.
2. Only accessible via SW-DP. Register is “reserved” via JTAG-DP.

DP debug port identification register (DP_DPIDR)

Address offset: 0x0

Reset value: 0x6BA0 2477

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
0x6: Rev 6

Bits 27:20 **PARTNO[7:0]**: part number for the debug port
0xBA

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

Bit 16 **MIN**: minimal debug port (MINDP) implementation
0x0: MINDP not implemented

Bits 15:12 **VERSION[3:0]**: debug port architecture version
0x2: DPv2

Bits 11:1 **DESIGNER[10:0]**: JEDEC designer identity code
0x23B: Arm® JEDEC code

Bit 0 Reserved, must be kept at reset value.

DP abort register (DP_ABORTR)

Address offset: 0x0

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.	ORUNERRCLR	WDERRCLR	STKERRCLR	Res.	DAPABORT
											w	w	w		w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **ORUNERRCLR**: overrun error clear
0: no effect
1: STICKYORUN bit cleared in DP_CTRL/STATR register

Bit 3 **WDERRCLR**: write data error clear
0: no effect
1: WDATAERR bit cleared in DP_CTRL/STATR register

Bit 2 **STKERRCLR**: sticky error clear
0: no effect
1: STICKYERR bit cleared in DP_CTRL/STATR register

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DAPABORT**: current AP transaction aborted if excessive number of WAIT responses returned
This bit indicates that the transaction is stalled.
0: no effect
1: transaction aborted

DP control and status register (DP_CTRLSTATR)

Address offset: 0x4

Reset value: 0x0000 0000

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x0.

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	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.	WDATAERR	READOK	STICKYERR	Res.	Res.	Res.	STICKYORUN	ORUNDETECT
								r	r	r				r	r

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **CDBGPWRUPACK**: debug power-up acknowledge
See description in *Section 41.2.5: Debug and low-power modes*.
0: DCLK gated
1: DCLK enabled

Bit 28 **CDBGPWRUPREQ**: debug power-up request
This bit controls the DCLK enable request signal.
0: requests DCLK gating
1: requests DCLK enable

Bits 27:8 Reserved, must be kept at reset value.

Bit 7 **WDATAERR**: write data error (read-only) in SW-DP
This bit indicates that there is a parity or framing error on the data phase of a write, or a write accepted by the DP is then discarded without being submitted to the AP.
This bit is reset by writing 1 to the ABORT.WDERRCLR bit.
0: no error
1: an error occurred
Note: This bit is reserved in JTAG-DP.

Bit 6 **READOK**: AP read response (read-only) in SW-DP
This bit indicates the response to the last AP read access.
0: read not OK
1: read OK
Note: This bit is reserved in JTAG-DP.

Bit 5 **STICKYERR**: transaction error (read-only in SW-DP, read/write in JTAG-DP)
This bit indicates that an error occurred in an AP transaction.It is reset by writing 1 to the DP_ABORTR.STKERRCLR bit (in SW-DP and JTAG-DP)
0: no error
1: an error occurred

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **STICKYORUN**: overrun (read-only in SW-DP, read/write in JTAG-DP).
This bit indicates that an overrun occurred (new transaction received before previous transaction completed). This bit is only set if the ORUNDETECT bit is set. it is reset by writing 1 to the DP_ABORTR.ORUNERRCLR bit (in SW-DP and JTAG-DP).
0: no overrun
1: an overrun occurred

Bit 0 **ORUNDETECT**: overrun detection mode enable.
0: disabled
1: enabled. In the event of an overrun, the STICKYORUN bit is set and subsequent transactions are blocked until the STICKYORUN bit is cleared.

DP data link control register (DP_DLCR)

Address offset: 0x4

Reset value: 0x0000 0000

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x1.

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

Bits 7:6 **WIREMODE[1:0]**: SW-DP mode
0x0: synchronous mode

Bits 5:0 Reserved, must be kept at reset value.

DP target identification register (DP_TARGETIDR)

Address offset: 0x4

Reset value: 0xXXXX 0041

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x2.

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

Bits 27:12 **TPARTNO[15:0]**: target part number
0x4740: STM32H503xx

Bits 11:1 **TDESIGNER[10:0]**: target designer JEDEC code
0x020: STMicroelectronics

Bit 0 Reserved, must be kept at reset value.

DP data link protocol identification register (DP_DLPIDR)

Address offset: 0x4

Reset value: 0x0000 0001

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x3.

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
this field defines the instance number for the 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

DP event status register (DP_RESENDR)

Address offset: 0x8

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_RDBUFF read
This register is used in the event of a corrupted read transfer.

DP access port select register (DP_SELECTR)

Address offset: 0x8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APSEL[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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.	APBANKSEL[3:0]				DPBANKSEL[3:0]			
								w	w	w	w	w	w	w	w

Bits 31:24 **APSEL[7:0]**: access port select
This field selects the access port for the next transaction.
0x00: AP0 - System debug access port (APB-AP)
0x01: AP1 - Cortex®-M33 debug access port (AHB-AP)
others: reserved

Bits 23:8 Reserved, must be kept at reset value.

Bits 7:4 **APBANKSEL[3:0]**: AP register bank select
This field selects the 4-word register bank on the active AP for the next transaction.

Bits 3:0 **DPBANKSEL[3:0]**: DP register bank select
This field selects the register at address 0x4 of the debug port.
0x0: DP_CTRLSTAT register
0x1: DP_DLCR register
0x2: DP_TARGETID register
0x3: DP_DLPIDR register
others: reserved

DP read buffer register (DP_RDBUFFR)

Address offset: 0xC

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]**: 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 that 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.

41.3.4 Debug port register map and reset values

These registers are not on the CPU memory bus. They are only accessed through the SW-DP and JTAG-DP debug interface.

The debug port address offset is 4 bits wide, where the two most significant bits are defined in the JTAG-DP register DPACC or SW-DP packet request A[3:2] field. The two least significant bits are 00.

Table 361. Debug port 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
0x0	**DP_DPIDR**	REVISION [3:0]				PARTNO[7:0]								Res.	Res.	Res.	MIN	VERSION [3:0]				DESIGNER[10:0]											Res.
	Reset value	0	1	1	0	1	0	1	1	1	0	1	0				0	0	0	1	0	0	1	0	0	0	1	1	1	0	1	1	1
0x0	**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
0x4(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
0x4(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 361. Debug port 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
0x4[3]	**DP_TARGETIDR**	TREVISION [3:0]				TPARTNO[15:0]																TDESIGNER[10:0]											Res.
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
0x4[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
0x8	**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
0x8	**DP_SELECTR**	APSEL[7:0]				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	x	x	x	x
0xC	**DP_RDBUFFR**	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[3:0] = 0x0.
2. DP_SELECTR.DPBANKSEL[3:0] = 0x1.
3. DP_SELECTR.DPBANKSEL[3:0] = 0x2.
4. DP_SELECTR.DPBANKSEL[3:0] = 0x3.

41.4 Access ports

There are two access ports (AP) attached to the DP.

- System debug access port (AP0): Enables access to the DBGMCU and the system ROM table via an APB bus.
- Cortex®-M33 debug access port (AP1): Enables access to the debug and trace features integrated in the Cortex®-M33 processor core via its internal AHB bus.

41.4.1 Access port registers

The access ports are of type MEM-AP: the debug and trace component registers are mapped in the address space of the AHB. The 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 363*.

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 access port select register (DP_SELECTR)*)
- bits [3:2]: content of the A(3:2) field of the APACC data register in the JTAG-DP (see *Table 356*), or of the SW-DP packet request (see *Table 357*), depending on the debug interface used
- bits [1:0]: always set to 0

The content of the DP_SELECTR.APSEL[3:0] field defines which MEM-AP is being accessed.

Table 362. MEM-AP registers

Address	APBANKSEL	A(3:2)	Name	Description
0x00	0x0	0	CSWR	Control/status word register
0x04	0x0	1	TAR	Transfer address register Target address for the bus transaction.
0x08	-	-	-	Reserved
0x0C	0x0	3	DRWR	Data read/write register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:0]
0x10	0x1	0	BD0R	Banked data 0 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] + 0x0.
0x14	0x1	1	BD1R	Banked data 1 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] + 0x4.
0x18	0x1	2	BD2R	Banked data 2 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] + 0x8.
0x1C	0x1	3	BD3R	Banked data 3 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] + 0xC.
0x20	-	-	-	Reserved
0x24 to 0xEC	-	-	-	Reserved
0xF0	-	-	-	Reserved
0xF4	0xF	1	CFGR	Configuration register (read only)
0xF8	0xF	2	BASER	Debug base address register (read only) Base address of the ROM table
0xFC	0xF	3	IDR	Identification register (read only)

The debugger can access the AP registers as follows:

1. Program the APSEL[3:0] field in the *DP access port select register (DP_SELECTR)* to choose the AP, and the APBANKSEL[3:0] field in DP_SELECTR to select the register bank to be accessed.
2. Program the A(3:2) field in the APACC data register, if using JTAG, 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 357*). The write data is sent in the data phase.

The debugger can access the memory mapped debug component registers through the AP registers (using the above AP register access procedure) as follows:

1. Program the transaction target address in the *APx transfer address register (APx_TAR) (x = 0, 1)*.
2. Program the *AP1 control/status word register (AP1_CSWR)*, if necessary, with the transfer parameters (AddrInc for example).
3. Write to or read from the *APx data read/write register (APx_DRWR) (x = 0, 1)* to initiate a bus transaction at the address held in AP_TAR. Alternatively, a read or write to the *APx banked data n register (APx_BDnR) (x = 0, 1)* triggers access to the TAR[31:4] + n address, 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]*.

AP0 control/status word register (AP0_CSWR)

Address offset: 0x0

Reset value: 0x8000 0042

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBGSWEN	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.	MODE[3:0]				TRINPROG	DEVICEEN	ADDRINC[1:0]		Res.	SIZE[2:0]		
				rw	rw	rw	rw	r	r	rw	rw		r	r	r

Bit 31 **DBGSWEN**: software access enable
Enables or disables software access to the APB bus
0: disable software access
1: enable software access

Bits 30:12 Reserved, must be kept at reset value.

Bits 11:8 **MODE[3:0]**: mode of operation
0b0000: normal download or upload
other: reserved

Bit 7 **TRINPROG**: transfer in progress (read only)
This field indicates whether a transfer is currently in progress on the APB master port
0: no APB transfer in progress
1: APB transfer in progress

Bit 6 **DEVICEEN**: device enable status (read only)
1: APB transfers always enabled

Bits 5:4 **ADDRINC[1:0]**: auto-increment mode
Defines whether TAR address is automatically incremented after a transaction.
0x0: no auto-increment
0x1: address incremented by the size in bytes of the transaction (SIZE field)
other: reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: size of next memory access transaction
0x2: word (32-bit)

AP1 control/status word register (AP1_CSWR)

Address offset: 0x0

Reset value: 0x43X0 00X2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SPROT	Res.	Res.	PROT[3:0]				SPISTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw			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.	TRINPROG	DBGSTATUS	ADDRINC[1:0]		Res.	SIZE[2:0]		
								r	r	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SPROT**: secure transfer request
This field sets the protection attribute HPROT[6] of the bus transfer.
0: Reserved
1: nonsecure transfer, HPROT[6] = high
If SPIDEN = 0 and SPROT = 0, no bus transfer occurs

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:24 **PROT[3:0]**: bus transfer protection
This field sets the protection attributes HPROT[3:0] of the bus transfer.
0bXXX0: instruction access
0bXXX1: data access
0bXX0X: user mode
0bXX1X: privilege mode
0bX0XX: non-bufferable
0bX1XX: bufferable
0b0XXX: non-shareable, no look-up, non-modifiable
0b1XXX: shareable, look-up, modifiable

Bit 23 **SPISTATUS**: secure debug authentication status
This field indicates the state of the SPIDEN signal
0: No secure AHB transfers allowed

Bits 22:8 Reserved, must be kept at reset value.

Bit 7 **TRINPROG**: transfer in progress (read only)
This field indicates whether a transfer is currently in progress on the APB master port
0: No AHB transfer in progress
1: AHB transfer in progress

Bit 6 **DBGSTATUS**: debug enable (DBGEN) status
0: AHB transfers blocked
1: AHB transfers enabled

Bits 5:4 **ADDRINC[1:0]**: auto-increment mode
Defines whether TAR address is automatically incremented after a transaction.
0x0: no auto-increment
0x1: address incremented by the size in bytes of the transaction (SIZE field). Single transfer.
0x2: address incremented by the size in bytes of the transaction (SIZE field). Packs four 8-bit transfers or two 16-bit transfers into a 32-bit DAP transfer. Multiple transactions are carried out on the AHB interface.
other: 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

APx transfer address register (APx_TAR) (x = 0, 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. In AP0, TA[1:0] are fixed at 0.

