YTM32B1LE0x Reference Manual

Support: YTM32B1LE05H0MLHT, YTM32B1LE05H0MLFT, YTM32B1LE05H0MLET, YTM32B1LE05H0MFMR, YTM32B1LE05H0MFMIR, YTM32B1LE04H0MLFT, YTM32B1LE04H0MLET, YTM32B1LE04H0MFMR

Yuntu Microelectronics

Document Number: YTM32B1LE0x RM

Rev.1.8, 2025/08/29

YUNTU reserves the right to change the production detail specifications as may be required to permit improvements in the design of its products.

Contents

I Introduction

1 Introduction

1.1 Overview . . . 2
1.2 Feature Summary . . . 2
1.3 Block Diagram . . . 4
1.4 Applications . . . 4

II Memory Map and Memories

2 Memory Map

2.1 Introduction . . . 7
2.2 System Memory Map . . . 7
2.3 APB Slot . . . 7

3 Embedded FLASH Module (EFM)

3.1 Introduction . . . 11
3.1.1 Features . . . 11
3.1.2 Memory Map . . . 11
3.1.3 Block Diagram . . . 11
3.2 Register Definition . . . 12
3.2.1 EFM Memory Map . . . 12
3.2.1.1 EFM CTRL Register . . . 12
3.2.1.2 EFM STS Register . . . 14
3.2.1.3 EFM CMD Register . . . 16
3.2.1.4 EFM AP0 Register . . . 16
3.2.1.5 EFM AP1 Register . . . 17
3.2.1.6 EFM CMD_UNLOCK Register . . . 18
3.2.1.7 EFM CUS_KEY Register . . . 19
3.2.1.8 EFM CTRL_OVRD Register . . . 20
3.3 Functional Description . . . 21
3.3.1 Program and Erase Sequences . . . 21
3.3.2 Disable JTAG and SWD . . . 22
3.3.3 Address Protection . . . 22
3.3.4 Flash Illegal Operation . . . 22
3.3.5 DFlash Operation . . . 22

Yuntu Microelectronics

III Signal Multiplexing and Pin Assignment

4 Pinmux
4.1 Pinmux Definition . . . 24

5 General Purpose Input/Output (GPIO)
5.1 Introduction . . . 26
5.1.1 Features . . . 26
5.1.2 Modes of Operation . . . 26
5.1.2.1 Active Mode . . . 26
5.1.2.2 Sleep Mode . . . 26
5.1.2.3 Deepsleep Mode . . . 26
5.1.2.4 Standby Mode . . . 26
5.2 Register Definition . . . 27
5.2.1 GPIO Memory Map . . . 27
5.2.1.1 GPIO PDOR Register . . . 28
5.2.1.2 GPIO PSOR Register . . . 28
5.2.1.3 GPIO PCOR Register . . . 29
5.2.1.4 GPIO PTOR Register . . . 30
5.2.1.5 GPIO PDIR Register . . . 31
5.2.1.6 GPIO POER Register . . . 32
5.2.1.7 GPIO PIER Register . . . 33
5.2.1.8 GPIO PIFR Register . . . 34
5.2.1.9 GPIO PCR Register . . . 35
5.3 Functional Description . . . 37
5.3.1 General-purpose Input . . . 37
5.3.2 General-purpose Output . . . 37
5.3.3 Port Interrupt and DMA Request . . . 37
5.3.4 IOPORT . . . 37

6 Port Control (PCTRL)
6.1 Introduction . . . 38
6.1.1 Overview . . . 38
6.1.2 Features . . . 38
6.1.3 Modes of Operation . . . 38
6.1.3.1 Active Mode . . . 38
6.1.3.2 Sleep Mode . . . 38
6.1.3.3 Deepsleep Mode . . . 38
6.1.3.4 Standby Mode . . . 38
6.2 Register Definition . . . 39
6.2.1 PCTRL Memory Map . . . 39
6.2.1.1 PCTRL PCR Register . . . 40
6.3 Functional Description . . . 41
6.3.1 Port Control . . . 41

IV Core and Systems

7 CM0+ Overview
7.1 ARM Cortex-M0+ Core Configuration . . . 43
7.2 Nested Vectored Interrupt Controller (NVIC) . . . 43

Yuntu Microelectronics

7.3 Debugger Configuration . . . 44

8 Direct Memory Access (DMA)
8.1 Chip Specific Information . . . 45
8.1.1 DMA Channel MUX table . . . 45
8.1.2 Power Mode Description . . . 47
8.2 Introduction . . . 47
8.2.1 Features . . . 47
8.2.2 Block Diagram . . . 48
8.2.3 DMA Channel Transfer Structure (CTS) . . . 48
8.2.4 DMA Trigger MUX . . . 49
8.3 Register Definition . . . 49
8.3.1 DMA Memory Map . . . 49
8.3.1.1 DMA CTRL Register . . . 51
8.3.1.2 DMA CHTLDIF Register . . . 53
8.3.1.3 DMA CHTLHDIF Register . . . 53
8.3.1.4 DMA CHEIF Register . . . 54
8.3.1.5 DMA CHEIE Register . . . 55
8.3.1.6 DMA ERS Register . . . 55
8.3.1.7 DMA REQEN Register . . . 57
8.3.1.8 DMA REQS Register . . . 58
8.3.1.9 DMA START Register . . . 58
8.3.1.10 DMA DONE Register . . . 58
8.3.1.11 DMA CHMUX Register . . . 59
8.3.1.12 DMA CTS_SADDR Register . . . 59
8.3.1.13 DMA CTS_SOFF Register . . . 60
8.3.1.14 DMA CTS_TCR Register . . . 60
8.3.1.15 DMA CTS_BCNT_BCNT Register . . . 61
8.3.1.16 DMA CTS_BCNT_LODIS Register . . . 62
8.3.1.17 DMA CTS_BCNT_LOEN Register . . . 62
8.3.1.18 DMA CTS_STO Register . . . 63
8.3.1.19 DMA CTS_DADDR Register . . . 64
8.3.1.20 DMA CTS_DOFF Register . . . 64
8.3.1.21 DMA CTS_TCNT_LKDIS Register . . . 65
8.3.1.22 DMA CTS_TCNT_LKEN Register . . . 65
8.3.1.23 DMA CTS_DTO_RLD_DTO Register . . . 66
8.3.1.24 DMA CTS_DTO_RLD_RLD Register . . . 67
8.3.1.25 DMA CTS_CSR Register . . . 67
8.3.1.26 DMA CTS_TCNTRV Register . . . 69
8.4 Functional Description . . . 69
8.4.1 Transfer Loop . . . 69
8.4.2 Trigger Loop . . . 69
8.4.3 Channel Arbitration . . . 69
8.4.4 DMA Channel Initialize . . . 70

9 Divide and Square Root (DIVSQRT)
9.1 Introduction . . . 71
9.1.1 Feature . . . 71

Yuntu Microelectronics

9.2 Register Definition . . . 71
9.2.1 DIVSQRT Memory Map . . . 71
9.2.1.1 DIVSQRT DEND Register . . . 72
9.2.1.2 DIVSQRT DSOR Register . . . 72
9.2.1.3 DIVSQRT CSR Register . . . 73
9.2.1.4 DIVSQRT RES Register . . . 76
9.2.1.5 DIVSQRT RCND Register . . . 76
9.3 Functional Description . . . 77
9.3.1 Modes of Operation . . . 77
9.3.2 Algorithms . . . 77
9.3.2.1 Integer Divide . . . 77
9.3.2.2 Integer Square Root . . . 78

10 Trigger Multiplexer Module (TMU)
10.1 Introduction . . . 79
10.2 Trigger Interconnectivity Table . . . 79
10.3 Register Definition . . . 82
10.3.1 TMU Memory Map . . . 82
10.3.1.1 TMU MUX Register . . . 83

11 Chip Integration Module (CIM)
11.1 Chip-specific CIM Information . . . 85
11.2 Introduction . . . 85
11.2.1 Features . . . 85
11.3 Register Definition . . . 85
11.3.1 CIM Memory Map . . . 85
11.3.1.1 CIM CTRL Register . . . 86
11.3.1.2 CIM SWTRIG Register . . . 87
11.3.1.3 CIM ETMROPT0 Register . . . 87
11.3.1.4 CIM ETMROPT1 Register . . . 89
11.3.1.5 CIM DIEINFO Register . . . 90
11.3.1.6 CIM CHIPINFO Register . . . 92

V Clock, Reset and Power Management

12 System Clock Unit (SCU)
12.1 Introduction . . . 96
12.1.1 Features . . . 96
12.1.2 Block Diagram . . . 97
12.2 Register Definition . . . 97
12.2.1 SCU Memory Map . . . 97
12.2.1.1 SCU CLKS Register . . . 98
12.2.1.2 SCU DIV Register . . . 99
12.2.1.3 SCU STS Register . . . 101
12.2.1.4 SCU DIVSTS Register . . . 102
12.2.1.5 SCU FIRC_CTRL Register . . . 104
12.2.1.6 SCU SIRC_CTRL Register . . . 105
12.2.1.7 SCU FXOSC_CTRL Register . . . 106
12.2.1.8 SCU SXOSC_CTRL Register . . . 107
12.2.1.9 SCU CLKO Register . . . 108

Yuntu Microelectronics

12.2.1.10 SCU FIRCTRIM Register . . . 109
12.2.1.11 SCU SIRCTRIM Register . . . 110
12.2.1.12 SCU LPOTRIM Register . . . 111
12.2.1.13 SCU CMU_CTRL Register . . . 112
12.2.1.14 SCU CMUSTS Register . . . 113
12.2.1.15 SCU CMUCMP_HIGH Register . . . 114
12.2.1.16 SCU CMUCMP_LOW Register . . . 115
12.3 Functional Description . . . 116
12.3.1 Clock Switcher . . . 116
12.3.2 System Clock Divider . . . 116
12.3.3 Clock Monitor Unit (CMU) . . . 117
12.3.4 Clocks after Reset . . . 117
12.3.5 Operation Modes . . . 117

13 Power Control Unit (PCU)
13.1 Introduction . . . 118
13.1.1 Block Diagram . . . 118
13.1.2 Features . . . 118
13.2 Register Definition . . . 119
13.2.1 PCU Memory Map . . . 119
13.2.1.1 PCU BORCR Register . . . 119
13.2.1.2 PCU LVDCR Register . . . 120
13.2.1.3 PCU CSR Register . . . 122
13.3 Functional Description . . . 123
13.3.1 Brown-out . . . 124
13.3.2 Low Voltage Detect . . . 124
13.3.3 Active Mode . . . 124
13.3.4 Sleep Mode . . . 124
13.3.5 Deepsleep Mode . . . 124
13.3.6 Standby Mode . . . 124

14 IP Clock Control (IPC)
14.1 Introduction . . . 125
14.1.1 Features . . . 125
14.1.2 Block Diagram . . . 125
14.2 Peripheral Clock Table . . . 126
14.3 Register Definition . . . 127
14.3.1 IPC Memory Map . . . 127
14.3.1.1 IPC CTRL Register . . . 130
14.4 Functional Description . . . 132

15 Reset Control Unit (RCU)
15.1 Introduction . . . 133
15.1.1 Features . . . 133
15.2 Register Definition . . . 133
15.2.1 RCU Memory Map . . . 133
15.2.1.1 RCU RSSR Register . . . 133
15.2.1.2 RCU RPFR Register . . . 135

Yuntu Microelectronics

16 Power Mode
16.1 Introduction . . . 137
16.1.1 Active Mode . . . 137
16.1.2 Sleep Mode . . . 137
16.1.3 Deepsleep Mode . . . 137
16.1.4 Standby Mode . . . 137
16.2 Power Modes Table . . . 137
16.3 Wake-up Sources . . . 139

VI Communication Interfaces

17 Flexible Controller Area Network (FlexCAN)
17.1 Chip-specific FlexCAN Information . . . 141
17.1.1 Instantiation Information . . . 141
17.2 Introduction . . . 141
17.2.1 Block Diagram . . . 141
17.2.2 Overview . . . 142
17.2.3 Features . . . 143
17.2.4 Mode of Operation . . . 144
17.3 Register Definition . . . 146
17.3.1 Register Access and Reset Information . . . 146
17.3.2 Memory Mapping . . . 148
17.3.3 CAN Memory Map . . . 149
17.3.3.1 CAN MCR Register . . . 175
17.3.3.2 CAN CTRL1 Register . . . 181
17.3.3.3 CAN TIMER Register . . . 185
17.3.3.4 CAN RXMGMASK Register . . . 186
17.3.3.5 CAN RX14MASK Register . . . 188
17.3.3.6 CAN RX15MASK Register . . . 189
17.3.3.7 CAN ECR Register . . . 190
17.3.3.8 CAN ESR1 Register . . . 192
17.3.3.9 CAN IMASK2 Register . . . 199
17.3.3.10 CAN IMASK1 Register . . . 200
17.3.3.11 CAN IFLAG2 Register . . . 201
17.3.3.12 CAN IFLAG1 Register . . . 201
17.3.3.13 CAN CTRL2 Register . . . 205
17.3.3.14 CAN ESR2 Register . . . 210
17.3.3.15 CAN CRCR Register . . . 211
17.3.3.16 CAN RXFGMASK Register . . . 212
17.3.3.17 CAN RXFIR Register . . . 214
17.3.3.18 CAN CBT Register . . . 215
17.3.3.19 CAN IMASK4 Register . . . 217
17.3.3.20 CAN IMASK3 Register . . . 218
17.3.3.21 CAN IFLAG4 Register . . . 219
17.3.3.22 CAN IFLAG3 Register . . . 220
17.3.3.23 CAN RAM Register . . . 221
17.3.3.24 CAN RXIMR Register . . . 221
17.3.3.25 CAN MECR Register . . . 223
17.3.3.26 CAN ERRIAR Register . . . 225
17.3.3.27 CAN ERRIDPR Register . . . 227

Yuntu Microelectronics

17.3.3.28 CAN ERRIPPR Register . . . 227
17.3.3.29 CAN RERRAR Register . . . 228
17.3.3.30 CAN RERRDR Register . . . 229
17.3.3.31 CAN RERRSYNR Register . . . 230
17.3.3.32 CAN ERRSR Register . . . 232
17.3.3.33 CAN CTRL1_PN Register . . . 233
17.3.3.34 CAN CTRL2_PN Register . . . 235
17.3.3.35 CAN WU_MTC Register . . . 236
17.3.3.36 CAN FLT_ID1 Register . . . 238
17.3.3.37 CAN FLT_DLC Register . . . 239
17.3.3.38 CAN PL1_LO Register . . . 239
17.3.3.39 CAN PL1_HI Register . . . 241
17.3.3.40 CAN FLT_ID2_IDMASK Register . . . 242
17.3.3.41 CAN PL2_PLMASK_LO Register . . . 243
17.3.3.42 CAN PL2_PLMASK_HI Register . . . 244
17.3.3.43 CAN WMB_WMBn_CS Register . . . 245
17.3.3.44 CAN WMB_WMBn_ID Register . . . 247
17.3.3.45 CAN WMB_WMBn_D03 Register . . . 247
17.3.3.46 CAN WMB_WMBn_D47 Register . . . 248
17.3.3.47 CAN EPRS Register . . . 250
17.3.3.48 CAN ENCBT Register . . . 251
17.3.3.49 CAN EDCBT Register . . . 252
17.3.3.50 CAN ETDC Register . . . 254
17.3.3.51 CAN FDCTRL Register . . . 255
17.3.3.52 CAN FDCBT Register . . . 259
17.3.3.53 CAN FDCRC Register . . . 261
17.3.3.54 CAN ERFCR Register . . . 262
17.3.3.55 CAN ERFIER Register . . . 267
17.3.3.56 CAN ERFSR Register . . . 268
17.3.3.57 CAN HR_TIME_STAMPn Register . . . 270
17.3.3.58 CAN ERFFELn Register . . . 271
17.3.4 Message Buffer Structure . . . 272
17.3.5 FlexCAN Memory Partition for CAN FD . . . 278
17.3.6 FlexCAN Message Buffer Memory Map . . . 280
17.3.7 Legacy Rx FIFO Structure . . . 286
17.3.8 Enhanced Rx FIFO Structure . . . 289
17.4 Functional Description . . . 291
17.4.1 Transmit Process . . . 292
17.4.2 Arbitration Process . . . 293
17.4.2.1 Lowest-Number Mailbox First . . . 293
17.4.2.2 Highest-Priority Mailbox First . . . 294
17.4.2.2.1 Local Priority Disabled . . . 294
17.4.2.2.2 Local Priority Enabled . . . 294
17.4.2.3 Arbitration Completion . . . 295
17.4.2.4 Arbitration Start and Stop Conditions . . . 295
17.4.3 Receive Process . . . 296
17.4.4 Matching Process . . . 297
17.4.5 Receive Process under Pretended Networking Mode . . . 301
17.4.6 Move Process . . . 304
17.4.6.1 Move-in . . . 304

Yuntu Microelectronics

17.4.6.2 Move-out . . . 305
17.4.7 Data Coherence . . . 306
17.4.7.1 Transmission Abort Mechanism . . . 306
17.4.7.2 Mailbox Inactivation . . . 307
17.4.7.3 Mailbox Lock Mechanism . . . 307
17.4.8 Enhanced Rx FIFO . . . 308
17.4.8.1 Enhanced Rx FIFO Filter Elements . . . 309
17.4.8.2 Enhanced Rx FIFO Matching Process . . . 310
17.4.8.3 Enhanced Rx FIFO under DMA Operation . . . 312
17.4.8.4 Enhanced Rx FIFO Clear Operation . . . 313
17.4.9 Legacy Rx FIFO . . . 313
17.4.9.1 Rx FIFO under DMA Operation . . . 314
17.4.9.2 Clear Legacy Rx FIFO Operation . . . 315
17.4.10 CAN Protocol Related Features . . . 315
17.4.10.1 CAN FD Frames . . . 315
17.4.10.2 Transceiver Delay Compensation . . . 320
17.4.10.3 Remote Frame Transmission . . . 323
17.4.10.4 Remote Frame Reception . . . 323
17.4.10.5 Overload Frames . . . 324
17.4.10.6 Message Buffer Time Stamp . . . 324
17.4.10.7 High-Resolution Time Stamp . . . 324
17.4.10.8 Protocol Timing . . . 325
17.4.10.9 Arbitration and Matching Timing . . . 330
17.4.10.10 Tx Arbitration Start Delay . . . 331
17.4.11 Clock Domains and Restrictions . . . 334
17.4.12 Modes of Operation Details . . . 338
17.4.12.1 Freeze Mode . . . 338
17.4.12.2 Module Disable Mode . . . 339
17.4.12.3 Doze Mode . . . 339
17.4.12.4 Deepsleep Mode . . . 340
17.4.12.5 Pretended Networking Mode . . . 342
17.4.13 Interrupts . . . 342
17.4.14 Bus Interface . . . 343
17.4.15 Detection and Correction of Memory Errors . . . 344
17.4.15.1 Sources of Memory Access . . . 344
17.4.15.2 Error Indication . . . 345
17.4.15.3 Error Reporting . . . 345
17.4.15.4 Response to Errors . . . 346
17.4.15.5 Error Injection . . . 347
17.5 Initialization / Application Information . . . 347
17.5.1 RAM Initialization . . . 347
17.5.2 FlexCAN Initialization Sequence . . . 348
17.6 FlexCAN Ram Usage And Initialization . . . 348
17.6.1 RAM Size . . . 349
17.6.2 RAM Usage . . . 349
17.6.3 RAM Initialization Addresses . . . 356
17.6.4 RAM Addresses for Error Injection and Reporting . . . 358
17.6.4.1 RAM Addresses for FlexCAN Standard Features . . . 358
17.6.4.2 RAM Addresses for FlexCAN Advanced Features . . . 359
17.6.4.2.1 Enhanced MB Memory Region . . . 360

Yuntu Microelectronics

17.6.4.2.2 High-Resolution Time Stamp Region . . . 360
17.6.4.2.3 Enhanced Rx FIFO Region . . . 361
17.6.4.2.4 Enhanced Rx FIFO Filter Elements Region . . . 365

18 Universal Asynchronous Receiver Transmitter (UART)
18.1 Chip-specific UART Information . . . 367
18.1.1 Instantiation Information . . . 367
18.1.2 Signal Description . . . 367
18.1.3 Clock Description . . . 367
18.1.4 Power Mode Description . . . 367
18.2 Introduction . . . 368
18.2.1 Features . . . 368
18.2.2 Block Diagram . . . 369
18.3 Register Definition . . . 369
18.3.1 UART Memory Map . . . 369
18.3.1.1 UART BAUD Register . . . 370
18.3.1.2 UART CTRL0 Register . . . 371
18.3.1.3 UART CTRL1 Register . . . 373
18.3.1.4 UART INTF Register . . . 375
18.3.1.5 UART INTE Register . . . 377
18.3.1.6 UART MATCH Register . . . 378
18.3.1.7 UART DATA Register . . . 379
18.3.1.8 UART TXFIFO Register . . . 380
18.3.1.9 UART RXFIFO Register . . . 381
18.4 Functional Description . . . 382
18.4.1 Baud Rate Generator . . . 382
18.4.1.1 Baud Rate Generator Diagram . . . 383
18.4.2 Transmitter Functional Description . . . 383
18.4.3 Receiver Functional Description . . . 383

19 Serial Port Interface (SPI)
19.1 Chip-specific SPI Information . . . 384
19.1.1 Instantiation Information . . . 384
19.2 Introduction . . . 384
19.2.1 Features . . . 384
19.2.2 Signal Description . . . 384
19.3 Register Definition . . . 385
19.3.1 SPI Memory Map . . . 385
19.3.1.1 SPI CTRL Register . . . 385
19.3.1.2 SPI TXCFG Register . . . 387
19.3.1.3 SPI CLK Register . . . 390
19.3.1.4 SPI STS Register . . . 390
19.3.1.5 SPI INTE Register . . . 392
19.3.1.6 SPI MATCH0 Register . . . 393
19.3.1.7 SPI MATCH1 Register . . . 394
19.3.1.8 SPI TXFIFO Register . . . 394
19.3.1.9 SPI RXFIFO Register . . . 395
19.3.1.10 SPI DATA Register . . . 396
19.4 Functional Description . . . 396
19.4.1 SPI Functional Clock . . . 396

Yuntu Microelectronics

19.4.2 Master Mode . . . 396
19.4.2.1 Tx and CMD FIFO . . . 396
19.4.2.2 Rx FIFO . . . 397
19.4.2.3 Data Match . . . 397
19.4.2.4 Pin Configuration . . . 397
19.4.2.5 Clock LoopBack . . . 397
19.4.2.6 Continuous Transfer . . . 397
19.4.2.6.1 Continuous Command . . . 397
19.4.3 Slave Mode . . . 397
19.4.3.1 Tx and CMD FIFO . . . 398
19.4.3.2 Receive FIFO . . . 398
19.4.3.3 Data Match . . . 398
19.4.3.4 Internal PCS . . . 398
19.4.3.5 Continuous Transfer . . . 398
19.4.3.5.1 Continuous Mode . . . 398

20 Inter-Integrated Circuit (I2C)
20.1 Chip-specific I2C Information . . . 399
20.1.1 Instantiation Information . . . 399
20.1.2 Signal Description . . . 399
20.1.3 Clock Description . . . 399
20.1.4 Reset Description . . . 399
20.1.5 Power Mode Description . . . 400
20.2 Introduction . . . 400
20.2.1 Features . . . 400
20.3 Register Definition . . . 400
20.3.1 I2C Memory Map . . . 400
20.3.1.1 I2C MCTRL Register . . . 401
20.3.1.2 I2C MCLKCFG Register . . . 403
20.3.1.3 I2C MFLTCFG Register . . . 403
20.3.1.4 I2C MSTS Register . . . 405
20.3.1.5 I2C MIE Register . . . 406
20.3.1.6 I2C TXFIFO Register . . . 408
20.3.1.7 I2C RXFIFO Register . . . 408
20.3.1.8 I2C MDATA Register . . . 409
20.3.1.9 I2C MATCH Register . . . 410
20.3.1.10 I2C TOCFG Register . . . 411
20.3.1.11 I2C HSCLK Register . . . 412
20.3.1.12 I2C SCTRL Register . . . 413
20.3.1.13 I2C SCLKCFG Register . . . 416
20.3.1.14 I2C SFLTCFG Register . . . 417
20.3.1.15 I2C SSTS Register . . . 418
20.3.1.16 I2C SIE Register . . . 421
20.3.1.17 I2C SDATA Register . . . 422
20.3.1.18 I2C SACK Register . . . 423
20.3.1.19 I2C SADDR Register . . . 423
20.3.1.20 I2C SRADDR Register . . . 424
20.4 Functional Description . . . 425
20.4.1 I2C Functional Clock . . . 425

Yuntu Microelectronics

20.4.2 Master Mode . . . 425
20.4.2.1 Tx and CMD FIFO . . . 425
20.4.2.2 Master Operations . . . 426
20.4.2.3 Rx FIFO and Data Matching . . . 426
20.4.3 Slave Mode . . . 426
20.4.3.1 Address Matching . . . 426
20.4.3.2 Data Processing . . . 427
20.4.3.3 Clock Stretching . . . 427

VII Timer

21 Periodic Timer (pTMR)
21.1 Chip-specific pTMR Information . . . 429
21.1.1 Instantiation Information . . . 429
21.2 Introduction . . . 429
21.2.1 Block Diagram . . . 429
21.2.2 Features . . . 429
21.3 Register Definition . . . 430
21.3.1 pTMR Memory Map . . . 430
21.3.1.1 pTMR MCR Register . . . 430
21.3.1.2 pTMR CH_TSV Register . . . 431
21.3.1.3 pTMR CH_TCV Register . . . 432
21.3.1.4 pTMR CH_TCR Register . . . 432
21.3.1.5 pTMR CH_TFR Register . . . 433
21.4 Functional Description . . . 434
21.4.1 Timers . . . 434
21.4.2 Chain Mode . . . 434
21.4.3 Debug Mode . . . 434
21.4.4 Interrupt . . . 434

22 Low Power Timer (lpTMR)
22.1 Introduction . . . 435
22.1.1 Features . . . 435
22.2 Register Definition . . . 435
22.2.1 lpTMR Memory Map . . . 435
22.2.1.1 lpTMR CTRL Register . . . 436
22.2.1.2 lpTMR PRS Register . . . 437
22.2.1.3 lpTMR DIE Register . . . 438
22.2.1.4 lpTMR STS Register . . . 439
22.2.1.5 lpTMR CMP Register . . . 440
22.2.1.6 lpTMR LCNT Register . . . 441
22.2.1.7 lpTMR CNT Register . . . 442
22.3 Functional Description . . . 443
22.3.1 lpTMR Clocking . . . 443
22.3.2 lpTMR Prescaler/Glitch Filter . . . 443
22.3.2.1 Time Counter . . . 444
22.3.2.2 Pulse Counter . . . 444
22.3.3 lpTMR Compare . . . 444
22.3.4 lpTMR Counter . . . 444
22.3.5 lpTMR Hardware Trigger . . . 444

Yuntu Microelectronics

22.3.6 lpTMR Interrupt . . . 444
22.3.7 Operation Modes . . . 445

23 Enhanced Timer (eTMR)
23.1 Chip-specific eTMR Information . . . 446
23.1.1 Instantiation Information . . . 446
23.1.2 Signal Description . . . 446
23.1.3 Clock Description . . . 447
23.1.4 DMA Description . . . 447
23.1.5 Interrupt Description . . . 447
23.1.6 Power Mode Description . . . 448
23.1.7 Channel Pair Description . . . 448
23.2 Introduction . . . 448
23.2.1 Overview . . . 448
23.2.2 Features . . . 449
23.2.3 Block Diagram . . . 450
23.3 Register Definition . . . 451
23.3.1 eTMR Memory Map . . . 451
23.3.1.1 eTMR STS Register . . . 452
23.3.1.2 eTMR IOSTS Register . . . 454
23.3.1.3 eTMR CNT Register . . . 456
23.3.1.4 eTMR QDCNT Register . . . 456
23.3.1.5 eTMR CTRL Register . . . 457
23.3.1.6 eTMR SYNC Register . . . 459
23.3.1.7 eTMR OTRIG Register . . . 461
23.3.1.8 eTMR INTE Register . . . 462
23.3.1.9 eTMR CHINIT Register . . . 464
23.3.1.10 eTMR CHSWV Register . . . 465
23.3.1.11 eTMR CHMASK Register . . . 466
23.3.1.12 eTMR CHFV Register . . . 469
23.3.1.13 eTMR MOD Register . . . 471
23.3.1.14 eTMR DT Register . . . 471
23.3.1.15 eTMR QDCTRL Register . . . 472
23.3.1.16 eTMR FAULT Register . . . 474
23.3.1.17 eTMR CH_CTRL Register . . . 476
23.3.1.18 eTMR CH_VAL0 Register . . . 478
23.3.1.19 eTMR CH_VAL1 Register . . . 479
23.3.1.20 eTMR CH_CVAL Register . . . 479
23.4 Functional Description . . . 480
23.4.1 Clock . . . 480
23.4.2 Counter . . . 480
23.4.2.1 Global Time Base . . . 481
23.4.3 PWM Mode . . . 481
23.4.3.1 Center Aligned PWM . . . 482
23.4.3.2 Edge Aligned PWM . . . 482
23.4.3.3 Asymmetrical PWM . . . 483
23.4.3.4 Software-generated PWM . . . 484
23.4.3.5 PWM Source . . . 484
23.4.3.6 Independent Mode . . . 484
23.4.3.7 Complementary Mode . . . 485

Yuntu Microelectronics

23.4.3.8 Complementary Mode with Dead-time Insertion . . . 486
23.4.3.9 Fault Detection and Protection . . . 487
23.4.3.9.1 Safe Value . . . 487
23.4.3.9.2 Fault Source Selection . . . 487
23.4.3.9.3 Fault Input Polarity . . . 487
23.4.3.9.4 Fault Filter . . . 487
23.4.3.9.5 Input Stretch . . . 487
23.4.3.9.6 Combinational Logic Fault . . . 488
23.4.3.9.7 Fault Recovery . . . 488
23.4.3.9.8 Fault Flag Clear . . . 488
23.4.3.10 Mask Control . . . 488
23.4.4 Output Compare Mode . . . 489
23.4.5 Input Capture Mode . . . 489
23.4.5.1 Channel Filter . . . 489
23.4.5.2 Capture Test . . . 490
23.4.6 Output Polarity . . . 490
23.4.7 Channels Initialization . . . 490
23.4.8 Loading Mechanism . . . 490
23.4.8.1 General Registers Loading . . . 491
23.4.8.2 Loading Frequency . . . 492
23.4.8.3 Special Registers loading . . . 492
23.4.8.4 Loading Trigger . . . 492
23.4.9 Quadrature Decoder . . . 492
23.4.9.1 Decoder Prescaler . . . 492
23.4.9.2 Count and Direction Decoder Mode . . . 493
23.4.9.3 Phase A and Phase B Decoder Mode . . . 493
23.4.9.4 Decoder Input Polarity . . . 493
23.4.9.5 Decoder Filter . . . 494
23.4.9.6 Decoder Counter Overflow . . . 494
23.4.10 Filter . . . 494
23.4.11 Output Trigger . . . 494
23.4.11.1 Width Fixed Pulse . . . 495
23.4.11.2 Width Adjustable Pulse . . . 495
23.4.11.3 Output Frequency . . . 495
23.4.12 Input Capture Source . . . 495
23.4.13 Hall Sensor Support . . . 495
23.4.14 Modulation Implementation . . . 496
23.4.15 Power Mode . . . 496
23.4.16 Interrupts . . . 496
23.4.17 DMA . . . 496
23.4.18 Reset . . . 496

24 Real Time Clock (RTC)

24.1 Introduction . . . 497
24.1.1 Features . . . 497
24.2 Register Definition . . . 497
24.2.1 RTC Memory Map . . . 497
24.2.1.1 RTC SEC Register . . . 497
24.2.1.2 RTC CMP Register . . . 498
24.2.1.3 RTC ALM Register . . . 500
24.2.1.4 RTC EN Register . . . 501

Yuntu Microelectronics

24.2.1.5 RTC CTRL Register . . . 501
24.2.1.6 RTC INTF Register . . . 503
24.2.1.7 RTC INTE Register . . . 504
24.3 Functional Description . . . 505
24.3.1 General . . . 505
24.3.1.1 Software Reset . . . 505
24.3.1.2 Time Counter . . . 505
24.3.1.3 Compensation . . . 505
24.3.1.4 Register Lock . . . 505
24.3.2 Interrupts . . . 506
24.3.3 Operation Modes . . . 506

VIII Analog

25 Analog-to-Digital Converter (ADC)
25.1 Chip-specific ADC Information . . . 508
25.1.1 Instantiation Information . . . 508
25.1.2 Signal Description . . . 508
25.1.3 Clock Description . . . 509
25.1.4 DMA Description . . . 509
25.1.5 Interrupt Description . . . 509
25.1.6 Power Mode Description . . . 509
25.2 Introduction . . . 509
25.2.1 Overview . . . 509
25.2.2 Features . . . 510
25.2.3 Block Diagram . . . 510
25.3 Register Definition . . . 511
25.3.1 ADC Memory Map . . . 511
25.3.1.1 ADC STS Register . . . 512
25.3.1.2 ADC INTE Register . . . 513
25.3.1.3 ADC CTRL Register . . . 515
25.3.1.4 ADC CFG0 Register . . . 516
25.3.1.5 ADC CFG1 Register . . . 518
25.3.1.6 ADC SMP Register . . . 519
25.3.1.7 ADC WDCTRL Register . . . 519
25.3.1.8 ADC WDTH Register . . . 520
25.3.1.9 ADC CHSEL Register . . . 521
25.3.1.10 ADC FIFO Register . . . 522
25.4 Functional Description . . . 523
25.4.1 Basic Configuration . . . 523
25.4.1.1 ADC On-off Control . . . 523
25.4.1.2 ADC Clock . . . 524
25.4.1.3 Channel Selection . . . 524
25.4.1.4 Configuring the ADC . . . 525
25.4.1.5 Starting Conversions . . . 525
25.4.1.6 Stopping an Ongoing Conversion . . . 526
25.4.2 Timings . . . 526
25.4.3 Functional Mode . . . 527
25.4.3.1 Single Conversion Mode . . . 527
25.4.3.2 Continuous Conversion Mode . . . 528

Yuntu Microelectronics

25.4.3.3 Discontinuous Mode . . . 528
25.4.3.4 Conversion on External Trigger . . . 528
25.4.3.5 Programmable Resolution . . . 529
25.4.3.6 End of Sampling/Conversion . . . 529
25.4.3.7 End of Conversion Sequence . . . 529
25.4.4 Data Management . . . 529
25.4.4.1 Data FIFO . . . 529
25.4.4.2 ADC Overrun . . . 530
25.4.4.3 Managing a Sequence of Data Converted Without Using the DMA . . . 530
25.4.4.4 Managing Converted Data Without Using the DMA and Overrun . . . 530
25.4.4.5 Managing Converted Data Using the DMA . . . 531
25.4.5 Low Power Mode . . . 531
25.4.5.1 Wait Mode . . . 531
25.4.5.2 Auto-off Mode . . . 531
25.4.6 Analog Window Watchdog . . . 531
25.4.7 Temperature Sensor . . . 532