APx data read/write register (APx_DRWR) (x = 0, 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

APx banked data n register (APx_BDnR) (x = 0, 1)

Address offset: 0x10 + 0x4 * n, (n = 0 to 3)

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]:** banked data of current transfer to address TA [31:4] + 4 * n.
Auto address incrementing is not performed on APx_BD0-3R. Banked transfers are only supported for word transfers.

APx base address register (APx_BASER) (x = 0, 1)

Address offset: 0xF8

Reset value: AP 0: 0xE00E 0003

Reset value: AP 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 the first ROM table
The 12 LSBs are zero since the ROM table must be aligned on a 4-Kbyte boundary.
0xE00E0: AP0
0xE00FE: AP1

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 present

APx identification register (APx_IDR) (x = 0, 1)

Address offset: 0xFC

Reset value: AP 0: 0x5477 0002

Reset value: AP 1: 0x8477 0001

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.	IDENTITY[7:0]							
r	r	r						r	r	r	r	r	r	r	r

Bits 31:28 **REVISION[3:0]**: revision number
0x5: r1p0
0x8: r0p9

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]**:
0x1: MEM-AP

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

Bits 7:0 **IDENTITY[7:0]**:
0x1: AHB-AP
0x2: APB-AP

41.4.2 Access port register map

These registers are not on the CPU memory bus. They are only accessed through SW-DP and JTAG-DP debug interfaces.

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 363. Access port 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	**AP0_CSWR**	DBGSWEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DEVICEEN	ADDRINC[1:0]		Res.	SIZE[2:0]		
	Reset value	1																				0	0	0	0	0	1	0	0		0	1	0
0x00	**AP1_CSWR**	Res.	SPROT	Res.	Res.	PROT[3:0]				SPISTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DBGSTATUS	ADDRINC[1:0]		Res.	SIZE[2:0]		
	Reset value		1			0	0	1	1	X												0	0	0	0	0	X	0	0		0	1	0
0x04	**APx_TAR**	TA[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
0x08	Reserved	Reserved																															
0x0C	**APx_DRWR**	TD[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
0x10	**APx_BD0R**	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
0x14	**APx_BD1R**	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
0x18	**APx_BD2R**	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
0x1C	**APx_BD3R**	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
0x20 to 0xF4	Reserved	Reserved																															
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	1	1	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

Table 363. Access port 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
0xFC	**AP0_IDR**	REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							CLASS[3:0]				Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
	Reset value	0	1	0	1	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0						0	0	0	0	0	0	1	0
0xFC	**AP0_IDR**	REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							CLASS[3:0]				Res.	Res.	Res	Res.	Res.	IDENTITY[7:0]							
	Reset value	1	0	0	0	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0						0	0	0	0	0	0	0	1

41.5 ROM tables

The ROM table is a CoreSight™ component that contains the base addresses of the CoreSight debug components accessible via the access port to which it is attached. These tables allow a debugger to discover the topology of the CoreSight system automatically.

There is one top level ROM table behind each access port, APn. The base address of this ROM table can be obtained by reading the APn_BASER register of the access port. The top level ROM table may point in turn to other ROM tables.

The system ROM table is pointed to by the AP0 base register, AP0_BASER. It contains the base address pointer for the DBGMCU.

The system ROM table occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00E 4000 to 0xE00E 4FFC, when accessed by the debugger. It can be accessed by the CPU at the address range 0x4402 4000 to 0x4402 4FFC.

Table 364. System ROM table

Address offset in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0x000	DBGMCU	0xE00E 4000 (debugger) 0x4402 4000 (CPU)	0x0000 4000	4	0x0000 4003
0x004	Top of table	-	-	-	0x0000 0000
0x008 to 0xFC8	Reserved	-	-	-	0x0000 0000
0xFCC to 0xFFC	ROM table registers	-	-	-	See *Table 367*

There are two ROM tables in the CPU subsystem. The MCU ROM table is pointed to by the AP1 base register, AP1_BASER. It contains the base-address pointer for the processor ROM table and for the TPIU registers.

The MCU ROM table (see the table below) occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00F E000 to 0xE00F EFFC.

Table 365. 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	Reserved	-	-	-	0x1FF0 2002
0x00C	Reserved	-	-	-	0x1FF0 2002
0x010	Top of table	-	-	-	0x0000 0000
0x014 to 0xFC8	Reserved	-	-	-	0x0000 0000
0xFCC to 0xFFC	ROM table registers	-	-	-	See *Table 368*

The processor ROM table contains the base-address pointer for the system control space (SCS) registers that allow the debugger to identify the CPU core, as well as for the BPU, DWT, ITM, ETM, and CTI.

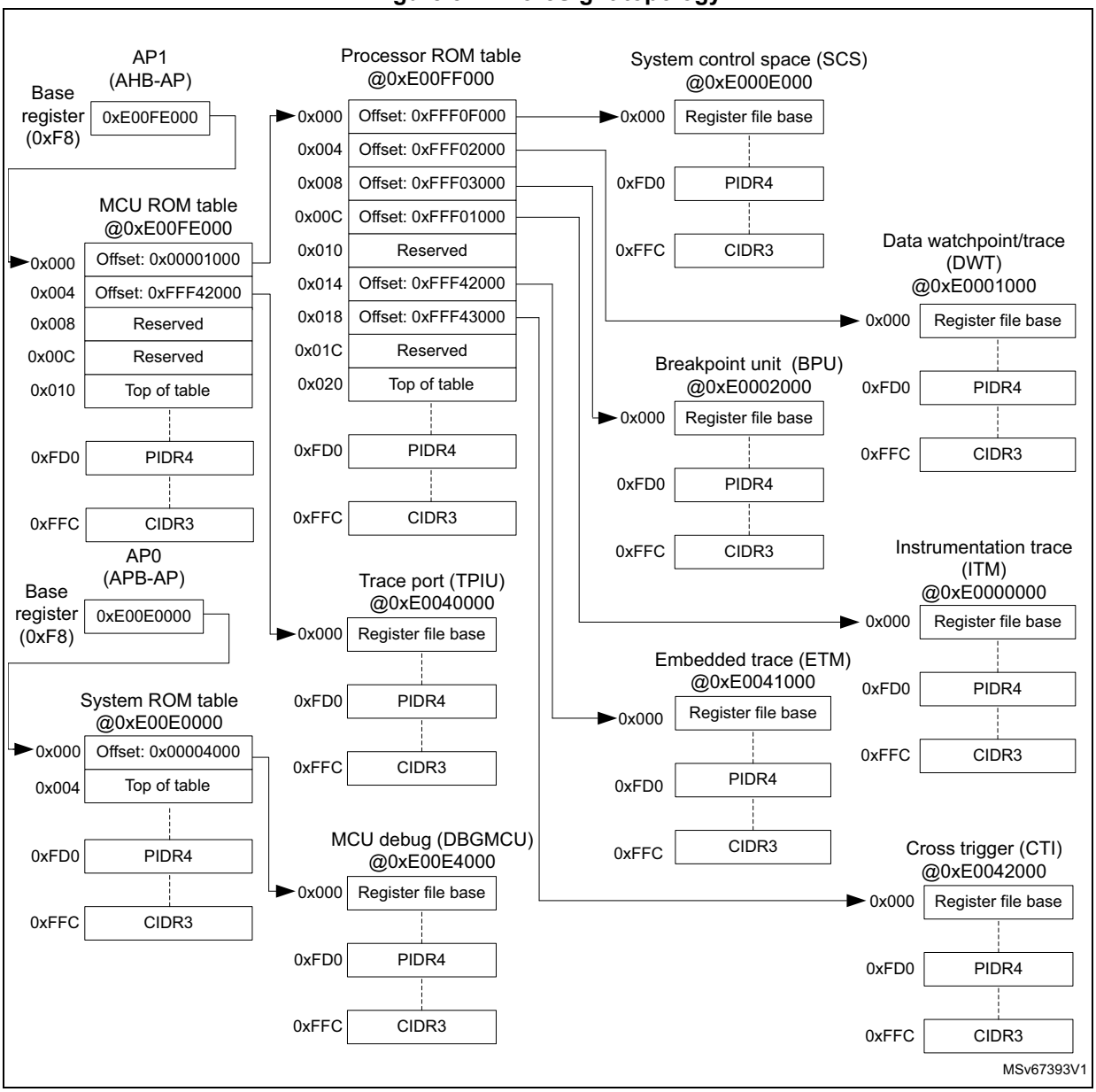
The processor ROM table (see the table below) occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00F F000 to 0xE00F FFFC.

Table 366. Processor ROM table

Address in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0xE00F F000	SCS	0xE000 E000	0xFFF0 F000	4	0xFFF0 F003
0xE00F F004	DWT	0xE000 1000	0xFFF0 2000	4	0xFFF0 2003
0xE00F F008	BPU	0xE000 2000	0xFFF0 3000	4	0xFFF0 3003
0xE00F F00C	ITM	0xE000 0000	0xFFF0 1000	4	0xFFF0 1003
0xE00F F010	Reserved	-	-	-	0xFFF4 1002
0xE00F F014	ETM	0xE004 1000	0xFFF4 2000	4	0xFFF4 2003
0xE00F F018	CTI	0xE004 2000	0xFFF4 3000	4	0xFFF4 3003
0xE00F F01C	Reserved	-	-	-	0xFFF4 4002
0xE00F F020	Top of table	-	-	-	0x0000 0000
0xE00F F024 to 0xE00F FFC8	Reserved	-	-	-	0x0000 0000
0xE00F FFCC to 0xE00F FFFC	ROM table registers	-	-	-	See *Table 369*

The topology for the CoreSight™ components is shown in the figure below.

Figure 517. CoreSight topology



41.5.1 System ROM table registers

System ROM memory type register (SYSROM_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
0x1: system memory present on this bus

System ROM 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.	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
0x0: STMicroelectronics JEDEC continuation code

System ROM CoreSight peripheral identity register 0 (SYSROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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]
0x74: STM32H503xx

System ROM CoreSight peripheral identity register 1(SYSROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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: STM32H503xx

System ROM 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 identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

System ROM 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

System ROM 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 identification bits [7:0]
0x0D: Common identification value

System ROM 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 identification bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

System ROM 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 identification bits [23:16]
0x05: common identification value

System ROM 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 identification bits [31:24]
0xB1: Common identification value

41.5.2 System ROM table register map

Table 367. System 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	**SYSROM_ 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	**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.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4 to FDC	Reserved	Reserved																															

Table 367. System ROM table 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
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																									X	X	X	X	X	X	X	X
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	X	X	X	X
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 *Table 364: System ROM table* for register boundary addresses.

41.5.3 MCU ROM table registers

MCU ROM memory type register (MCUROM_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

0x1: system memory present on this bus

MCU ROM CoreSight peripheral identity register 4 (MCUROM_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.	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
0x0: STMicroelectronics JEDEC continuation code

MCU ROM CoreSight peripheral identity register 0 (MCUROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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]
0x74: STM32H503xx

MCU ROM CoreSight peripheral identity register 1(MCUROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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: STM32H503xx

MCU ROM CoreSight peripheral identity register 2 (MCUROM_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 identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

MCU ROM CoreSight peripheral identity register 3 (MCUROM_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

MCU ROM CoreSight component identity register 0 (MCUROM_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

MCU ROM CoreSight peripheral identity register 1 (MCUROM_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 identification bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

MCU ROM CoreSight component identity register 2 (MCUROM_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

MCU ROM CoreSight component identity register 3 (MCUROM_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

41.5.4 MCU ROM table register map

Table 368. MCU 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	**MCUROM_ 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	**MCUROM_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	0	0	0
0xFD4 to FDC	Reserved	Reserved																															
0xFE0	**MCUROM_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																									X	X	X	X	X	X	X	X
0xFE4	**MCUROM_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	X	X	X	X
0xFE8	**MCUROM_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	**MCUROM_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	**MCUROM_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	**MCUROM_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	MCUROM_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	MCUROM_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 365: MCU ROM table* for register boundary addresses.

41.5.5 Processor ROM table registers

CPU ROM memory type register (CPUROM_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

CPU ROM CoreSight peripheral identity register 4 (CPUROM_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 continuation code

CPU ROM CoreSight peripheral identity register 0 (CPUROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00C9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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]
0xC9: Cortex®-M33

CPU ROM CoreSight peripheral identity register 1 (CPUROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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]
0x4: Cortex®-M33

CPU ROM CoreSight peripheral identity register 2 (CPUROM_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: 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]
0x3: Arm® JEDEC code

CPU ROM CoreSight peripheral identity register 3 (CPUROM_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

CPU ROM CoreSight component identity register 0 (CPUROM_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

CPU ROM CoreSight peripheral identity register 1 (CPUROM_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 identification bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

CPU ROM CoreSight component identity register 2 (CPUROM_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

CPU ROM CoreSight component identity register 3 (CPUROM_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

41.5.6 Processor ROM table register map

Table 369. CPU 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	**CPUROM_ 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
0xFD4 to FDC	Reserved	Reserved																															

Table 369. CPU ROM table 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
0xFD0	**CPUROM_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
0xFE0	**CPUROM_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	1	0	0	1	0	0	1
0xFE4	**CPUROM_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	0	1	0	0
0xFE8	**CPUROM_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	**CPUROM_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	**CPUROM_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	**CPUROM_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	**CPUROM_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	**CPUROM_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 366: Processor ROM table* for register boundary addresses.