26 Analog Comparator (ACMP)
26.1 Chip-specific ACMP Information . . . 533
26.1.1 Instantiation Information . . . 533
26.1.2 Signal Description . . . 533
26.1.3 Clock Description . . . 534
26.1.4 DMA Description . . . 534
26.1.5 Interrupt Description . . . 534
26.1.6 Power Mode Description . . . 534
26.2 Introduction . . . 534
26.2.1 Overview . . . 534
26.2.2 Features . . . 535
26.2.3 Block Diagram . . . 535
26.3 Register Definition . . . 536
26.3.1 ACMP Memory Map . . . 536
26.3.1.1 ACMP STS Register . . . 536
26.3.1.2 ACMP INTE Register . . . 539
26.3.1.3 ACMP DAC Register . . . 540
26.3.1.4 ACMP MUX Register . . . 541
26.3.1.5 ACMP FILT Register . . . 542
26.3.1.6 ACMP CONT Register . . . 543
26.3.1.7 ACMP CTRL Register . . . 544
26.3.1.8 ACMP EN Register . . . 546
26.3.1.9 ACMP DMACR Register . . . 547
26.4 Functional Description . . . 547
26.4.1 Input Control . . . 547
26.4.1.1 Input Selection . . . 547
26.4.1.2 DAC (Digital to Analog Converter) . . . 548
26.4.2 Output Control . . . 548
26.4.2.1 Polarity . . . 548
26.4.2.2 Filter . . . 548
26.4.3 Function Mode . . . 548
26.4.3.1 Common Mode . . . 548
26.4.3.2 Sample Mode . . . 548

Yuntu Microelectronics

26.4.3.3 Window Mode . . . 548
26.4.3.4 Continuous Mode . . . 549
26.4.3.4.1 Channel Fix . . . 549
26.4.3.4.2 Channel Select . . . 549
26.4.3.4.3 Timing . . . 549
26.4.3.4.4 Scan Result . . . 549
26.4.3.4.5 Loop Mode . . . 550
26.4.3.4.6 One-shot Mode . . . 550
26.4.4 Modes of Operation . . . 550
26.4.5 Interrupts . . . 550
26.4.6 DMA Usage . . . 550
26.4.7 Reset . . . 550

27 Temperature Sensor (TEMPSENSOR)
27.1 Introduction . . . 551

IX Security, Integrity and Safety

28 Cyclic Redundancy Check (CRC)
28.1 Introduction . . . 553
28.1.1 Features . . . 553
28.1.2 Block Diagram . . . 553
28.2 Register Definition . . . 554
28.2.1 CRC Memory Map . . . 554
28.2.1.1 CRC CTRL Register . . . 554
28.2.1.2 CRC INIT Register . . . 555
28.2.1.3 CRC DATA_IN32 Register . . . 556
28.2.1.4 CRC DATA_IN16 Register . . . 557
28.2.1.5 CRC DATA_IN8 Register . . . 557
28.2.1.6 CRC RESULT Register . . . 557
28.3 Functional Description . . . 558
28.3.1 CRC Initialization . . . 558
28.3.2 CRC Calculation . . . 558
28.3.3 Bit Swap and Inversion . . . 559

29 Watch Dog (WDG)
29.1 Introduction . . . 560
29.1.1 Features . . . 560
29.2 Register Definition . . . 560
29.2.1 WDG Memory Map . . . 560
29.2.1.1 WDG SVCR Register . . . 561
29.2.1.2 WDG CR Register . . . 561
29.2.1.3 WDG LR Register . . . 563
29.2.1.4 WDG TOVR Register . . . 564
29.2.1.5 WDG WVR Register . . . 564
29.2.1.6 WDG CNTCVR Register . . . 565
29.2.1.7 WDG INTF Register . . . 566
29.3 Functional Description . . . 567
29.3.1 General . . . 567
29.3.1.1 Operation Modes . . . 567

Yuntu Microelectronics

29.3.2 Interrupts . . . 568

30 ECC Management Unit (EMU)
30.1 Chip-specific EMU Information . . . 569
30.2 Introduction . . . 569
30.2.1 Error Injection . . . 569
30.2.2 Error Report . . . 569
30.3 Register Definition . . . 569
30.3.1 Programming Model Access . . . 569
30.3.2 Error Injection Registers . . . 570
30.3.3 Error Report Registers . . . 570
30.3.4 EMU Memory Map . . . 570
30.3.4.1 EMU EIGCR Register . . . 571
30.3.4.2 EMU EICHCR Register . . . 571
30.3.4.3 EMU EICHD_ADDR Register . . . 572
30.3.4.4 EMU EICHD_CHKBIT Register . . . 572
30.3.4.5 EMU EICHD_DATA Register . . . 573
30.3.4.6 EMU ERIE Register . . . 574
30.3.4.7 EMU ERIF Register . . . 574
30.3.4.8 EMU ERINFO_ADDR Register . . . 575
30.3.4.9 EMU ERINFO_SYND Register . . . 575
30.3.4.10 EMU ERINFO_CNT Register . . . 576
30.4 Functional Description . . . 577
30.4.1 General . . . 577
30.4.2 Error Injection . . . 577
30.4.3 Error Report . . . 577

X Appendix

A
A.1 General Changes . . . 579

B
B.1 Revision History . . . 580
B.2 Release Notes . . . 581
B.2.1 Introduction . . . 581
B.2.2 Memory Map . . . 581
B.2.3 Embedded Flash Module (EFM) . . . 581
B.2.4 Pinmux . . . 581
B.2.5 General Purpose Input/Output (GPIO) . . . 581
B.2.6 Port Control (PCTRL) . . . 581
B.2.7 CM0+ Overview . . . 582
B.2.8 Direct Memory Access (DMA) . . . 582
B.2.9 Divide and Square Root (DIVSQRT) . . . 582
B.2.10 Trigger Multiplexer Module (TMU) . . . 582
B.2.11 Chip Integration Module (CIM) . . . 582
B.2.12 System Clock Unit (SCU) . . . 582
B.2.13 Power Control Unit (PCU) . . . 583
B.2.14 IP Clock Control (IPC) . . . 583
B.2.15 Reset Control Unit (RCU) . . . 583

Yuntu Microelectronics

B.2.16 Power Mode . . . 583
B.2.17 Flexible Controller Area Network (FlexCAN) . . . 583
B.2.18 Universal Asynchronous Receiver Transmitter (UART) . . . 583
B.2.19 Serial Port Interface (SPI) . . . 584
B.2.20 Inter-Integrated Circuit (I2C) . . . 584
B.2.21 Periodic Timer (pTMR) . . . 584
B.2.22 Low Power Timer (lpTMR) . . . 584
B.2.23 Enhanced Timer (eTMR) . . . 584
B.2.24 Real Time Clock Module (RTC) . . . 584
B.2.25 Analog-to- Digital Converter (ADC) . . . 584
B.2.26 Analog Comparator (ACMP) . . . 585
B.2.27 Temperature Sensor (TEMPSENSOR) . . . 585
B.2.28 Cyclic Redundancy Check (CRC) . . . 585
B.2.29 Watch Dog (WDG) . . . 585
B.2.30 ECC Management Unit (EMU) . . . 585

Yuntu Microelectronics

Part I

Introduction

Yuntu Microelectronics

Chapter 1

Introduction

1.1 Overview

The YTM32B1LE0x product series further extends the highly scalable portfolio of ARM® Cortex®-M0+ MCUs in the automotive industry. It introduces higher memory options alongside a richer peripheral set extending capability into a variety of automotive applications. With a 2.97~5.5 V supply and focus on automotive environment robustness, the YTM32B1LE0x product series devices are well suited to a wide range of applications in electrically harsh environments, and are optimized for cost-sensitive applications offering low pin-count options.

The YTM32B1LE0x product series offers a broad range of memory, peripherals, and package options. It shares common peripherals and pin counts, allowing developers to migrate easily within an MCU family or among the MCU families to take advantage of more memory or feature integration. This scalability allows developers to use the YTMicro product series as the standard for their end product platforms, maximizing hardware and software reuse and reducing time to market.

1.2 Feature Summary

- AEC-Q100 qualified
- ASIL-B compliant
- RoHS compliant
- Core and Architecture
  - ARM Cortex-M0+ running up to 48MHz
    - Single-cycle access to I/O: Up to 50 percent faster than standard I/O, improves reaction time to external events allowing bit manipulation and software protocol emulation
    - Two-stage pipeline: Reduced number of cycles per instruction (CPI), enabling faster branch instruction and ISR entry, and reducing power consumption
    - Excellent code density in comparison to 8-bit and 16-bit MCUs: Reduced flash size, system cost and power consumption
    - Optimized access to program memory: Accesses on alternate cycles reduces power consumption
    - 100 percent compatible with ARM Cortex-M0 and a subset ARM Cortex-M3/M4: Reuse existing compilers and debug tools
    - Simplified architecture: 56 instructions and 17 registers enable easy programming and efficient packaging of 8/16/32-bit data in memory
    - Linear 4 GB address space removes the need for paging/banking, reducing software

Yuntu Microelectronics

    complexity
    * ARM third-party ecosystem support: Software and tools to help minimize development time/cost
  - System clock running up to 48MHz
  - Support systick with a 8-bit divider and enablement
- Memory
  - Up to 128 KB PFlash with ECC which supports single error correction (SEC) and double error detection (DED)
  - Up to 16 KB SRAM with ECC which supports single error correction (SEC) and double error detection (DED)
- Clocks
  - Fast IRC (FIRC), up to 48 MHz
  - Fast crystal oscillator (FXOSC), range 4~40 MHz
  - Slow IRC (SIRC), up to 2 MHz
  - Slow crystal oscillator (SXOSC), run at 32.768 KHz
  - Low power oscillator (LPO), run at 750 Hz
- 4 DMA channels with up to 41 peripheral hardware trigger sources
- DIVSQRT module with 32-bit integer divide and square root operations
- Mixed-signal analog
  - One Analog-to-Digital Converter (ADC) with 12-bit 1Msps sample rate, it has 16 channels and supports temperature sensor
  - One Analog Comparator (ACMP) with 8-bit DAC
- Enhanced Timer (eTMR), Periodic Timer (pTMR) and Low Power Timer (lpTMR) for a broad range of applications including motor control
- Serial communication interfaces such as UART, SPI, I2C and FlexCAN with FD
- High security and safety with internal Watchdog (WDG), programmable CRC module, ECC Management Unit (EMU) and Clock Monitor Unit (CMU)
- Single power supply (2.97~5.5 V) with full functional flash program/erase/read operations
- Power management
  - Low-power ARM Cortex-M0+ core with excellent energy efficiency
  - Support four power modes: Active, Sleep, Deepsleep and Standby
  - Support clock gating for unused modules, and specific peripherals remain working in Sleep, Deepsleep and Standby modes
- Up to 58 GPIO pins with interrupt functionality
- Support 64/48/32 LQFP and 32 QFN packages
- Temperature parameters:
  - Ambient operation temperature range: -40°C ~ 125°C

Yuntu Microelectronics

1.3 Block Diagram

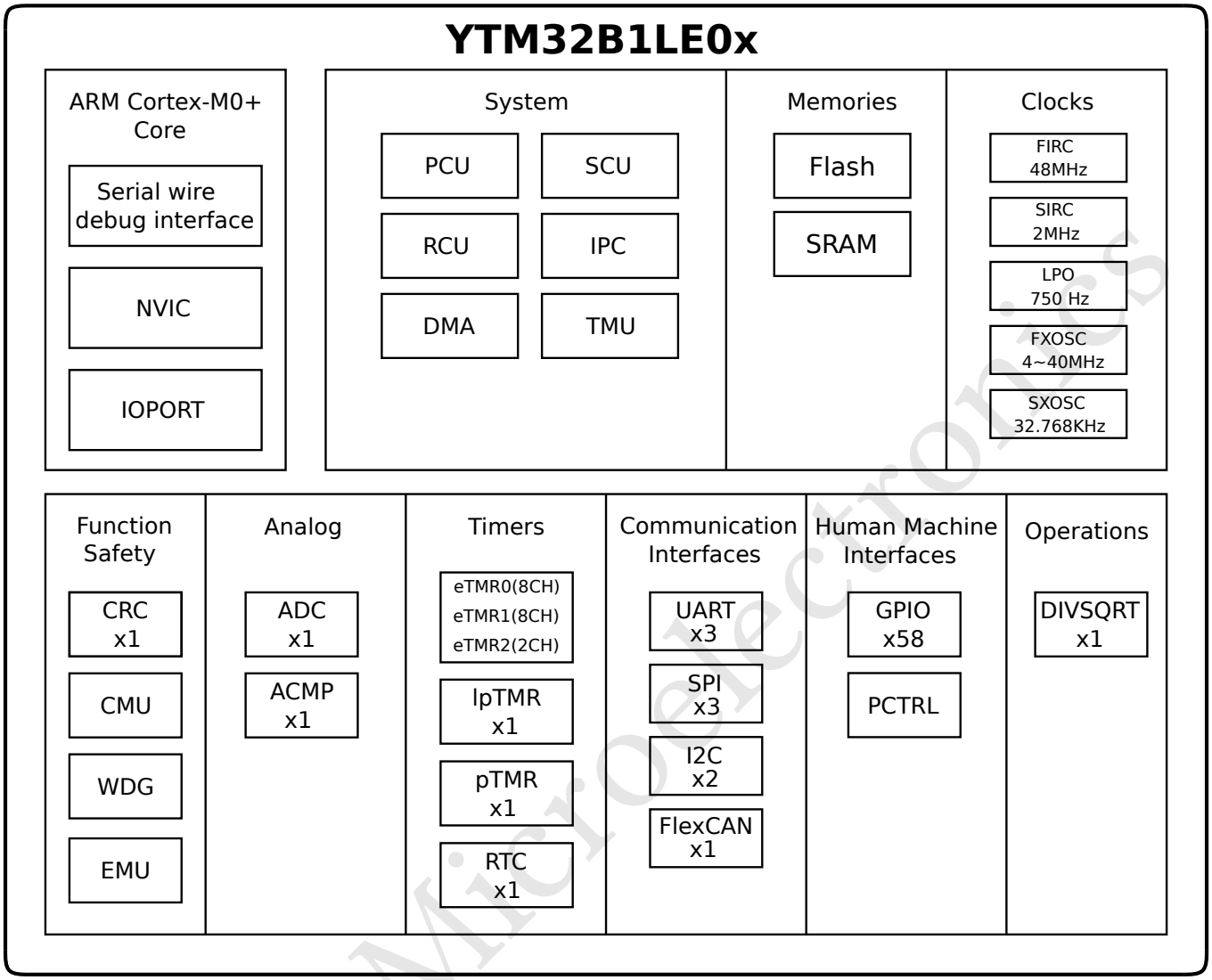


Figure 1.1: YTM32B1LE0 Block Diagram

1.4 Applications

The YTM32B1LE0x product series are the ideal choices for general purpose automotive applications, which include but not limited to:

- Exterior and interior lighting
- HVAC
- Door/Window/Wiper/Seat controller
- BLDC/PMSM motor control
- E-shifter
- TPMS
- Real time control in infotainment system
- Human machine interface such as touch sense control
- Secured vehicle data transfer
- Safety controller

Yuntu Microelectronics

- Over the air update

The YTM32B1LE0x offer small packages: 10*10 mm 64 LQFP, 7*7 mm 48 LQFP, 7*7 mm 32 LQFP and 5*5 mm 32 QFN, which are suitable for broad range of size sensitive applications in vehicle, such as various sensor controllers.

In addition to automotive applications, the YTM32B1LE0x product series can also be used in challenging environments found in industrial, automation, communications, transportation, medical and A & D applications which need high level quality, reliability and safety.

Yuntu Microelectronics

Part II

Memory Map and Memories

Yuntu Microelectronics

Chapter 2

Memory Map

2.1 Introduction

This chip contains various memories and memory-mapped peripherals that are located in one 32-bit contiguous memory space. This chapter describes the memory and peripheral locations within memory space.

2.2 System Memory Map

Table 2.1: YTM32B1LE05x System Memory Map

Module	Start Address	End Address	Size
PFlash	0x0000_0000	0x0001_FFFF	128 KB
DFlash	0x1000_0000	0x1000_07FF	2 KB
OTP NVR	0x1000_0800	0x1000_09FF	512 B
CUS NVR	0x1000_0A00	0x1000_0BFF	512 B
SRAM	0x2000_0000	0x2000_3FFF	16 KB
APB	0x4000_0000	0x4007_FFFF	512 KB

Table 2.2: YTM32B1LE04x System Memory Map

Module	Start Address	End Address	Size
PFlash	0x0000_0000	0x0000_FFFF	64 KB
DFlash	0x1000_0000	0x1000_07FF	2 KB
OTP NVR	0x1000_0800	0x1000_09FF	512 B
CUS NVR	0x1000_0A00	0x1000_0BFF	512 B
SRAM	0x2000_0000	0x2000_1FFF	8 KB
APB	0x4000_0000	0x4007_FFFF	512 KB

2.3 APB Slot

Yuntu Microelectronics

Table 2.3: APB Module Table

Slot	Start Address	End Address	Module
0	0x4000_0000	0x4000_0FFF	-
1	0x4000_1000	0x4000_1FFF	-
2	0x4000_2000	0x4000_2FFF	-
3	0x4000_3000	0x4000_3FFF	-
4	0x4000_4000	0x4000_4FFF	-
5	0x4000_5000	0x4000_5FFF	-
6	0x4000_6000	0x4000_6FFF	-
7	0x4000_7000	0x4000_7FFF	-
8	0x4000_8000	0x4000_8FFF	DMA
9	0x4000_9000	0x4000_9FFF	-
10	0x4000_A000	0x4000_AFFF	-
11	0x4000_B000	0x4000_BFFF	-
12	0x4000_C000	0x4000_CFFF	-
13	0x4000_D000	0x4000_DFFF	-
14	0x4000_E000	0x4000_EFFF	DIVSQRT
15	0x4000_F000	0x4000_FFFF	GPIO
16	0x4001_0000	0x4001_0FFF	-
17	0x4001_1000	0x4001_1FFF	-
18	0x4001_2000	0x4001_2FFF	-
19	0x4001_3000	0x4001_3FFF	-
20	0x4001_4000	0x4001_4FFF	-
21	0x4001_5000	0x4001_5FFF	-
22	0x4001_6000	0x4001_6FFF	-
23	0x4001_7000	0x4001_7FFF	-
24	0x4001_8000	0x4001_8FFF	EMU
25	0x4001_9000	0x4001_9FFF	-
26	0x4001_A000	0x4001_AFFF	-
27	0x4001_B000	0x4001_BFFF	-
28	0x4001_C000	0x4001_CFFF	-
29	0x4001_D000	0x4001_DFFF	-
30	0x4001_E000	0x4001_EFFF	-
31	0x4001_F000	0x4001_FFFF	-
32	0x4002_0000	0x4002_0FFF	EFM
33	0x4002_1000	0x4002_1FFF	-
34	0x4002_2000	0x4002_2FFF	-
35	0x4002_3000	0x4002_3FFF	-
36	0x4002_4000	0x4002_4FFF	FlexCAN-FD
37	0x4002_5000	0x4002_5FFF	-
38	0x4002_6000	0x4002_6FFF	-
39	0x4002_7000	0x4002_7FFF	-
40	0x4002_8000	0x4002_8FFF	-
41	0x4002_9000	0x4002_9FFF	-
42	0x4002_A000	0x4002_AFFF	-
43	0x4002_B000	0x4002_BFFF	-

Yuntu Microelectronics

Table 2.3 continued from previous page

Slot	Start Address	End Address	Module
44	0x4002_C000	0x4002_CFFF	SPI0
45	0x4002_D000	0x4002_DFFF	SPI1
46	0x4002_E000	0x4002_EFFF	SPI2
47	0x4002_F000	0x4002_FFFF	-
48	0x4003_0000	0x4003_0FFF	-
49	0x4003_1000	0x4003_1FFF	-
50	0x4003_2000	0x4003_2FFF	CRC
51	0x4003_3000	0x4003_3FFF	-
52	0x4003_4000	0x4003_4FFF	-
53	0x4003_5000	0x4003_5FFF	-
54	0x4003_6000	0x4003_6FFF	-
55	0x4003_7000	0x4003_7FFF	pTMR0
56	0x4003_8000	0x4003_8FFF	eTMR0
57	0x4003_9000	0x4003_9FFF	eTMR1
58	0x4003_A000	0x4003_AFFF	eTMR2
59	0x4003_B000	0x4003_BFFF	ADC0
60	0x4003_C000	0x4003_CFFF	-
61	0x4003_D000	0x4003_DFFF	RTC
62	0x4003_E000	0x4003_EFFF	-
63	0x4003_F000	0x4003_FFFF	-
64	0x4004_0000	0x4004_0FFF	lpTMR0
65	0x4004_1000	0x4004_1FFF	-
66	0x4004_2000	0x4004_2FFF	-
67	0x4004_3000	0x4004_3FFF	-
68	0x4004_4000	0x4004_4FFF	-
69	0x4004_5000	0x4004_5FFF	-
70	0x4004_6000	0x4004_6FFF	-
71	0x4004_7000	0x4004_7FFF	-
72	0x4004_8000	0x4004_8FFF	CIM
73	0x4004_9000	0x4004_9FFF	PCTRL A
74	0x4004_A000	0x4004_AFFF	PCTRL B
75	0x4004_B000	0x4004_BFFF	PCTRL C
76	0x4004_C000	0x4004_CFFF	PCTRL D
77	0x4004_D000	0x4004_DFFF	PCTRL E
78	0x4004_E000	0x4004_EFFF	-
79	0x4004_F000	0x4004_FFFF	-
80	0x4005_0000	0x4005_0FFF	-
81	0x4005_1000	0x4005_1FFF	-
82	0x4005_2000	0x4005_2FFF	WDG
83	0x4005_3000	0x4005_3FFF	-
84	0x4005_4000	0x4005_4FFF	-
85	0x4005_5000	0x4005_5FFF	-
86	0x4005_6000	0x4005_6FFF	-
87	0x4005_7000	0x4005_7FFF	-
88	0x4005_8000	0x4005_8FFF	-

Yuntu Microelectronics

Table 2.3 continued from previous page

Slot	Start Address	End Address	Module
89	0x4005_9000	0x4005_9FFF	-
90	0x4005_A000	0x4005_AFFF	-
91	0x4005_B000	0x4005_BFFF	-
92	0x4005_C000	0x4005_CFFF	-
93	0x4005_D000	0x4005_DFFF	-
94	0x4005_E000	0x4005_EFFF	-
95	0x4005_F000	0x4005_FFFF	-
96	0x4006_0000	0x4006_0FFF	-
97	0x4006_1000	0x4006_1FFF	-
98	0x4006_2000	0x4006_2FFF	-
99	0x4006_3000	0x4006_3FFF	TMU
100	0x4006_4000	0x4006_4FFF	SCU
101	0x4006_5000	0x4006_5FFF	IPC
102	0x4006_6000	0x4006_6FFF	I2C0
103	0x4006_7000	0x4006_7FFF	I2C1
104	0x4006_8000	0x4006_8FFF	-
105	0x4006_9000	0x4006_9FFF	-
106	0x4006_A000	0x4006_AFFF	UART0
107	0x4006_B000	0x4006_BFFF	UART1
108	0x4006_C000	0x4006_CFFF	UART2
109	0x4006_D000	0x4006_DFFF	-
110	0x4006_E000	0x4006_EFFF	-
111	0x4006_F000	0x4006_FFFF	-
112	0x4007_0000	0x4007_0FFF	-
113	0x4007_1000	0x4007_1FFF	-
114	0x4007_2000	0x4007_2FFF	-
115	0x4007_3000	0x4007_3FFF	ACMP0
116	0x4007_4000	0x4007_4FFF	-
117	0x4007_5000	0x4007_5FFF	-
118	0x4007_6000	0x4007_6FFF	-
119	0x4007_7000	0x4007_7FFF	-
120	0x4007_8000	0x4007_8FFF	-
121	0x4007_9000	0x4007_9FFF	-
122	0x4007_A000	0x4007_AFFF	-
123	0x4007_B000	0x4007_BFFF	-
124	0x4007_C000	0x4007_CFFF	-
125	0x4007_D000	0x4007_DFFF	PCU
126	0x4007_E000	0x4007_EFFF	-
127	0x4007_F000	0x4007_FFFF	RCU

Yuntu Microelectronics

Chapter 3

Embedded FLASH Module (EFM)

3.1 Introduction

The Embedded FLASH Module (EFM) is organized as a 128K Bytes main array that supports sector erase and word program. The sector erase operation erases all bytes within a sector(512bytes).

The Flash IP program and erase operations are performed under CPU control through a command-driven interface to an internal state machine. It is not permitted to read from a FLASH physical block while the block is under programming or erasing.

3.1.1 Features

The EFM includes the following features:

- 128K bytes PFlash + 2K bytes DFlash of FLASH memory
- Automated program and erase operation
- Optional interrupt on command completion
- Security for single-chip applications
- Single power supply (system VDD) used for all module operations

3.1.2 Memory Map

Table 3.1: Memory Map

Module	Start Address	End Address	Size
PFlash	0x0000_0000	0x0001_FFFF	128 KB
DFlash	0x1000_0000	0x1000_07FF	2 KB
OTP NVR	0x1000_0800	0x1000_09FF	512 B
CUS NVR	0x1000_0A00	0x1000_0BFF	512 B
UID[0]	0x1000_0FE0	0x1000_0FE3	4 B
UID[1]	0x1000_0FE4	0x1000_0FE5	2 B
UID[2]	0x1000_0FE8	0x1000_0FE8	1 B
UID[3]	0x1000_0FEC	0x1000_0FED	2 B

3.1.3 Block Diagram

The following figure shows the EFM block diagram.

Yuntu Microelectronics

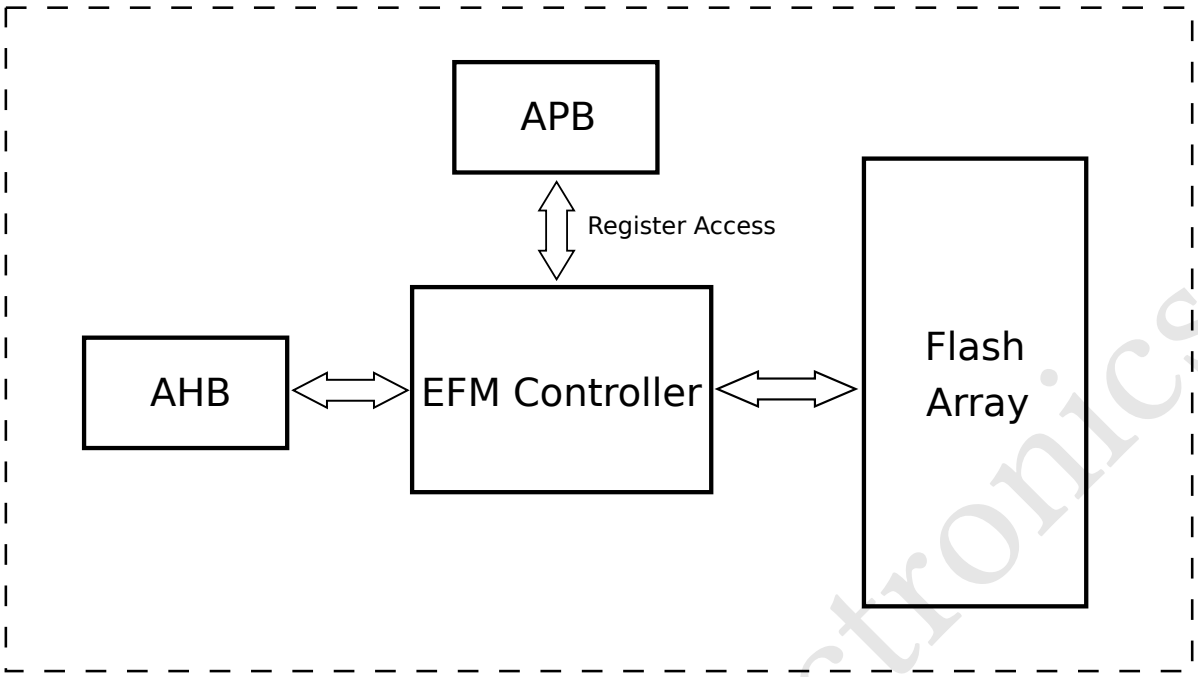


Figure 3.1: EFM Block Diagram

3.2 Register Definition

3.2.1 EFM Memory Map

EFM base address 40020000h

Table 3.2: EFM Memory Map

Offset	Register	Width	Access	Reset Value
0h	Control Register (CTRL)	32	RW	00180100h
4h	Status Register (STS)	32	RW	00000080h
8h	Command Register (CMD)	32	RW	00000000h
24h	Address Protect 0 (AP0)	32	RW	FFFFFFFFh
28h	Address Protect 1 (AP1)	32	RW	FFFFFFFFh
200h	EFM Command Unlock (CMD_UNLOCK)	32	WO	00000000h
204h	EFM Customer Key (CUS_KEY)	32	WO	00000000h
300h	Override Control Register (CTRL_OVRD)	32	RW	00000000h

3.2.1.1 EFM CTRL Register

Offset: 0h

Yuntu Microelectronics

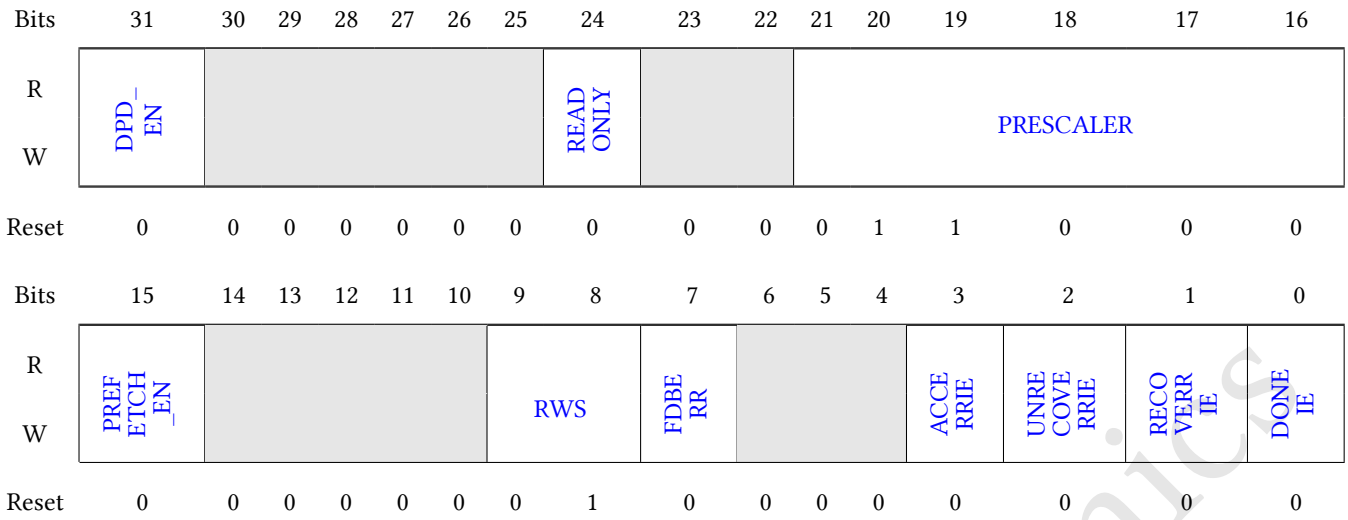


Table 3.3: EFM CTRL Register Description

Field	Function
31 DPD_EN	**Deep Power Down Enable** This DPD_EN bit determines whether the flash will enter deep power-down mode when MCU is entering Deepsleep mode. 0b - Flash will stay active when MCU enters Deepsleep mode 1b - Flash will enter deep power-down mode when MCU enters Deepsleep mode.
24 READONLY	**Flash Readonly** Program and erase operation will be aborted immediately, and only read access can be executed. Note that this bit is write-only, always get back 0 when read this bit. 0b - Read, program and erase function are enabled. 1b - Only read operation to Flash is enabled.
21 - 16 PRESCALER	**Prescaler** These PRESCALER bits are used for flash operation timing control. Please make sure PRESCALER to be the value of slow bus frequency.
15 PREFETCH_EN	**Prefetch Enable** This PREFETCH_EN bit is used for enabling prefetch read functionality.
9 - 8 RWS	**Read Wait State** These RWS bits determine clock cycles number that needs to be waited to get the flash data out after flash received the read access signal. Please use the following equation to calculate the RWS value: (core frequency/(RWS+1)) <= 24MHz If core frequency is less than 24MHz, the RWS should be set to 0. Please note that once the core frequency is changed, this field needs to be set to the appropriate value.