41.6 Data watchpoint and trace unit (DWT)

The DWT provides four comparators that can be used as one of the following:

- watchpoint
- ETM trigger
- PC sampling trigger
- data address sampling trigger
- data comparator (COMP 1 only)
- clock cycle counter comparator (COMP 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 comparator x register (DWT_COMPxR)* with one of the following:

- a data address
- an instruction address
- a data value
- the cycle-count value, for COMP 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]*.

41.6.1 DWT registers

The DWT registers are located at address range 0xE000 1000 to 0xE000 1FFC.

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]				POSTRESET[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 disabled secure.
Controls whether the cycle counter is disabled in secure mode.
0: no effect
1: reserved

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 to be used as a timer for periodic PC sample packet generation
0: disabled
1: enabled

Bits 11:10 **SYNCTAP[1:0]**: position of the synchronization packet counter tap on the CYCCNT counter
This field 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 bits must be reset prior to writing POSTINIT.

Bits 4:1 **POSTRESET[3:0]**: reload value of the POSTCNT counter

Bit 0 **CYCCNTENA**: enable CYCCNT counter
0: disabled
1: enabled

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

DWT CPI count register (DWT_CPICNTR)

Address offset: 0x008

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.	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.

DWT exception count register (DWT_EXCCNTR)

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.	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.

DWT sleep count register (DWT_SLPCNTR)

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.	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).

DWT LSU count register (DWT_LSUCNTR)

Address offset: 0x014

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.	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.

DWT fold count register (DWT_FOLDCNTR)

Address offset: 0x018

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.	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.

DWT program counter sample register (DWT_PCSR)

Address offset: 0x01C

Reset value: 0xXXXX XXXX

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.

DWT comparator x register (DWT_COMPxR)

Address offset: 0x020 + 0x010 * x, (x = 0 to 3)

Reset value: 0xXXXX XXXX

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

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: 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 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: 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 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: 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 2.
For possible values of this field, refer to *[4]*

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® limited.

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

DWT device type register (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

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.	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

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: Cortex®-M33 DWT part number

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: Cortex®-M33 DWT part number

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 identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

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

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 identification bits [7:0]
0x0D: Common identification value

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 identification bits [15:12] - component class
0x9: debug component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

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 identification bits [23:16]
0x05: common identification value

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 identification bits [31:24]
0xB1: common identification value

41.6.2 DWT register map

The DWT registers are located at address range 0xE000 1000 to 0xE000 1FFC.

Table 370. 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	POSTINIT[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

Table 370. 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
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																									X	X	X	X	X	X	X	X
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																									X	X	X	X	X	X	X	X
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																									X	X	X	X	X	X	X	X
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																									X	X	X	X	X	X	X	X
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																									X	X	X	X	X	X	X	X
0x01C	**DWT_PCSR**	EIASAMPLE[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
0x020	**DWT_COMP0R**	COMP[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
0x024	Reserved	Reserved																															
0x028	**DWT_FUNCTR0**	ID[4:0]					Res.	Res.	MATCHED	Res.	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	1			0													0	0					0	0	0	0	0	0
0x02C	Reserved	Reserved																															
0x030	**DWT_COMP1R**	COMP[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
0x034	Reserved	Reserved																															
0x038	**DWT_FUNCTR1**	ID[4:0]					Res.	Res.	MATCHED	Res.	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	0	1	0			0													0	0					0	0	0	0	0	0
0x03C	Reserved	Reserved																															
0x040	**DWT_COMP2R**	COMP[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
0x044	Reserved	Reserved																															
0x048	**DWT_FUNCTR2**	ID[4:0]					Res.	Res.	MATCHED	Res.	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																															

Table 370. 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
0x050	**DWT_COMP3R**	COMP[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
0x054	Reserved	Reserved																															
0x058	**DWT_FUNCTR3**	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**	ARCHITECT[10:0]											PRESENT	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[3:0]			
	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.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	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

Table 370. 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
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 366: Processor ROM table* for register boundary addresses.

41.7 Instrumentation trace macrocell (ITM)

The ITM generates trace information in packets. Three sources 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 the following:

- Software trace

  The software can write directly to any 32 x 32-bit ITM stimulus register 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 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.
- 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.
- Local time-stamping

  The ITM contains a 21-bit counter clocked by the (pre-divided) 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 *[4]*.

41.7.1 ITM registers

The ITM registers are located at address range 0xE000 0000 to 0xE000 0FFC.

ITM stimulus register x (ITM_STIMRx)

Address offset: 0x000 + 0x004 * x, (x = 0 to 31)

Reset value: 0xXXXX XXXX

Condition: when writing

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[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	rw	rw

Bits 31:0 **STIMULUS[31:0]**: trace output data
write data is output on the trace bus as a software event packet.

ITM stimulus register x [alternate] (ITM_STIMRx)

Address offset: 0x000 + 0x004 * x, (x = 0 to 31)

Reset value: 0xXXXX XXXX

Condition: when reading

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DISABLE**: Disable flag
0: stimulus port and ITM enabled
1: stimulus port and ITM disabled

Bit 0 **FIFO_READY**: FIFO ready indicator
0: stimulus port buffer is full (or port is disabled)
1: stimulus port can accept new write data

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]**: stimulus port enable

Each bit x(0 to 31) enables the stimulus port associated with the ITM_STIMRx register.

0: port disabled

1: port enabled

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]**: disable unprivileged access to ITM stimulus ports

Each bit controls eight stimulus ports.

XXX0: unprivileged access permitted on ports 0 to 7

XXX1: only privileged access permitted on ports 0 to 7

XX0X: unprivileged access permitted on ports 8 to 15

XX1X: only privileged access permitted on ports 8 to 15

X0XX: unprivileged access permitted on ports 16 to 23

X1XX: only privileged access permitted on ports 16 to 23

0XXX: unprivileged access permitted on ports 24 to 31

1XXX: only privileged access permitted on ports 24 to 31

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 identifiers 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
0x0: no prescaling
0x1: Divide by 4.
0x2: Divide by 16.
0x3: Divide 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 data trace packets.

Bit 4 **SWOENA**: SWO enable
Enables asynchronous clocking of the timestamp counter (read only).
0: Timestamp counter uses processor clock.

Bit 3 **TXENA**: transmit enable
Enables forwarding of hardware event packets from the DWT unit to the trace port.
0: disabled
1: enabled

Bit 2 **SYNCENA**: synchronization packet transmission enable
The debugger setting this bit must also configure the DWT_CTRLR.SYNCTAP field for the correct synchronization speed.
0: disabled
1: enabled

Bit 1 **TSENA**: local timestamp generation enable
0: disabled
1: enabled

Bit 0 **ITMENA**: ITM enable
0: disabled
1: enabled

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® limited.

Bit 20 **PRESENT**: DEVARCH register presence
0x1: present

Bits 19:16 **REVISION[3:0]**: architecture revision
0x0: ITM architecture v2.0

Bits 15:12 **ARCHVER[3:0]**: architecture version
0x1: ITM architecture v2.0

Bits 11:0 **ARCHPART[11:0]**: architecture part
0xA01: ITM architecture

ITM device type register (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

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.	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

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

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

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 identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

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

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 identification bits [7:0]
0x0D: Common identification value

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 identification bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

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 identification bits [23:16]
0x05: Common identification value

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 identification bits [31:24]
0xB1: Common identification value

41.7.2 ITM register map

The ITM registers are located at address range 0xE000 0000 to 0xE000 0FFC.

Table 371. 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
0x07C to 0xDFC	Reserved	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

Table 371. 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
0xE04 to 0xE3C	Reserved	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	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	Reserved																															
0xFBC	**ITM_DEVARCHR**	ARCHITECT[10:0]											PRESEN	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[3:0]											
	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	Reserved																															
0xFCC	**ITM_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[3:0]			
	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.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	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
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 *Table 366: Processor ROM table* for register boundary addresses.

41.8 Breakpoint unit (BPU)

The BPU allows the user to set hardware breakpoints. 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 breakpoint unit and how to use it, refer to *[4]*.

41.8.1 BPU registers

The BPU registers are located at address range 0xE0002000 to 0xE0002FFC.

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
0x08: 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 FPB_CTRLR register is ignored if this bit is not set to 1.

Bit 0 **ENABLE**: FPB enable
0: disabled
1: enabled

BPU comparator x register (BPU_COMPxR)

Address offset: 0x008 + 0x004 * 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

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® limited.

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

BPU device type register (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

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.	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

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

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

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
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm ® JEDEC code

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

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 identification bits [7:0]
0x0D: common identification value

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 identification bits [15:12] - component class
0x9: debug component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

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 identification bits [23:16]
0x05: common identification value

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 identification bits [31:24]
0xB1: common identification value

41.8.2 BPU register map

The BPU registers are located at address range 0xE000 2000 to 0xE000 2FFC.

Table 372. 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**	REV[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	Reserved																															

Table 372. BPU 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 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
0x028 to 0xFB8	Reserved	Reserved																															
0xFBC	**BPU_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	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	1
0xFC0 to 0xFC8	Reserved	Reserved																															
0xFCC	**BPU_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[3:0]			
	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.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	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 *Table 366: Processor ROM table* for register boundary addresses.

41.9 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 the STM32H503xx, 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]*.

41.9.1 ETM registers

The ETM registers are located at 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 (including normal and special keys)
0x1: one conditional instruction right-hand key 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

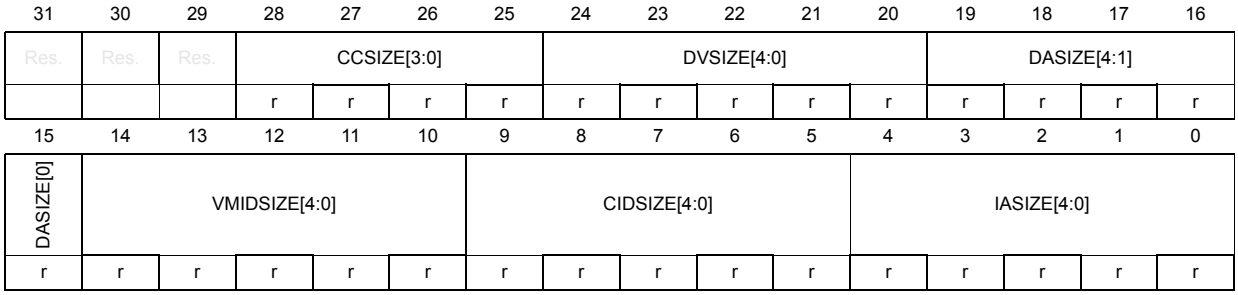
Bits 7:4 **TRCARCHMIN[3:0]**: minor trace unit architecture version number
0x2: minor revision 2

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



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

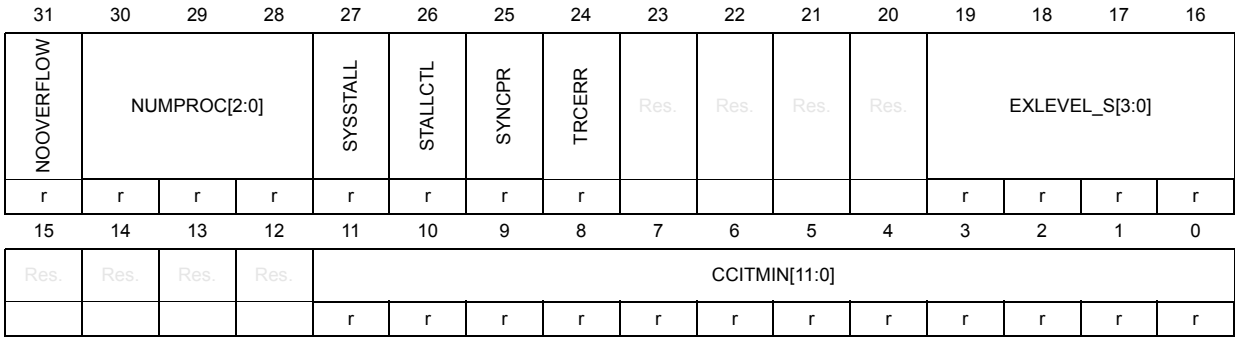
Bits 9:5 **CIDSIZE[4:0]**: context ID size
0x0: context ID tracing not implemented

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



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 noninvasive debug
0x2: secure noninvasive debug disabled

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 noninvasive debug
0x2: nonsecure noninvasive debug disabled
0x3: nonsecure noninvasive 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

41.9.2 ETM register map

The ETM registers are accessed by the debugger at address range 0xE0041000 to 0xE0041FFC.