Yuntu Microelectronics

Field	Function
7 FDBERR	**Force Double Bit Ecc Error** This FDBERR bit will generate unrecovery ecc error for each read access. 0b - Disable generating unrecovery ecc error 1b - Enable generating unrecovery ecc error
3 ACCERRIE	**Access Error Interrupt Enable** This ACCERRIE bit determines whether the interrupt will be generated if ACCERR flag is asserted. 0b - Disable generating interrupt if ACCERR flag is asserted 1b - Enable generating interrupt if ACCERR flag is asserted
2 UNRECOVERRIE	**Unrecovery Error Interrupt Enable** This UNRECOVERRIE bit determines whether the interrupt will be generated if UNRECOVERR flag is asserted. 0b - Disable generating interrupt if UNRECOVERR flag is asserted 1b - Enable generating interrupt if UNRECOVERR flag is asserted
1 RECOVERRIE	**Recovery Error Interrupt Enable** This RECOVERRIE bit determines whether the interrupt will be generated if RECOVERR flag is asserted. 0b - Disable generating interrupt if RECOVERR flag is asserted 1b - Enable generating interrupt if RECOVERR flag is asserted
0 DONEIE	**Command Done Interrupt Enable** This DONEIE bit determines whether the interrupt will be generated if DONE flag is asserted. 0b - Disable generating interrupt if DONE flag is asserted 1b - Enable generating interrupt if DONE flag is asserted

Yuntu Microelectronics

3.2.1.2 EFM STS Register

Offset: 4h

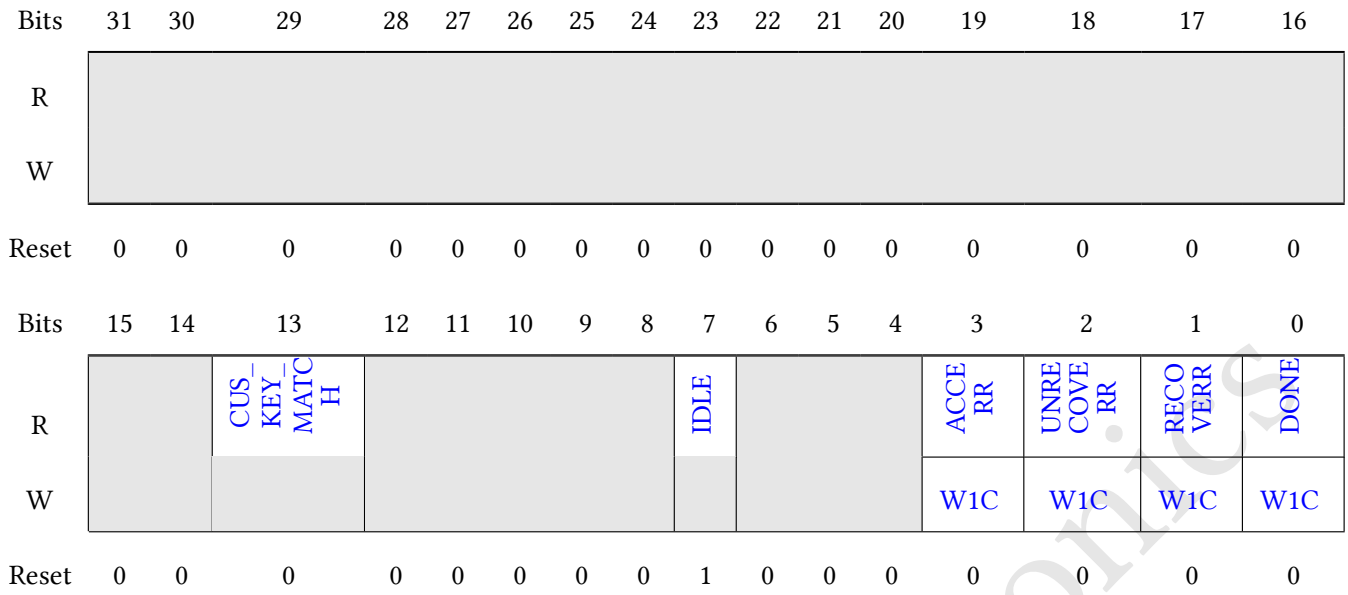


Table 3.4: EFM STS Register Description

Field	Function
13 CUS_KEY_ MATCH	**Customer Key Match Status** This bit indicates whether the customer key is matched. 0b - Customer key is not matched 1b - Customer key is matched
7 IDLE	**Command IDLE Status** This IDLE bit indicates whether flash is executing a command. 0b - Flash is currently executing a command 1b - No command is currently being executed
3 ACCERR	**Access Error Status** This ACCERR bit indicates whether an access error been detected. Accerr error includes below scenarios: 1.Access flash when IDLE is de-asserted. 2.Launch a new command when IDLE is de-asserted. 3.Lack of address or data information before launching the program or erase command. 0b - Access error is not detected 1b - Access error is detected
2 UNRECOVE RR	**Unrecovery Error Status** This UNRECOVERR bit indicates whether an unrecoverable ecc error been detected. 0b - Unrecoverry ecc error is not detected 1b - Unrecoverry ecc error is detected
1	**Recovery Error Status**

Yuntu Microelectronics

Field	Function
RECOVERR	This RECOVERR bit indicates whether a recoverable ecc error been detected. 0b - Recovery ecc error is not detected 1b - Recovery ecc error is detected
0 DONE	**Command Done Status** This DONE bit indicates whether the current command is completed. 0b - Current command is in progress 1b - Current command is completed

3.2.1.3 EFM CMD Register

Offset: 8h

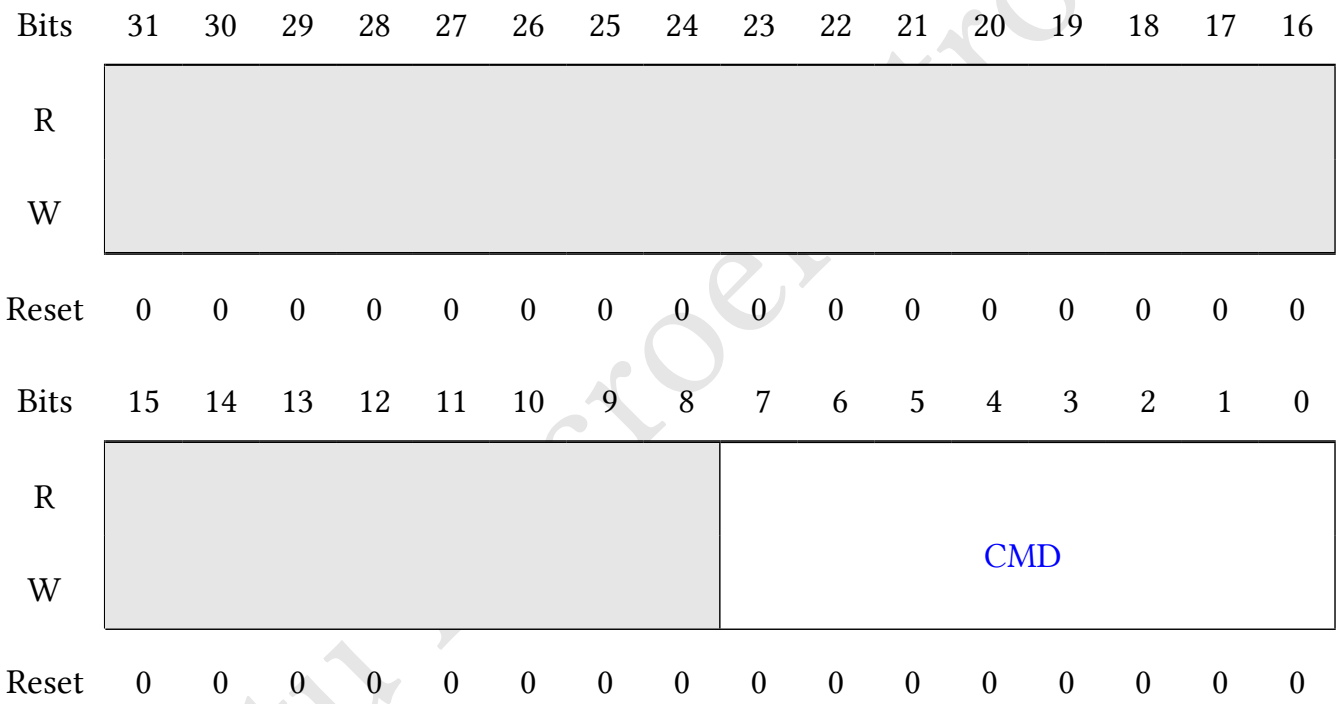


Table 3.5: EFM CMD Register Description

Field	Function
7 - 0 CMD	**Command Control** These CMD bits determine which commands are available. 00000010b (0x02) - Program 32 bits 00010000b (0x10) - Erase sector 00010010b (0x12) - Erase Block (only main array)

3.2.1.4 EFM AP0 Register

Offset: 24h

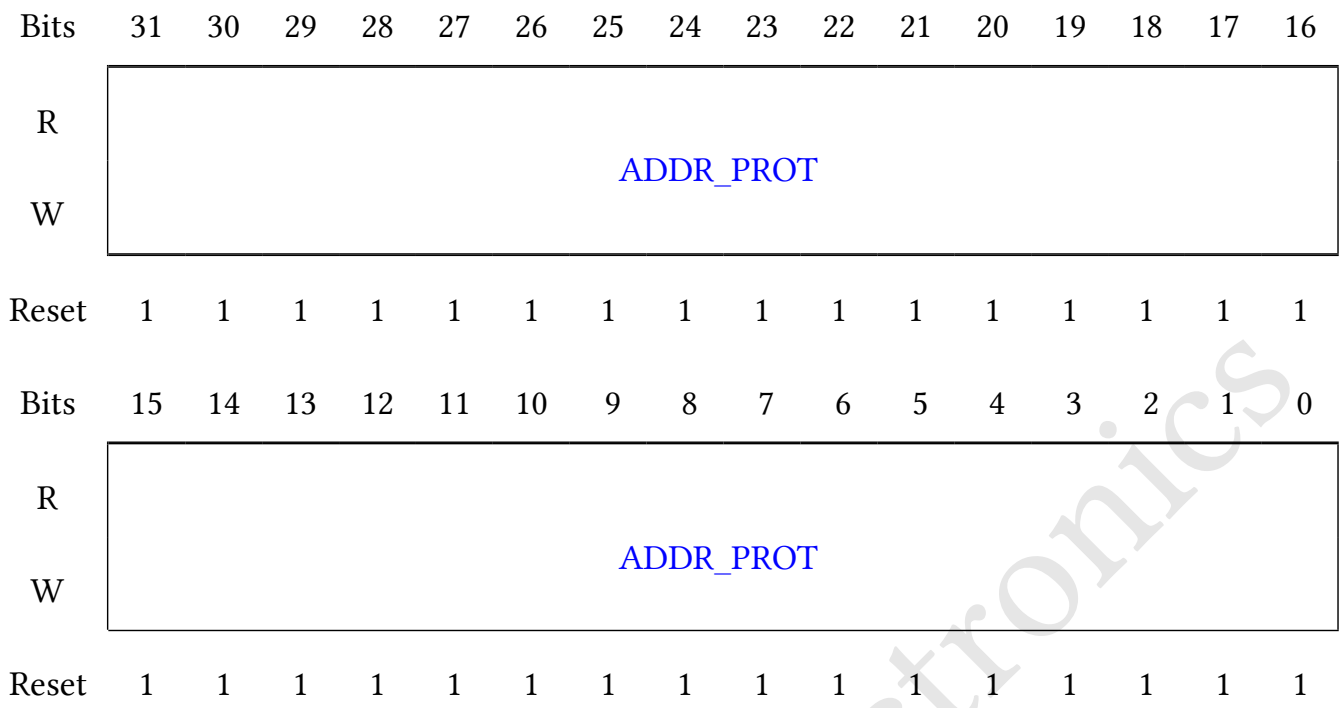


Table 3.6: EFM AP0 Register Description

Field	Function
31 - 0 ADDR_PROT	**Address Protect** Each bit protects 2K bytes address area. if the bit is set to 0, the corresponding 2K area is under protection.

Yuntu Microelectronics

3.2.1.5 EFM AP1 Register

Offset: 28h

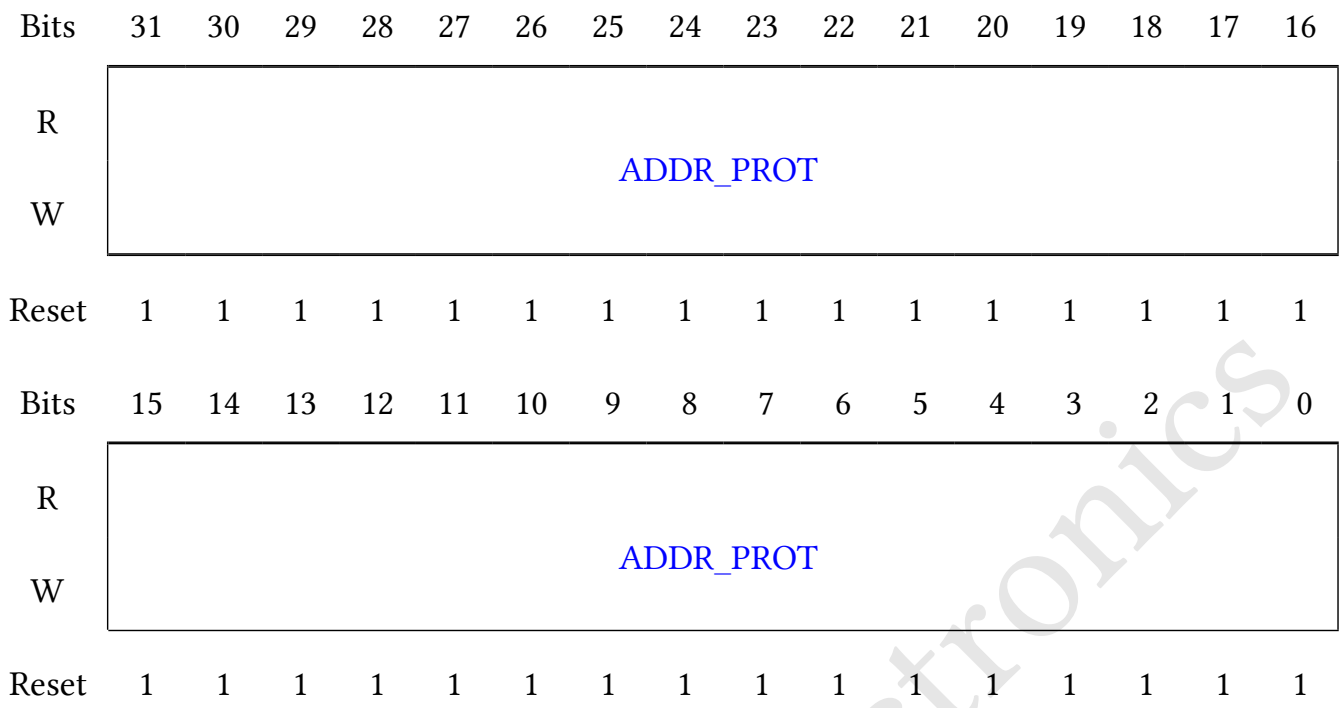


Table 3.7: EFM AP1 Register Description

Field	Function
31 - 0 ADDR_PROT	**Address Protect** Each bit protects 2K bytes address area. if the bit is set to 0, the corresponding 2K area is under protection.

Yuntu Microelectronics

3.2.1.6 EFM CMD_UNLOCK Register

Offset: 200h

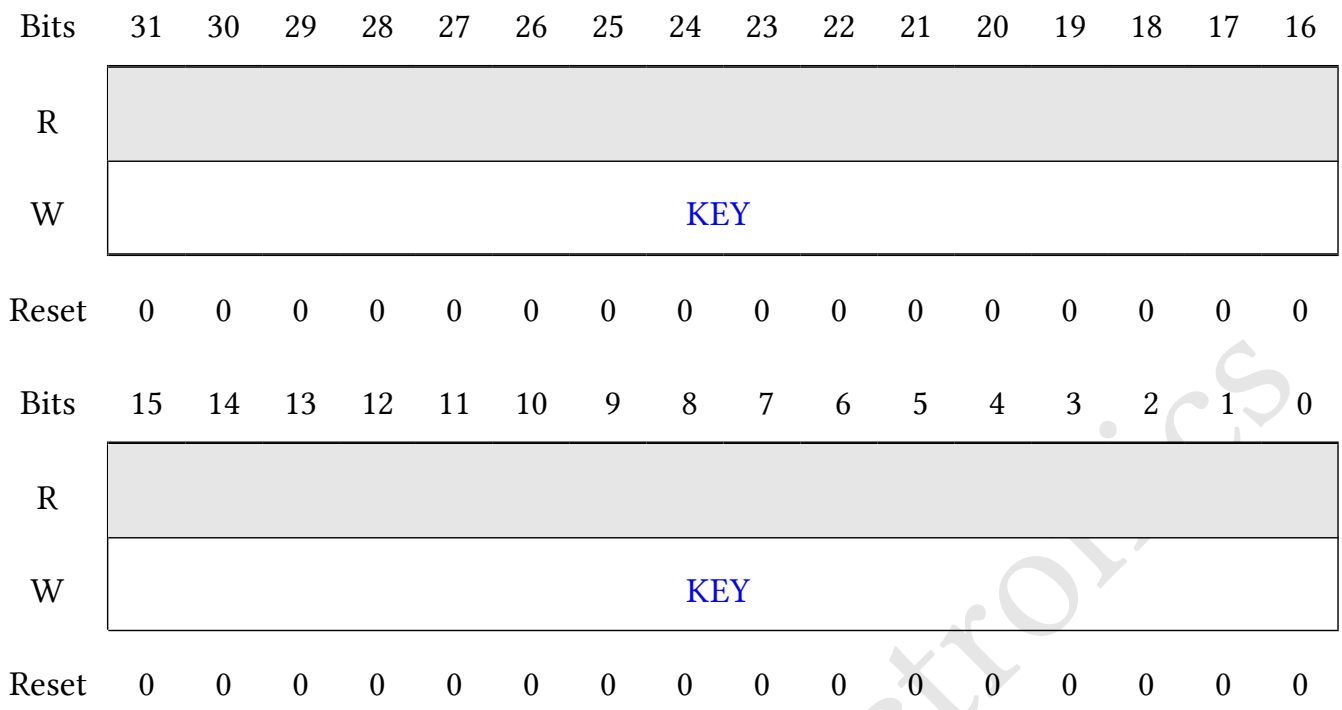


Table 3.8: EFM CMD_UNLOCK Register Description

Field	Function
31 - 0 KEY	**EFM Command Unlock Key** Please write 0xfd9573f5 before launching any command

Yuntu Microelectronics

3.2.1.7 EFM CUS_KEY Register

Offset: 204h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	KEY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	KEY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.9: EFM CUS_KEY Register Description

Field	Function
31 - 0 KEY	**EFM Customer Key** Please write Customer key (0x4dff32) before accessing to Customer NVR. If the key is not matched, accessing to Customer NVR will generate HARDFAULT.

Yuntu Microelectronics

3.2.1.8 EFM CTRL_OVRD Register

Offset: 300h

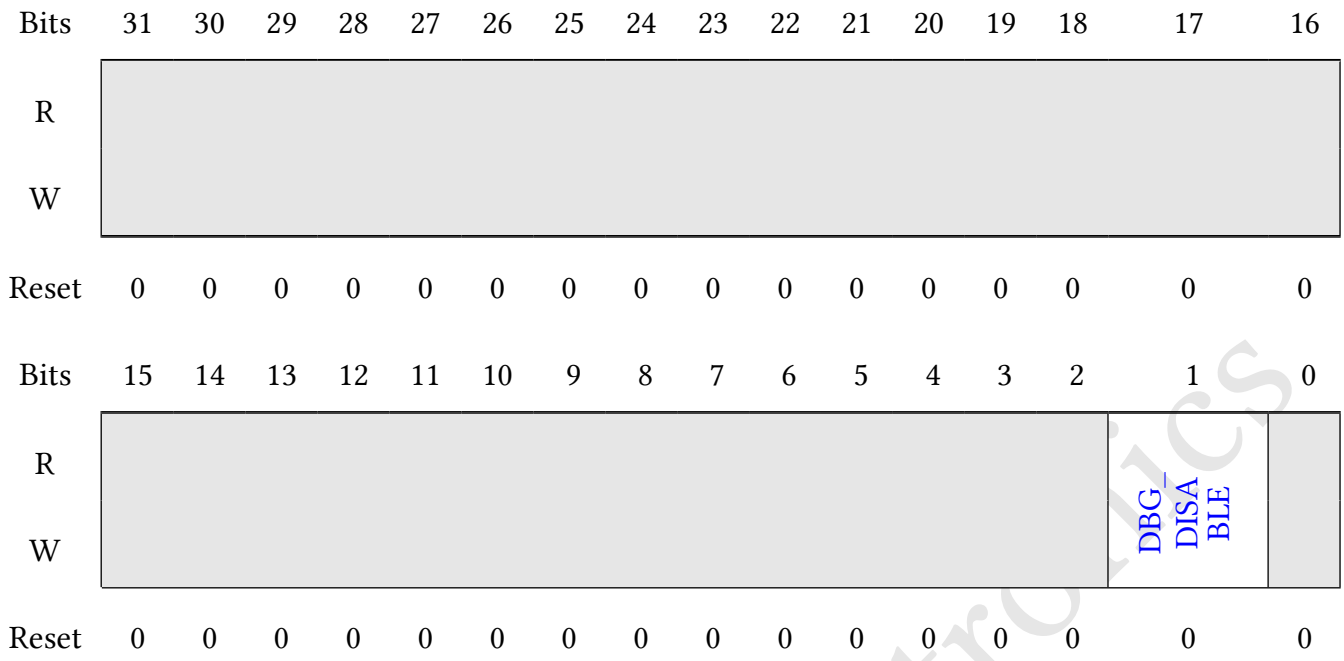


Table 3.10: EFM CTRL_OVRD Register Description

Field	Function
1 DBG_DISABLE	**Debugger Disable** This bit determines whether the debugger is override disabled. 0b - Debugger is not override disabled 1b - Debugger is override disabled

3.3 Functional Description

3.3.1 Program and Erase Sequences

This three-step command write sequence must be strictly followed. No intermediate writes to the EFM module are permitted between these three steps.

- Read STS[IDLE] bit. If STS[IDLE] is set, the address, data, and command are ready for a new command sequence to begin.
- Write the word to be programmed to its location in the EFM array. The address and data will be stored in internal buffers. All address bits are valid for program commands. The value of the data that is written for erase commands is ignored. For chip erase, the address can be any location of the EFM array. For sector erase, address bits[8:0] are ignored.
- Write the valid command to CMD register to launch the command. STS[IDLE] will be cleared automatically when the command is launched.

When command execution is completed, the FLASH state machine will set the STS[IDLE] and STS[DONE] flags that indicate the address, data, and command are ready for a new command sequence to begin.

STS[DONE] should be cleared before commencing another command.

Yuntu Microelectronics

The FLASH state machine will set STS[ACCERR] in command write sequences if there are any illegal operations exist. The STS[ACCERR] must be cleared before commencing another command.

3.3.2 Disable JTAG and SWD

Program 0x5A5A5A5A to 0x10000A00 which is the first address of CUS_NVR, after reset, the connection between chip and debugger will be disabled.

Please be caution, this feature should be used carefully, once the connection disabled, customers will not be able to re-enabled it.

3.3.3 Address Protection

There are 2 registers (EFM_PROTn) that protect flash memory which means the main array is divided into 64 parts and each part is 2KB.

The initial value of EFM_PROTn is 0xFFFFFFFF, if user wants to protect any address, the corresponding bit should be set to 0.

Once any bit is set to 0, corresponding part will be protected, any program or sector erase command to this part will be canceled.

Once any bit is set to 0, block erase command to this block will be canceled as well.

Write 0 to EFM_PROTn is valid but write 1 to EFM_PROTn will not be effective.

There are 2 ways to configure EFM_PROTn:

- Write data to EFM_PROTn registers
- Program data to 0x10000A04 and 0x10000A08 will set initial values for EFM_PROTn (effective after reset).

3.3.4 Flash Illegal Operation

The STS[ACCERR] bit will be set during a command write sequence if any of the illegal operations below are performed.

- Writing or reading the EFM array when the command sequence is busy(STS[IDLE] = 0) will not be effective and set STS[ACCERR].
- Writing a new command to EFMCMD when the command sequence is busy(STS[IDLE] = 0) will not be effective and set STS[ACCERR]. The command register will not be updated.
- Program or sector erase NVR when not permitted will not be effective and set STS[ACCERR].
- Trying to program or sector erase the foundry NVR will not be effective and set STS[ACCERR].

3.3.5 DFlash Operation

Please follow below setup configurations before using DFlash:

- Disable all the interrupts
- Do not access to PFlash when operating DFlash

Yuntu Microelectronics

Part III

Signal Multiplexing and Pin Assignment

Yuntu Microelectronics

Chapter 4

Pinmux

4.1 Pinmux Definition

The following table is for YTM32B1LE0x device.

Table 4.1: Pinmux Table

64 LQFP	48 LQFP	32 LQFP	32 QFN	NAME	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
1	1	1	-	PTD_1	-	PTD_1	eTMR0_CH3	SPI1_SIN	-	I2C1_SCL	-	TMU_OUT2
2	2	2	-	PTD_0	-	PTD_0	eTMR0_CH2	SPI1_SCK	-	I2C1_SDA	-	TMU_OUT1
3	-	-	-	PTE_11	-	PTE_11	SPI2_PCS0	lpTMR0_ALT1	-	-	-	TMU_OUT5
4	-	-	-	PTE_10	-	PTE_10	CLKOUT	SPI2_PCS1	-	-	-	TMU_OUT4
5	3	-	1	PTE_5	-	PTE_5	TCLK_IN2	-	eTMR1_CH1	CAN0_TX	-	-
6	4	-	2	PTE_4	-	PTE_4	CLKOUT	-	eTMR1_CH0	CAN0_RX	-	-
7	5	3	3	VDD	VDD	-	-	-	-	-	-	-
8	6	4	-	VDDA	VDDA	-	-	-	-	-	-	-
9	-	-	-	VREFH	VREFH	-	-	-	-	-	-	-
-	-	5	-	VREFL	VREFL	-	-	-	-	-	-	-
10	7	6	4	VSS	VSS	-	-	-	-	-	-	-
11	8	7	5	PTB_7	EXTAL	PTB_7	I2C0_SCL	-	UART2_TX	-	-	TMU_OUT2
12	9	8	6	PTB_6	XTAL	PTB_6	I2C0_SDA	-	UART2_RX	-	-	TMU_OUT1
13	-	-	-	PTE_3	-	PTE_3	eTMR0_FLT0	SPI1_SIN	-	-	TMU_IN6	ACMP0_OUT
14	10	-	7	PTD_16	EXTAL32K	PTD_16	eTMR0_CH1	-	SPI0_SIN	ACMP0_ACTIVE	-	-
15	11	-	8	PTD_15	XTAL32K	PTD_15	eTMR0_CH0	-	SPI0_SCK	-	-	-
16	12	-	-	PTE_9	-	PTE_9	eTMR0_CH7	SPI1_SCK	-	-	-	-
17	13	-	9	PTE_8	ACMP0_IN3	PTE_8	eTMR0_CH6	-	-	-	-	-
18	14	9	10	PTB_5	-	PTB_5	eTMR0_CH5	SPI0_PCS1	SPI0_PCS0	CLKOUT	TMU_IN0	-
19	15	10	11	PTB_4	-	PTB_4	eTMR0_CH4	SPI0_SOUT	-	-	TMU_IN1	-
20	16	11	12	PTC_3	ADC0_SE11 ACMP0_IN4	PTC_3	eTMR0_CH3	CAN0_TX	UART0_TX	-	-	-
21	17	12	13	PTC_2	ADC0_SE10 ACMP0_IN5	PTC_2	eTMR0_CH2	CAN0_RX	UART0_RX	-	-	-
22	-	-	-	PTD_7	ACMP0_IN6	PTD_7	UART2_TX	eTMR0_CH3	-	-	-	-
23	-	-	-	PTD_6	ACMP0_IN7	PTD_6	UART2_RX	eTMR0_CH2	-	-	-	-
24	18	-	14	PTD_5	-	PTD_5	-	lpTMR0_ALT2	-	-	TMU_IN7	-

Yuntu Microelectronics

64 LQFP	48 LQFP	32 LQFP	32 QFN	NAME	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
25	19	13	-	PTC_1	ADC0_SE9	PTC_1	eTMR0_CH1	SPI2_SOUT	SPI2_SIN	-	eTMR1_CH7	-
26	-	14	-	PTC_0	ADC0_SE8	PTC_0	eTMR0_CH0	SPI2_SIN	SPI2_PCS1	-	eTMR1_CH6	-
27	-	-	-	PTC_17	ADC0_SE15	PTC_17	eTMR1_FLT3	-	SPI2_PCS2	-	-	-
28	20	-	-	PTC_16	ADC0_SE14	PTC_16	eTMR1_FLT2	-	SPI2_SCK	-	-	-
29	21	-	-	PTC_15	ADC0_SE13	PTC_15	eTMR1_CH3	SPI2_SCK	SPI2_SOUT	-	TMU_IN8	-
30	22	-	-	PTC_14	ADC0_SE12	PTC_14	eTMR1_CH2	-	SPI2_PCS0	-	TMU_IN9	-
31	23	15	15	PTB_3	ADC0_SE7	PTB_3	eTMR1_CH1	SPI0_SIN	eTMR1_QD_PHA	-	TMU_IN2	-
32	24	16	16	PTB_2	ADC0_SE6	PTB_2	eTMR1_CH0	SPI0_SCK	eTMR1_QD_PHB	-	TMU_IN3	-
33	25	17	17	PTB_1	ADC0_SE5	PTB_1	UART0_TX	SPI0_SOUT	TCLK_IN0	CAN0_TX	-	-
34	26	18	18	PTB_0	ADC0_SE4	PTB_0	UART0_RX	SPI0_PCS0	lpTMR0_ALT3	CAN0_RX	-	-
35	27	-	-	PTC_9	-	PTC_9	UART1_TX	eTMR1_FLT1	-	-	-	-
36	28	-	-	PTC_8	-	PTC_8	UART1_RX	eTMR1_FLT0	-	-	-	-
37	29	19	19	PTA_7	ADC0_SE3	PTA_7	eTMR0_FLT2	-	RTC_CLKIN	-	-	-
38	-	20	-	PTA_6	ADC0_SE2	PTA_6	eTMR0_FLT1	SPI1_PCS1	-	-	-	-
39	-	-	-	PTE_7	-	PTE_7	eTMR0_CH7	SPI1_PCS2	-	-	-	-
40	30	-	20	VSS	VSS	-	-	-	-	-	-	-
41	31	-	21	VDD	VDD	-	-	-	-	-	-	-
42	32	-	-	PTB_13	-	PTB_13	eTMR0_CH1	-	-	-	-	-
43	-	-	-	PTB_12	-	PTB_12	eTMR0_CH0	-	-	-	-	-
44	-	-	-	PTD_4	-	PTD_4	eTMR0_FLT3	-	-	-	-	-
45	33	21	22	PTD_3	-	PTD_3	-	SPI1_PCS0	I2C1_SCL	eTMR2_CH0	TMU_IN4	NMI_b
46	34	22	-	PTD_2	-	PTD_2	-	SPI1_SOUT	I2C1_SDA	eTMR2_CH1	TMU_IN5	-
47	35	23	23	PTA_3	-	PTA_3	eTMR2_CH0	I2C0_SCL	SPI2_SCK	-	UART0_TX	-
48	36	24	24	PTA_2	-	PTA_2	eTMR2_CH1	I2C0_SDA	SPI2_SIN	-	UART0_RX	-
49	37	25	25	PTA_1	ADC0_SE1 ACMP0_IN1	PTA_1	eTMR1_CH1	-	SPI2_SOUT	eTMR1_QD_PHA	-	TMU_OUT0
50	38	26	26	PTA_0	ADC0_SE0 ACMP0_IN0	PTA_0	eTMR1_CH0	-	SPI2_PCS0	-	-	TMU_OUT3
51	39	27	-	PTC_7	-	PTC_7	UART1_TX	-	SPI2_PCS1	CAN0_TX	eTMR1_QD_PHA	-
52	40	28	-	PTC_6	-	PTC_6	UART1_RX	-	-	CAN0_RX	eTMR1_QD_PHB	-
53	-	-	-	PTE_6	-	PTE_6	SPI0_PCS2	-	-	-	-	-
54	-	-	-	PTE_2	-	PTE_2	SPI0_SOUT	lpTMR0_ALT3	-	-	-	-
55	41	-	-	PTA_13	-	PTA_13	eTMR1_CH7	-	-	UART2_TX	-	-
56	42	-	-	PTA_12	-	PTA_12	eTMR1_CH6	-	-	UART2_RX	-	-
57	43	-	27	PTA_11	-	PTA_11	eTMR1_CH5	-	I2C1_SCL	ACMP0_ACTIVE	-	-
58	44	-	28	PTA_10	-	PTA_10	eTMR1_CH4	-	I2C1_SDA	-	-	-
59	-	-	-	PTE_1	-	PTE_1	SPI0_SIN	-	I2C1_SCL	SPI1_PCS0	eTMR1_FLT1	-
60	-	-	-	PTE_0	-	PTE_0	SPI0_SCK	TCLK_IN1	I2C1_SDA	SPI1_SOUT	eTMR1_FLT2	-
61	45	29	29	PTC_5	-	PTC_5	eTMR2_CH0	RTC_CLKOUT	eTMR1_CH1	-	-	-
62	46	30	30	PTC_4	ACMP0_IN2	PTC_4	eTMR1_CH0	RTC_CLKOUT	-	-	eTMR1_QD_PHB	SWD_CLK
63	47	31	31	PTA_5	-	PTA_5	-	TCLK_IN1	-	-	-	RESET_b
64	48	32	32	PTA_4	-	PTA_4	-	-	ACMP0_OUT	-	-	SWD_IO

Yuntu Microelectronics

Chapter 5

General Purpose Input/Output (GPIO)

5.1 Introduction

5.1.1 Features

Features of the GPIO module include:

- Port Data Input register.
- Port Data Output register with corresponding set/clear/toggle registers
- Port Data Output enable register
- Digital filter for data inputs.
- Inversion for data inputs
- Interrupt flag and enable registers for each pin
- Support for edge sensitive (rising, falling, both) or level sensitive (low, high)
- Asynchronous wake-up in low-power modes
- Pin interrupt is functional in all digital pin muxing modes

5.1.2 Modes of Operation

5.1.2.1 Active Mode

In Active mode, the GPIO module operates normally.

5.1.2.2 Sleep Mode

In Sleep mode, the GPIO module continues to operates normally and can be configured to wake up the chip from Sleep mode if an enable interrupt is detected.

5.1.2.3 Deepsleep Mode

In Deepsleep mode, the GPIO configuration is static and the GPIO can be configured to wake up the chip from Deepsleep mode via an asynchronous wakeup signal if an enabled interrupt is detected.

5.1.2.4 Standby Mode

In Standby mode, the GPIO configuration is static and the GPIO can be configured to wake up the chip from Standby mode via an asynchronous wakeup signal if an enabled interrupt is detected.

Yuntu Microelectronics

5.2 Register Definition

5.2.1 GPIO Memory Map

GPIOA base address 4000f000h
GPIOB base address 4000f100h
GPIOC base address 4000f200h
GPIOD base address 4000f300h
GPIOE base address 4000f400h

Table 5.1: GPIO Memory Map

Offset	Register	Width	Access	Reset Value
0h	Port Data Output Register (PDOR)	32	RW	00000000h
4h	Port Set Output Register (PSOR)	32	WO	00000000h
8h	Port Clear Output Register (PCOR)	32	WO	00000000h
ch	Port Toggle Output Register (PTOR)	32	WO	00000000h
10h	Port Data Input Register (PDIR)	32	RO	00000000h
14h	Port Output Enable Register (POER)	32	RW	00000000h
18h	Port Input Enable Register (PIER)	32	RW	FFFFFFFFh
1ch	Port Interrupt Status Flag Register (PIFR)	32	RW	00000000h
80h	Port Control Register (PCR0)	32	RW	00000000h
84h	Port Control Register (PCR1)	32	RW	00000000h
88h	Port Control Register (PCR2)	32	RW	00000000h
8ch	Port Control Register (PCR3)	32	RW	00000000h
90h	Port Control Register (PCR4)	32	RW	00000000h
94h	Port Control Register (PCR5)	32	RW	00000000h
98h	Port Control Register (PCR6)	32	RW	00000000h
9ch	Port Control Register (PCR7)	32	RW	00000000h
a0h	Port Control Register (PCR8)	32	RW	00000000h
a4h	Port Control Register (PCR9)	32	RW	00000000h
a8h	Port Control Register (PCR10)	32	RW	00000000h
ach	Port Control Register (PCR11)	32	RW	00000000h
b0h	Port Control Register (PCR12)	32	RW	00000000h
b4h	Port Control Register (PCR13)	32	RW	00000000h
b8h	Port Control Register (PCR14)	32	RW	00000000h
bch	Port Control Register (PCR15)	32	RW	00000000h
c0h	Port Control Register (PCR16)	32	RW	00000000h
c4h	Port Control Register (PCR17)	32	RW	00000000h
c8h	Port Control Register (PCR18)	32	RW	00000000h
cch	Port Control Register (PCR19)	32	RW	00000000h
d0h	Port Control Register (PCR20)	32	RW	00000000h
d4h	Port Control Register (PCR21)	32	RW	00000000h
d8h	Port Control Register (PCR22)	32	RW	00000000h
dch	Port Control Register (PCR23)	32	RW	00000000h
e0h	Port Control Register (PCR24)	32	RW	00000000h
e4h	Port Control Register (PCR25)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
e8h	Port Control Register (PCR26)	32	RW	00000000h
ech	Port Control Register (PCR27)	32	RW	00000000h
f0h	Port Control Register (PCR28)	32	RW	00000000h
f4h	Port Control Register (PCR29)	32	RW	00000000h
f8h	Port Control Register (PCR30)	32	RW	00000000h
fch	Port Control Register (PCR31)	32	RW	00000000h

5.2.1.1 GPIO PDOR Register

Offset: 0h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PDO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PDO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5.2: GPIO PDOR Register Description

Field	Function
31 - 0 PDO	**Port Data Output** When configured for general-purpose output, it shows the logic level on the corresponding pin.

5.2.1.2 GPIO PSOR Register

Offset: 4h

Yuntu Microelectronics

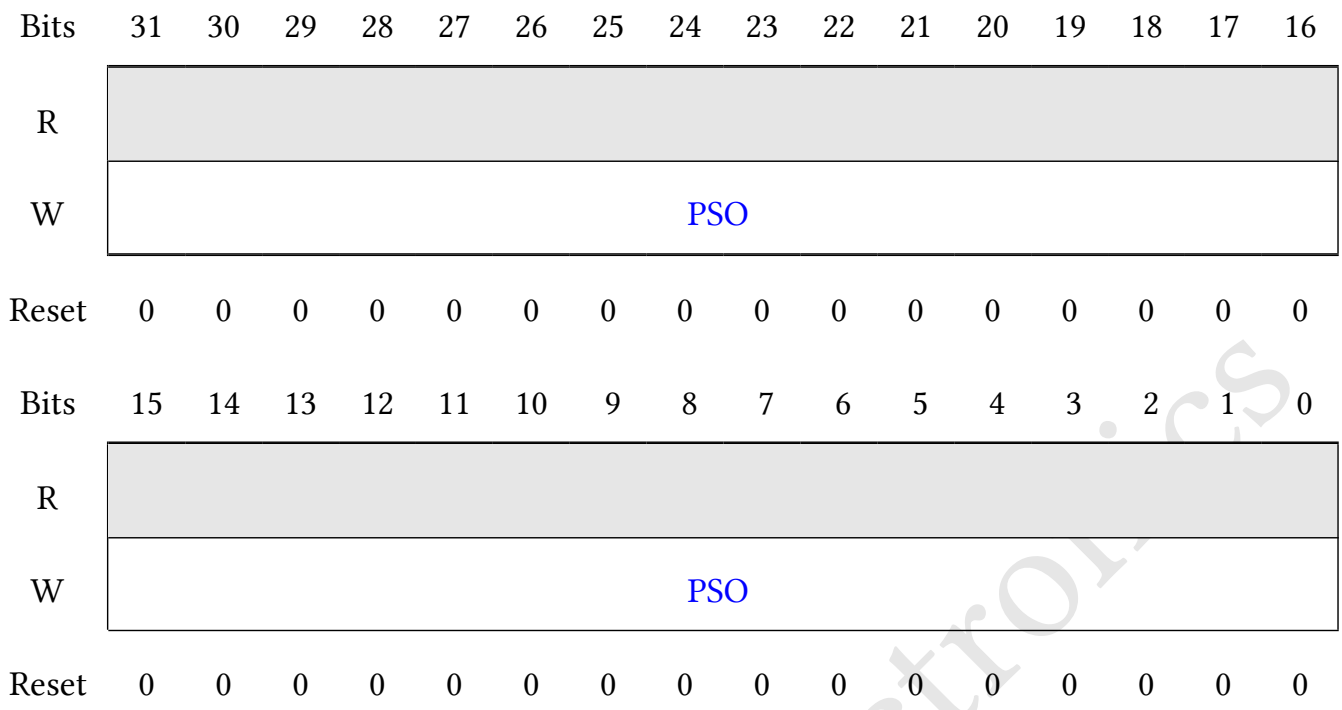


Table 5.3: GPIO PSOR Register Description

Field	Function
31 - 0 PSO	**Port Set Output** Writing to this register will update the contents of the corresponding bit in PDOR. 0b - No change. 1b - Corresponding bit in PDOR is set to logic 1.

Yuntu Microelectronics

5.2.1.3 GPIO PCOR Register

Offset: 8h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	PCO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	PCO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5.4: GPIO PCOR Register Description

Field	Function
31 - 0 PCO	**Port Clear Output** Writing to this register will update the contents of the corresponding bit in PDOR. 0b - No change. 1b - Corresponding bit in PDOR is set to logic 0.

Yuntu Microelectronics

5.2.1.4 GPIO PTOR Register

Offset: ch

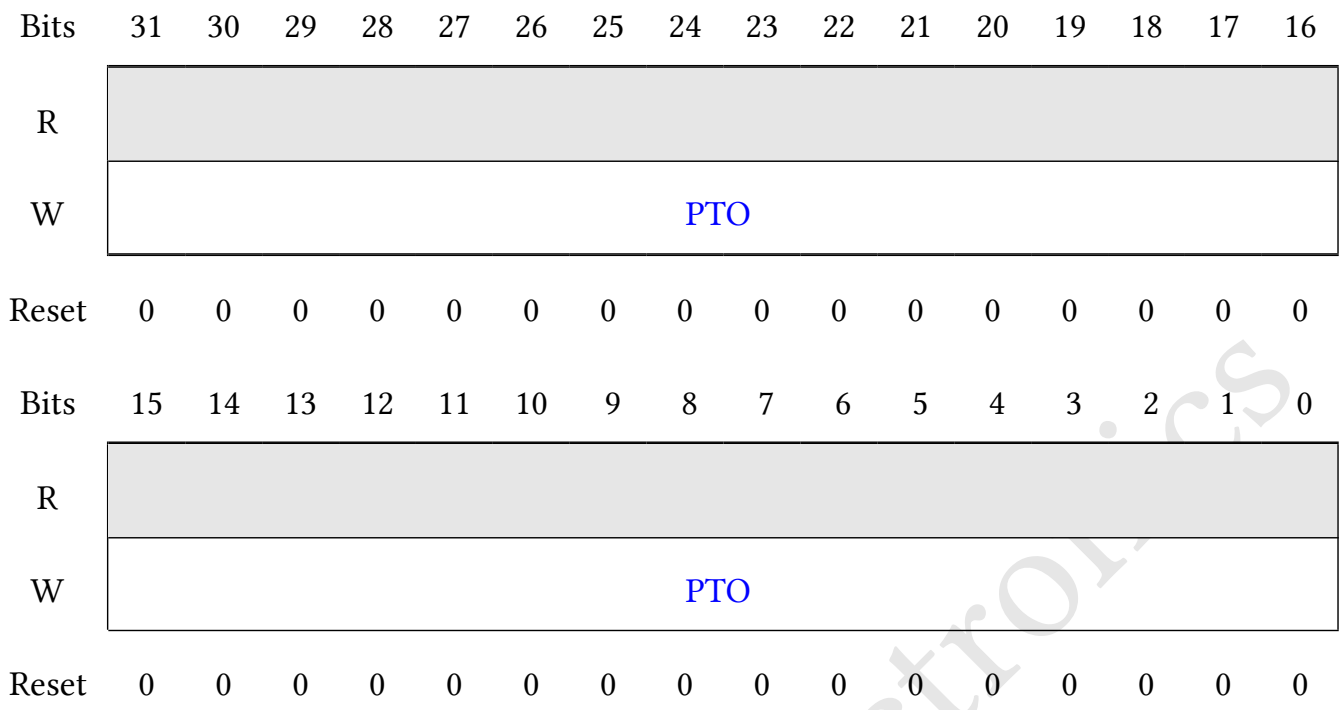


Table 5.5: GPIO PTOR Register Description

Field	Function
31 - 0 PTO	**Port Toggle Output** Writing to this register will update the contents of the corresponding bit in PDOR. 0b - No change. 1b - Corresponding bit in PDOR is set to the inverse of its existing logic state.

Yuntu Microelectronics

5.2.1.5 GPIO PDIR Register

Offset: 10h

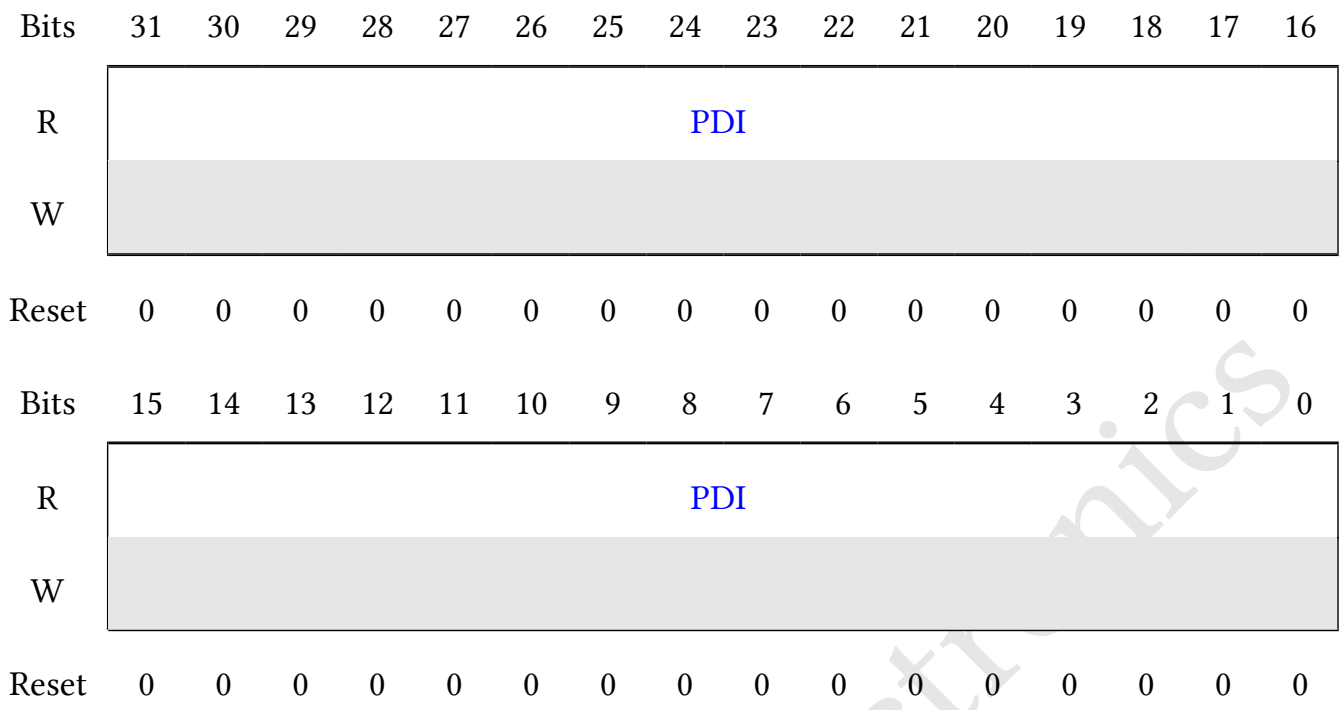


Table 5.6: GPIO PDIR Register Description

Field	Function
31 - 0 PDI	**Port Data Input** When configured for digital function, it shows the logic level on the corresponding pin. When configured for analog function, it always reads 0.

Yuntu Microelectronics

5.2.1.6 GPIO POER Register

Offset: 14h

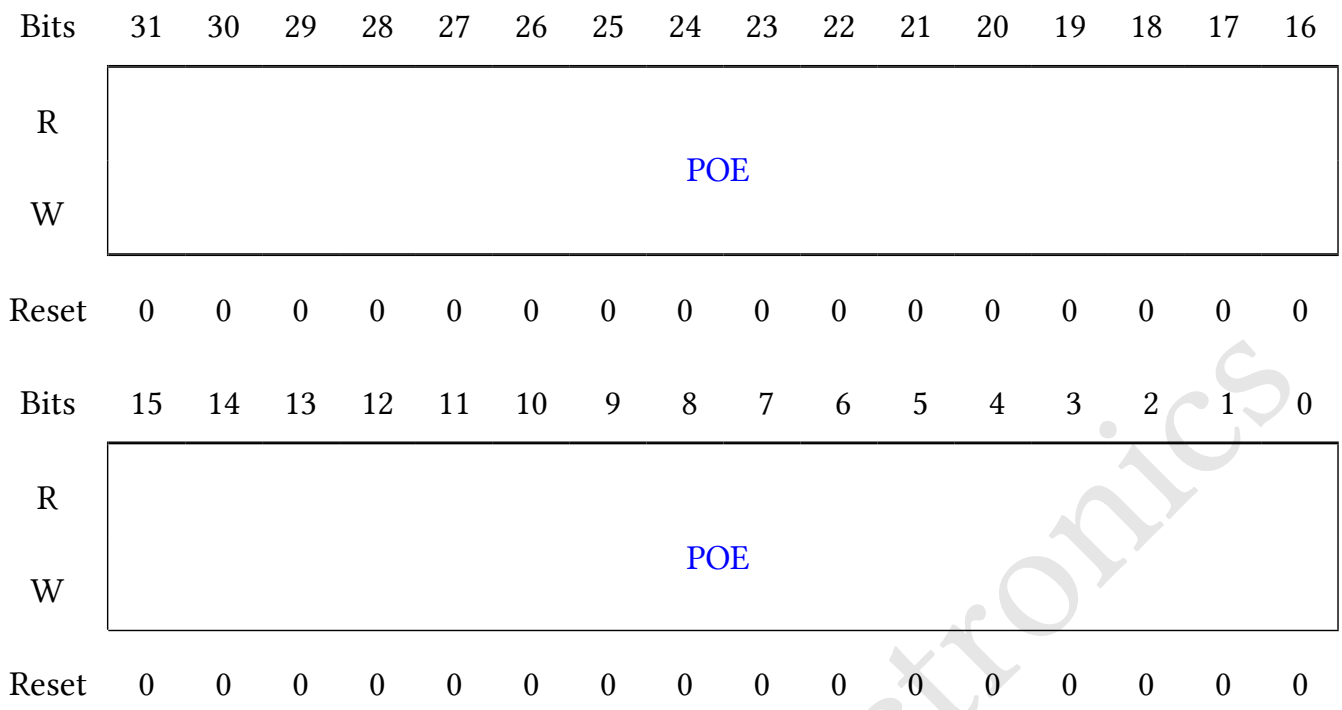


Table 5.7: GPIO POER Register Description

Field	Function
31 - 0 POE	**Port Output Enable** Configure the corresponding pin for output. 0b - Port output disable 1b - Port output enable

Yuntu Microelectronics

5.2.1.7 GPIO PIER Register

Offset: 18h

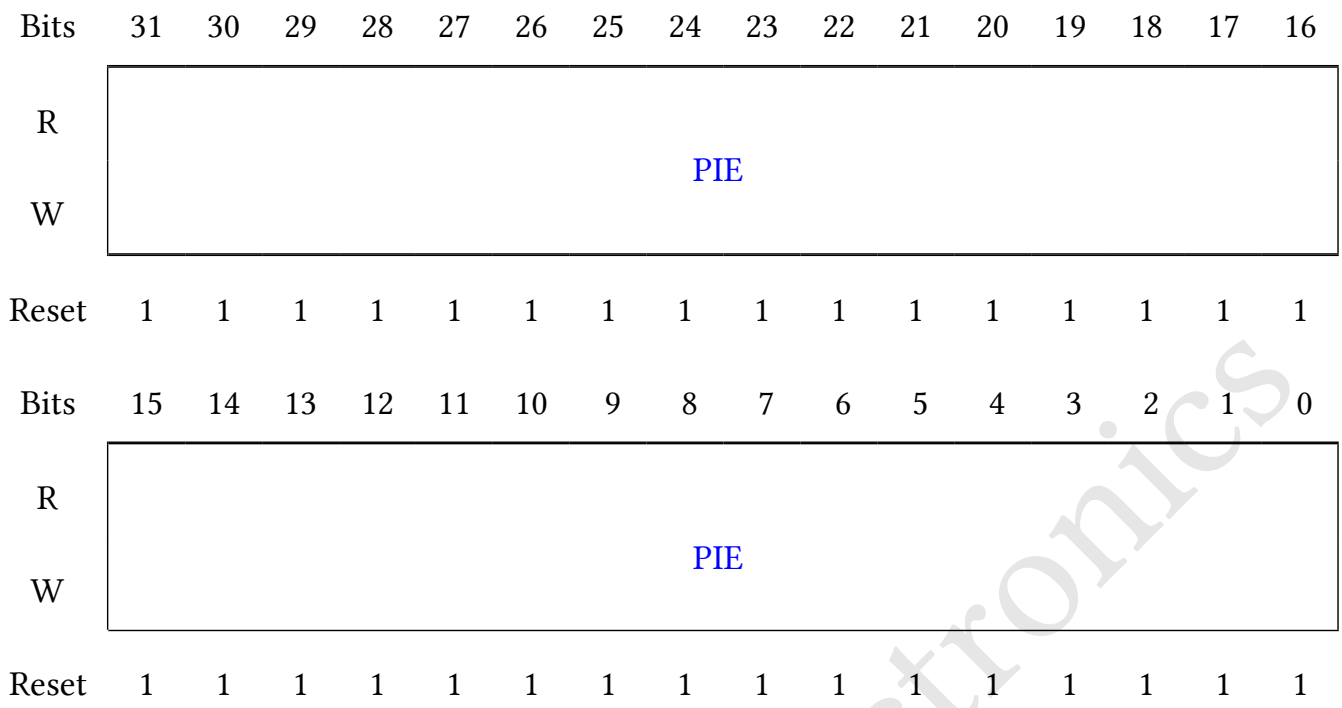


Table 5.8: GPIO PIER Register Description

Field	Function
31 - 0 PIE	**Port Input Enable** Configure the corresponding pin for input. 0b - Port input disable 1b - Port input enable

Yuntu Microelectronics

5.2.1.8 GPIO PIFR Register

Offset: 1ch

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PIF															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PIF															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5.9: GPIO PIFR Register Description

Field	Function
31 - 0 PIF	**Port Interrupt Status Flag** Indicate the detection for the defined event in GPIO[PCR] of the corresponding pin. 0b - No event detected. 1b - The event defined in GPIO[PCR] is detected.

Yuntu Microelectronics

5.2.1.9 GPIO PCR Register

Offset: 80h + n * 4h

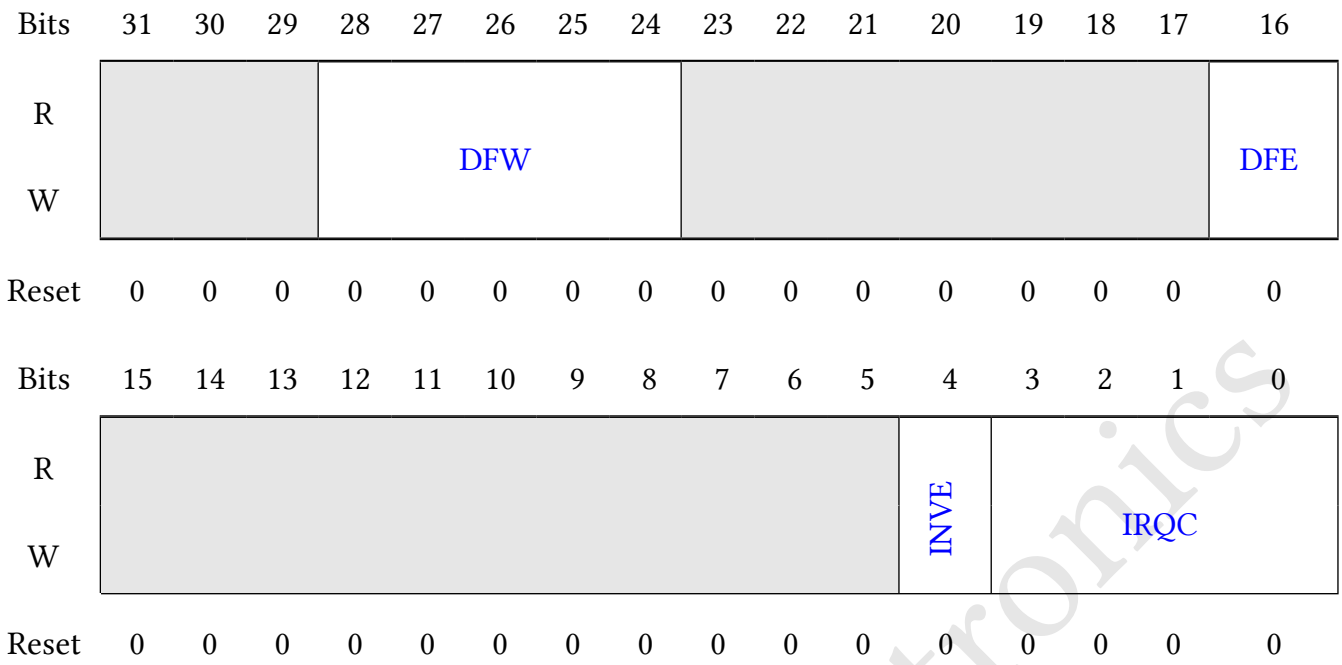


Table 5.10: GPIO PCR Register Description

Field	Function
28 - 24 DFW	**Digital Filter Width** The digital filter configuration is valid in all digital pin muxing modes. Glitches that are longer than this register setting will pass through the digital filter, and glitches that are equal to or less than this register setting are filtered. Changing the filter length must be done only after all filters are disabled.
16 DFE	**Digital Filter Enable** **NOTE**. Digital filter is not available in low power mode.
4 INVE	**Invert Enable** Invert the input logic level when configured for digital function.
3 - 0	**Interrupt Configuration**

Yuntu Microelectronics

Field	Function
IRQC	0000b - Disabled. 0001b - DMA request on rising edge. 0010b - DMA request on falling edge. 0011b - DMA request on either edge. 0100b - Reserved. 0101b - Reserved. 0110b - Reserved. 0111b - Reserved. 1000b - Interrupt when logic 0. 1001b - Interrupt on rising-edge. 1010b - Interrupt on falling-edge. 1011b - Interrupt on either edge. 1100b - Interrupt when logic 1. 1101b - Reserved. 1110b - Reserved. 1111b - Reserved.

5.3 Functional Description

5.3.1 General-purpose Input

Reading the PDIR register gets the logic state of GPIO pins when the pin is configured as GPIO input enable.

Glitch filter is provided to filter the narrow pulse, configure PCR[DFW] and PCR[DFE] to enable the digital glitch filter. Refer to the IP Control module (IPC) for the clock source of digital glitch filter.

5.3.2 General-purpose Output

Writing the PDOR registers controls the logic state of GPIO pins when the pin is configured as GPIO output enable.

The GPIO module supports bit manipulation on the GPIO outputs including pin data set, pin data clear and pin data toggle.

5.3.3 Port Interrupt and DMA Request

Each of port pin can generate DMA request or interrupt with controlled by PCR[IRQC].

5.3.4 IOPORT

The GPIO registers are also aliased to the IOPORT interface. Accesses via the IOPORT interface occur in parallel with any instruction fetches and will therefore complete in a single cycle. If another bus master attempts to access the GPIO registers on the same cycle as an IOPORT access, then the DMA access will stall until any IOPORT accesses have completed.

Yuntu Microelectronics

Chapter 6

Port Control (PCTRL)

6.1 Introduction

6.1.1 Overview

The Port Control (PCTRL) module provides support for port control. The functions should be configured independently for each pin in the 32-bit port and affect the pin regardless of its pin muxing state. Not all pins within each port are implemented.

6.1.2 Features

The PCTRL module has the following port control features:

- Individual pull control fields with pullup, pulldown, and pull-disable support
- Individual slew rate field supporting fast and slow slew rates
- Individual input passive filter field supporting enable and disable of the individual input passive filter on selected pins
- Individual mux control field supporting analog or pin disabled, GPIO, and up to 6 chip-specific digital functions
- Individual high drive strength on selected pins

6.1.3 Modes of Operation

6.1.3.1 Active Mode

In Active mode, the PCTRL module operates normally.

6.1.3.2 Sleep Mode

In sleep mode, the PCTRL module continues to operates normally.

6.1.3.3 Deepsleep Mode

In deepsleep mode, the PCTRL module will keep registers settings and operates normally.

6.1.3.4 Standby Mode

In Standby mode, the PCTRL module will keep registers settings and operates normally.

Yuntu Microelectronics

6.2 Register Definition

6.2.1 PCTRL Memory Map

PCTRLA base address 40049000h
PCTRLB base address 4004a000h
PCTRLC base address 4004b000h
PCTRLD base address 4004c000h
PCTRLE base address 4004d000h

Table 6.1: PCTRL Memory Map

Offset	Register	Width	Access	Reset Value
0h	Port Control Register (PCR0)	32	RW	see description
4h	Port Control Register (PCR1)	32	RW	see description
8h	Port Control Register (PCR2)	32	RW	see description
ch	Port Control Register (PCR3)	32	RW	see description
10h	Port Control Register (PCR4)	32	RW	see description
14h	Port Control Register (PCR5)	32	RW	see description
18h	Port Control Register (PCR6)	32	RW	see description
1ch	Port Control Register (PCR7)	32	RW	see description
20h	Port Control Register (PCR8)	32	RW	see description
24h	Port Control Register (PCR9)	32	RW	see description
28h	Port Control Register (PCR10)	32	RW	see description
2ch	Port Control Register (PCR11)	32	RW	see description
30h	Port Control Register (PCR12)	32	RW	see description
34h	Port Control Register (PCR13)	32	RW	see description
38h	Port Control Register (PCR14)	32	RW	see description
3ch	Port Control Register (PCR15)	32	RW	see description
40h	Port Control Register (PCR16)	32	RW	see description
44h	Port Control Register (PCR17)	32	RW	see description
48h	Port Control Register (PCR18)	32	RW	see description
4ch	Port Control Register (PCR19)	32	RW	see description
50h	Port Control Register (PCR20)	32	RW	see description
54h	Port Control Register (PCR21)	32	RW	see description
58h	Port Control Register (PCR22)	32	RW	see description
5ch	Port Control Register (PCR23)	32	RW	see description
60h	Port Control Register (PCR24)	32	RW	see description
64h	Port Control Register (PCR25)	32	RW	see description
68h	Port Control Register (PCR26)	32	RW	see description
6ch	Port Control Register (PCR27)	32	RW	see description
70h	Port Control Register (PCR28)	32	RW	see description
74h	Port Control Register (PCR29)	32	RW	see description
78h	Port Control Register (PCR30)	32	RW	see description
7ch	Port Control Register (PCR31)	32	RW	see description

Yuntu Microelectronics

6.2.1.1 PCTRL PCR Register

Port control register

Offset: 0h + n * 4h

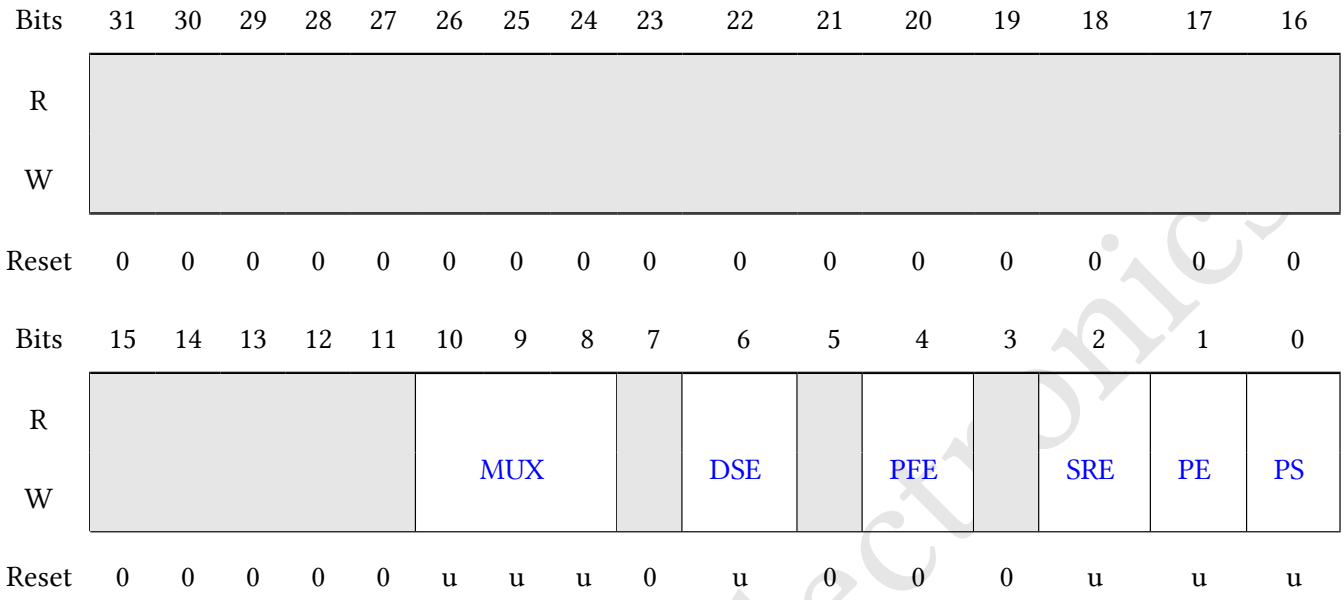


Table 6.2: PCTRL PCR Register Description

Field	Function
10 - 8 MUX	**Port Function Selection** Not all pin muxing slots are implemented. Unimplemented pin muxing slots are reserved. Please refer to pinmux table for detailed peripheral functions. The corresponding pin is configured in the following pin muxing slot as follows: 000b - Alternative 0, analog function, digital function is disabled. 001b - Alternative 1, GPIO function. 010b - Alternative 2. 011b - Alternative 3. 100b - Alternative 4. 101b - Alternative 5. 110b - Alternative 6. 111b - Alternative 7.
6 DSE	**Drive Strength Enable** Drive strength configuration for digital pin muxing. 0b - Low drive strength, if the pin is configured as a digital output. 1b - High drive strength, if the pin is configured as a digital output.
4 PFE	**Passive Filter Enable** Passive filter configuration for digital pin muxing. 0b - Passive input filter is disabled. 1b - Passive input filter is enabled, if the pin is configured as a digital input.

Yuntu Microelectronics

Field	Function
2 SRE	**Slew Rate Enable** Slew rate configuration for digital pin muxing. 0b - Fast slew rate, if the pin is configured as a digital output. 1b - Slow slew rate, if the pin is configured as a digital output.
1 PE	**Pull Enable** Pull configuration for digital pin muxing. 0b - Internal pullup or pulldown resistor is not enabled. 1b - Internal pullup or pulldown resistor is enabled.
0 PS	**Pull Select** Pull configuration for digital pin muxing. 0b - Internal pulldown resistor is enabled, if PE field is set. 1b - Internal pullup resistor is enabled, if PE field is set.

6.3 Functional Description

6.3.1 Port Control

Each port pin has a corresponding Port Control register. The Port Control register configures the following functions for each pin within the 32-bit port.

- Pullup or pulldown enable
- Slew rate configuration
- Passive input filter enable on selected pins
- Pin Muxing mode
- High drive strength enable

Yuntu Microelectronics

Part IV

Core and Systems

Yuntu Microelectronics

Chapter 7

CM0+ Overview

7.1 ARM Cortex-M0+ Core Configuration

This section summarizes how the module has been configured in the chip. Full documentation for this module is provided by ARM and can be found at www.arm.com.

7.2 Nested Vectored Interrupt Controller (NVIC)

Table 7.1: Interrupt Table

ID	Interrupt Vector	Description	Wake up
0	__StackTop	SP	-
1	Reset	PC	-
2	NMI	NMI	Y
3	HardFault	Hard Fault	-
4	-	-	-
5	-	-	-
6	-	-	-
7	-	-	-
8	-	-	-
9	-	-	-
10	-	-	-
11	SVC	SVCall	-
12	-	-	-
13	-	-	-
14	PendSV	PendableSrvReq	-
15	SysTick	SysTick	-
16	DMA0	DMA channel 0 transfer complete	-
17	DMA1	DMA channel 1 transfer complete	-
18	DMA2	DMA channel 2 transfer complete	-
19	DMA3	DMA channel 3 transfer complete	-
20	DMA_Error	DMA_Error	-
21	EMU	EMU	-
22	RTC	RTC alarm interrupt, RTC seconds interrupt	Y

Yuntu Microelectronics

Table 7.1 continued from previous page

ID	Interrupt Vector	Description	Wake up
23	SPI2	SPI2	Y
24	lpTMR	lpTMR	Y
25	GPIO	GPIOA/B/C/D/E	Y
26	CAN0_ORed	CAN0	Y
27	CAN0_ORed_0_15_MB	CAN0_MB_0_15	-
28	eTMR0_Ch0_Ch7	eTMR0	-
29	eTMR0_Fault	eTMR0	-
30	eTMR0_Ovf	eTMR0	-
31	eTMR1	eTMR1	-
32	eTMR2	eTMR2	-
33	CAN0_ORed_16_31_MB	CAN0_MB_16_31	-
34	EFM	EFM	-
35	-	-	-
36	pTMR	pTMR	-
37	PCU	PCU/CMU	Y
38	WDG	WDG	Y
39	UART2	UART2	Y
40	I2C0	I2C0	Y
41	I2C1	I2C1	Y
42	SPI0	SPI0	Y
43	SPI1	SPI1	Y
44	ADC0	ADC0	Y
45	ACMP0	ACMP0	Y
46	UART1	UART1	Y
47	UART0	UART0	Y

7.3 Debugger Configuration

The debug capability on this device is based on the Arm CoreSight™ architecture and this device only supports Serial Wire Debug(SWD) interface.

Yuntu Microelectronics

Chapter 8

Direct Memory Access (DMA)

8.1 Chip Specific Information

YTM32B1LE0x has 1 DMA instance with 4 channels, each channel support 41 hardware mux selection.

8.1.1 DMA Channel MUX table

DMA module has internal channel mux for each channel, which dma channel mux, each dma channel could support up to 41 hardware request triggers. With hardware request trigger support, DMA could process data transfer from memory to peripherals, memory to memory, peripherals to memory and peripherals to peripherals without CPU.