Table 373. 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	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 to 0x01C	Reserved	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	Reserved																															

Table 373. 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
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	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.	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	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 to 0x07C	Reserved	Reserved																															
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	SSSTATUS	Res.	EVENT[7:0]							
	Reset value													X	X	X	X					X	X	X		X	X	X	X	X	X	X	X
0x084 to 0x13C	Reserved	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	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 to 0x1BC	Reserved	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 to 0x1DC	Reserved	Reserved																															

Table 373. 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
0x1E0	**ETM_IDR0**	Res.	Res.	COMMOPT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCEXDATA	QSUPP[1:0]			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
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 to 0x204	Reserved	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 to 0x27C	Reserved	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 to 0x29C	Reserved	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 to 0x2BC	Reserved	Reserved																															

Table 373. 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
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 to 0x30C	Reserved	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 to 0xF9C	Reserved	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
0xFA8 to 0xFB4	Reserved	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
0xFC0to 0xFC8	Reserved	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
0xFD4to 0xFDC	Reserved	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

Table 373. ETM register map and reset values (continued)

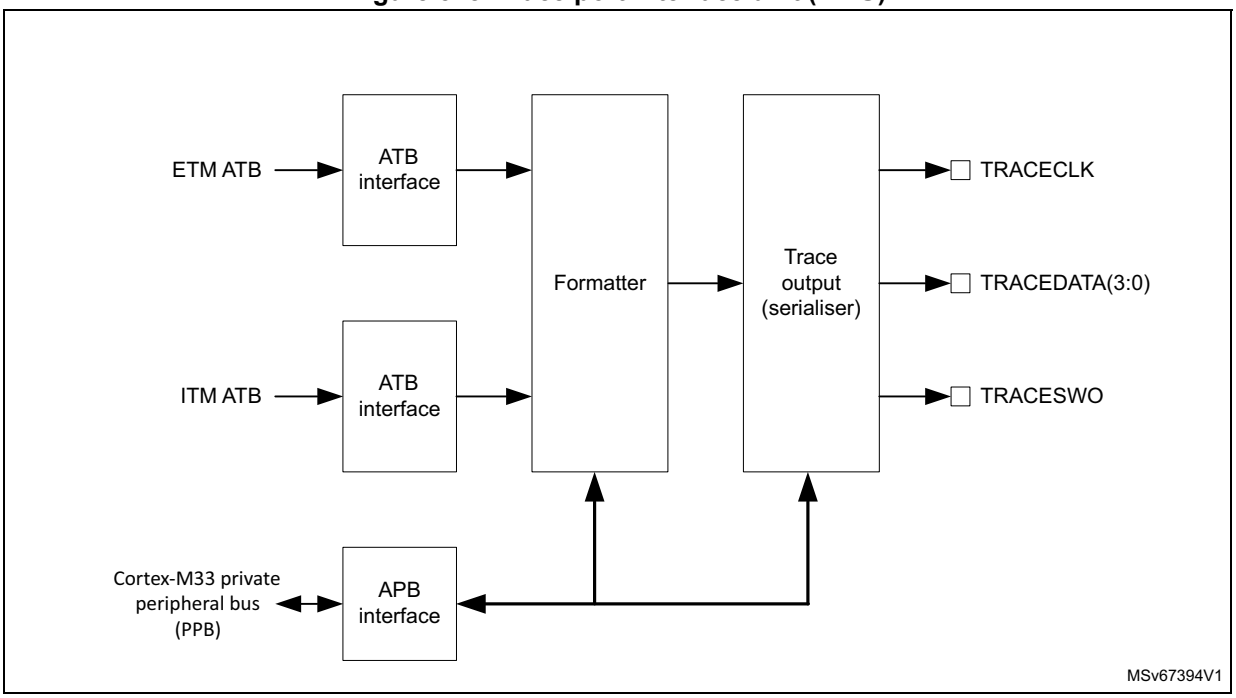
Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF0	**ETM_CIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									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 *Table 366: Processor ROM table* for register boundary addresses.

41.10 Trace port interface unit (TPIU)

The TPIU formats the trace stream and outputs it on the external trace port signals. As shown in the figure below, the TPIU has two ATB slave ports for incoming trace data from the ETM and ITM respectively. The trace port is 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 IOs 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 518. Trace port interface unit (TPIU)



Trace data can also be output on the serial-wire output, TRACESWO.

For more information on the trace port interface in the Cortex®-M33, refer to the Arm® Cortex®-M33 Technical Reference Manual *[5]*.

41.10.1 TPIU registers

TPIU supported port size register (TPIU_SSPSR)

Address offset: 0x000

Reset value: 0x0000 000F

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]**: trace port sizes, from 1 to 32 pins

Bit n-1 when set, indicates that port size n is supported.

0x0000 000F: port sizes 1 to 4 supported

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]															
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]**: 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 sizes (1-4). Only one bit can be set, or unpredictable behavior may result.

This register must only be modified when the formatter is stopped.

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]**: baud rate for the asynchronous output, TRACESWO
The baud rate is given by the TRACELKIN frequency divided by (PRESCALER +1).

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]**: protocol used for trace output
0x0: parallel trace port mode
0x1: asynchronous SWO using Manchester encoding
0x2: asynchronous SWO using NRZ encoding
0x3: reserved

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**: formatter stop

Indicates whether formatter can be stopped or not.

1: The formatter cannot be stopped.

Bit 2 **TCPRESENT**: TRACECTL output pin availability

Indicates whether the optional TRACECTL output pin is available for use.

0: TRACECTL pin is not present in this device.

Bit 1 **FTSTOPPED**: formatter stop

The formatter has 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: The formatter has not stopped.

Bit 0 **FLINPROG**: flush in progress

Indicates whether a flush on the ATB slave port is in progress. This bit 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

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: Generates a flush.

Bits 5:2 Reserved, must be kept at reset value.

Bit 1 **ENFCONT:** continuous formatting enable

Setting this bit to zero in SWO mode bypasses the formatter and only ITM/DWT trace is output, ETM trace is discarded.

0: continuous formatting disabled

1: continuous formatting enabled

Bit 0 Reserved, must be kept at reset value.

TPIU periodic synchronization counter register (TPIU_PSCR)

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.

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]**: 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.

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]**: 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.

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]			CLKRELAT	MAXNUM[4:0]				
				r	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 support
0x1: supported

Bit 10 **SWOMAN**: Serial-wire output, Manchester encoded format, support
0x1: supported

Bit 9 **TCLKDATA**: trace clock plus data support
0x0: supported

Bits 8:6 **FIFOSIZE[2:0]**: FIFO size in powers of 2
0x2: FIFO size = 4 bytes

Bit 5 **CLKRELAT**: ATB clock and TRACECLKIN relationship (synchronous or asynchronous)
0x1: asynchronous

Bits 4:0 **MAXNUM[4:0]**: number/type of ATB input port multiplexing
0x1: two input ports

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

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.	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

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

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

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 identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

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

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 identification bits [7:0]
0x0D: common identification value

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 identification bits [11:8]
0x0: common identification value

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 identification bits [23:16]
0x05: common identification value

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 identification bits [31:24]
0xB1: common identification value

41.10.2 TPIU register map

Table 374. 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	1	1	1	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	Reserved	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	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	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_PSCR**	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

Table 374. TPIU register map and reset values (continued)

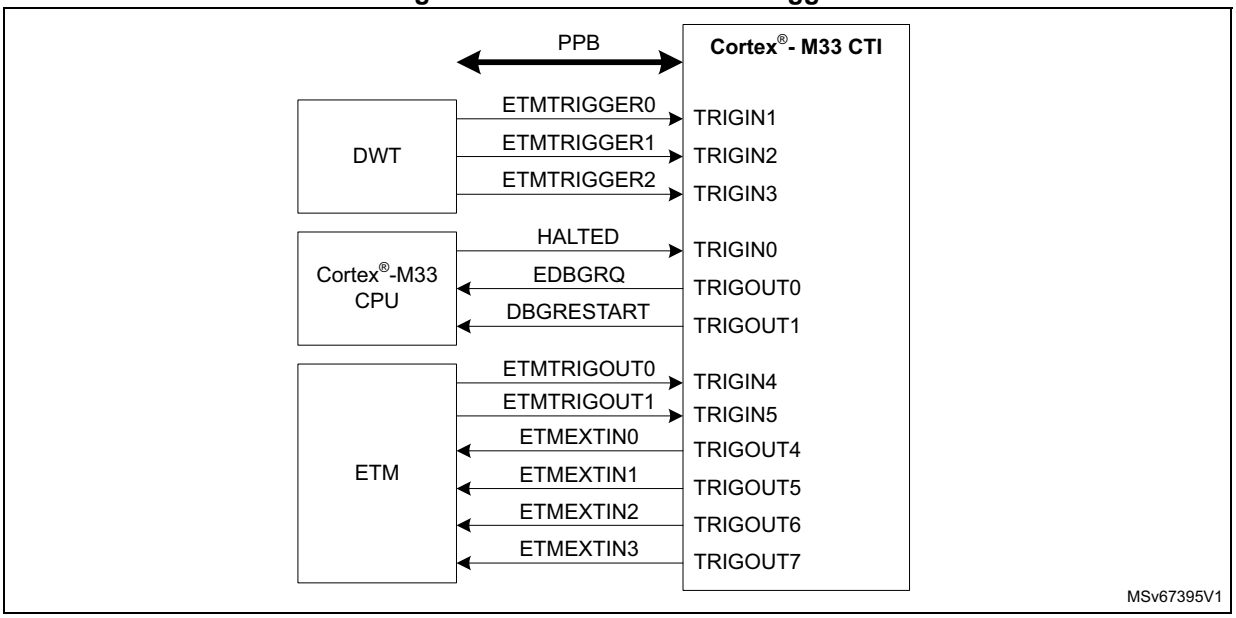
Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
030C to 0xF9C	Reserved	Reserved																															
0xFA0	**TPIU_CLAIMSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0FA8 to 0xFC4	Reserved	Reserved																															
0xFC8	**TPIU_DEVIDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MaXNUM[4:0]				
	Reset value																					1	1	0	0	1	0	1	0	0	0	0	1
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.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
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 *Table 365: MCU ROM table* for register boundary addresses.

41.11 Cross-trigger interface (CTI)

The CTI allows cross triggering between the processor and the ETM (see the figure below).

Figure 519. Embedded cross trigger



The CTI enables events from various sources to trigger debug and/or trace activity. For example, a watchpoint reached in the processor can start or stop the code trace, or a trace comparator can halt the processor.

The trigger input and output signals for the CTI are listed in the tables below.

Table 375. CTI inputs

Number	Source signal	Source component	Comments
0	HALTED	CPU	Processor halted - 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 376. CTI outputs

Number	Source signal	Destination component	Comments
0	EDBGRQ	CPU	CPU halt request - Puts CPU in debug mode
1	DBGRESTART	CPU	CPU restart request - CPU exits debug mode

Table 376. CTI outputs (continued)

Number	Source signal	Destination component	Comments
2	ETMEXTIN0	ETM	ETM event input 0
3	ETMEXTIN1	ETM	ETM event input 1
4	ETMEXTIN2	ETM	ETM event input 2
5	ETMEXTIN3	ETM	ETM event input 3
6	-	-	Not used
7	-	-	Not used

For more information on the cross-trigger interface CoreSight™ component, refer to the Arm® CoreSight™ SoC-400 Technical Reference Manual *[2]*.

41.11.1 CTI registers

The register file base address for the CTI is 0xE004 2000.

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 CTI enable
0: disabled
1: enabled

CTI trigger acknowledge register (CTI_INTACKR)

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.	Res.	Res.	Res.	Res.	Res.	INTACK[7:0]							
								w	w	w	w	w	w	w	w

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 CTITRIGOUT output. When a 1 is written to a bit in this register, the corresponding CTITRIGOUT output is acknowledged, causing it 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
XXX0: 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:
XXX0: no effect
XXX0: Sets event on channel 0.
XX0X: no effect
XX1X: Sets event on channel 1.
X0XX: no effect
X1XX: Sets event on channel 2.
0XXX: no effect
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]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPCLEAR[3:0]**: channel event clear
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: 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.	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 input x enable register (CTI_INENxR)

Address offset: 0x020 + 0x004 * 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]**: trigger input event enable

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 channels.

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 output x enable register (CTI_OUTENxR)

Address offset: 0x0A0 + 0x004 * 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]**: trigger output event enable

For each channel, defines whether an event on that channel generates a trigger on CTITRIGOUTx (x = 0 to 7).

0000: Channel events do not generate triggers on trigger outputs.

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 input 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 CTITRIGINx 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 output 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 CTITRIGOUT 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 input 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 output 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 available
0x4: four channels

Bits 15:8 **NUMTRIG[7:0]**: number of ECT triggers available
0x8: height trigger inputs and height 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: cross-triggering component.