Table 8.1: DMA Channel Mux Table

No.	Module	DMA_Request Description
0	Disabled	RESERVED
1	-	-
2	UART0_RX	Receive DMA Request
3	UART0_TX	Transmit DMA Request
4	UART1_RX	Receive DMA Request
5	UART1_TX	Transmit DMA Request
6	UART2_RX	Receive DMA Request
7	UART2_TX	Transmit DMA Request
8	I2C1_RX	DMA RX Request
9	I2C1_TX	DMA TX Request
10	-	-
11	-	-
12	-	-
13	-	-
14	SPI0_RX	DMA RX Request
15	SPI0_TX	DMA TX Request
16	SPI1_RX	DMA RX Request
17	SPI1_TX	DMA TX Request
18	SPI2_RX	DMA RX Request
19	SPI2_TX	DMA TX Request

Yuntu Microelectronics

Table 8.1 continued from previous page

No.	Module	DMA_Request Description
20	eTMR1_CH0	Channel 0 DMA transfer request
21	eTMR1_CH1	Channel 1 DMA transfer request
22	eTMR1_CH2	Channel 2 DMA transfer request
23	eTMR1_CH3	Channel 3 DMA transfer request
24	eTMR1_CH4	Channel 4 DMA transfer request
25	eTMR1_CH5	Channel 5 DMA transfer request
26	eTMR1_CH6	Channel 6 DMA transfer request
27	eTMR1_CH7	Channel 7 DMA transfer request
28	eTMR2_CH0	Channel 0 DMA transfer request
29	eTMR2_CH1	Channel 1 DMA transfer request
30	-	-
31	-	-
32	-	-
33	-	-
34	-	-
35	-	-
36	eTMR0	'OR' of eTMR0 channel 0 - 7
37	-	-
38	-	-
39	-	-
40	-	-
41	-	-
42	ADC0	DMA request
43	-	-
44	I2C0_RX	DMA RX Request
45	I2C0_TX	DMA TX Request
46	-	-
47	-	-
48	ACMP0	DMA request
49	GPIOA	PORTA DMA request
50	GPIOB	PORTB DMA request
51	GPIOC	PORTC DMA request
52	GPIOD	PORTD DMA request
53	GPIOE	PORTE DMA request
54	FlexCAN0	FlexCAN DMA request
55	-	-
56	-	-
57	-	-
58	-	-
59	-	-
60	-	-
61	-	-
62	Always On	Always On
63	Always On	Always On

Yuntu Microelectronics

8.1.2 Power Mode Description

Table 8.2: DMA Power Mode Table

Power Mode	Description
Active	Full function
Sleep	Function disabled
Deepsleep	Function disabled
Debug	Support pause DMA transfer when CPU halt under debug mode

8.2 Introduction

Direct Memory Access (DMA) module allows hardware access SoC memory without CPU instructions. With DMA modules, CPU only have to configuration DMA channel transfer structure(CTS), and DMA will start the transfer automatically after given trigger comes.

In order to support various peripherals in chip, the DMA has an internal channel mux module to support hardware trigger selection from peripherals.

8.2.1 Features

- All address range data transfer from source to destination
- Support separate source/destination data size configuration
- Word(32-bit), half word(16-bit), byte(8-bit) transfer size
- Support separate source/destination address offset configuration
- Address increase/decrease/keep selectable
- Up to 4 DMA channels
- Fix priority and round-robin arbitration
- Support channel to channel link
- Software/Hardware/Link trigger
- Up to 41 peripheral hardware triggers
- Internal data fifo for data transfer
- Support update DMA transfer information from system memory after transfer complete
- Support data transfer loop and trigger loop

Yuntu Microelectronics

8.2.2 Block Diagram

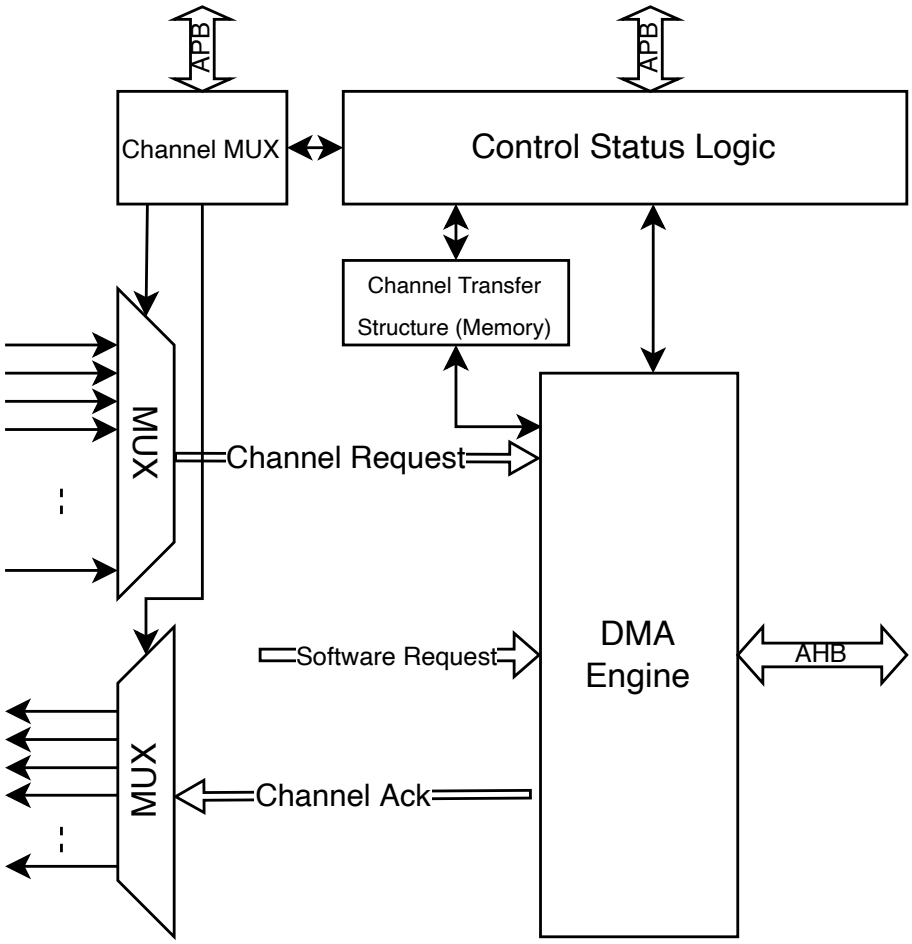


Figure 8.1: DMA Block Diagram

8.2.3 DMA Channel Transfer Structure (CTS)

In order to support different data transfer configuration, DMA use Channel Transfer Structure(CTS) for each transfer channel.

Table 8.3: DMA Channel Transfer Structure Table

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
Source address																															
SMOD					SSIZE			DMOD					DSIZE			SOFF															
Byte couter and loop done offset control																															
Source Trigger loop done offset																															
DADDR																															
Trigger counter and Trigger Loop Link information																DOFF															
Destination trigger loop done offset or ram reload address																															
Trigger counter and Trigger Loop Link information reload value																Channel control information															

NOTE. CTS information is stored in DMA internal memory, CTS reset value is random, so user have to initialize all CTS registers before using DMA.

8.2.4 DMA Trigger MUX

In order to support more hardware trigger source, DMA has an internal trigger MUX to mux hardware trigger source to DMA channel request.

NOTE. DMA MUX not support connect one hardware to different DMA channels simultaneously.

8.3 Register Definition

8.3.1 DMA Memory Map

DMA base address 40008000h

Table 8.4: DMA Memory Map

Offset	Register	Width	Access	Reset Value
0h	Control Register (CTRL)	32	RW	00003000h
4h	Channel Trigger Loop Done Interrupt Flag Register (CHTLDIF)	32	RW	00000000h
8h	Channel Trigger Loop Half Done Interrupt Flag Register (CHTLHDIF)	32	RW	00000000h
ch	DMA Channel Error Interrupt Flag Register (CHEIF)	32	RW	00000000h
10h	DMA Channel Error Interrupt Enable Register (CHEIE)	32	RW	00000000h
14h	Error Status Register (ERS)	32	RW	00000000h
18h	Hardware Request Enable Register (REQEN)	32	RW	00000000h
1ch	Request Status Register (REQS)	32	RW	00000000h
20h	Channel Start Register (START)	32	RW	00000000h
24h	Channel Done Register (DONE)	32	RW	00000000h
40h	Channel Mux Register (CHMUX0)	8	RW	00h
41h	Channel Mux Register (CHMUX1)	8	RW	00h
42h	Channel Mux Register (CHMUX2)	8	RW	00h
43h	Channel Mux Register (CHMUX3)	8	RW	00h
44h	Channel Mux Register (CHMUX4)	8	RW	00h
45h	Channel Mux Register (CHMUX5)	8	RW	00h
46h	Channel Mux Register (CHMUX6)	8	RW	00h
47h	Channel Mux Register (CHMUX7)	8	RW	00h
48h	Channel Mux Register (CHMUX8)	8	RW	00h
49h	Channel Mux Register (CHMUX9)	8	RW	00h
4ah	Channel Mux Register (CHMUX10)	8	RW	00h
4bh	Channel Mux Register (CHMUX11)	8	RW	00h
4ch	Channel Mux Register (CHMUX12)	8	RW	00h
4dh	Channel Mux Register (CHMUX13)	8	RW	00h
4eh	Channel Mux Register (CHMUX14)	8	RW	00h
4fh	Channel Mux Register (CHMUX15)	8	RW	00h
800h	Source Address Register (CTS0_SADDR)	32	RW	see description
804h	Source Address Offset Register (CTS0_SOFF)	16	RW	see description
806h	Transfer Control Register (CTS0_TCR)	16	RW	see description

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
c10h	Transfer Loop Byte Count (CTS0_BCNT_BCNT)	32	RW	see description
c10h	Transfer Loop Offset Disable Mapping (CTS0_BCNT_LODIS)	32	RW	see description
c10h	Transfer Loop Offset Enable Mapping (CTS0_BCNT_LOEN)	32	RW	see description
80ch	Source Trigger Loop Offset Register (CTS0_STO)	32	RW	see description
810h	Destination Address Register (CTS0_DADDR)	32	RW	see description
814h	Destination Address Offset Register (CTS0_DOFF)	16	RW	see description
c2ch	Trigger Count with Link Disable (CTS0_TCNT_LKDIS)	16	RW	see description
c2ch	Trigger Count with Link Enable (CTS0_TCNT_LKEN)	16	RW	see description
c30h	Destination Trigger Loop Offset Register (CTS0_DTO_RLD_DTO)	32	RW	see description
c30h	RAM Reload Address Register (CTS0_DTO_RLD_RLD)	32	RW	see description
81ch	Control and Status Register (CTS0_CSR)	16	RW	see description
81eh	TCNT Reload Value Register (CTS0_TCNTRV)	16	RW	see description
820h	Source Address Register (CTS1_SADDR)	32	RW	see description
824h	Source Address Offset Register (CTS1_SOFF)	16	RW	see description
826h	Transfer Control Register (CTS1_TCR)	16	RW	see description
c30h	Transfer Loop Byte Count (CTS1_BCNT_BCNT)	32	RW	see description
c30h	Transfer Loop Offset Disable Mapping (CTS1_BCNT_LODIS)	32	RW	see description
c30h	Transfer Loop Offset Enable Mapping (CTS1_BCNT_LOEN)	32	RW	see description
82ch	Source Trigger Loop Offset Register (CTS1_STO)	32	RW	see description
830h	Destination Address Register (CTS1_DADDR)	32	RW	see description
834h	Destination Address Offset Register (CTS1_DOFF)	16	RW	see description
c4ch	Trigger Count with Link Disable (CTS1_TCNT_LKDIS)	16	RW	see description
c4ch	Trigger Count with Link Enable (CTS1_TCNT_LKEN)	16	RW	see description
c50h	Destination Trigger Loop Offset Register (CTS1_DTO_RLD_DTO)	32	RW	see description
c50h	RAM Reload Address Register (CTS1_DTO_RLD_RLD)	32	RW	see description
83ch	Control and Status Register (CTS1_CSR)	16	RW	see description
83eh	TCNT Reload Value Register (CTS1_TCNTRV)	16	RW	see description
840h	Source Address Register (CTS2_SADDR)	32	RW	see description
844h	Source Address Offset Register (CTS2_SOFF)	16	RW	see description
846h	Transfer Control Register (CTS2_TCR)	16	RW	see description
c50h	Transfer Loop Byte Count (CTS2_BCNT_BCNT)	32	RW	see description
c50h	Transfer Loop Offset Disable Mapping (CTS2_BCNT_LODIS)	32	RW	see description
c50h	Transfer Loop Offset Enable Mapping (CTS2_BCNT_LOEN)	32	RW	see description
84ch	Source Trigger Loop Offset Register (CTS2_STO)	32	RW	see description
850h	Destination Address Register (CTS2_DADDR)	32	RW	see description
854h	Destination Address Offset Register (CTS2_DOFF)	16	RW	see description

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
c6ch	Trigger Count with Link Disable (CTS2_TCNT_LKDIS)	16	RW	see description
c6ch	Trigger Count with Link Enable (CTS2_TCNT_LKEN)	16	RW	see description
c70h	Destination Trigger Loop Offset Register (CTS2_DTO_RLD_DTO)	32	RW	see description
c70h	RAM Reload Address Register (CTS2_DTO_RLD_RLD)	32	RW	see description
85ch	Control and Status Register (CTS2_CSR)	16	RW	see description
85eh	TCNT Reload Value Register (CTS2_TCNTRV)	16	RW	see description
860h	Source Address Register (CTS3_SADDR)	32	RW	see description
864h	Source Address Offset Register (CTS3_SOFF)	16	RW	see description
866h	Transfer Control Register (CTS3_TCR)	16	RW	see description
c70h	Transfer Loop Byte Count (CTS3_BCNT_BCNT)	32	RW	see description
c70h	Transfer Loop Offset Disable Mapping (CTS3_BCNT_LODIS)	32	RW	see description
c70h	Transfer Loop Offset Enable Mapping (CTS3_BCNT_LOEN)	32	RW	see description
86ch	Source Trigger Loop Offset Register (CTS3_STO)	32	RW	see description
870h	Destination Address Register (CTS3_DADDR)	32	RW	see description
874h	Destination Address Offset Register (CTS3_DOFF)	16	RW	see description
c8ch	Trigger Count with Link Disable (CTS3_TCNT_LKDIS)	16	RW	see description
c8ch	Trigger Count with Link Enable (CTS3_TCNT_LKEN)	16	RW	see description
c90h	Destination Trigger Loop Offset Register (CTS3_DTO_RLD_DTO)	32	RW	see description
c90h	RAM Reload Address Register (CTS3_DTO_RLD_RLD)	32	RW	see description
87ch	Control and Status Register (CTS3_CSR)	16	RW	see description
87eh	TCNT Reload Value Register (CTS3_TCNTRV)	16	RW	see description

8.3.1.1 DMA CTRL Register

Offset: 0h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ACTIVE			ACTCH									CX	ECX	PAUSE	POE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R						LOEN	CLM							DBGDIS		
W															CLRAD	STACH
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Yuntu Microelectronics

Table 8.5: DMA CTRL Register Description

Field	Function
31 ACTIVE	**DMA Active** Shows current DMA active state 0b - DMA is idle 1b - DMA is active
28 - 24 ACTCH	**DMA Active Channel** Shows current DMA active channel. This field only valid when DMA active, when DMA is idle ACTCH = 0
19 CX	**Cancel DMA Transfer** Setting this bit will terminate DMA data transfer after DMA finished current write operation. This bit is cleared by hardware after DMA data transfer terminated. 0b - DMA in normal operation 1b - Cancel DMA data transfer
18 ECX	**Error Cancel Transfer** Setting this bit will terminate DMA data transfer after DMA finished current write operation and set error flag. This bit is cleared by hardware after DMA data transfer terminated. 0b - DMA in normal operation 1b - Cancel DMA data transfer and set error flag
17 PAUSE	**Pause DMA Transfer** Pause DMA following channel transfer, active channel transfer is allowed to complete, following channel transfer can be resumed after this bit cleared. 0b - DMA in normal operation 1b - Pause DMA transfer
16 POE	**Pause DMA Transfer on Error** Pause DMA following channel transfer when any DMA error happened 0b - DMA in normal operation 1b - DMA transfer is paused as DMA error happens, software need to handle the error and clear this bit to continue following data transfer
10 LOEN	**Loop Offset Enable** Enable loop offset adjustment, this bit will affect all DMA channels transfer loop setting. 0b - DMA loop offset is disabled 1b - DMA loop is enabled and CTS[BCNT] split into SLOEN, DLOEN, OFFSET and BCNT
9	**Continuous Link Mode**

Yuntu Microelectronics

Field	Function
CLM	**NOTE**. Only set this bit when more than one DMA channel enabled 0b - DMA will serve other channel even if DMA channel linking to itself. 1b - DMA will restart current channel transfer if DMA channel linking to itself.
2 DBGDIS	**Debug Mode Disable** Setting this bit will pause DMA transfer when CPU is halted 0b - DMA in normal operation 1b - DMA transfer is paused when CPU is halted
1 CLRAD	**Clear All DMA Done** Setting this bit will clear all DMA channel done flags 0b - No effects 1b - Clear all DMA channel done flags
0 STACH	**Start All DMA Channel** Setting this bit will start all DMA channel 0b - No effects 1b - Start all DMA channels

8.3.1.2 DMA CHTLDIF Register

Offset: 4h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R													CH			
W													W1C			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8.6: DMA CHTLDIF Register Description

Field	Function
3 - 0 CH	**DMA Channel Trigger Loop Done Interrupt Flag** CH[n] is channel n trigger loop done interrupt flag, write 1 to clear channel interrupt flag.

8.3.1.3 DMA CHTLHDIF Register

Offset: 8h

Yuntu Microelectronics

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R													CH			
W													W1C			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8.7: DMA CHTLHDIF Register Description

Field	Function
3 - 0 CH	**DMA Channel Trigger Loop Half Done Interrupt Flag** CH[n] is channel n trigger loop half done interrupt flag, write 1 to clear channel interrupt flag.

8.3.1.4 DMA CHEIF Register

Offset: ch

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R													CH			
W													W1C			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8.8: DMA CHEIF Register Description

Field	Function
3 - 0	**DMA Channel Error Interrupt Flag**

Yuntu Microelectronics

Field	Function
CH	CH[n] is channel n error interrupt flag

8.3.1.5 DMA CHEIE Register

Offset: 10h

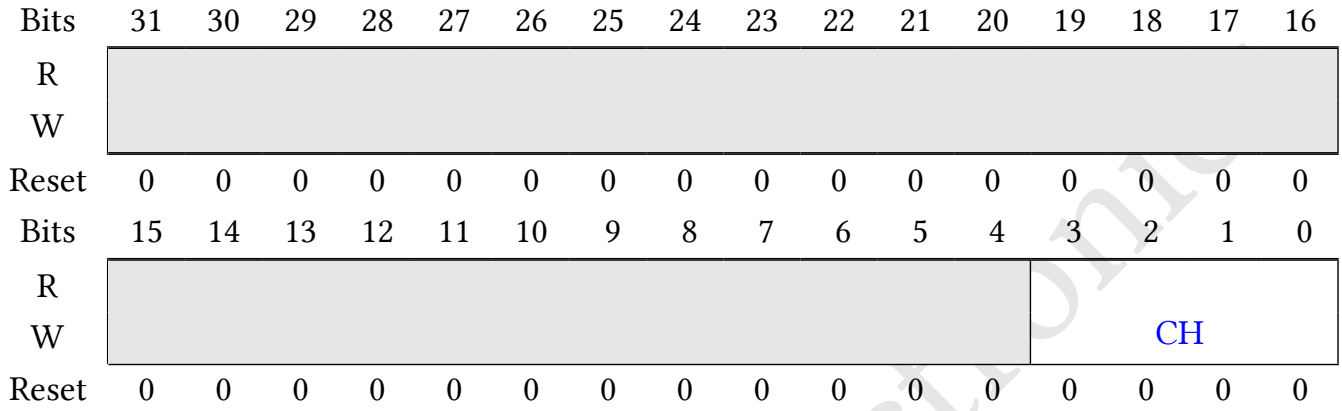
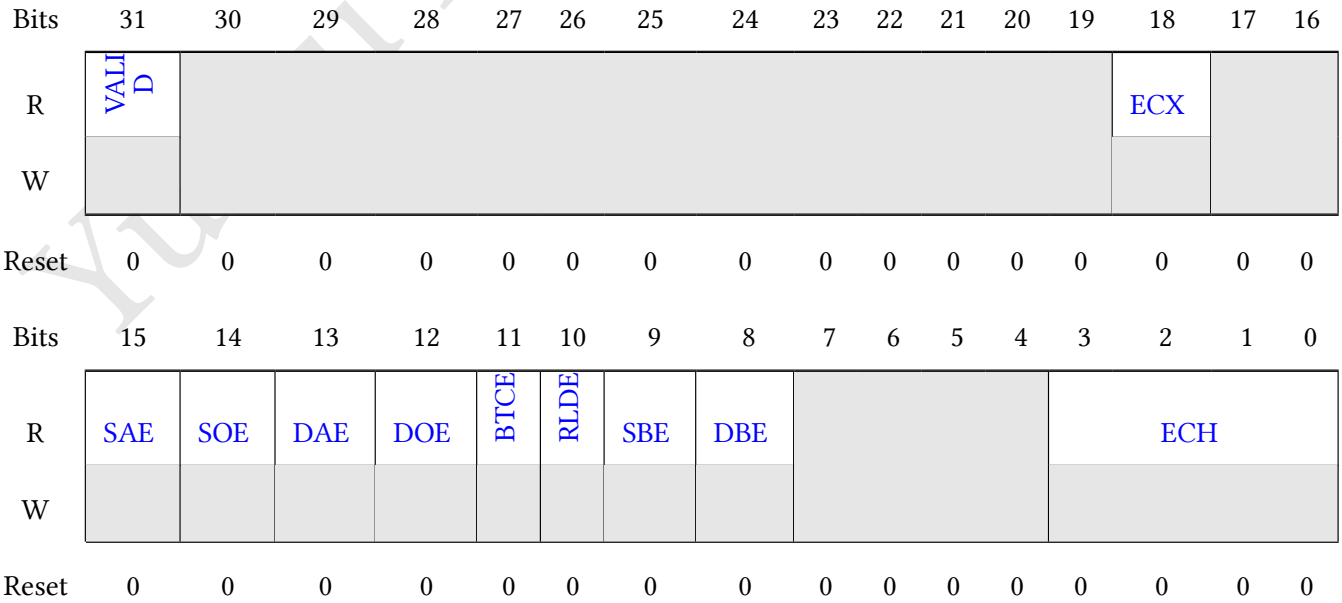


Table 8.9: DMA CHEIE Register Description

Field	Function
3 - 0 CH	**DMA Channel Error Interrupt Enable** CH[n] is channel n error interrupt enable control bit

8.3.1.6 DMA ERS Register

Offset: 14h



Yuntu Microelectronics

Table 8.10: DMA ERS Register Description

Field	Function
31 VALID	**DMA Configuration Valid** All DMA channel configurations are valid. 0b - No DMA error happens 1b - DMA error happens on at least one DMA channel
18 ECX	**Error Cancel Transfer** 0b - No DMA error happens 1b - DMA error happens because and data transfer error happened.
15 SAE	**Source Address Error** DMA channel source address configuration error 0b - No DMA source address error happens 1b - At least one DMA channel has source address configuration error. Check CTSn_SADDR and CTSn_TCR configuration.
14 SOE	**Source Offset Error** DMA source offset configuration error 0b - No DMA source offset error happens 1b - At least one DMA channel has source offset configuration error. Check CTSn_SOFF and CTSn_TCR configuration.
13 DAE	**Destination Address Error** DMA channel destination address configuration error 0b - No DMA destination address error happens 1b - At least one DMA channel has destination address configuration error. Check CTSn_DADDR and CTSn_TCR configuration.
12 DOE	**Destination Offset Error** DMA channel destination offset configuration error 0b - No DMA destination offset error happens 1b - At least one DMA channel has destination offset configuration error. Check CTSn_DOFF and CTSn_TCR configuration.
11 BTCE	**Byte/Trigger Count Configuration Error** Byte count or trigger count configuration not match DMA transfer size 0b - No DMA byte/trigger count error happens 1b - At least one DMA channel has byte/trigger count configuration error. Check CTSn_BCNT, CTSn_TCNT, CTSn_TCR configuration.
10	**Configuration RAM Reload Error**

Yuntu Microelectronics

Field	Function
RLDE	Target reload address is not aligned with 32 bytes 0b - No DMA ram reload error happens 1b - At least one DMA channel has ram reload configuration error. Check CTSn_RLDADDR(should be 32 byte align)
9 SBE	**Source Bus Error** Bus error happened when DMA reading data from source address. 0b - No DMA source bus error happens 1b - At least one DMA channel has source bus data access error. System bus error happens when DMA reading data from source address.
8 DBE	**Destination Bus Error** Bus error happened when DMA writing data to destination address. 0b - No DMA destination bus error happens 1b - At least one DMA channel has destination bus data access error. System bus error happened when dma write data to destination address.
3 - 0 ECH	**Error Channel Index** Channel index of error captured **NOTE**. This field is valid only when DMA error happened.

8.3.1.7 DMA REQEN Register

Offset: 18h

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

Table 8.11: DMA REQEN Register Description

Field	Function
3 - 0 CH	**DMA Channel Request Enable** Hardware DMA request for channel [n] enable control bit, one bit for one channel.

Yuntu Microelectronics

8.3.1.8 DMA REQS Register

Offset: 1ch

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

Table 8.12: DMA REQS Register Description

Field	Function
3 - 0 CH	**DMA Channel Request Status** Hardware DMA request status for channel [n], one bit for one channel.

8.3.1.9 DMA START Register

Offset: 20h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R													CH			
W													W1C			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8.13: DMA START Register Description

Field	Function
3 - 0 CH	**DMA Channel Start** Start DMA channel [n], one bit for one channel.

8.3.1.10 DMA DONE Register

Offset: 24h

Yuntu Microelectronics

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R													CH			
W													W1C			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8.14: DMA DONE Register Description

Field	Function
3 - 0 CH	**DMA Channel Done** Trigger loop done for channel [n], one bit for one channel.

8.3.1.11 DMA CHMUX Register

Offset: 40h + n * 1h

Bits	7	6	5	4	3	2	1	0
R								
W			CH					
Reset	0	0	0	0	0	0	0	0

Table 8.15: DMA CHMUX Register Description

Field	Function
5 - 0 CH	**Hardware Channel** DMA hardware request input index, please refer to DMA MUX table for MUX detail

8.3.1.12 DMA CTS_SADDR Register

Offset: 800h + n * 20h

Yuntu Microelectronics

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

Table 8.16: DMA CTS_SADDR Register Description

Field	Function
31 - 0 SADDR	**Source Address** Source address of DMA transfer.

8.3.1.13 DMA CTS_SOFF Register

Offset: 804h + n * 20h

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SOFF															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Table 8.17: DMA CTS_SOFF Register Description

Field	Function
15 - 0 SOFF	**Source Address Offset** Source address will add offset value(support signed-short type) after each DMA transfer size completed.

8.3.1.14 DMA CTS_TCR Register

Offset: 806h + n * 20h

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SMOD					SSIZE			DMOD					DSIZE		
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Yuntu Microelectronics

Table 8.18: DMA CTS_TCR Register Description

Field	Function
15 - 11 SMOD	**Source Address Modulo Value** Source address will add offset value(support signed-short type) and modulo SMOD after each DMA transfer size completed.
10 - 8 SSIZE	**Source Data Transfer Size** Source data transfer size is DMA access source data width. 000b - 8-bit 001b - 16-bit 010b - 32-bit
7 - 3 DMOD	**Destination Address Modulo** Destination address will add offset value(support signed-short type) and modulo SMOD after each DMA transfer size completed.
2 - 0 DSIZE	**Destination Data Transfer Size** Destination data transfer size is DMA write destination data width. 000b - 8-bit 001b - 16-bit 010b - 32-bit

8.3.1.15 DMA CTS_BCNT_BCNT Register

Transfer loop byte count is the number of bytes to be transferred in each trigger loop.

Offset: c10h + n * 20h

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

Table 8.19: DMA CTS_BCNT_BCNT Register Description

Field	Function
31 - 0 BCNT	**Transfer Loop Byte Count** Valid for CTRL[LOEN] = 0, Transfer loop byte count is the number of bytes to be transferred in each trigger loop.

Yuntu Microelectronics

8.3.1.16 DMA CTS_BCNT_LODIS Register

Offset: c10h + n * 20h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SLOE	DLOE	BCNT													
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BCNT															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Table 8.20: DMA CTS_BCNT_LODIS Register Description

Field	Function
31 SLOE	**Source Loop Offset Enable** 0b - Source loop offset disabled 1b - Source loop offset enable, refer LOEN for register information
30 DLOE	**Destination Minor Loop Offset Enable** 0b - Destination minor loop offset disabled 1b - Destination minor loop offset enable, refer LOEN for register information
29 - 0 BCNT	**Byte Transfer Count**

8.3.1.17 DMA CTS_BCNT_LOEN Register

Offset: c10h + n * 20h

Yuntu Microelectronics

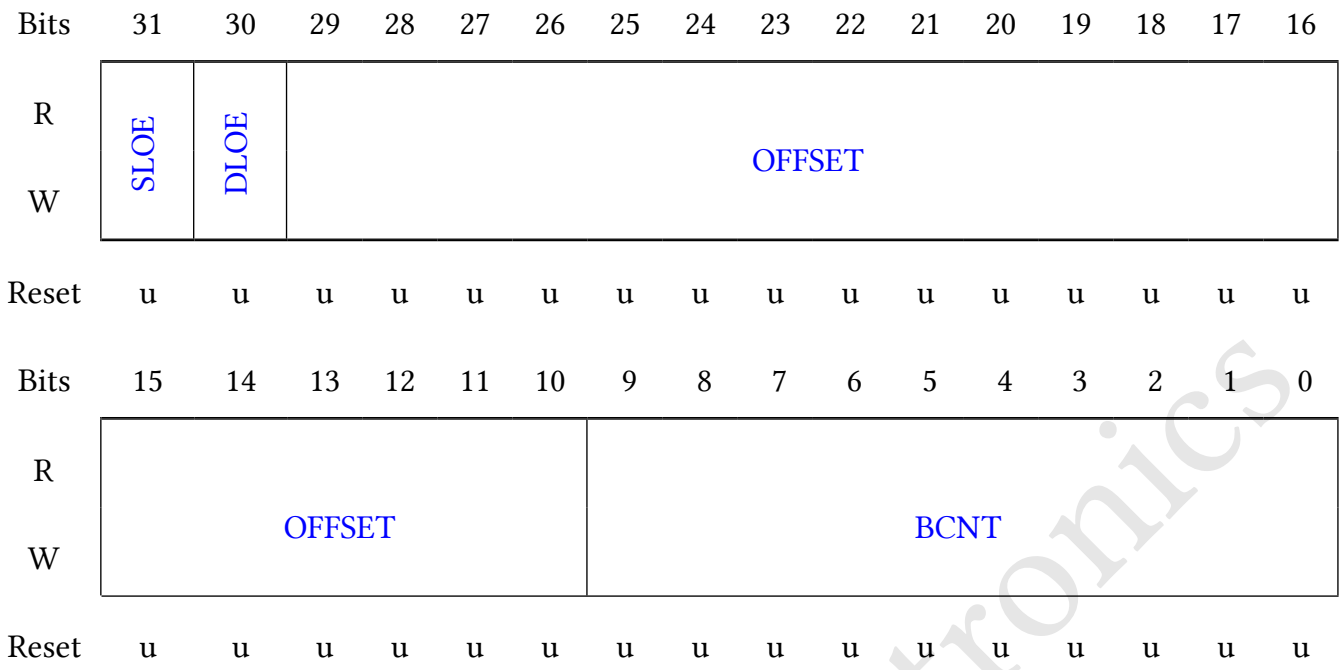


Table 8.21: DMA CTS_BCNT_LOEN Register Description

Field	Function
31 SLOE	**Source Loop Offset Enable** 0b - Source loop offset disabled, refer LODIS for register information 1b - Source loop offset enable
30 DLOE	**Destination Loop Offset Enable** 0b - Destination loop offset disabled, refer LODIS for register information 1b - Destination loop offset enable
29 - 10 OFFSET	**Loop Offset** Source or/and destination address will add offset value(support signed-short type) after each DMA loop completed.
9 - 0 BCNT	**Byte Transfer Count** Total byte count to be transferred in each trigger loop.

Yuntu Microelectronics

8.3.1.18 DMA CTS_STO Register

Offset: 80ch + n * 20h

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

Table 8.22: DMA CTS_STO Register Description

Field	Function
31 - 0 STLOFF	Source address will add offset value(support signed-short type) after DMA channel trigger loop completed.

8.3.1.19 DMA CTS_DADDR Register

Offset: 810h + n * 20h

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

Table 8.23: DMA CTS_DADDR Register Description

Field	Function
31 - 0 DADDR	**Destination Address** Destination address where DMA will transfer data to.

8.3.1.20 DMA CTS_DOFF Register

Offset: 814h + n * 20h

Yuntu Microelectronics

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DOFF															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Table 8.24: DMA CTS_DOFF Register Description

Field	Function
15 - 0 DOFF	**Destination Address Offset** Destination address will add offset value(support signed-short type) after each DMA transfer (write operation with DSIZE) completed.

8.3.1.21 DMA CTS_TCNT_LKDIS Register

Offset: c2ch + n * 20h

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LKEN	TCNT														
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Table 8.25: DMA CTS_TCNT_LKDIS Register Description

Field	Function
15 LKEN	**Link Enable** Enable channel to channel linking after DMA one trigger loop complete. This bit should be 0 when channel to channel link mode is disabled. 0b - Channel to channel linking is disabled. 1b - Channel to channel linking is enabled, refer LKEN for register information
14 - 0 TCNT	**Trigger Count** Trigger counter decrease 1 for every DMA trigger

8.3.1.22 DMA CTS_TCNT_LKEN Register

Offset: c2ch + n * 20h

Yuntu Microelectronics

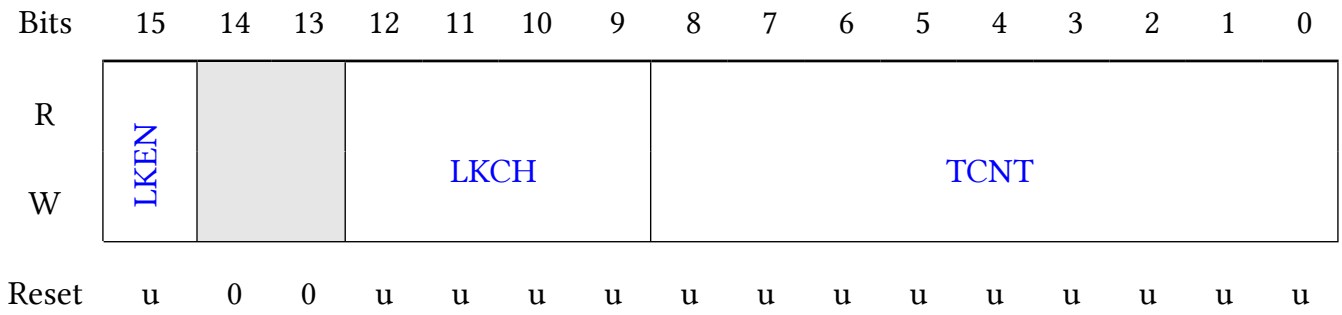


Table 8.26: DMA CTS_TCNT_LKEN Register Description

Field	Function
15 LKEN	**Link Enable** 0b - Channel to channel linking is disabled, refer LKDIS for register information 1b - Channel to channel linking is enabled.
12 - 9 LKCH	**Link Channel** DMA next trigger channel index after current dma trigger loop is done(one trigger)
8 - 0 TCNT	**Trigger Count** Trigger counter decrease 1 for every DMA trigger, when TCNT is 0, channel done flag set. **NOTE**. If new trigger happens and hardware request is enabled, TCNT will reload TCNT and serve new trigger request.

8.3.1.23 DMA CTS_DTO_RLD_DTO Register

Offset: c30h + n * 20h

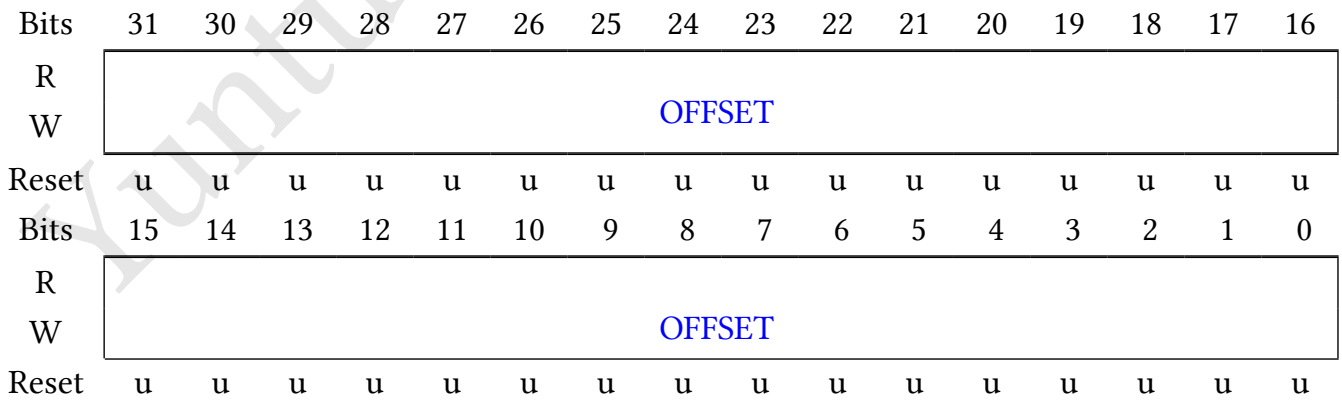


Table 8.27: DMA CTS_DTO_RLD_DTO Register Description

Field	Function
31 - 0	**Destination Trigger Loop Offset**

Yuntu Microelectronics

Field	Function
OFFSET	Destination address will add offset value(support signed-short type) after DMA channel trigger loop completed (trigger count reaches zero).

8.3.1.24 DMA CTS_DTO_RLD_RLD Register

Offset: c30h + n * 20h

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

Table 8.28: DMA CTS_DTO_RLD_RLD Register Description

Field	Function
31 - 0 ADDR	**RAM Reload Address** DMA will load new channel configuration structure(CTS) from system memory after trigger loop completed (trigger count reaches zero). **NOTE**. RAM reload address should be 32 bytes aligned. **NOTE**. CTSn_CR[RLEDN] should be 1 when RAM reload function is used.

8.3.1.25 DMA CTS_CSR Register

Offset: 81ch + n * 20h

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TDINT	THDINT			TLKCH				TLKEN	RLDEN	BWC		DREQ	ACTIVE	DONE	START
W																
Reset	u	u	0	0	u	u	u	u	u	u	u	u	u	u	u	u

Table 8.29: DMA CTS_CSR Register Description

Field	Function
15	**Trigger Loop Done Interrupt Enable**

Yuntu Microelectronics

Field	Function
TDINT	Enable an interrupt when channel all trigger loop done 0b - Channel trigger loop done interrupt not enabled 1b - Channel trigger loop done interrupt enabled
14 THDINT	**Trigger Half Done Interrupt Enable** Enable an interrupt when half trigger loop done 0b - Channel trigger loop half done interrupt not enabled 1b - Channel trigger loop half done interrupt enabled
11 - 8 TLKCH	**Trigger Loop Link Channel Index** Setting this bit will let current DMA channel trigger linked DMA channel after current DMA channel all trigger loop completed
7 TLKEN	**Trigger Loop Link Enable** Enable channel to channel linking on all trigger loop complete 0b - Channel trigger loop done link not enabled 1b - Channel trigger loop done link enabled
6 RLDEN	**RAM Reload Enable** Setting this bit will current DMA channel load new configurations from system memory map after trigger loop completed 0b - Channel trigger loop done ram reload not enabled 1b - Channel trigger loop done ram reload enabled
5 - 4 BWC	**Bandwidth Control** Bandwidth control let DMA insert wait cycles between each R/W operation, this allows user to balance bus loadings. **NOTE**. If DMA need to transfer large amount of data, it is recommended to set BWC to 2. otherwise, CPU may need to wait DMA transfer for bus access. 00b - DMA transfer data continuous 01b - Reserved 10b - DMA engine wait 4 cycles after each R/W 11b - DMA engine wait 8 cycles after each R/W
3 DREQ	**Disable Request** Setting this bit will let DMA clear channel hardware request enable bit after trigger loop completed (TCNT decrease to zero). 0b - Channel trigger loop done won't clear channel hardware request enable bit, DMA will continue serve hardware request until even if TCNT already decrease down to zero. 1b - Channel trigger loop done will clear channel hardware request enable bit
2 ACTIVE	**Channel Active** 0b - DMA channel is not active 1b - DMA channel is active
1	**Trigger Loop Done**

Yuntu Microelectronics

Field	Function
DONE	0b - DMA channel trigger loop not done 1b - DMA channel trigger loop done
0 START	**Channel Start** 0b - DMA channel is not started 1b - DMA channel is started and waiting for trigger or active

8.3.1.26 DMA CTS_TCNTRV Register

Offset: 81eh + n * 20h

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RV															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Table 8.30: DMA CTS_TCNTRV Register Description

Field	Function
15 - 0 RV	**Reload Value** DMA will reload channel's TCNT with TCNTRV after DMA TCNT decreased to zero. This register share same register map with TCNT register.

8.4 Functional Description

8.4.1 Transfer Loop

Transfer loop means data transferred after one DMA channel trigger. Transfer loop use BCNT as data transfer byte counter, BCNT is a counting down counter, when BCNT reaches zero, one transfer loop completed. So BCNT is total transfer data size in bytes of one DMA channel trigger.

8.4.2 Trigger Loop

Trigger loop means DMA channel could accept how many DMA channel triggers(include software and hardware trigger). Trigger count is define in TCNT field, TCNT is also a counting down counter, when TCNT reaches zero, one trigger loop completed.

8.4.3 Channel Arbitration

DMA module use round-robin arbitration when multi-channel trigger comes simultaneously, Round-robin arbitration is a fair algorithm, it lets every DMA channel have a chance to be served.

Yuntu Microelectronics

8.4.4 DMA Channel Initialize

1. Flush DMA CTS memory with channel default, for example clear CTS memory with 0.
2. Set DMA trigger source if using hardware trigger source.
3. Enable DMA channel error interrupt if needed.
4. Update DMA channel configuration structure with DMA channel configuration
5. Enable DMA hardware request

Yuntu Microelectronics

Chapter 9

Divide and Square Root (DIVSQRT)

9.1 Introduction

The Divide and Square Root (DIVSQRT) module supports the integer divide operation and an unsigned integer square root operation. The supported integer divide operation includes the 32/32 signed (SDIV) and unsigned (UDIV) calculations.

9.1.1 Feature

The DIVSQRT includes the following features:

- Lightweight implementation of 32-bit integer divide and square root arithmetic operations.
  - Supports 32/32 signed and unsigned divide (or remainder) calculation
  - Supports 32-bit unsigned square root calculation
- Simple programming model includes registers of input data, result, control, status.
- Programming model interface optimized for activation from inline code or software library call.
  - "Fast Start" configuration minimizes the memory-mapped register write overhead.
  - Supports two methods to determine when the result is valid, including software polling.
  - Configurable divide-by-zero response
- Pipelined design process 2bits per cycle with early termination exit for minimum execution time.

9.2 Register Definition

9.2.1 DIVSQRT Memory Map

DIVSQRT base address 4000e000h

Table 9.1: DIVSQRT Memory Map

Offset	Register	Width	Access	Reset Value
0h	Dividend Register (DEND)	32	RW	00000000h
4h	Divisor Register (DSOR)	32	RW	00000000h
8h	Control and Status Register (CSR)	32	RW	00000000h
ch	Result Register (RES)	32	RW	00000000h
10h	Radicand Register (RCND)	32	RW	00000000h

Yuntu Microelectronics

9.2.1.1 DIVSQRT DEND Register

This register should be loaded with input dividend operand before the divide operation is launched.
This register will be updated by hardware during the calculation.
Access to this register, while the calculation is not completed, will cause the access to be stalled until the calculation is done.

Offset: 0h

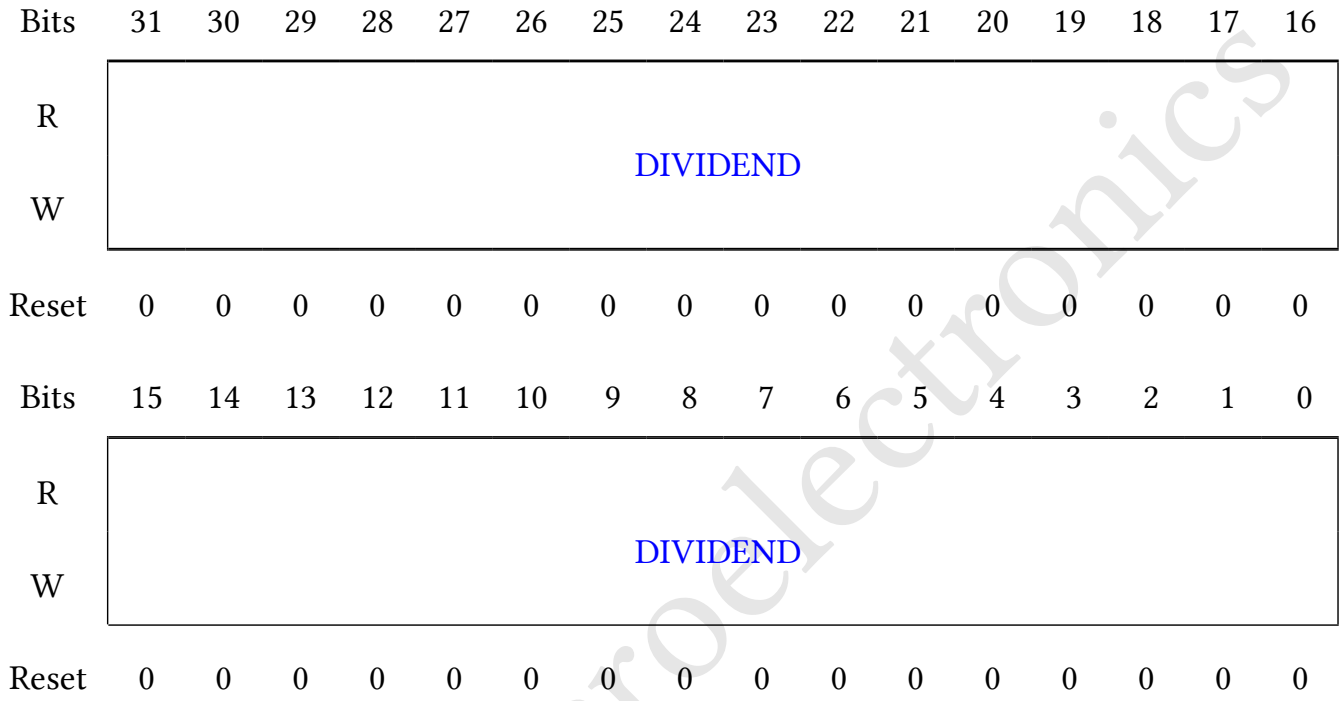


Table 9.2: DIVSQRT DEND Register Description

Field	Function
31 - 0 DIVIDEND	**Dividend** This is the input dividend operand for divide calculations.

9.2.1.2 DIVSQRT DSOR Register

This register should be loaded with input dividend operand before the divide operation is launched.
Writing to this register will launch a divide operation if CSR[DFS] is 0.
Access to this register, while the calculation is not completed, will cause the access to be stalled until the calculation is done.

Offset: 4h

Yuntu Microelectronics

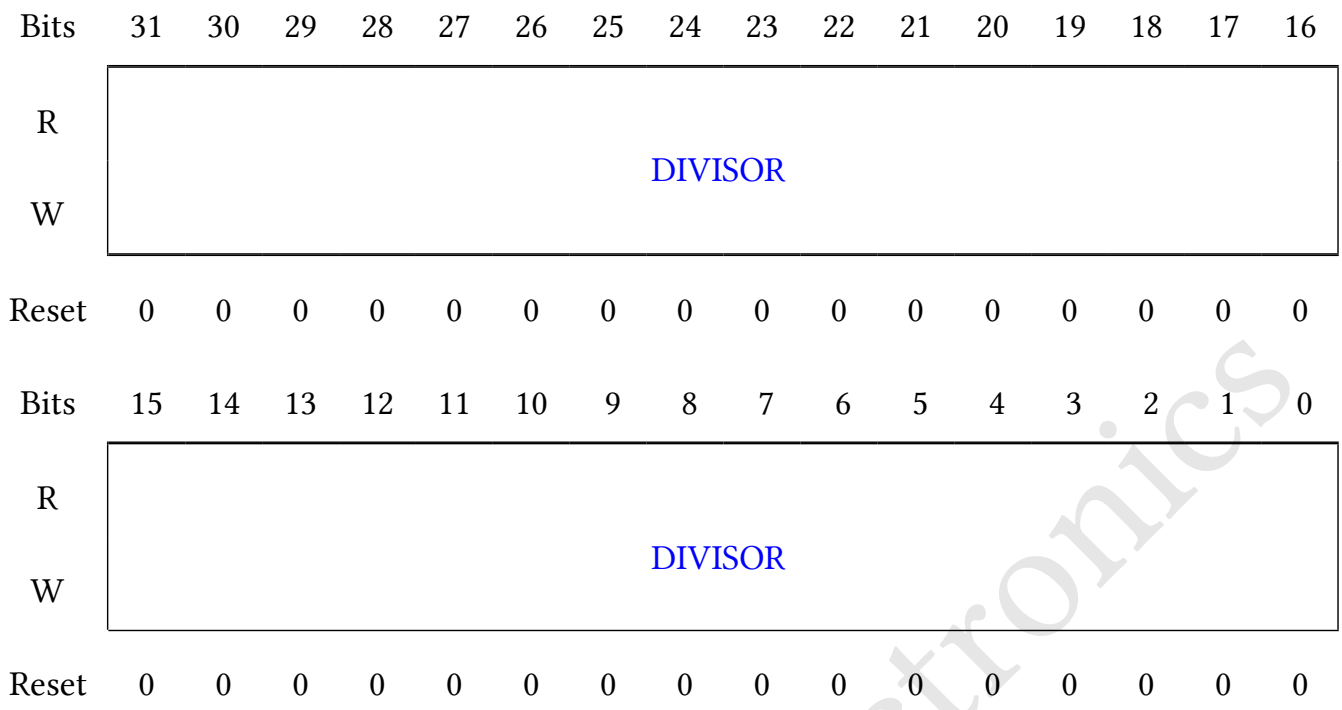


Table 9.3: DIVSQRT DSOR Register Description

Field	Function
31 - 0 DIVISOR	**Divisor** This is the input divisor operand for divide calculations.

Yuntu Microelectronics

9.2.1.3 DIVSQRT CSR Register

This register includes the configurations and status.
Access to this register, while the calculation is not completed, will cause the access to be stalled until the calculation is done.

Offset: 8h

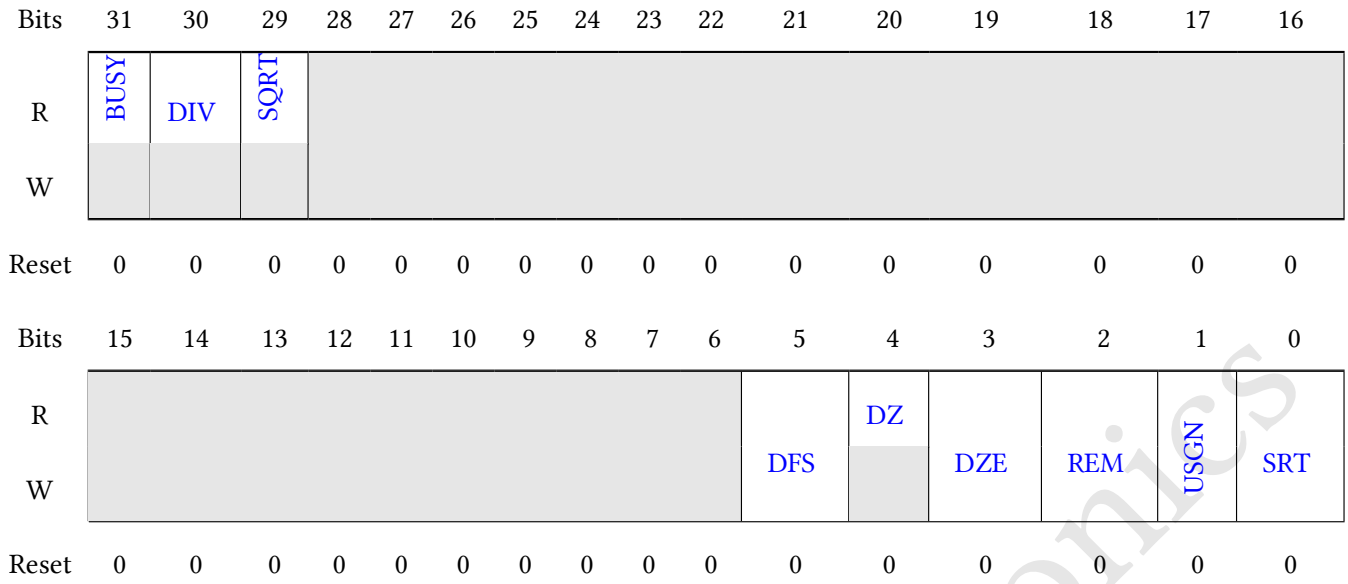


Table 9.4: DIVSQRT CSR Register Description

Field	Function
31 BUSY	**Busy Status** This read-only bit is asserted when DIVSQRT is running a divide or square root calculation. When a calculation is initiated, this bit will be asserted. When a calculation is completed, this bit will be cleared automatically. This bit could be used to poll the DIVSQRT's execution status. The combined [BUSY, DIV, SQRT] indicates the module status - if 3'b001, DIVSQRT is idle and the last calculation is square root - if 3'b010, DIVSQRT is idle and the last calculation is divide - if 3'b101, DIVSQRT is busy processing a square root calculation - if 3'b110, DIVSQRT is busy processing a divide calculation The remaining status of [BUSY, DIV, SQRT] is reserved. `0b` - DIVSQRT is idle `1b` - DIVSQRT is busy
30 DIV	**Divide Operation** This read-only bit indicates the current or last DIVSQRT operation was a divide operation. `0b` - Current or last DIVSQRT operation was not a divide operation `1b` - Current or last DIVSQRT operation was a divide operation
29 SQRT	**Square Root Operation** This read-only bit indicates the current or last DIVSQRT operation was a square root `0b` - Current or last DIVSQRT operation was not a square root operation `1b` - Current or last DIVSQRT operation was a square root operation
5	**Disable Fast Start**

Yuntu Microelectronics

Field	Function
DFS	There are 2 ways for initiating a divide operation. The default way is writing the DSOR register to initiate the divide operation. The other way is writing the CSR[SRT] bit to initiate the divide operation. 0b - A divide operation is initiated by writing DSOR register 1b - A divide operation is initiated by writing CSR[SRT] bit
4 DZ	**Divided by Zero** This read-only bit indicates the last divide operation had a zero divisor (DSOR == 0). Once this bit is asserted, the read of RES register returns zero or terminates this read with an error if CSR[DZE] is set. This bit will be cleared automatically once the new operation is initiated. 0b - The last divide operation had a non-zero divisor 1b - The last divide operation had a zero divisor
3 DZE	**Divided by Zero Enable** This bit controls the response to the divided by zero calculation. If both CSR[DZ] and CSR[DZE] are asserted, the next read of RES register will be terminated with an error. 0b - Read of RES register returns the register content 1b - If CSR[DZ] is set, the next read of RES register will be terminated with an error, else the RES register content are returned
2 REM	**Remainder** This bit indicates whether the quotient or the remainder is returned in RES register. The combined [REM, USGN] bits define the following divide operation if 2'b00, perform a signed divide operation and return the quotient if 2'b01, perform a unsigned divide operation and return the quotient if 2'b10, perform a signed divide operation and return the remainder if 2'b11, perform a unsigned divide operation and return the remainder 0b - Return the quotient of divide operation in RES register 1b - Return the remainder of divide operation in RES register
1 USGN	**Unsigned Calculation** This bit indicates whether a signed or unsigned divide operation is performed. Please see the description of CSR[REM] for the definition of divide operation. 0b - Perform a signed divide operation 1b - Perform an unsigned divide operation
0 SRT	**Start** If CSR[DFS] is set, writing 1 to this bit will initiate a divide operation, else is ignored. This bit always reads as a zero. 0b - No operation initialed 1b - IF CSR[DFS] is set, then initiate a divide operation, else will be ignored

Yuntu Microelectronics

9.2.1.4 DIVSQRT RES Register

This register is loaded with the result of the divide or square root calculation.
It will be updated once the calculation is completed.
When a square root operation is performed (on an unsigned 32-bit number), the result is limited to a 16-bit value with RES[31:16] is zero.
Any access to the RES register while the module is busy calculating will cause the access to be stalled until the calculation is done and a new result is written into this register.

Offset: ch

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RESULT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RESULT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9.5: DIVSQRT RES Register Description

Field	Function
31 - 0 RESULT	**Result** This is the output result for a divide or square root calculation.

9.2.1.5 DIVSQRT RCND Register

This write-only register is loaded with the square number.
Writing this register will initiate a square root calculation.
Any access to this register while the module is busy calculating will cause the access to be stalled until the calculation is done.
Any read of this read register will cause an error.

Offset: 10h

Yuntu Microelectronics

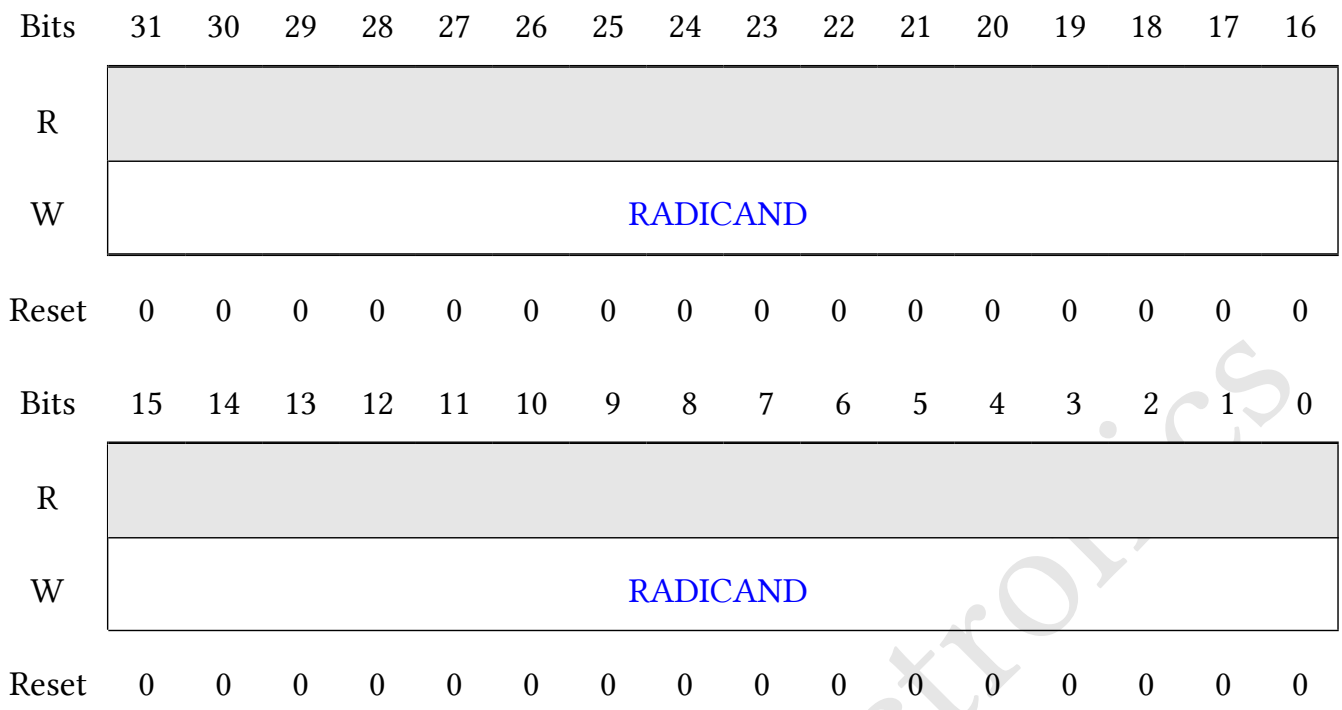


Table 9.6: DIVSQRT RCND Register Description

Field	Function
31 - 0 RADICAND	**Radicand** This is the input radicand for a square root calculation, that is, the input "square" number.

9.3 Functional Description

9.3.1 Modes of Operation

The DIVSQRT does not support any special operation modes. As a memory-mapped device with the AHB system bus port, DIVSQRT responds based strictly on memory addresses to its programming model.

9.3.2 Algorithms

9.3.2.1 Integer Divide

The DIVSQRT module implements a "shift, test, and restore" radix-2 algorithm for unsigned integer divide operations.

When performing a signed divide calculation, negative input operands are converted into 2's complement positive numbers first, an unsigned divide performed, and the sign of the results based on the input operand signs, namely:

- The sign of the remainder is the same as the sign of the dividend
- The quotient is negated if the signs of the dividend and divisor are different

Yuntu Microelectronics

The hardware implementation processes two bits per machine cycle and includes “early termination” logic where the execution time is data-dependent, based on the magnitude of the positive dividend.

9.3.2.2 Integer Square Root

The unsigned square root algorithm begins by creating a 32-bit “one-hot” bit vector signaling the highest power of four of the Radicand register’s contents (RCND).

It then iterates through an algorithm involving magnitude comparisons of the RCND register versus the working result plus bit vector summation, conditional decrementing of the radicand, a 1-bit right shift of the result, and a 2-bit right shift of the one-hot bit vector.

Processing two bits of the radicand per cycle, the result register finishes with the square root calculation’s integer portion. The module includes early termination logic so that the execution time is data-dependent, based on the input radicand’s magnitude.

Yuntu Microelectronics

Chapter 10

Trigger Multiplexer Module (TMU)

10.1 Introduction

The Trigger Multiplexer Unit (TMU) provides an extremely flexible mechanism for connecting various trigger sources to multiple pins/peripherals.

With the TMU, each peripheral that accepts external triggers usually has one specific 8-bit trigger control register. Each control register supports up to 128 triggers, and each trigger can be selected from the available input triggers.

10.2 Trigger Interconnectivity Table

Table 10.1: Trigger MUX Table

Slot	TRIG_IN	TRIG_OUT
0	VSS	
1	VDD	
2	TMU_IN0	
3	TMU_IN1	
4	TMU_IN2	TMU_OUT0
5	TMU_IN3	TMU_OUT1
6	TMU_IN4	TMU_OUT2
7	TMU_IN5	TMU_OUT3
8	TMU_IN6	TMU_OUT4
9	TMU_IN7	TMU_OUT5
10	TMU_IN8	
11	TMU_IN9	
12		ADC0_ADHWT_TRIG
13		
14	ACMP0_OUT	
15		
16		
17	PTMR_CH0	
18	PTMR_CH1	
19	PTMR_CH2	

Yuntu Microelectronics

Table 10.1 continued from previous page

Slot	TRIG_IN	TRIG_OUT
20	PTMR_CH3	
21	lpTMR0	
22		
23	eTMR0_EXT_TRIG	
24		
25	eTMR1_EXT_TRIG	
26		
27	eTMR2_EXT_TRIG	
28		ACMP0_SAMPLE_INPUT
29		
30	ADC0_EOC	
31	ADC0_AWD	
32		
33		
34		
35		
36		
37		
38		
39		
40		eTMR0_HWTRIG0
41		eTMR0_FAULT0
42		eTMR0_FAULT1
43	RTC_ALARM	eTMR0_FAULT2
44	RTC_SECOND	eTMR1_HWTRIG0
45		eTMR1_FAULT0
46		eTMR1_FAULT1
47		eTMR1_FAULT2
48		eTMR2_HWTRIG0
49	UART0_RX_DATA	eTMR2_FAULT0
50	UART0_TX_DATA	eTMR2_FAULT1
51	UART0_RX_IDLE	eTMR2_FAULT2
52	UART1_RX_DATA	
53	UART1_TX_DATA	
54	UART1_RX_IDLE	
55	I2C0_MASTER_TRIG	
56	I2C0_SLAVE_TRIG	
57		
58		
59	SPI0_FRAME	
60	SPI0_RX_DATA	
61	SPI1_FRAME	
62	SPI1_RX_DATA	
63	SW_TRIG	
64	SPI2_FRAME	

Yuntu Microelectronics

Table 10.1 continued from previous page

Slot	TRIG_IN	TRIG_OUT
65	SPI2_RX_DATA	
66		
67	I2C1_MASTER_TRIG	
68		
69		
70		
71		
72		
73		
74		
75		
76		
77	UART2_RX_DATA	
78	UART2_TX_DATA	
79	UART2_RX_IDLE	
80		
81		
82		
83		
84		I2C0_TRIG
85		
86		
87		
88		
89		
90		
91		
92		
93		
94		
95		
96		
97		
98		
99		
100		lpTMR0_ALT0
101		
102		
103		
104		
105		
106		
107		
108		I2C1_TRIG
109		

Yuntu Microelectronics

Table 10.1 continued from previous page

Slot	TRIG_IN	TRIG_OUT
110		
111		
112		
113		
114		
115		
116		
117		
118		
119		
120		
121		
122		
123		
124		
125		
126		
127		

10.3 Register Definition

This subsection provides a description of the memory map and registers.

10.3.1 TMU Memory Map

TMU base address 40063000h

Table 10.2: TMU Memory Map

Offset	Register	Width	Access	Reset Value
0h	MUX (MUX0)	32	RW	00000000h
4h	MUX (MUX1)	32	RW	00000000h
8h	MUX (MUX2)	32	RW	00000000h
ch	MUX (MUX3)	32	RW	00000000h
10h	MUX (MUX4)	32	RW	00000000h
14h	MUX (MUX5)	32	RW	00000000h
18h	MUX (MUX6)	32	RW	00000000h
1ch	MUX (MUX7)	32	RW	00000000h
20h	MUX (MUX8)	32	RW	00000000h
24h	MUX (MUX9)	32	RW	00000000h
28h	MUX (MUX10)	32	RW	00000000h
2ch	MUX (MUX11)	32	RW	00000000h
30h	MUX (MUX12)	32	RW	00000000h
34h	MUX (MUX13)	32	RW	00000000h
38h	MUX (MUX14)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
3ch	MUX (MUX15)	32	RW	00000000h
40h	MUX (MUX16)	32	RW	00000000h
44h	MUX (MUX17)	32	RW	00000000h
48h	MUX (MUX18)	32	RW	00000000h
4ch	MUX (MUX19)	32	RW	00000000h
50h	MUX (MUX20)	32	RW	00000000h
54h	MUX (MUX21)	32	RW	00000000h
58h	MUX (MUX22)	32	RW	00000000h
5ch	MUX (MUX23)	32	RW	00000000h
60h	MUX (MUX24)	32	RW	00000000h
64h	MUX (MUX25)	32	RW	00000000h
68h	MUX (MUX26)	32	RW	00000000h
6ch	MUX (MUX27)	32	RW	00000000h
70h	MUX (MUX28)	32	RW	00000000h
74h	MUX (MUX29)	32	RW	00000000h
78h	MUX (MUX30)	32	RW	00000000h
7ch	MUX (MUX31)	32	RW	00000000h

10.3.1.1 TMU MUX Register

Offset: 0h + n * 4h

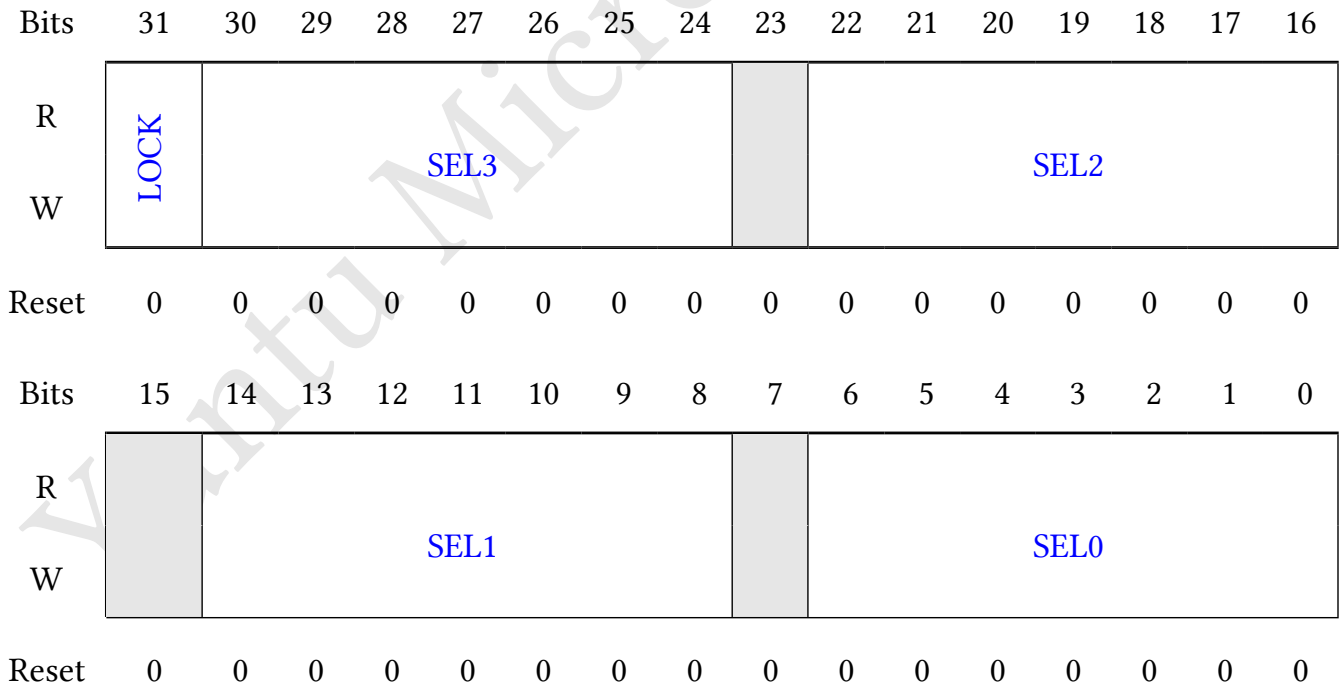


Table 10.3: TMU MUX Register Description

Field	Function
31	**Trigger MUX Locked**

Yuntu Microelectronics

Field	Function
LOCK	Lock the MUX select for peripheral trigger input
30 - 24 SEL3	**Trigger MUX Input Source Select 3** This read/write bit field is used to configure the MUX select for peripheral trigger input.
22 - 16 SEL2	**Trigger MUX Input Source Select 2** This read/write bit field is used to configure the MUX select for peripheral trigger input.
14 - 8 SEL1	**Trigger MUX Input Source Select 1** This read/write bit field is used to configure the MUX select for peripheral trigger input.
6 - 0 SEL0	**Trigger MUX Input Source Select 0** This read/write bit field is used to configure the MUX select for peripheral trigger input.