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.	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

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 identification 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 identification bits [7:0]
0x0D: common identification 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 identification bits [15:12] - component class
0x9: CoreSight™ component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification 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 identification bits [23:16]
0x05: common identification 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 identification bits [31:24]
0xB1: common identification value

41.11.2 CTI register map

Table 377. 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	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																									X	X	X	X	X	X	X	X

Table 377. 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
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																													X	X	X	X
0x020 to 0x03C	**CTI_INEN0R to CTI_INEN7R**	Res.	Res.	Res.	Res.	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	Reserved																															
0x0A0 to 0x0BC	**CTI_OUTEN0R to CTI_OUTEN7R**	Res.	Res.	Res.	Res.	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	Reserved																															
0x130	**CTI_TRGISTSR**	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_TRGOSTSR**	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.	CHINSTATUS [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.	CHOUTSTATUS[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	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
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.	SUB[3:0]				MAJOR[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.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															

Table 377. 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
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	0	1	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 *Table 366: Processor ROM table* for register boundary addresses.

41.12 Microcontroller debug unit (DBGMCU)

The 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 software) to:

- maintain the clock and power to the processor cores when in low-power modes (sleep, stop or standby)
- maintain the clock and power to the system debug and trace components when in low-power modes
- stop the clock to certain peripherals (SMBUS timeout, watchdogs, timers, RTC) when either processor core is stopped in debug mode

41.12.1 Device ID

The DBGMCU includes an identity code register, DBGMCU_IDCODE. This register contains the ID code for the device. Debug tools can locate this register via the CoreSight™ discovery procedure described in *Section 41.5: ROM tables*.

41.12.2 Low-power mode emulation

When the device enters either stop mode (clocks are stopped) or standby mode (core power is switched off), the debugger can no longer access the debug access port and loses the connection with the device. To avoid this, the debugger (or software) can set the

DBG_STANDBY and/or DBG_STOP bits in the *DBGMCU configuration register (DBGMCU_CR)*. These bits, when set, maintain the clock and power to the processor while the device is in the corresponding low-power mode. The processor remains in Sleep mode, and exits the low-power mode in the normal way. However, peripheral devices continue to operate, so the device behavior may not be identical to that of the actual low-power mode.

41.12.3 Peripheral clock freeze

The DBGMCU peripheral clock freeze registers allow the operation of certain peripherals to be suspended in debug mode. The peripheral units, which support this feature are listed in the table below.

Table 378. Peripheral clock freeze control bits

Bus	Control register	Peripheral	Description
APB1L	DBGMCU_APB1LFZR	I3C1	I3C1 SCL stall timeout counter
		I2C2	I2C2 SMBUS timeout
		I2C1	I2C1 SMBUS timeout
		IWDG	Independent watchdog
		WWDG	Window watchdog
		TIM7	General purpose timer 7
		TIM6	General purpose timer 6
		TIM3	General purpose timer 3
		TIM2	General purpose timer 2
APB1H	DGBMCU_APB1HFZR	LPTIM2	Low power timer 2
APB2	DBGMCU_APB2FZR	TIM1	General purpose timer 1
APB3	DBGMCU_APB3FZR	RTC	Real time clock
		LPTIM1	Low power timer 1
		I3C2	I3C2 SCL stall timeout counter
AHB1	DBGMCU_AHB1FZR	GPDMA2 0 to 7	General purpose DMA2 channels 0 to 7
		GPDMA1 0 to 7	General purpose DMA1 channels 0 to 7

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_zzzFZR registers, where zzz 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 379. Peripheral behaviour in debug mode

HALTED	DBG_xxx_STOP	Peripheral behavior
0	X	The operation continues.
1	0	The operation continues.
1	1	The operation is suspended.

41.12.4 DBGMCU registers

The DBGMCU registers are not reset by a system reset, only by a power-on reset. They are accessible to the debugger via the AHB access port at base address 0xE00E 4000, and to software at base address 0x4402 4000.

DBGMCU identity code register (DBGMCU_IDCODE)

Address offset: 0x00

Reset value: 0xXXXX 6XXX

This register is always accessible.

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
This field indicates the revision of the device.
0x1000: Revision A
0x1001: Revision Z
0x1002: Revision Y

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID[11:0]**: device identification
The device ID is 0x474: STM32H503

DBGMCU configuration register (DBGMCU_CR)

Address offset: 0x04

Reset value: 0x0000 0000

This register is accessible to the debugger after successful authentication. Prior to this, debugger accesses are ignored. It is always accessible to software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCRT
															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.	TRACE_MODE[1:0]		TRACE_EN	TRACE_IOEN	Res.	DBG_STANDBY	DBG_STOP	Res.
								rw	rw	rw	rw		rw	rw	

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DCRT**: Debug credentials reset type
This bit selects which type of reset is used to revoke the debug authentication credentials
0: System reset
1: Power reset

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **TRACE_MODE[1:0]**: trace pin assignment
0x0: trace pins assigned for asynchronous mode (TRACESWO)
0x1: trace pins assigned for synchronous mode with a port width of 1 (TRACECK, TRACED0)
0x2: trace pins assigned for synchronous mode with a port width of 2 ((TRACECK, TRACED0-1)
0x3: trace pins assigned for synchronous mode with a port width of 4 ((TRACECK, TRACED0-3)

Bit 5 **TRACE_EN**: trace port and clock enable.
This bit enables the trace port clock, TRACECK.
0: disabled
1: enabled

Bit 4 **TRACE_IOEN**: trace pin enable
0: disabled - trace pins not assigned
1: enabled - trace pins assigned according to the value of TRACE_MODE field

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DBG_STANDBY**: Allows debug in standby mode

0: normal operation

All clocks are disabled and the core powered down automatically in standby mode.

1: automatic clock stop/power down disabled

All active clocks and oscillators continue to run during standby mode, and the core supply is maintained, allowing full debug capability. On exit from standby mode, a system reset is performed.

Bit 1 **DBG_STOP**: Allows debug in stop mode

0: normal operation

All clocks are disabled automatically in stop mode.

1: automatic clock stop disabled

All active clocks and oscillators continue to run during stop mode, allowing full debug capability. On exit from stop mode, the clock settings are set to the stop mode exit state.

Bit 0 Reserved, must be kept at reset value.

DBGMCU APB1L peripheral freeze register (DBGMCU_APB1LFZR)

Address offset: 0x08

Reset value: 0x0000 0000

This register is accessible to the debugger after successful authentication. Prior to this, debugger accesses are ignored. It is always accessible to software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I3C1_STOP	DBG_I2C2_STOP	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.
								rw	rw	rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	Res.	Res.	Res.	Res.	Res.	DBG_TIM7_STOP	DBG_TIM6_STOP	Res.	Res.	DBG_TIM3_STOP	DBG_TIM2_STOP
			rw	rw						rw	rw			rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **DBG_I3C1_STOP**: I3C1 SCL stall counter stop in debug

0: normal operation. I3C1 SCL stall timeout counter continues to operate while CPU is in debug mode.

1: stop in debug. I3C1 SCL stall timeout counter is frozen while CPU is in debug mode.

Bit 22 **DBG_I2C2_STOP**: I2C2 SMBUS timeout stop in debug

0: normal operation. I2C2 SMBUS timeout continues to operate while CPU is in debug mode.

1: stop in debug. I2C2 SMBUS timeout is frozen while CPU is in debug mode.

Bit 21 **DBG_I2C1_STOP**: I2C1 SMBUS timeout stop in 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 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.

Bit 11 **DBG_WWDG_STOP**: WWDG stop in debug
0: normal operation. WWDG continues to operate while CPU is in debug mode.
1: stop in debug. WWDG is frozen while CPU is in debug mode.

Bits 10:6 Reserved, must be kept at reset value.

Bit 5 **DBG_TIM7_STOP**: TIM7 stop in debug
0: normal operation. TIM7 continues to operate while CPU is in debug mode.
1: stop in debug. TIM7 is frozen while CPU is in debug mode.

Bit 4 **DBG_TIM6_STOP**: TIM6 stop in debug
0: normal operation. TIM6 continues to operate while CPU is in debug mode.
1: stop in debug. TIM6 is frozen while CPU is in debug mode.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **DBG_TIM3_STOP**: TIM3 stop in debug
0: normal operation. TIM3 continues to operate while CPU is in debug mode.
1: stop in debug. TIM3 is frozen while CPU is in debug mode.

Bit 0 **DBG_TIM2_STOP**: TIM2 stop in 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: 0x0C

Reset value: 0x0000 0000

This register is accessible to the debugger after successful authentication. Prior to this, debugger accesses are ignored. It is always accessible to software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 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: 0x10

Reset value: 0x0000 0000

This register is accessible to the debugger after successful authentication. Prior to this, debugger accesses are ignored. It is always accessible to software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DBG_TIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw											

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **DBG_TIM1_STOP**: TIM1 stop in debug
0: normal operation. TIM1 continues to operate while CPU is in debug mode.
1: stop in debug. TIM1 is frozen while CPU is in debug mode.

Bits 10:0 Reserved, must be kept at reset value.

DBGMCU APB3 peripheral freeze register (DBGMCU_APB3FZR)

Address offset: 0x14

Reset value: 0x0000 0000

This register is accessible to the debugger after successful authentication. Prior to this, debugger accesses are ignored. It is always accessible to software.

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.	DBG_I3C2_STOP	Res.	Res.	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 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 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 16:13 Reserved, must be kept at reset value.

Bit 12 **DBG_I3C2_STOP**: I3C2 SCL stall counter stop in debug

0: normal operation. I3C2 SCL stall timeout counter continues to operate while CPU is in debug mode.

1: stop in debug. I3C2 SCL stall timeout counter is frozen while CPU is in debug mode.

Bits 11:0 Reserved, must be kept at reset value.

DBGMCU AHB1 peripheral freeze register (DBGMCU_AHB1FZR)

Address offset: 0x20

Reset value: 0x0000 0000

This register is accessible to the debugger after successful authentication. Prior to this, debugger accesses are ignored. It is always accessible to software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_GPDMA2_7_STOP	DBG_GPDMA2_6_STOP	DBG_GPDMA2_5_STOP	DBG_GPDMA2_4_STOP	DBG_GPDMA2_3_STOP	DBG_GPDMA2_2_STOP	DBG_GPDMA2_1_STOP	DBG_GPDMA2_0_STOP
								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.	DBG_GPDMA1_7_STOP	DBG_GPDMA1_6_STOP	DBG_GPDMA1_5_STOP	DBG_GPDMA1_4_STOP	DBG_GPDMA1_3_STOP	DBG_GPDMA1_2_STOP	DBG_GPDMA1_1_STOP	DBG_GPDMA1_0_STOP
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **DBG_GPDMA2_7_STOP**: GPDMA2 channel 7 stop in debug
0: normal operation. GPDMA2 channel 7 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 7 is frozen while CPU is in debug mode.

Bit 22 **DBG_GPDMA2_6_STOP**: GPDMA2 channel 6 stop in debug
0: normal operation. GPDMA2 channel 6 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 6 is frozen while CPU is in debug mode.

Bit 21 **DBG_GPDMA2_5_STOP**: GPDMA2 channel 5 stop in debug
0: normal operation. GPDMA2 channel 5 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 5 is frozen while CPU is in debug mode.

Bit 20 **DBG_GPDMA2_4_STOP**: GPDMA2 channel 4 stop in debug
0: normal operation. GPDMA2 channel 4 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 4 is frozen while CPU is in debug mode.

Bit 19 **DBG_GPDMA2_3_STOP**: GPDMA2 channel 3 stop in debug
0: normal operation. GPDMA2 channel 3 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 3 is frozen while CPU is in debug mode.

Bit 18 **DBG_GPDMA2_2_STOP**: GPDMA2 channel 2 stop in debug
0: normal operation. GPDMA2 channel 2 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 2 is frozen while CPU is in debug mode.

Bit 17 **DBG_GPDMA2_1_STOP**: GPDMA2 channel 1 stop in debug
0: normal operation. GPDMA2 channel 1 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 1 is frozen while CPU is in debug mode.

Bit 16 **DBG_GPDMA2_0_STOP**: GPDMA2 channel 0 stop in debug
0: normal operation. GPDMA2 channel 0 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 0 is frozen while CPU is in debug mode.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **DBG_GPDMA1_7_STOP**: GPDMA1 channel 7 stop in debug
0: normal operation. GPDMA1 channel 7 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 7 is frozen while CPU is in debug mode.

Bit 6 **DBG_GPDMA1_6_STOP**: GPDMA1 channel 6 stop in debug
0: normal operation. GPDMA1 channel 6 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 6 is frozen while CPU is in debug mode.

Bit 5 **DBG_GPDMA1_5_STOP**: GPDMA1 channel 5 stop in debug
0: normal operation. GPDMA1 channel 5 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 5 is frozen while CPU is in debug mode.

Bit 4 **DBG_GPDMA1_4_STOP**: GPDMA1 channel 4 stop in debug
0: normal operation. GPDMA1 channel 4 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 4 is frozen while CPU is in debug mode.