Yuntu Microelectronics

Chapter 11

Chip Integration Module (CIM)

11.1 Chip-specific CIM Information

The Chip Integration Module (CIM) provides system control and chip configuration registers. There is a 128-bit UID information display (address@ 0x10000fe0 ~ 0x10000fef) in YTM32B1LE0x device.

11.2 Introduction

11.2.1 Features

- System function configuration
- ADC/ACMP trigger synchronize selection
- Software trigger generate
- eTMR external clock and fault selection
- eTMR channel input/output selection
- eTMR hardware trigger 1 generation
- Die information display
- Chip information display

11.3 Register Definition

The CIM includes registers to control and configure the chip.

11.3.1 CIM Memory Map

CIM base address 40048000h

Table 11.1: CIM Memory Map

Offset	Register	Width	Access	Reset Value
0h	Control Register (CTRL)	32	RW	00000001h
4h	Software Trigger Register (SWTRIG)	32	RW	00000000h
8h	ETMR Option 0 Register (ETMROPT0)	32	RW	00000000h
ch	ETMR Option 1 Register (ETMROPT1)	32	RW	00000000h
10h	Die Information Register (DIEINFO)	32	RO	01042000h
14h	Chip Information Register (CHIPINFO)	32	RO	see description

Yuntu Microelectronics

11.3.1.1 CIM CTRL Register

Offset: 0h

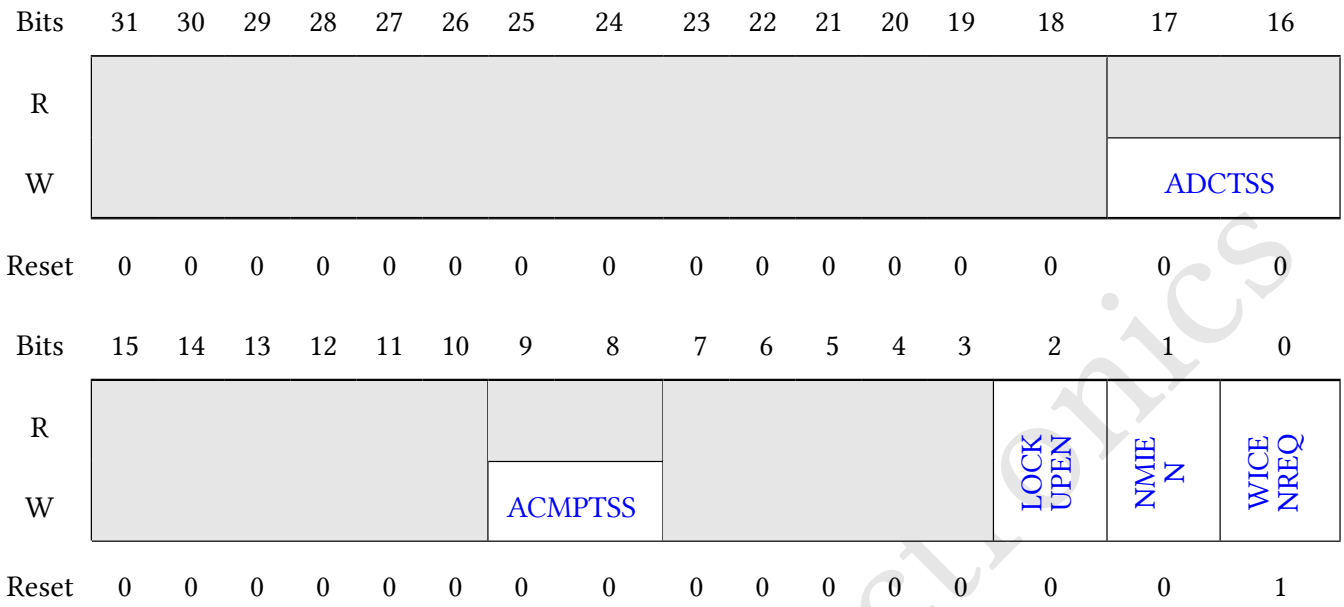


Table 11.2: CIM CTRL Register Description

Field	Function
17 - 16 ADCTSS	**ADC Trigger Sync Select** Select ADC input trigger synchronized mode. 00b - Input trigger is raw signal. 01b - Input trigger is synchronized signal, synchronized by bus clock. 10b - Input trigger is synchronized signal, synchronized by functional clock. 11b - Reserved.
9 - 8 ACMPTSS	**ACMP Trigger Sync Select** Select ACMP input trigger synchronized mode. 00b - Input trigger is raw signal. 01b - Input trigger is synchronized signal, synchronized by bus clock. 10b - Input trigger is synchronized signal, synchronized by functional clock. 11b - Reserved.
2 LOCKUPEN	**CPU Lockup Reset Enable** Enable CPU Lockup Reset. 0b - CPU Lockup reset is disabled. 1b - CPU Lockup reset is enabled.
1 NMIEN	**Non-mask Interrupt Enable** Enable NMI function. 0b - NMI is disabled. 1b - NMI is enabled.
0	**WIC Enable**

Yuntu Microelectronics

Field	Function
WICENREQ	Enable WIC to wakeup low power mode. 0b - Disable WIC to wakeup low power mode. 1b - Enable WIC to wakeup low power mode.

11.3.1.2 CIM SWTRIG Register

Offset: 4h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									SWTCNT							SWT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11.3: CIM SWTRIG Register Description

Field	Function
7 - 1 SWTCNT	**Software Trigger Count Register** Define software trigger valid length (multiplied with bus clock cycle), SWTCNT value will decrease until equal to zero when SWT set.
0 SWT	**Software Trigger** Generate software trigger. 0b - Software trigger is disabled. 1b - Generate software trigger and it keeps until SWTCNT equals to zero.

11.3.1.3 CIM ETMROPT0 Register

Offset: 8h

Yuntu Microelectronics

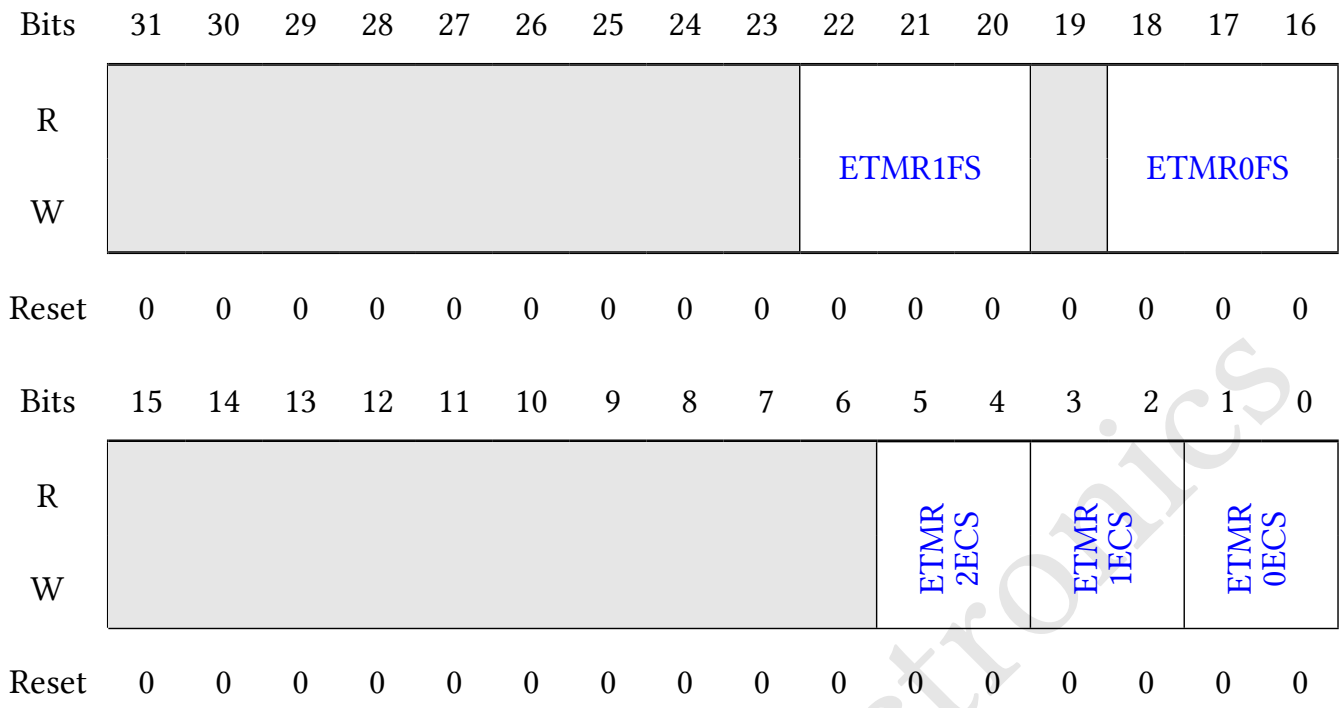


Table 11.4: CIM ETMROPT0 Register Description

Field	Function
22 - 20 ETMR1FS	**ETMR1 Fault Select** Select fault source for ETMR1. 0b000 - ETMR1 fault 0 input from pad. 0b001 - ETMR1 fault 0 input from TMU. 0b000 - ETMR1 fault 1 input from pad. 0b010 - ETMR1 fault 1 input from TMU. 0b000 - ETMR1 fault 2 input from pad. 0b100 - ETMR1 fault 2 input from TMU.
18 - 16 ETMR0FS	**ETMR0 Fault Select** Select fault source for ETMR0. 0b000 - ETMR0 fault 0 input from pad. 0b001 - ETMR0 fault 0 input from TMU. 0b000 - ETMR0 fault 1 input from pad. 0b010 - ETMR0 fault 1 input from TMU. 0b000 - ETMR0 fault 2 input from pad. 0b100 - ETMR0 fault 2 input from TMU.
5 - 4 ETMR2ECS	**ETMR2 External Clock Select** Select external clock source for ETMR2. 00b - Select TCLK_IN0 input as external clock for ETMR2. 01b - Select TCLK_IN1 input as external clock for ETMR2. 10b - Select TCLK_IN2 input as external clock for ETMR2. 11b - Reserved.
3 - 2	**ETMR1 External Clock Select**

Yuntu Microelectronics

Field	Function
ETMR1ECS	Select external clock source for ETMR1. 00b - Select TCLK_IN0 input as external clock for ETMR1. 01b - Select TCLK_IN1 input as external clock for ETMR1. 10b - Select TCLK_IN2 input as external clock for ETMR1. 11b - Reserved.
1 - 0 ETMR0ECS	**ETMR0 External Clock Select** Select external clock source for ETMR0. 00b - Select TCLK_IN0 input as external clock for ETMR0. 01b - Select TCLK_IN1 input as external clock for ETMR0. 10b - Select TCLK_IN2 input as external clock for ETMR0. 11b - Reserved.

11.3.1.4 CIM ETMROPT1 Register

Offset: ch

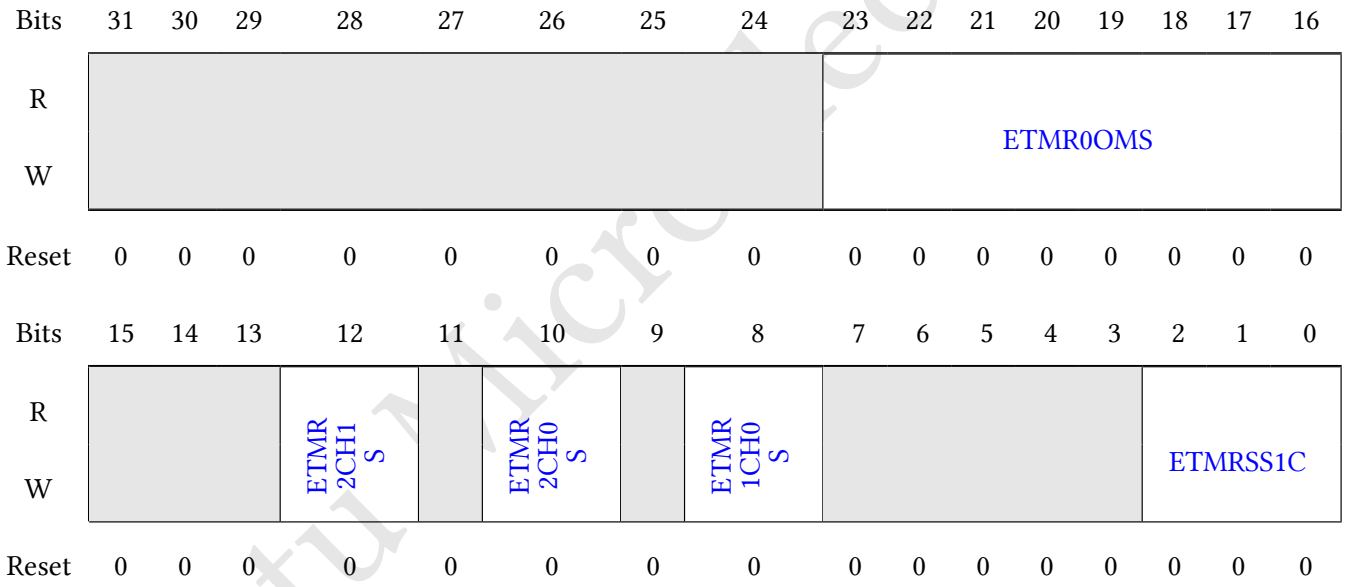


Table 11.5: CIM ETMROPT1 Register Description

Field	Function
23 - 16	**ETMR0 Output Modulation Select**

Yuntu Microelectronics

Field	Function
ETMR0OMS	Select modulation with ETMR1_CH1 for ETMR0 channels. 0b00000000 - ETMR0 channel 0 output to pad directly. 0b00000001 - ETMR0 channel 0 output modulation with ETMR1_CH1 output. 0b00000000 - ETMR0 channel 1 output to pad directly. 0b00000010 - ETMR0 channel 1 output modulation with ETMR1_CH1 output. 0b00000000 - ETMR0 channel 2 output to pad directly. 0b00000100 - ETMR0 channel 2 output modulation with ETMR1_CH1 output. 0b00000000 - ETMR0 channel 3 output to pad directly. 0b00001000 - ETMR0 channel 3 output modulation with ETMR1_CH1 output. 0b00000000 - ETMR0 channel 4 output to pad directly. 0b00010000 - ETMR0 channel 4 output modulation with ETMR1_CH1 output. 0b00000000 - ETMR0 channel 5 output to pad directly. 0b00100000 - ETMR0 channel 5 output modulation with ETMR1_CH1 output. 0b00000000 - ETMR0 channel 6 output to pad directly. 0b01000000 - ETMR0 channel 6 output modulation with ETMR1_CH1 output. 0b00000000 - ETMR0 channel 7 output to pad directly. 0b10000000 - ETMR0 channel 7 output modulation with ETMR1_CH1 output.
12 ETMR2CH1S	**ETMR2 Channel 1 Select** Select input source for ETMR2 channel 1. 0b - ETMR2 channel 1 input from pad. 1b - ETMR2 channel 1 input from XOR result(ETMR2_CH0, ETMR2_CH1, ETMR1_CH1).
10 ETMR2CH0S	**ETMR2 Channel 0 Select** Select input source for ETMR2 channel 0. 0b - ETMR2 channel 0 input from pad. 1b - ETMR2 channel 0 input from ACMP out.
8 ETMR1CH0S	**ETMR1 Channel 0 Select** Select input source for ETMR1 channel 0. 0b - ETMR1 channel 0 input from pad. 1b - ETMR1 channel 0 input from ACMP out.
2 - 0 ETMRSS1C	**ETMR Synchronize Source 1 Control** Generate synchronize signal for ETMR trigger 1. 0b000 - Don’t generate synchronize signal for ETMR0 trigger 1. 0b001 - Generate synchronize signal for ETMR0 trigger 1. 0b000 - Don’t generate synchronize signal for ETMR1 trigger 1. 0b010 - Generate synchronize signal for ETMR1 trigger 1. 0b000 - Don’t generate synchronize signal for ETMR2 trigger 1. 0b100 - Generate synchronize signal for ETMR2 trigger 1.

11.3.1.5 CIM DIEINFO Register

Offset: 10h

Yuntu Microelectronics

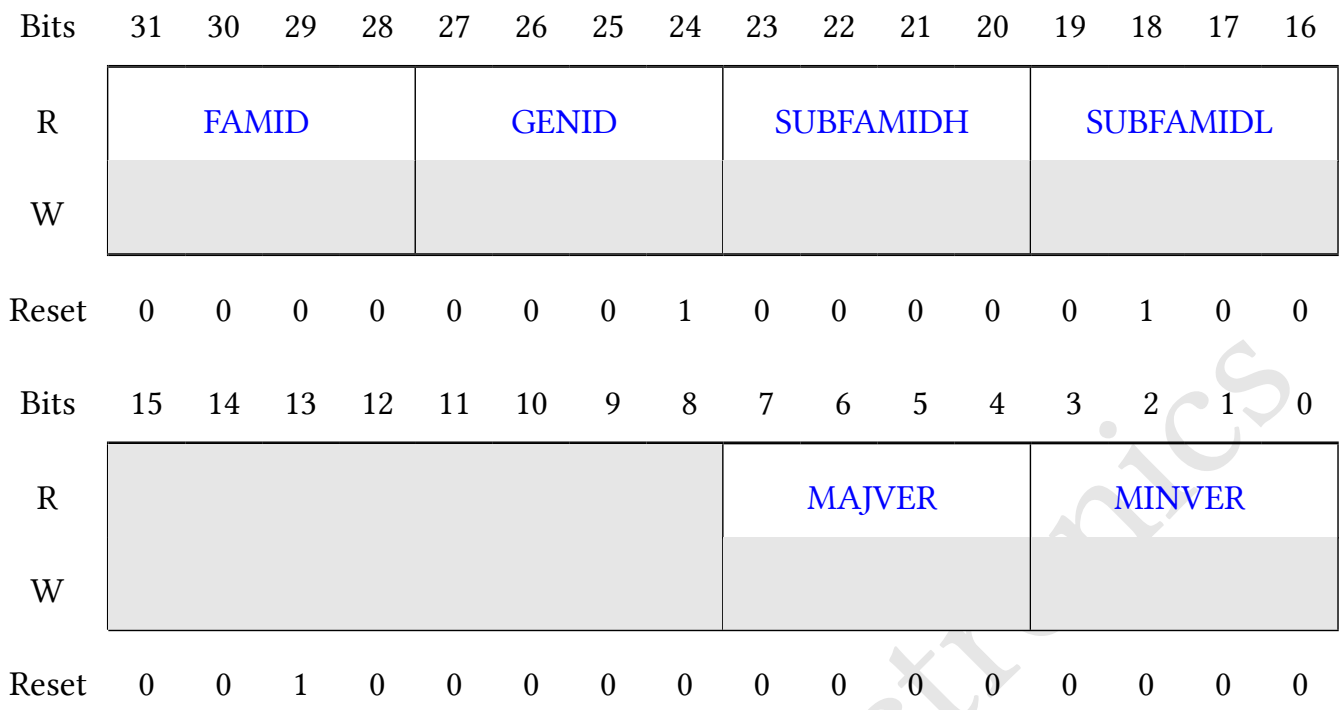


Table 11.6: CIM DIEINFO Register Description

Field	Function
31 - 28 FAMID	**Family ID** Describe the product family information. 0000b - B - Basic(General) 0001b - D - Dashboard 0010b - P - Powertrain 0011b - V - Vision 0100b - N - Networks
27 - 24 GENID	**Generation ID** Describe the product generation information. 0001b - 1st generation production 0010b - 2nd generation production 0011b - 3rd generation production
23 - 20 SUBFAMID H	**Sub Family ID High** Describe the product sub family information. 0000b - L - Low end 0001b - M - Middle end 0010b - H - High end
19 - 16	**Sub Family ID Low**

Yuntu Microelectronics

Field	Function
SUBFAMID L	Describe the product max Flash size information. 0000b - A - 64KB 0001b - B - 128KB 0010b - C - 256KB 0011b - D - 512KB 0100b - E - 1MB
7 - 4 MAJVER	**Major Version Information** Describe the major version information.
3 - 0 MINVER	**Minor Version Information** Describe the minor version information.

11.3.1.6 CIM CHIPINFO Register

Offset: 14h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R								FLEXCAN_FD_EN	PKGS				SRAMS			
W																
Reset	0	0	0	0	0	0	0	u	u	u	u	u	0	1	1	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DFLASHS				PFLASHS											
W																
Reset	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0

Table 11.7: CIM CHIPINFO Register Description

Field	Function
24 FLEXCAN_ FD_EN	**FlexCAN FD Feature Enable** FlexCAN FD Feature Enable/Disable 0b - FlexCAN FD Feature is disabled 1b - FlexCAN FD Feature is enable
23 - 20	**Package Size**

Yuntu Microelectronics

Field	Function
PKGS	Describe chip package type information. 0000b - 16 TSOP 0001b - 20 TSOP/32 LQFP 0010b - 32 QFN 0011b - 48 LQFP 0100b - 64 LQFP 1000b - 100 LQFP/MAPBGA 1010b - 144 LQFP 1100b - 176 LQFP 1101b - 257 BGA 1111b - 289 BGA
19 - 16 SRAMS	**SRAM Size** Describe sram size information. 0010b - 1KB 0011b - 2KB 0100b - 4KB 0101b - 8KB 0110b - 16KB 0111b - 32KB 1000b - 64KB 1001b - 128KB 1010b - 256KB 1011b - 512KB 1100b - 1MB 1101b - 2MB 1110b - 4MB 1111b - 8MB
15 - 12 DFLASHS	**DFLASH Size** Describe data flash size information. 0000b - 1KB 0001b - 2KB 0010b - 4KB 0011b - 8KB 0100b - 16KB 0101b - 32KB 0110b - 64KB 0111b - 128KB 1000b - 256KB 1001b - 512KB
11 - 8	**PFLASH Size**

Yuntu Microelectronics

Field	Function
PFLASHS	Describe program flash size information. 0000b - 1KB 0001b - 2KB 0010b - 4KB 0011b - 8KB 0100b - 16KB 0101b - 32KB 0110b - 64KB 0111b - 128KB 1000b - 256KB 1001b - 512KB 1010b - 1MB 1011b - 2MB 1100b - 4MB 1101b - 8MB 1110b - 16MB 1111b - 32MB

Yuntu Microelectronics

Part V

Clock, Reset and Power Management

Yuntu Microelectronics

Chapter 12

System Clock Unit (SCU)

12.1 Introduction

The system clock unit(SCU) provides multiple clock source choices for the MCU. The module contains fast internal RC oscillator(FIRC), slow internal RC oscillator(SIRC), fast crystal oscillator(FXOSC), slow crystal oscillator(SXOSC) and low power oscillator(LPO).

- The FIRC, SIRC and FXOSC can be selected as system clock source.
- The SIRC is always on unless it is forced to be disabled in deepsleep and standby mode.
- The SXOSC is not used as system clock source, it provides accurate clock to watchdog(WDG) and real time clock(RTC).
- The system clock unit(SCU) contains clock monitor block.
- Multiple clock sources can be selected to output to chip IO.

12.1.1 Features

- Fast internal RC oscillator(FIRC)
  - Up to 48 MHz
  - Default system boot clock source
  - Support trim for temperature and process
- Slow internal RC oscillator(SIRC)
  - Up to 2 MHz
  - Can be selected as system clock source
  - Always on unless it is forced to be disabled in deepsleep and standby mode
  - Support trim for temperature and process
- Fast crystal oscillator(FXOSC)
  - Support 4~40 MHz crystal
  - Can be selected as system clock source
  - Support bypass mode
- Slow crystal oscillator(SXOSC)
  - 32.768 KHz real time oscillator
  - Can't be selected as system clock
  - Provides accurate clock to watchdog(WDG) and real time clock(RTC)
- Low power oscillator(LPOCLK)
  - 750 Hz always on clock source
- Clock monitor unit(CMU)

Yuntu Microelectronics

  - SCU contains 2 CMU blocks
  - CMU monitors FIRC and FXOSC clock
  - SIRC clock is used as reference clock of CMU
  - CMU can detect frequency out of range, loss of checked clock and loss of reference clock
- SCU provides glitch free switcher to select system clock source
- SCU provides system clock dividers to generate core clock, fast bus clock and slow bus clock.

12.1.2 Block Diagram

The following figure shows the SCU block diagram.

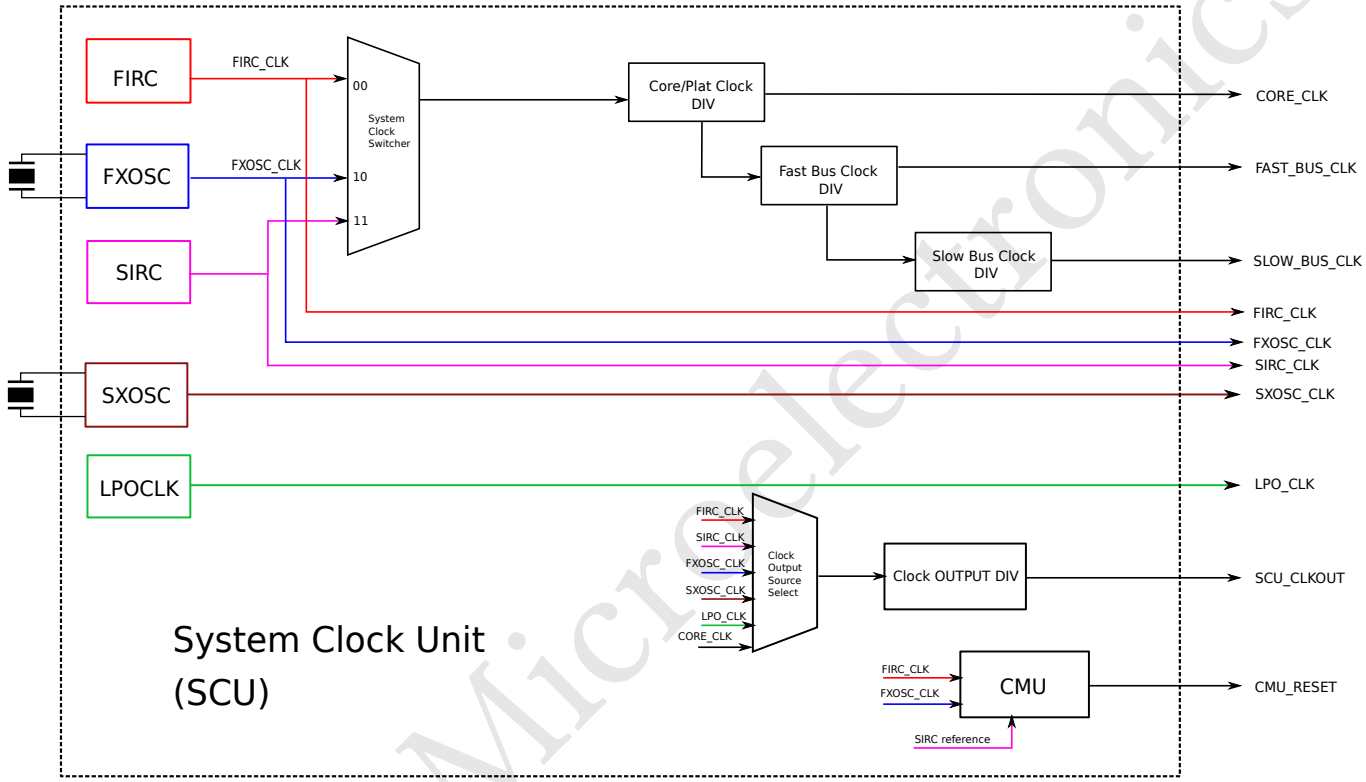


Figure 12.1: SCU Block Diagram

12.2 Register Definition

12.2.1 SCU Memory Map

SCU base address 40064000h

Table 12.1: SCU Memory Map

Offset	Register	Width	Access	Reset Value
0h	Clock Selection Register (CLKS)	32	RW	00000000h
4h	System Clock Divider (DIV)	32	RW	00000001h
8h	Status Register (STS)	32	RO	00000004h
ch	Clock Divider Status Register (DIVSTS)	32	RO	00000001h
10h	FIRC Control Register (FIRC_CTRL)	32	RW	00000000h
14h	SIRC Control Register (SIRC_CTRL)	32	RW	00000001h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
18h	FXOSC Control Register (FXOSC_CTRL)	32	RW	00002700h
1ch	SXOSC Control Register (SXOSC_CTRL)	32	RW	00000000h
2ch	Clock Output Control Register (CLKO)	32	RW	00000000h
30h	FIRC Trim Controller (FIRCTRIM)	32	RW	see description
34h	SIRC Trim Register (SIRCTRIM)	32	RW	see description
38h	LPOCLK Trim Register (LPOTRIM)	32	RW	see description
40h	Clock Monitor Unit Control Register (CMU_CTRL)	32	RW	00000000h
44h	Clock Monitor Unit Status Register (CMUSTS)	32	RW	00000000h
90h	Clock Monitor Unit High Comparison Value Register (CMUCMP0_HIGH)	32	RW	0000FFFFh
94h	Clock Monitor Unit Low Comparison Value Register (CMUCMP0_LOW)	32	RW	00000000h
98h	Clock Monitor Unit High Comparison Value Register (CMUCMP1_HIGH)	32	RW	0000FFFFh
9ch	Clock Monitor Unit Low Comparison Value Register (CMUCMP1_LOW)	32	RW	00000000h

12.2.1.1 SCU CLKS Register

The register contains clock select bits.