Bit 3 **DBG_GPDMA1_3_STOP**: GPDMA1 channel 3 stop in debug
0: normal operation. GPDMA1 channel 3 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 3 is frozen while CPU is in debug mode.

Bit 2 **DBG_GPDMA1_2_STOP**: GPDMA1 channel 2 stop in debug
0: normal operation. GPDMA1 channel 2 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 2 is frozen while CPU is in debug mode.

Bit 1 **DBG_GPDMA1_1_STOP**: GPDMA1 channel 1 stop in debug
0: normal operation. GPDMA1 channel 1 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 1 is frozen while CPU is in debug mode.

Bit 0 **DBG_GPDMA1_0_STOP**: GPDMA1 channel 0 stop in debug
0: normal operation. GPDMA1 channel 0 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 0 is frozen while CPU is in debug mode.

DBGMCU status register (DBGMCU_SR)

Address offset: 0xFC

Reset value: 0x0001 XX03

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AP_ENABLED[15:0]															
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
AP_PRESENT[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **AP_ENABLED[15:0]**: Bit n identifies whether access port AP n is open (can be accessed via the debug port) or locked (debug access to the AP is blocked)
Bit n = 0: APn locked
Bit n = 1: APn enabled

Bits 15:0 **AP_PRESENT[15:0]**: Bit n identifies whether access port AP n is present in device
Bit n = 0: APn absent
Bit n = 1: APn present

DBGMCU debug authentication mailbox host register (DBGMCU_DBG_AUTH_HOST)

Address offset: 0x100

Reset value: 0xXXXX XXXX

This register is read only when accessed by the CPU, writes have no effect.

This register can be written and read by an external debugger.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MESSAGE[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
MESSAGE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **MESSAGE[31:0]**: Debug host to device mailbox message.
During debug authentication the debug host communicates with the device via this register.

DBGMCU debug authentication mailbox device register (DBGMCU_DBG_AUTH_DEVICE)

Address offset: 0x104

Reset value: 0xXXXX XXXX

This register is read only when accessed via the debug port, writes have no effect.

This register can be written and read by the CPU.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MESSAGE[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
MESSAGE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MESSAGE[31:0]**: Device to debug host mailbox message.
During debug authentication the device communicates with the debug host via this register.

DBGMCU debug authentication mailbox acknowledge register (DBGMCU_DBG_AUTH_ACK)

Address offset: 0x108

Reset value: 0x0000 0000

This register is always accessible.

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DEV_ACK**: Device to host acknowledge.
The host sets this bit to indicate that it has placed a message in the DBGMCU_DBG_AUTH_HOST register. It is reset by the device after reading the message

Bit 0 **HOST_ACK**: Host to device acknowledge.
The device sets this bit to indicate that it has placed a message in the DBGMCU_DBG_AUTH_DEVICE register. It should be reset by the host after reading the message

DBGMCU CoreSight peripheral identity register 4 (DBGMCU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x0: STMicroelectronics JEDEC code

DBGMCU CoreSight peripheral identity register 0 (DBGMCU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0000

This register is always accessible.

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

This register is always accessible.

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

This register is always accessible.

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

This register is always accessible.

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

This register is always accessible.

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

DBGMCU CoreSight component identity register 1 (DBGMCU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00F0

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0xF: Non-CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

DBGMCU CoreSight component identity register 2 (DBGMCU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

This register is always accessible.

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

DBGMCU CoreSight component identity register 3 (DBGMCU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

This register is always accessible.

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

41.12.5 DBGMCU register map

Table 380. 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

Table 380. 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
0x004	**DBGMCU_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCRT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACE_MODE [1:0]		TRACE_EN	TRACE_IOEN	Res.	DBG_STANDBY	DBG_STOP	Res.
	Reset value																0									0	0	0	0		0	0	
0x008	**DBGMCU_ APB1LFZR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I3C1_STOP	DBG_I2C2_STOP	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	Res.	Res.	Res.	Res.	Res.	DBG_TIM7_STOP	DBG_TIM6_STOP	Res.	Res.	DBG_TIM3_STOP	DBG_TIM2_STOP
	Reset value									0	0	0									0	0						0	0			0	0
0x0C0	**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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																					0											
0x04	**DBGMCU_ APB3FZR**	Res.	DBG_RTC_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM1_STOP	Res.	Res.	Res.	Res.	DBG_I3C2_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0													0					0												
0x018 to 0x01C	Reserved	Reserved																															
0x020	**DBGMCU_ AHB1FZR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_GPDMA2_7_STO	DBG_GPDMA2_6_STO	DBG_GPDMA2_5_STO	DBG_GPDMA2_4_STO	DBG_GPDMA2_3_STO	DBG_GPDMA2_2_STO	DBG_GPDMA2_1_STO	DBG_GPDMA2_0_STO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_GPDMA1_7_STO	DBG_GPDMA1_6_STO	DBG_GPDMA1_5_STO	DBG_GPDMA1_4_STO	DBG_GPDMA1_3_STO	DBG_GPDMA1_2_STO	DBG_GPDMA1_1_STO	DBG_GPDMA1_0_STO
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x024 to 0x0F8	Reserved	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	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0x100	**DBGMCU_DBG_ AUTH_HOST**	MESSAGE[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

Table 380. 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
0x104	**DBGMCU_DBG_AUTH_DEVICE**	MESSAGE[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	**DBGMCU_DBG_AUTH_ACK**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEV_ACK	HOST_ACK
	Reset value																															0	0
0x10C to 0xFBC	Reserved	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.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4 to 0xFDC	Reserved	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 2.2* for register boundary addresses.

41.13 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® CoreSight™ ETM-M33 r0p1 Technical Reference Manual, Issue 0001-00, 3 February 2017

42 Device electronic signature

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 automatically match to the characteristics of the devices.

42.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited:

- for use as serial numbers (for example USB string serial numbers or other end applications)
- for use as part of the security keys in order to increase the security of code in Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the internal Flash memory
- to activate secure boot processes

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits cannot be altered by the user.

Base address: 0x08FF F800

Address offset: 0x00

Read only = 0xXXXX XXXX where X is factory-programmed

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

Address offset: 0x04

Read only = 0xXXXX XXXX where 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 31:8 **UID[63:40]:** LOT_NUM[23:0]
Lot number (ASCII encoded)

Bits 7:0 **UID[39:32]:** WAF_NUM[7:0]
Wafer number (8-bit unsigned number)

Address offset: 0x08

Read only = 0xXXXX XXXX where 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 (ASCII encoded)

42.2 Flash size data register

Base address: 0x08FF F80C

Address offset: 0x00

Read only = 0xXXXX where X is factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_SIZE															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **FLASH_SIZE[15:0]**: Flash memory size
This field indicates the size of the device Flash memory expressed in Kbytes.
As an example, 0x080 corresponds to 128 Kbytes.

42.3 Package data register

Base address: 0x08FF F80E

Address offset: 0x00

Read only = 0xXXXX where X is factory-programmed

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 15:5 Reserved, must be kept at reset value.

Bits 4:0 **PKG[4:0]**: Package type
00000: LQFP64
00101: LQFP48
01001: UFQFPN32
01111: WLCSP25
10000: UFQFPN48
Others: reserved

43 Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST's customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on *www.st.com* for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers should re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers should not assume it is certified.
- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.
- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.
- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.
- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.

44 Revision history

Table 381. Document revision history

Date	Revision	Changes
01-Mar-2023	1	First release.
16-Mar-2023	2	Title updated.
20-Nov-2024	3	**Documentation conventions:** Added *Section 1.3: Register reset value*. **System secuirty:** Updated *Figure 4: Device life-cycle security*. Updated *Figure 5: Product life-cycle*. **RAMCFG:** Updated *Section 5.3.2: Error code correction (SRAM1, SRAM2, BKPSRAM)*. **Flash:** Updated *Table 26: Option-byte organization*. Updated *Section 7.10.11: FLASH option status register (FLASH_OPTSR_CUR)*. **ICACHE:** Updated *Section 8.4: ICACHE functional description* introduction. Updated *Section 8.4.6: Cacheable and noncacheable traffic*. Updated *Section 8.4.8: ICACHE maintenance*. Updated *Section 8.7.1: ICACHE control register (ICACHE_CR)*. **PWR:** Updated *Section 9.4.7: Backup domain*. **PLL:** Updated *Section 10.4.5: PLL description*. Updated *Section 10.4.6: LSE clock*. Updated *Section 10.4.10: Clock security system (CSS)*. Updated *Section 10.8.7: RCC PLL clock source selection register (RCC_PLL1CFGR)*. Updated *Section 10.8.8: RCC PLL clock source selection register (RCC_PLL2CFGR)*. **CRS:** Updated *Section 11.1: CRS introduction*. Updated *Section 11.2: CRS main features*.

Table 381. Document revision history

Date	Revision	Changes
20-Nov-2024	3 (continued)	Updated *Table 72: CRS internal input/output signals*. Updated *Figure 38: CRS block diagram*. Updated *Section 11.4.3: Synchronization input*. Updated *Section 11.5: CRS in low-power modes*. Updated *Section 11.7.1: CRS control register (CRS_CR)*. Updated *Section 11.7.2: CRS configuration register (CRS_CFGR)*. **GPIO:** Updated *Section 12.3.2: I/O pin alternate function multiplexer and mapping*. **SBS:** Updated *Section 13.5.7: SBS memory erase status register (SBS_MESR)*. Updated *Section 13.5.8: SBS compensation cell for I/Os control and status register (SBS_CCCSR)*. **GPDMA:** Updated *Section 15.2: GPDMA main features*. Updated *Section 15.3.3: GPDMA in low-power modes*. Updated *Section : GPDMA data handling: byte-based reordering, packing/unpacking, padding/truncation, sign extension and left/right alignment*. Updated *Section : GPDMA arbitration and bandwidth*. Deleted Section : GPDMA autonomous mode. Updated *Section 15.6: GPDMA in low-power modes*. Updated *Section 15.8.7: GPDMA channel x transfer register 1 (GPDMA_CxTR1)*. Updated *Section 15.8.8: GPDMA channel x transfer register 2 (GPDMA_CxTR2)*. Updated *Section 15.8.12: GPDMA channel x destination address register (GPDMA_CxDAR)*. Updated *Section 15.8.15: GPDMA channel x linked-list address register (GPDMA_CxLLR)*. Updated *Section 15.8.16: GPDMA channel x alternate linked-list address register (GPDMA_CxLLR)*. **NVIC:** Updated *Table 97: STM32H503xx vector table*. **EXTI:** Updated *Table 100: EXTI line connections*.

Table 381. Document revision history

Date	Revision	Changes
20-Nov-2024	3 (continued)	**ADC:** Updated *Section 19.3: ADC implementation*. Updated *Figure 81: Enabling / disabling the ADC*. Updated *Section 19.4.11: Channel selection (SQRx, JSQRx)*. Updated *Section 19.4.12: Channel-wise programmable sampling time (SMPR1, SMPR2)*. Updated *Figure 83: Analog-to-digital conversion time*. Updated *Figure 99: Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 1)*. Updated *Section : Analog watchdog threshold control*. **DAC:** Updated *Figure 134: Dual-channel DAC block diagram*. Updated *Section 21.4.6: DAC conversion*. Updated *Section 21.4.7: DAC output voltage*. Updated *Section 21.4.9: DMA requests*. Updated *Section 21.4.13: DAC channel buffer calibration*. **RNG:** Updated *Section 24.1: Introduction*. Updated *Section : Noise source*. Updated *Section 24.3.8: RNG low-power use*. Updated *Table 154: RNG interrupt requests*. Updated *Section 24.5: RNG processing time*. Added *Table 155: RNG initialization times*. Updated *Section 24.6.2: Validation conditions*. Updated *Table 156: RNG configurations*. Updated *Table 157: Configuration selection*. Updated *Section 24.7.1: RNG control register (RNG_CR)*. Added *Section 24.7.4: RNG noise source control register (RNG_NSCR)*. **HASH:** Updated *Section 25.4.3: About secure hash algorithms*. Updated *Section 25.4.6: Message padding*. Deleted Section : HASH interrupts. Updated *Section 25.6.1: HASH control register (HASH_CR)*. Updated *Section 25.6.2: HASH data input register (HASH_DIN)*.

Table 381. Document revision history

Date	Revision	Changes
20-Nov-2024	3 (continued)	Updated *Section 25.6.3: HASH start register (HASH_STR)*. **TIM1:** Updated *Figure 161: Advanced-control timer block diagram*. Updated *Figure 166: TIM internal input/output signals*. Updated *Figure 184: tim_ti2 external clock connection example*. Updated *Section 26.3.25: Encoder interface mode*. Updated *Section 26.6: TIM1 registers* introduction. Updated *Section 26.6.3: TIM1 slave mode control register (TIM1_SMCR)*. Updated *Section 26.6.20: TIM1 break and dead-time register (TIM1_BDTR)* introduction. **TIM2/3:** Updated *Figure 251: General-purpose timer block diagram*. Updated *Figure 272: tim_ti2 external clock connection example*. Updated *Figure 291: OCREF_CLR input selection multiplexer*. Updated *Section 27.5.3: TIMx slave mode control register (TIMx_SMCR)(x = 2, 3)*. Updated *Section 27.5.10: TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2, 3)*. **LPTIM:** Updated *Figure 347: LPTIM1/2 block diagram*[(1)]. Updated *Section 29.4.14: Timer counter reset*. Updated *Section 29.4.20: DMA requests*. Updated *Section 29.7.2: LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1, 2)*. Updated *Section 29.7.11: LPTIM counter register (LPTIM_CNT)*. **IWDG:** Whole section re-edited. **RTC:** Updated *Section 32.3.5: Clock and prescalers*. Updated *Section 32.3.18: Tamper and alarm output*. Updated *Section 32.6.19: RTC status register (RTC_SR)*. Updated *Section 32.6.20: RTC masked interrupt status register (RTC_MISR)*. **TAMP:** Updated *Section 33.3.2: TAMP pins and internal signals*.

Table 381. Document revision history

Date	Revision	Changes
20-Nov-2024	3 (continued)	Updated *Section 33.3.8: TAMP backup registers and other device secrets erase*. Updated *Section : Trigger output generation on tamper event*. Updated *Section : Level detection with filtering on tamper inputs (passive mode)*. Added *Figure 372: Tamper sampling with precharge pulse*. Added *Figure 373: Low level detection with precharge and filtering*. Updated *Section 33.4: TAMP low-power modes*. Updated *Section 33.6.17: TAMP backup x register (TAMP_BKPxR)*. **I2C:** Whole section re-edited. **I3C:** Updated *Section 35.14: I3C in low-power modes*. Updated *Section 35.16.28: I3C extended provisioned ID register (I3C_EPIDR)*. **USART/UART:** Whole section re-edited. **LPUART:** Whole section re-edited. **SAI:** Whole section re-edited. **FDCAN:** Whole section re-edited. **USB:** Updated *Figure 512: USB peripheral block diagram*. Added *Section 40.4.2: USB pins and internal signals*. Added *Section 40.4.3: USB reset and clocks*. Updated *Section 40.4.5: Description of USB blocks used in both Device and Host modes*. Updated *Figure 513: Packet buffer areas with examples of buffer description table locations*. Updated *Section 40.6: USB registers* introduction. Updated *Section 40.6.6: USB battery charging detector (USB_BCDR)* . Updated *Section 40.6.7: USB endpoint/channel n register (USB_CHEPnR)* . Deleted Section : Buffer descriptor table.

Table 381. Document revision history

Date	Revision	Changes
20-Nov-2024	3 (continued)	Added *Section 40.7: USBSRAM registers*. **DEBUG:** Updated *Section 41.5: ROM tables*. Updated *Section 41.12.4: DBGMCU registers* introduction.

Index

A

ADC_AWD2CR 596
ADC_AWD3CR 596
ADC_CALFACT 598
ADC_CCR 599
ADC_CFGR 577
ADC_CFGR2 581
ADC_CR 573
ADC_DIFSEL 597
ADC_DR 591
ADC_HWCFGR0 600
ADC_IER 571
ADC_IPDR 601
ADC_ISR 569
ADC_JDRy 595
ADC_JSQR 591
ADC_OFRy 594
ADC_OR 598
ADC_SIDR 602
ADC_SMPR1 583
ADC_SMPR2 584
ADC_SQR1 587
ADC_SQR2 588
ADC_SQR3 589
ADC_SQR4 590
ADC_TR1 585
ADC_TR2 586
ADC_TR3 587
ADC_VERR 601
AP0_CSWR 1687
AP1_CSWR 1688
APx_BASER 1690
APx_BDnR 1690
APx_DRWR 1690
APx_IDR 1691
APx_TAR 1689

B

BPU_CIDR0 1746
BPU_CIDR1 1746
BPU_CIDR2 1747
BPU_CIDR3 1747
BPU_COMPxR 1742
BPU_CTRLR 1741
BPU_DEVARCHR 1742
BPU_DEVTYPER 1743
BPU_PIDR0 1744
BPU_PIDR1 1744
BPU_PIDR2 1745
BPU_PIDR3 1745
BPU_PIDR4 1743

C

C1ROM_CIDR3 . . . . . 1700, 1706, 1711
COMP_CFGR1 671
COMP_CFGR2 673
COMP_ICFR 671
COMP_SR 670
CPUROM_CIDR0 1710
CPUROM_CIDR1 1710
CPUROM_CIDR2 1711
CPUROM_CIDR3 1711
CPUROM_MEMTYPER 1707
CPUROM_PIDR0 1708
CPUROM_PIDR1 1708
CPUROM_PIDR2 1709
CPUROM_PIDR3 1709
CPUROM_PIDR4 1707
CRC_CR 500
CRC_DR 499
CRC_IDR 499
CRC_INIT 501
CRC_POL 501
CRS_CFGR 338
CRS_CR 336
CRS_ICR 341
CRS_ISR 339
CTI_APPCLEAR 1794
CTI_APPPULSER 1794
CTI_APPSETR 1793
CTI_CHINSTSR 1797
CTI_CHOUTSTSR 1797
CTI_CIDR0 1802
CTI_CIDR1 1802
CTI_CIDR2 1803
CTI_CIDR3 1803
CTI_CONTROLR 1792
CTI_DEVIDR 1798
CTI_DEVTYPER 1799
CTI_GATER 1798
CTI_INENxR 1795
CTI_INTACKR 1792
CTI_OUTENxR 1795
CTI_PIDR0 1800
CTI_PIDR1 1800

CTI_PIDR2 1801
CTI_PIDR3 1801
CTI_PIDR4 1799
CTI_TRGISTSR 1796
CTI_TRGOSTSR 1796

D

DAC_CCR 655
DAC_CR 644
DAC_DHR12L1 649
DAC_DHR12L2 650
DAC_DHR12LD 652
DAC_DHR12R1 648
DAC_DHR12R2 650
DAC_DHR12RD 651
DAC_DHR8R1 649
DAC_DHR8R2 651
DAC_DHR8RD 652
DAC_DOR1 653
DAC_DOR2 653
DAC_MCR 656
DAC_SHHR 658
DAC_SHRR 659
DAC_SHSR1 657
DAC_SHSR2 658
DAC_SR 654
DAC_SWTRGR 648
DBGMCU_AHB1FZR 1812
DBGMCU_APB1HFZR 1810
DBGMCU_APB1LFZR 1809
DBGMCU_APB2FZR 1811
DBGMCU_APB3FZR 1811
DBGMCU_CIDR0 1819
DBGMCU_CIDR1 1819
DBGMCU_CIDR2 1820
DBGMCU_CIDR3 1820
DBGMCU_CR 1808
DBGMCU_DBG_AUTH_ACK 1816
DBGMCU_DBG_AUTH_DEVICE 1815
DBGMCU_DBG_AUTH_HOST 1815
DBGMCU_IDCODE 1807
DBGMCU_PIDR0 1817
DBGMCU_PIDR1 1817
DBGMCU_PIDR2 1818
DBGMCU_PIDR3 1818
DBGMCU_PIDR4 1816
DBGMCU_SR 1814
DP_ABORTR 1679
DP_CTRLSTATR 1680
DP_DLCR 1681
DP_DLPIDR 1682
DP_DPIDR 1678
DP_RDBUFFR 1684
DP_RESENDR 1683
DP_SELECTR 1683
DP_TARGETIDR 1682
DTS_CFGR1 615
DTS_DR 618
DTS_ICIFR 621
DTS_ITENR 620
DTS_ITR1 618
DTS_OR 622
DTS_RAMPVALR 617
DTS_SR 619
DTS_T0VALR1 617
DWT_CIDR0 1726
DWT_CIDR1 1726
DWT_CIDR2 1727
DWT_CIDR3 1727
DWT_COMPxR 1718
DWT_CPICNTR 1715
DWT_CTRLR 1713
DWT_CYCCNTR 1715
DWT_DEVARCHR 1722
DWT_DEVTYPER 1723
DWT_EXCCNTR 1716
DWT_FOLDCNTR 1717
DWT_FUNCTR0 1718
DWT_FUNCTR1 1719
DWT_FUNCTR2 1720
DWT_FUNCTR3 1721
DWT_LSUCNTR 1717
DWT_PCSR 1717
DWT_PIDR0 1724
DWT_PIDR1 1724
DWT_PIDR2 1725
DWT_PIDR3 1725
DWT_PIDR4 1723
DWT_SLPCNTR 1716

E

ETM_AUTHSTATR 1768
ETM_CCCTLR 1754
ETM_CIDR0 1772
ETM_CIDR1 1773
ETM_CIDR2 1773
ETM_CIDR3 1774
ETM_CLAIMCLR 1767
ETM_CLAIMSETR 1767
ETM_CNTRLDVR0 1755
ETM_CONFIGR 1750
ETM_DEVARCHR 1769

ETM_DEVTYPER 1769
ETM_EVENTCTL0R 1751
ETM_EVENTCTL1R 1752
ETM_IDR0 1758
ETM_IDR1 1759
ETM_IDR10 1756
ETM_IDR11 1757
ETM_IDR12 1757
ETM_IDR13 1757
ETM_IDR2 1760
ETM_IDR3 1760
ETM_IDR4 1761
ETM_IDR5 1762
ETM_IDR8 1756
ETM_IDR9 1756
ETM_IMSPECR0 1758
ETM_PDCR 1766
ETM_PDSR 1766
ETM_PIDR0 1770
ETM_PIDR1 1771
ETM_PIDR2 1771
ETM_PIDR3 1772
ETM_PIDR4 1770
ETM_PRGCTLR 1749
ETM_RSCTLR2 1763
ETM_RSCTLR3 1763
ETM_SSCCR0 1764
ETM_SSCSR0 1765
ETM_SSPCICR0 1765
ETM_STALLCTLR 1752
ETM_STATR 1750
ETM_SYNCPR 1753
ETM_TRACEIDR 1754
ETM_VICTLR 1754
EXTI_EMR1 491
EXTI_EMR2 493
EXTI_EXTICR1 485
EXTI_EXTICR2 486
EXTI_EXTICR3 487
EXTI_EXTICR4 488
EXTI_FPR1 479
EXTI_FPR2 483
EXTI_FTSR1 477
EXTI_FTSR2 481
EXTI_IMR1 490
EXTI_IMR2 492
EXTI_PRIVCFGR1 479
EXTI_PRIVCFGR2 484
EXTI_RPR1 478
EXTI_RPR2 482
EXTI_RTSR1 476
EXTI_RTSR2 480
EXTI_SWIER1 478
EXTI_SWIER2 481

F

FDCAN_CCCR 1591
FDCAN_CKDIV 1616
FDCAN_CREL 1588
FDCAN_DBTP 1588
FDCAN_ECR 1596
FDCAN_ENDN 1588
FDCAN_HPMS 1607
FDCAN_IE 1602
FDCAN_ILE 1605
FDCAN_ILS 1604
FDCAN_IR 1599
FDCAN_NBTP 1592
FDCAN_PSR 1597
FDCAN_RWD 1590
FDCAN_RXF0A 1609
FDCAN_RXF0S 1608
FDCAN_RXF1A 1610
FDCAN_RXF1S 1609
FDCAN_RXGFC 1605
FDCAN_TDCR 1599
FDCAN_TEST 1589
FDCAN_TOCC 1595
FDCAN_TOCV 1596
FDCAN_TSCC 1594
FDCAN_TSCV 1594
FDCAN_TXBAR 1612
FDCAN_TXBC 1610
FDCAN_TXBCF 1614
FDCAN_TXBCIE 1615
FDCAN_TXBCR 1613
FDCAN_TXBRP 1611
FDCAN_TXBTIE 1614
FDCAN_TXBTO 1613
FDCAN_TXEFA 1616
FDCAN_TXEFS 1615
FDCAN_TXFQS 1611
FDCAN_XIDAM 1607
FLASH_ACR 162
FLASH_ECCCORR 183
FLASH_ECCDETR 184
FLASH_ECCDR 185
FLASH_HDP1R_CUR 182
FLASH_HDP1R_PRG 182
FLASH_HDP2R_CUR 187
FLASH_HDP2R_PRG 187
FLASH_HDPEXTR 172
FLASH_NSBOOTR_CUR 178

FLASH_NSBOOTR_PRG 179
FLASH_NSCCR 171
FLASH_NSCR 168
FLASH_NSKEYR 163
FLASH_NSSR 167
FLASH_OPSR 165
FLASH_OPTCR 165
FLASH_OPTKEYR 164
FLASH_OPTSR_CUR 173
FLASH_OPTSR_PRG 175
FLASH_OPTSR2_CUR 176
FLASH_OPTSR2_PRG 177
FLASH_OTPBLR_CUR 179
FLASH_OTPBLR_PRG 180
FLASH_PRIVBB1R1 180
FLASH_PRIVBB2R1 185
FLASH_PRIVCFGR 172
FLASH_WRP1R_CUR 181
FLASH_WRP1R_PRG 181
FLASH_WRP2R_CUR 185
FLASH_WRP2R_PRG 186

G

GPDMA_CxBR1 452-453
GPDMA_CxBR2 459
GPDMA_CxCR 442
GPDMA_CxDAR 457
GPDMA_CxFCR 440
GPDMA_CxLBAR 439
GPDMA_CxLLR 460, 462
GPDMA_CxSAR 456
GPDMA_CxSR 441
GPDMA_CxTR1 444
GPDMA_CxTR2 448
GPDMA_CxTR3 458
GPDMA_MISR 439
GPDMA_PRIVCFGR 438
GPIOx_AFRH 357
GPIOx_AFRL 357
GPIOx_BRR 358
GPIOx_BSRR 355
GPIOx_HSLVR 359
GPIOx_IDR 354
GPIOx_LCKR 356
GPIOx_MODER 352
GPIOx_ODR 355
GPIOx_OSPEEDR 353
GPIOx_OTYPER 352
GPIOx_PUPDR 354
GTZC1_MPCBBz_PRIVCFGRx 116
GTZC1_TZSC_MPCWM4ACFGR 113
GTZC1_TZSC_MPCWM4AR 114
GTZC1_TZSC_PRIVCFGR1 109
GTZC1_TZSC_PRIVCFGR2 111
GTZC1_TZSC_PRIVCFGR3 112

H

HASH_CR 717
HASH_CSRx 725
HASH_DIN 719
HASH_HRAx 722
HASH_HRx 722
HASH_IMR 723
HASH_SR 723
HASH_STR 720

I

I2C_CR1 1210
I2C_CR2 1213
I2C_ICR 1221
I2C_ISR 1218
I2C_OAR1 1215
I2C_OAR2 1215
I2C_PECR 1222
I2C_RXDR 1222
I2C_TIMEOUTR 1217
I2C_TIMINGR 1216
I2C_TXDR 1223
I3C_BCR 1324
I3C_CEVR 1312
I3C_CFGR 1291
I3C_CR 1287, 1289
I3C_CRCAPR 1327
I3C_DCR 1325
I3C_DEVR0 1314
I3C_DEVRx 1316
I3C_EPIDR 1330
I3C_EVR 1306
I3C_GETCAPR 1326
I3C_GETMXDSR 1328
I3C_IBIDR 1300
I3C_IER 1310
I3C_MAXRLR 1318
I3C_MAXWLR 1319
I3C_RDR 1296
I3C_RDWR 1296
I3C_RMR 1305
I3C_SER 1303
I3C_SR 1302
I3C_TDR 1297
I3C_TDWR 1298
I3C_TGTTDR 1301

I3C_TIMINGR0 1320
I3C_TIMINGR1 1321
I3C_TIMINGR2 1323
ICACHE_CR 200
ICACHE_FCR 202
ICACHE_HMONR 203
ICACHE_IER 202
ICACHE_MMONR 203
ICACHE_SR 201
ITM_CIDR0 1738
ITM_CIDR1 1738
ITM_CIDR2 1739
ITM_CIDR3 1739
ITM_DEVARCHR 1734
ITM_DEVTYPER 1735
ITM_PIDR0 1736
ITM_PIDR1 1736
ITM_PIDR2 1737
ITM_PIDR3 1737
ITM_PIDR4 1735
ITM_STIMRx 1731
ITM_TCR 1733
ITM_TER 1732
ITM_TPR 1732
IWDG_EWCR 1063
IWDG_KR 1060
IWDG_PR 1060
IWDG_RLR 1061
IWDG_SR 1061
IWDG_WINR 1063

L

LPTIM_ARR 1042
LPTIM_CCMR1 1045
LPTIM_CCR1 1042
LPTIM_CCR2 1047
LPTIM_CFGR 1038
LPTIM_CFGR2 1043
LPTIM_CNT 1043
LPTIM_CR 1041
LPTIM_RCR 1044
LPTIMx_DIER 1035-1036
LPTIMx_ICR 1032-1033
LPTIMx_ISR 1028, 1030
LPUART_BRR 1467
LPUART_CR1 . . . . . 1453, 1456
LPUART_CR2 1459
LPUART_CR3 . . . . . 1461, 1464
LPUART_ICR 1475
LPUART_ISR 1468, 1472
LPUART_PRESC 1477
LPUART_RDR 1476
LPUART_RQR 1467
LPUART_TDR 1477

M

MCUROM_CIDR0 1704
MCUROM_CIDR1 1704
MCUROM_CIDR2 1705
MCUROM_CIDR3 1705
MCUROM_MEMTYPER 1701
MCUROM_PIDR0 1702
MCUROM_PIDR1 1703
MCUROM_PIDR2 1703
MCUROM_PIDR3 1703
MCUROM_PIDR4 1702

O

OPAMP1_CSR 685
OPAMP1_HSOTR 688
OPAMP1_OTR 687

P

PWR_BDCR 236
PWR_BDSR 237
PWR_DBPCR 237
PWR_IORETR 243
PWR_PMCR 232
PWR_PMSR 234
PWR_PRIVCFGR 243
PWR_SCCR 238
PWR_VMCR 239
PWR_VMSR 240
PWR_VOSCR 234
PWR_VOSSR 235
PWR_WUCR 242
PWR_WUSCR 241
PWR_WUSR 241

R

RAMCFG_M2WPR1 100
RAMCFG_MxCR 96
RAMCFG_MxDEAR 99
RAMCFG_MxECCKEYR 100
RAMCFG_MxERKEYR 101
RAMCFG_MxICR 99
RAMCFG_MxIER 97
RAMCFG_MxISR 98
RAMCFG_MxSEAR 98
RCC_AHB1ENR 302

RCC_AHB1LPENR 309
RCC_AHB1RSTR 296
RCC_AHB2ENR 303
RCC_AHB2LPENR 311
RCC_AHB2RSTR 297
RCC_APB1HENR 307
RCC_APB1HLPENR 314
RCC_APB1HRSTR 300
RCC_APB1LENR 305
RCC_APB1LLPENR 312
RCC_APB1LRSTR 298
RCC_APB2ENR 307
RCC_APB2LPENR 315
RCC_APB2RSTR 300
RCC_APB3ENR 308
RCC_APB3LPENR 316
RCC_APB3RSTR 301
RCC_BDCR 322
RCC_CCIPR1 317
RCC_CCIPR2 318
RCC_CCIPR3 318
RCC_CCIPR4 319
RCC_CCIPR5 321
RCC_CFGR1 278
RCC_CFGR2 281
RCC_CICR 294
RCC_CIER 292
RCC_CIFR 293
RCC_CR 274
RCC_CRRCR 277
RCC_CSICFGR 278
RCC_HSICFGR 277
RCC_PLL1CFGR 284
RCC_PLL1DIVR 288
RCC_PLL1FRACR 289
RCC_PLL2CFGR 286
RCC_PLL2DIVR 290
RCC_PLL2FRACR 291
RCC_PRIVCFGR 325
RCC_RSR 324
RNG_CR 700
RNG_DR 703
RNG_HTCR 705
RNG_NSCR 704
RNG_SR 702
RTC_ALRABINR 1118
RTC_ALRBBINR 1118
RTC_ALRMAR 1110
RTC_ALRMASSR 1111
RTC_ALRMBR 1112
RTC_ALRMBSSR 1113
RTC_CALR 1105
RTC_CR 1099
RTC_DR 1095
RTC_ICSR 1096
RTC_MISR 1116
RTC_PRER 1098
RTC_PRIVCFGR 1103
RTC_SCR 1117
RTC_SHIFTR 1107
RTC_SR 1114
RTC_SSR 1096
RTC_TR 1094
RTC_TSDR 1109
RTC_TSSSR 1109
RTC_TSTR 1108
RTC_WPR 1105
RTC_WUTR 1099

S

SBS_CCCSR 373
SBS_CCSWCR 375
SBS_CCVALR 374
SBS_CFGR2 375
SBS_CNSLCKR 376
SBS_DBGCR 370
SBS_DBGLOCKR 371
SBS_ECCNMIR 377
SBS_FPUIMR 372
SBS_HDPLCR 369
SBS_HDPLSR 369
SBS_MESR 372
SBS_PMCR 371
SPI_CFG1 1538
SPI_CFG2 1541
SPI_CR1 1535
SPI_CR2 1537
SPI_CRCPOLY 1549
SPI_I2SCFGR 1551
SPI_IER 1543
SPI_IFCR 1547
SPI_RXCRC 1550
SPI_RXDR 1548
SPI_SR 1544
SPI_TXCRC 1549
SPI_TXDR 1548
SPI_UDRDR 1551
SYSROM_CIDR0 1699
SYSROM_CIDR1 1699
SYSROM_CIDR2 1700
SYSROM_CIDR3 1700
SYSROM_MEMTYPER 1696
SYSROM_PIDR0 1697

SYSROM_PIDR1 1697
SYSROM_PIDR2 1698
SYSROM_PIDR3 1698
SYSROM_PIDR4 1696

T

TAMP_ATCR1 1143
TAMP_ATCR2 1147
TAMP_ATOR 1146
TAMP_ATSEEDR 1145
TAMP_BKPxR 1158
TAMP_CFGR 1148
TAMP_COUNT1R 1157
TAMP_CR1 1137
TAMP_CR2 1139
TAMP_CR3 1140
TAMP_FLTCR 1142
TAMP_IER 1150
TAMP_MISR 1153
TAMP_PRIVCFGR 1149
TAMP_RPCFGR 1157
TAMP_SCR 1155
TAMP_SR 1151
TIM1_AF1 860
TIM1_AF2 863
TIM1_ARR 846
TIM1_BDTR 850
TIM1_CCER 841
TIM1_CCMR1 . . . . . 832, 834
TIM1_CCMR2 837-838
TIM1_CCMR3 856
TIM1_CCR1 847
TIM1_CCR2 847
TIM1_CCR3 848
TIM1_CCR4 849
TIM1_CCR5 854
TIM1_CCR6 855
TIM1_CNT 845
TIM1_CR1 818
TIM1_CR2 819
TIM1_DCR 865
TIM1_DIER 827
TIM1_DMAR 867
TIM1_DTR2 857
TIM1_ECR 858
TIM1_EGR 831
TIM1_PSC 845
TIM1_RCR 846
TIM1_SMCR 823
TIM1_SR 828
TIM1_TISEL 859
TIM2_ARR 969
TIM2_CCR1 971
TIM2_CCR2 972
TIM2_CCR3 974
TIM2_CCR4 976
TIM2_CNT 968
TIM3_ARR 969
TIM3_CCR1 970
TIM3_CCR2 971
TIM3_CCR3 973
TIM3_CCR4 975
TIM3_CNT 967
TIMx_AF1 979
TIMx_AF2 980
TIMx_ARR 1003
TIMx_CCER 966
TIMx_CCMR1 . . . . . 958, 960
TIMx_CCMR2 962-963
TIMx_CNT 1002
TIMx_CR1 . . . . . 947, 999
TIMx_CR2 . . . . . 948, 1001
TIMx_DCR 981
TIMx_DIER 954, 1001
TIMx_DMAR 982
TIMx_ECR 977
TIMx_EGR 957, 1002
TIMx_PSC 968, 1003
TIMx_SMCR 950
TIMx_SR 955, 1002
TIMx_TISEL 978
TPIU_ACPR 1780
TPIU_CIDR0 1787
TPIU_CIDR1 1788
TPIU_CIDR2 1788
TPIU_CIDR3 1789
TPIU_CLAIMCLR 1783
TPIU_CLAIMSETR 1783
TPIU_CSPSR 1779
TPIU_DEVIDR 1784
TPIU_DEVTYPER 1784
TPIU_FFCR 1781
TPIU_FFSR 1781
TPIU_PIDR0 1785
TPIU_PIDR1 1786
TPIU_PIDR2 1786
TPIU_PIDR3 1787
TPIU_PIDR4 1785
TPIU_PSCR 1782
TPIU_SPPR 1780
TPIU_SSPSR 1779

U

USART_BRR 1405
USART_CR1 . . . . . .1386, 1390
USART_CR2 1393
USART_CR3 . . . . . .1397, 1401
USART_GTPR 1405
USART_ICR 1419
USART_ISR 1408, 1414
USART_PRESC 1421
USART_RDR 1420
USART_RQR 1407
USART_RTOR 1406
USART_TDR 1421
USB_BCDR 1651
USB_CHEP_RXTXBD_n 1665-1666
USB_CHEP_TXRXBD_n 1663
USB_CHEPnR 1652
USB_CNTR 1642
USB_DADDR 1649
USB_FNR 1649
USB_ISTR 1645
USB_LPMCSR 1650

W

WWDG_CFR 1071
WWDG_CR 1070
WWDG_SR 1072

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. 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.

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 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.

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.

© 2024 STMicroelectronics – All rights reserved