Offset: 0h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W															CLKSRCSEL	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12.2: SCU CLKS Register Description

Field	Function
1 - 0	**System Clock Source Selection**

Field	Function
CLKSRCSEL	System Clock Source selection bits. 00b - Select FIRC as system clock source 01b - Reserved 10b - Select FXOSC as system clock source 11b - Select SIRC as system clock source

12.2.1.2 SCU DIV Register

System Clock Divider

Offset: 4h

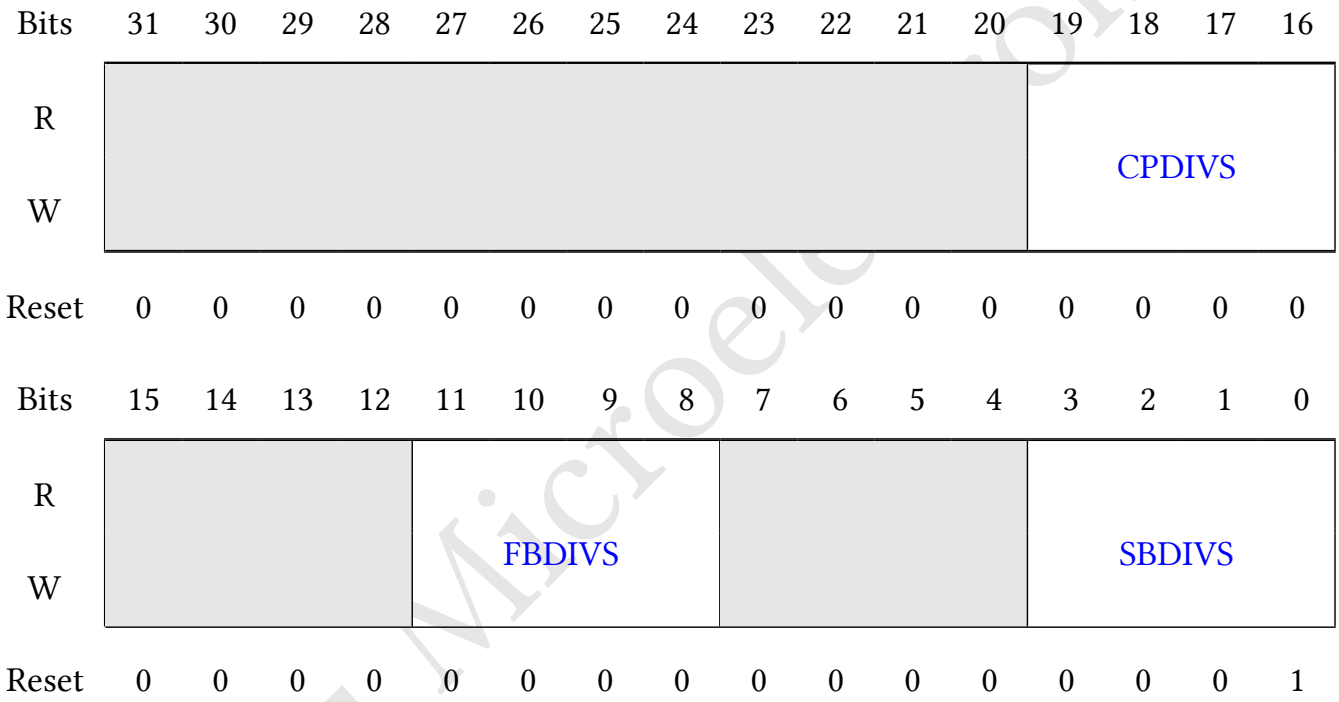


Table 12.3: SCU DIV Register Description

Field	Function
19 - 16	**Core Platform Clock Divider Value**

Yuntu Microelectronics

Field	Function
CPDIVS	Core Platform Clock Divider Value 0000b - Divide by 1 0001b - Divide by 2 0010b - Divide by 3 0011b - Divide by 4 0100b - Divide by 5 0101b - Divide by 6 0110b - Divide by 7 0111b - Divide by 8 1000b - Divide by 9 1001b - Divide by 10 1010b - Divide by 11 1011b - Divide by 12 1100b - Divide by 13 1101b - Divide by 14 1110b - Divide by 15 1111b - Divide by 16
11 - 8 FBDIVS	**Fast Bus Clock Divider Value** Fast Bus Clock Divider Value 0000b - Divide by 1 0001b - Divide by 2 0010b - Divide by 3 0011b - Divide by 4 0100b - Divide by 5 0101b - Divide by 6 0110b - Divide by 7 0111b - Divide by 8 1000b - Divide by 9 1001b - Divide by 10 1010b - Divide by 11 1011b - Divide by 12 1100b - Divide by 13 1101b - Divide by 14 1110b - Divide by 15 1111b - Divide by 16
3 - 0	**Slow Bus Clock Divider Value**

Yuntu Microelectronics

Field	Function
SBDIVS	Slow Bus Clock Divider Value 0000b - Divide by 1 0001b - Divide by 2 0010b - Divide by 3 0011b - Divide by 4 0100b - Divide by 5 0101b - Divide by 6 0110b - Divide by 7 0111b - Divide by 8 1000b - Divide by 9 1001b - Divide by 10 1010b - Divide by 11 1011b - Divide by 12 1100b - Divide by 13 1101b - Divide by 14 1110b - Divide by 15 1111b - Divide by 16

12.2.1.3 SCU STS Register

System Clock Unit Status register

Offset: 8h

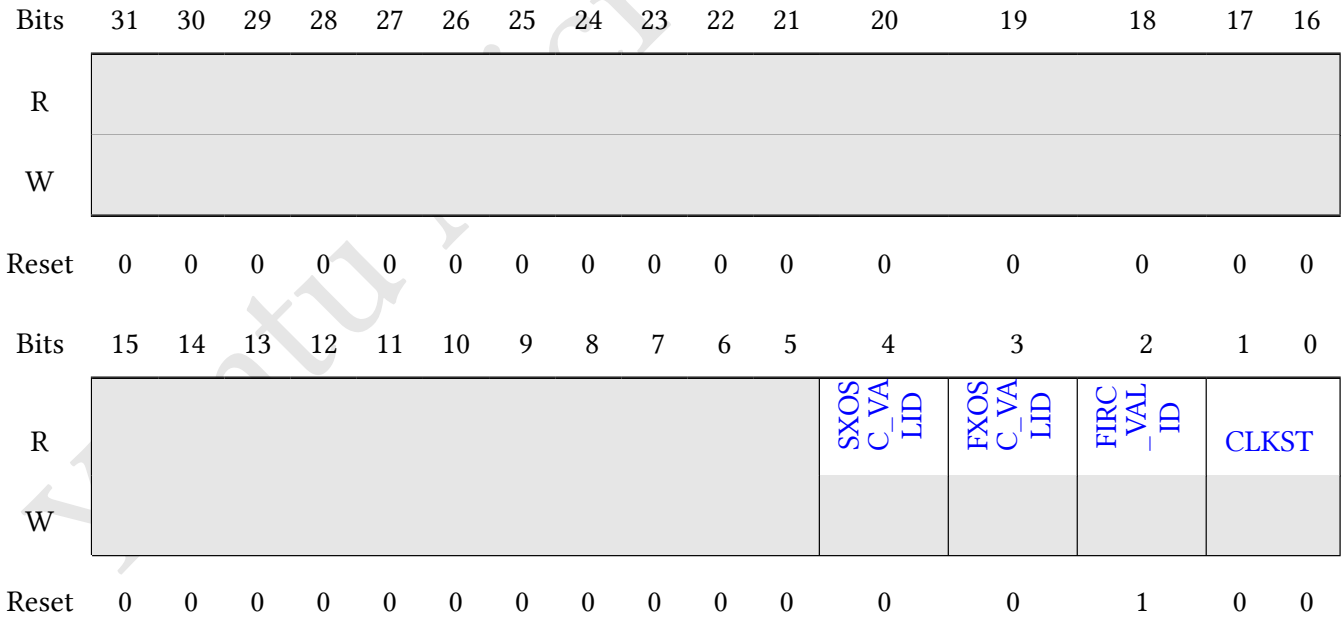


Table 12.4: SCU STS Register Description

Field	Function
4	**SXOSC Clock Valid.**

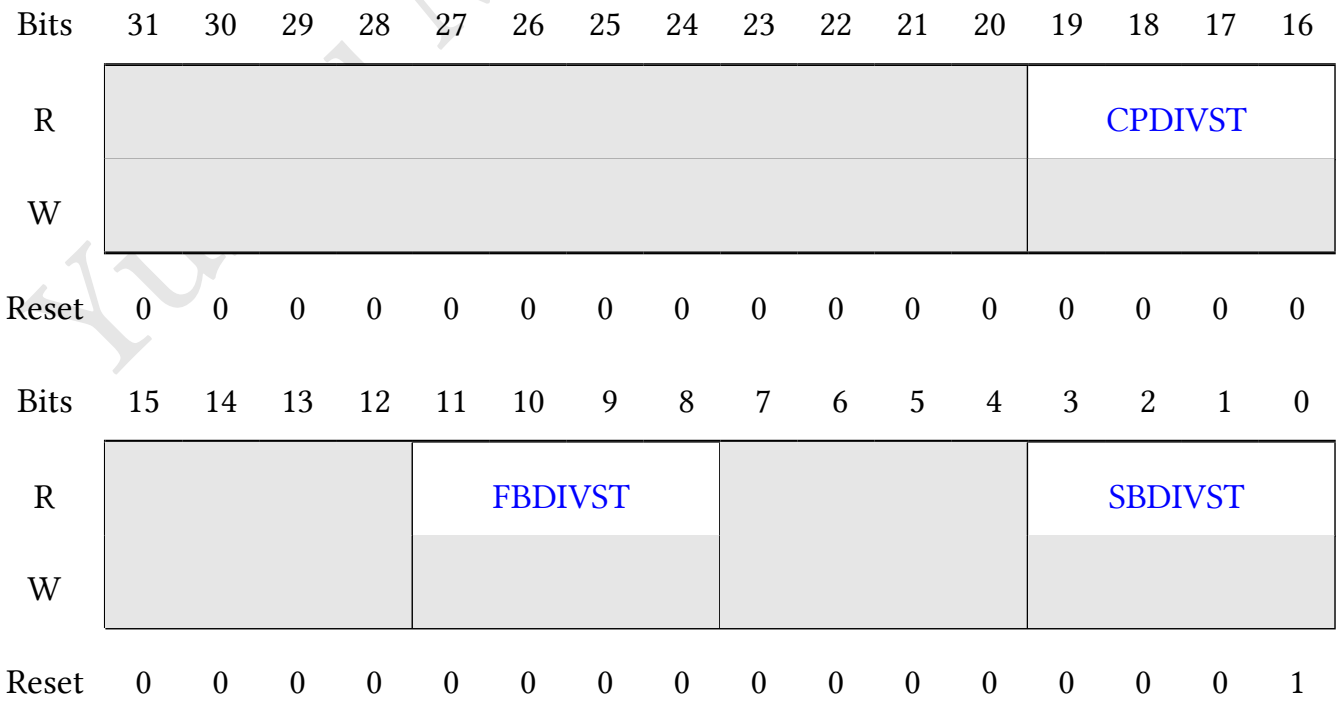
Yuntu Microelectronics

Field	Function
SXOSC_VALID	SXOSC Clock Valid. 0b - SXOSC Clock invalid 1b - SXOSC Clock Valid
3 FXOSC_VALID	**FXOSC Clock Valid.** FXOSC Clock Valid. 0b - FXOSC Clock invalid 1b - FXOSC Clock valid
2 FIRC_VALID	**FIRC Clock Valid.** FIRC Clock Valid. 0b - FIRC Clock invalid 1b - FIRC Clock valid
1 - 0 CLKST	**System Clock Source Selection Status Bits.** System Clock Source Selection status bits. 00b - FIRC is selected as system clock source 01b - Reserved 10b - FXOSC is selected as system clock source 11b - SIRC is selected as system clock source

12.2.1.4 SCU DIVSTS Register

System Clock Unit clock divider status

Offset: ch



Yuntu Microelectronics

Table 12.5: SCU DIVSTS Register Description

Field	Function
19 - 16 CPDIVST	**Core Platform Clock Divider Value** Core Platform Clock Divider Value 0000b - Divide by 1 0001b - Divide by 2 0010b - Divide by 3 0011b - Divide by 4 0100b - Divide by 5 0101b - Divide by 6 0110b - Divide by 7 0111b - Divide by 8 1000b - Divide by 9 1001b - Divide by 10 1010b - Divide by 11 1011b - Divide by 12 1100b - Divide by 13 1101b - Divide by 14 1110b - Divide by 15 1111b - Divide by 16
11 - 8 FBDIVST	**Fast Bus Clock Divider Value** Fast Bus Clock Divider Value 0000b - Divide by 1 0001b - Divide by 2 0010b - Divide by 3 0011b - Divide by 4 0100b - Divide by 5 0101b - Divide by 6 0110b - Divide by 7 0111b - Divide by 8 1000b - Divide by 9 1001b - Divide by 10 1010b - Divide by 11 1011b - Divide by 12 1100b - Divide by 13 1101b - Divide by 14 1110b - Divide by 15 1111b - Divide by 16
3 - 0	**Slow Bus Clock Divider Value**

Yuntu Microelectronics

Field	Function
SBDIVST	Slow Bus Clock Divider Value 0000b - Divide by 1 0001b - Divide by 2 0010b - Divide by 3 0011b - Divide by 4 0100b - Divide by 5 0101b - Divide by 6 0110b - Divide by 7 0111b - Divide by 8 1000b - Divide by 9 1001b - Divide by 10 1010b - Divide by 11 1011b - Divide by 12 1100b - Divide by 13 1101b - Divide by 14 1110b - Divide by 15 1111b - Divide by 16

12.2.1.5 SCU FIRC_CTRL Register

FIRC control register

Offset: 10h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R													FIRC_DLY_EN		FIRC_DS_EN	FIRC_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12.6: SCU FIRC_CTRL Register Description

Field	Function
3	**FIRC Delay Enable**

Yuntu Microelectronics

Field	Function
FIRC_DLY _EN	Delay FIRC clock out. 0b - Disable FIRC clock out delay 1b - Enable FIRC clock out delay
1 FIRC_DS_ EN	**FIRC Deepsleep Enable** Enable FIRC when chip enters Deepsleep mode. 0b - Disable FIRC when chip enters Deepsleep mode 1b - Enable FIRC when chip enters Deepsleep mode
0 FIRC_EN	**FIRC Enable** Enable FIRC regardless it is selected as system clock source. 0b - Disable FIRC if it is not selected as system clock source 1b - Enable FIRC regardless it is selected as system clock or not

12.2.1.6 SCU SIRC_CTRL Register

SIRC control register

Offset: 14h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R														SIRC _STB _EN	SIRC _DS_ EN	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 12.7: SCU SIRC_CTRL Register Description

Field	Function
2 SIRC_STB _EN	**SIRC Standby Enable** Enable SIRC when chip enters Standby mode. 0b - Disable SIRC when chip enters Standby mode 1b - Enable SIRC when chip enters Standby mode

Yuntu Microelectronics

Field	Function
1 SIRC_DS_ EN	**SIRC Deepsleep Enable** Enable SIRC when chip enters Deepsleep mode. 0b - Disable SIRC when chip enters Deepsleep mode 1b - Enable SIRC when chip enters Deepsleep mode

12.2.1.7 SCU FXOSC_CTRL Register

FXOSC control register

Offset: 18h

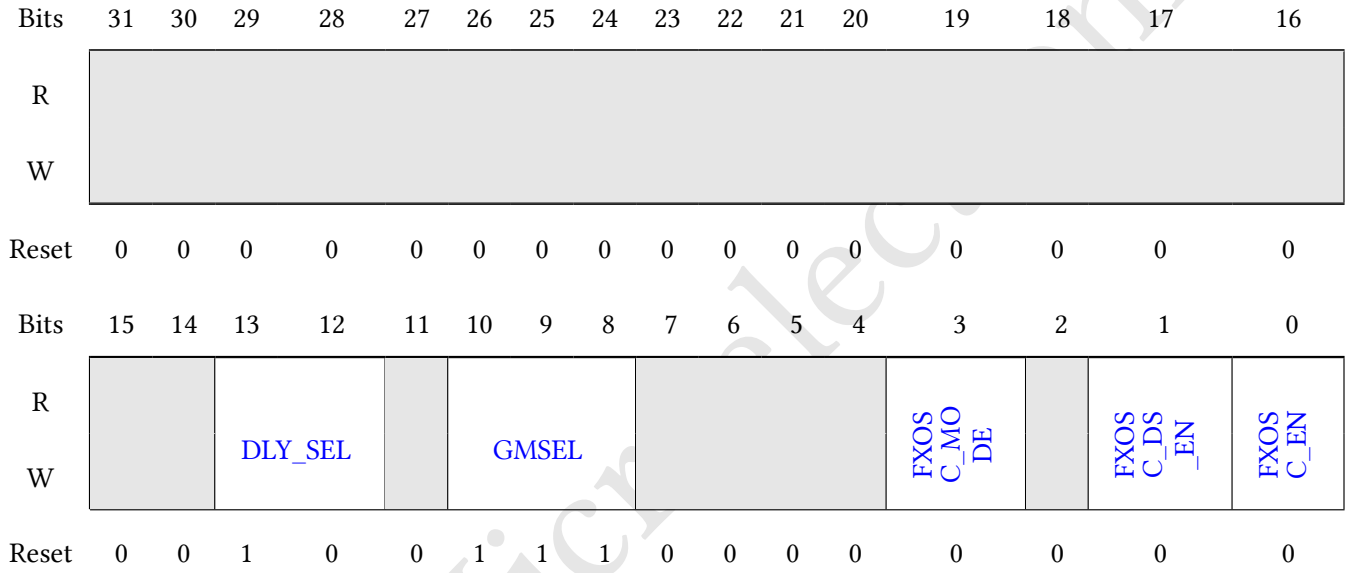


Table 12.8: SCU FXOSC_CTRL Register Description

Field	Function
13 - 12 DLY_SEL	**FXOSC clock Output Delay Select.** FXOSC clock output delay select. 00b - 4096 cycles 01b - 8192 cycles 10b - 16384 cycles (default) 11b - 32768 cycles
10 - 8	**FXOSC Gain Select**

Yuntu Microelectronics

Field	Function
GMSEL	FXOSC Gain Select for different oscillators 000b - Reserved 001b - used for 4MHz-6MHz oscillator 010b - used for 6MHz-8MHz oscillator 011b - used for 8MHz-12MHz oscillator 100b - Reserved 101b - used for 12MHz-20MHz oscillator 110b - used for 20MHz-32MHz oscillator 111b - used for 32MHz-40MHz oscillator NOTE: The above data represents recommended values for typical applications of FXOSC. For more specific details, please refer to the Datasheet, calculate the required Gain Setting, and perform crystal oscillator impedance matching tests.
3 FXOSC_MODE	**FXOSC Mode Selection** Select FXOSC oscillator mode or bypass mode 0b - FXOSC bypass mode 1b - FXOSC oscillator mode
1 FXOSC_DS_EN	**FXOSC Deepsleep Enable** Enable FXOSC when chip enters Deepsleep mode. 0b - Disable FXOSC when chip enters Deepsleep mode 1b - Enable FXOSC when chip enters Deepsleep mode
0 FXOSC_EN	**FXOSC Enable** Enable Fast crystal osc regardless it is selected as system clock source. 0b - Disable FXOSC if it is not selected as system clock source 1b - Enable FXOSC regardless it is selected as system clock or not

12.2.1.8 SCU SXOSC_CTRL Register

SXOSC control register

Offset: 1ch

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W			SXOSC_DLY_SEL					SXOSC_HIGAIN_EN					SXOSC_MODE	SXOSC_STB_EN	SXOSC_DS_EN	SXOSC_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Yuntu Microelectronics

Table 12.9: SCU SXOSC_CTRL Register Description

Field	Function
13 - 12 SXOSC_DLY_SEL	**SXOSC Clock Output Delay Select.** SXOSC clock output delay select. 00b - 1024 cycles (default) 01b - 2048 cycles 10b - 4096 cycles 11b - 8192 cycles
8 SXOSC_HIGAIN_EN	**SXOSC High Gain Enable/Disable** Enable/Disable SXOSC High Gain 0b - Disable SXOSC High Gain 1b - Enable SXOSC High Gain
3 SXOSC_MODE	**SXOSC Mode Selection** Select SXOSC oscillator mode or bypass mode 0b - SXOSC bypass mode 1b - SXOSC oscillator mode
2 SXOSC_STB_EN	**SXOSC Standby Enable** Enable SXOSC when chip enters Standby mode. 0b - Disable SXOSC when chip enters Standby mode 1b - Enable SXOSC when chip enters Standby mode
1 SXOSC_DS_EN	**SXOSC Deepsleep Enable** Enable SXOSC when chip enters Deepsleep mode. 0b - Disable SXOSC when chip enters Deepsleep mode 1b - Enable SXOSC when chip enters Deepsleep mode
0 SXOSC_EN	**SXOSC Enable** Enable/Disable SXOSC 0b - Disable SXOSC 1b - Enable SXOSC

Yuntu Microelectronics

12.2.1.9 SCU CLKO Register

Clock Output Control Register

Offset: 2ch

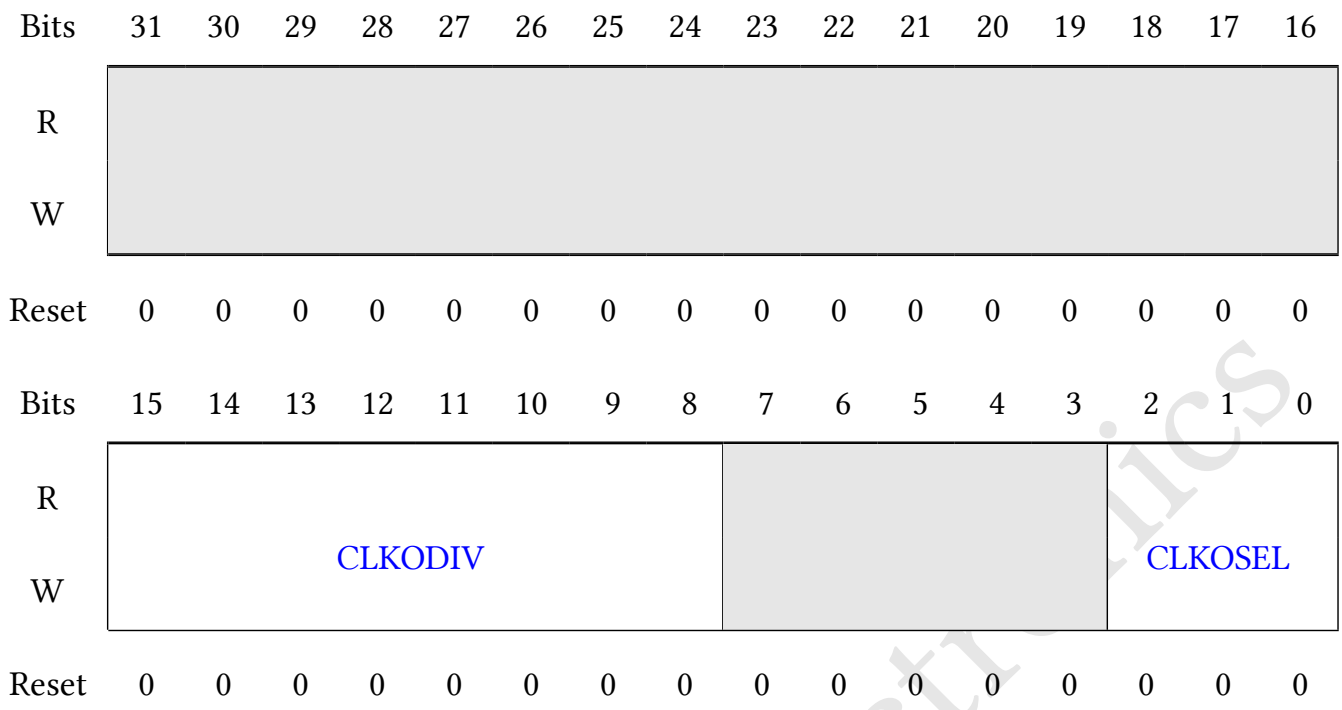


Table 12.10: SCU CLKO Register Description

Field	Function
15 - 8 CLKODIV	**Clock Output Divider** Clock output divider value. 0b00000000 - DIV disable 0b00000001 - DIV disable 0b00000010 - DIV as the value
2 - 0 CLKOSEL	**Clock Output Source Select** Clock output source select 000b - None clock select to output 001b - FIRC clock output 010b - SIRC clock output 011b - FXOSC clock output 100b - SXOSC clock output 101b - LPO clock output 110b - Reserved 111b - Core clock output

12.2.1.10 SCU FIRCTRIM Register

System Clock Unit FIRC trim controller

Offset: 30h

Yuntu Microelectronics

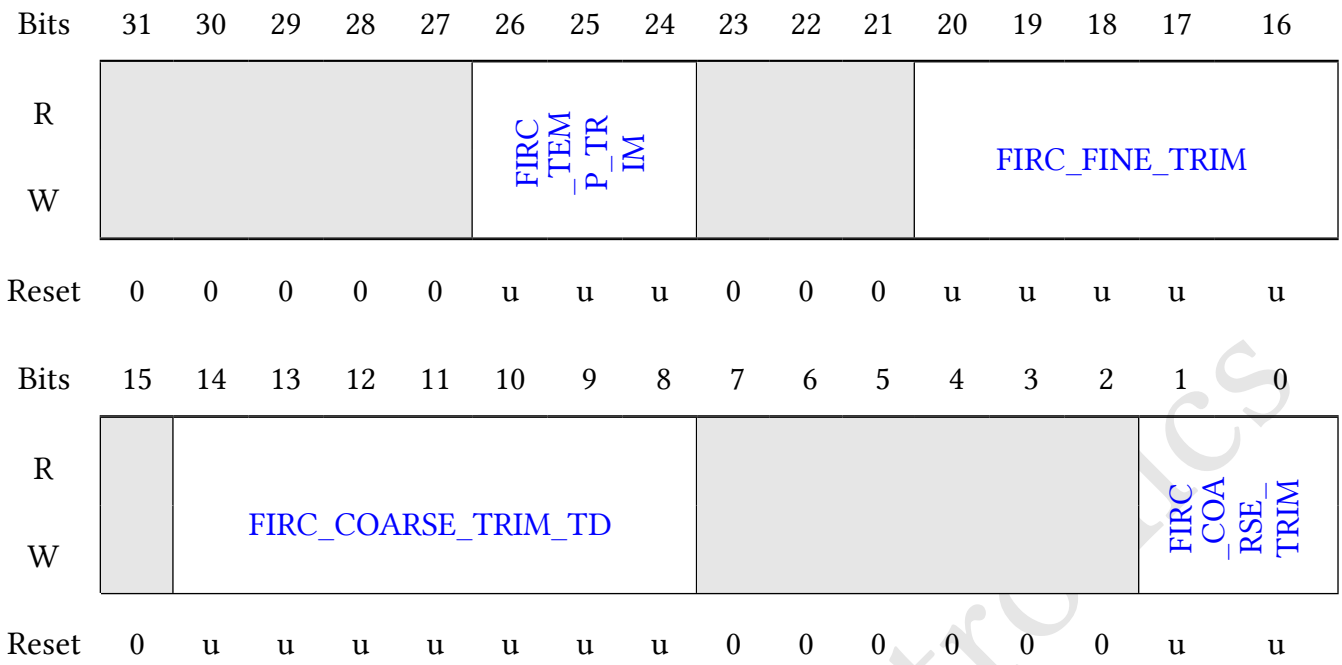


Table 12.11: SCU FIRCTRIM Register Description

Field	Function
26 - 24 FIRC_TEMP_TRIM	**FIRC Temperature Trim Bit** FIRC Temperature Trim bit
20 - 16 FIRC_FINE_TRIM	**FIRC Fine TRIM** FIRC Fine TRIM
14 - 8 FIRC_COARSE_TRIM_TD	**FIRC COARSE Trim TD Bit** FIRC COARSE Trim TD bit
1 - 0 FIRC_COARSE_TRIM	**FIRC COARSE Trim Bit** FIRC COARSE Trim bit

Yuntu Microelectronics

12.2.1.11 SCU SIRCTRIM Register

System Clock Unit SIRC trim register

Offset: 34h

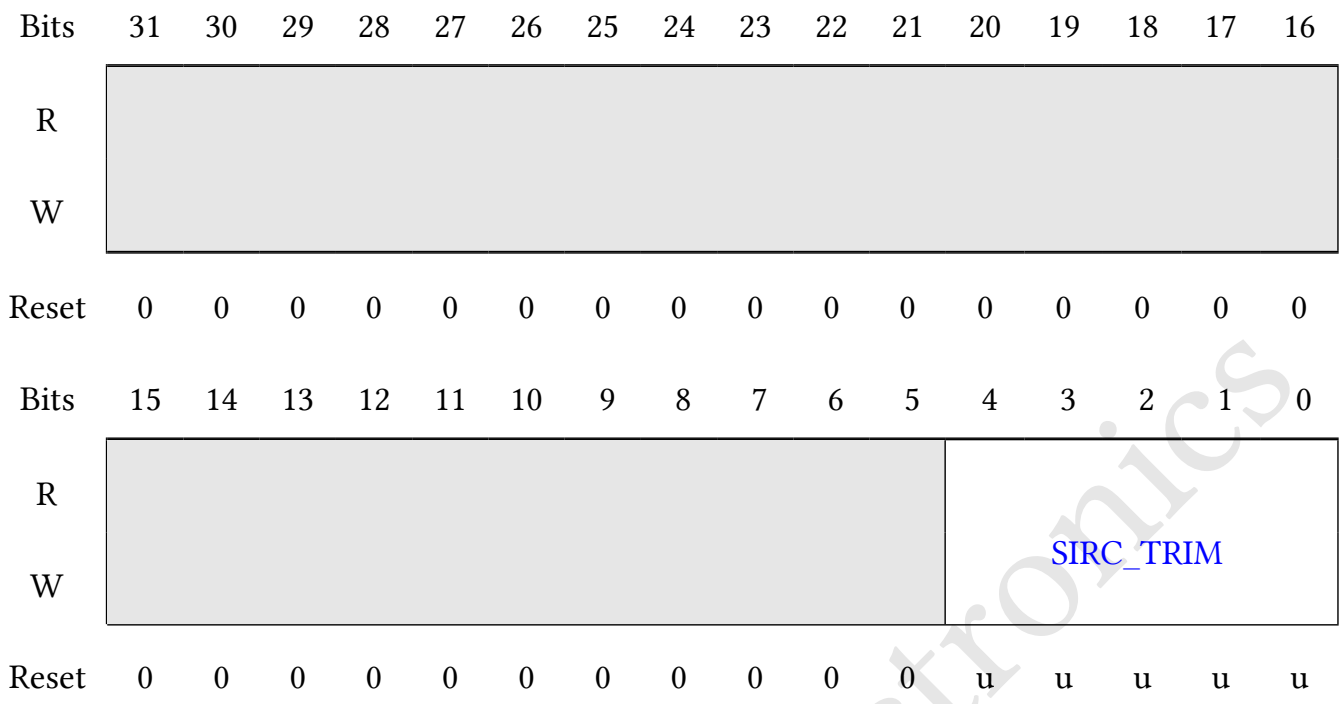


Table 12.12: SCU SIRCTRIM Register Description

Field	Function
4 - 0 SIRC_TRIM	**SIRC TRIM Bit** SIRC TRIM bit

Yuntu Microelectronics

12.2.1.12 SCU LPOTRIM Register

System Clock Unit LPOCLK trim register

Offset: 38h

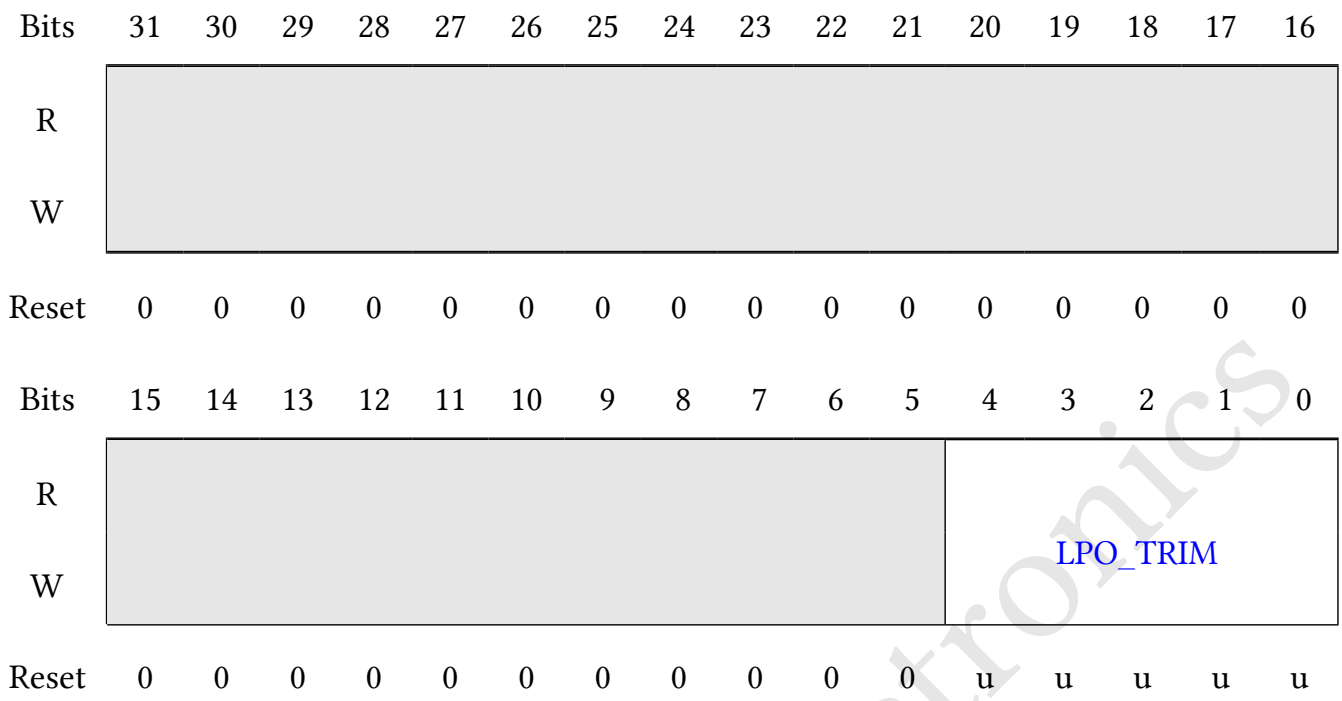


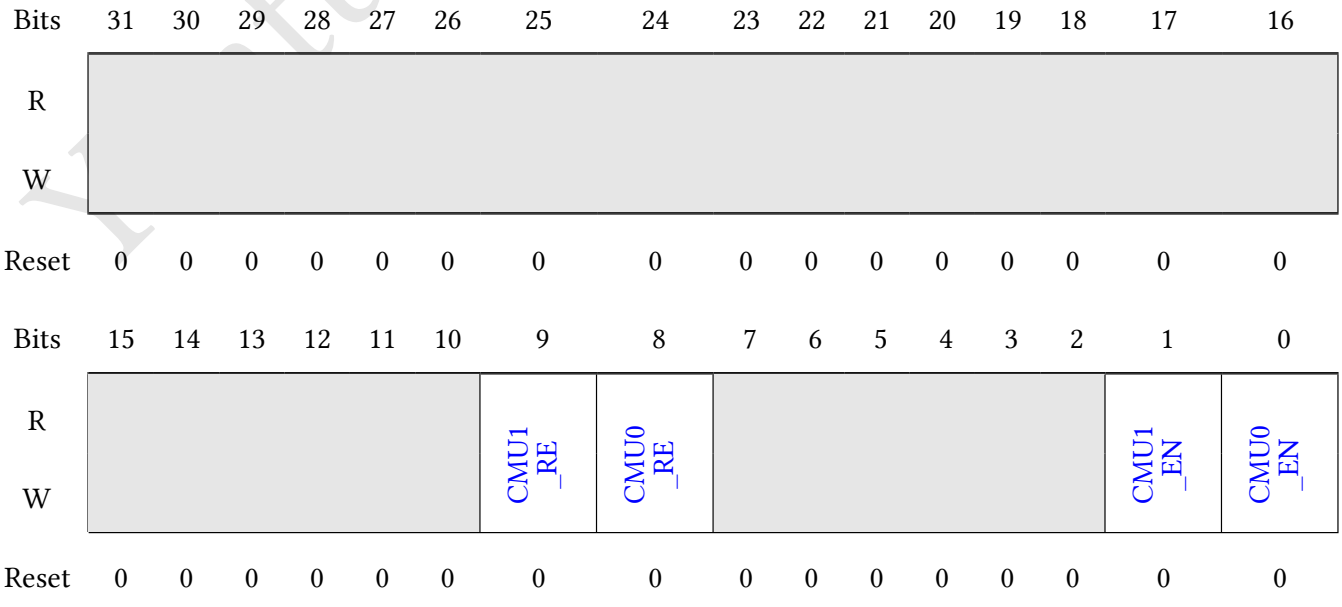
Table 12.13: SCU LPOTRIM Register Description

Field	Function
4 - 0 LPO_TRIM	**LPO Trim Bit** LPO Trim bit

12.2.1.13 SCU CMU_CTRL Register

System Clock Unit Clock Monitor Unit control register

Offset: 40h



Yuntu Microelectronics

Table 12.14: SCU CMU_CTRL Register Description

Field	Function
9 CMU1_RE	**CMU1 Reset Enable** CMU1 reset enable 0b - Reset disable when CMU event happen 1b - Reset enable when CMU event happen
8 CMU0_RE	**CMU0 Reset Enable** CMU0 reset enable 0b - Reset disable when CMU event happen 1b - Reset enable when CMU event happen
1 CMU1_EN	**CMU1 Enable** CMU1 enable 0b - CMU disable 1b - CMU enable
0 CMU0_EN	**CMU0 Enable** CMU0 enable 0b - CMU disable 1b - CMU enable

12.2.1.14 SCU CMUSTS Register

System Clock Unit Clock Monitor Unit status register

Offset: 44h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R															CMU1_LOSC	CMU0_LOSC
W															W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R							CMU1_LOSR	CMU0_LOSR							CMU1_OUTRNG	CMU0_OUTRNG
W							W1C	W1C							W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Yuntu Microelectronics

Table 12.15: SCU CMUSTS Register Description

Field	Function
17 CMU1_LOSC	**CMU1 Loss of Clock** CMU1 loss of clock
16 CMU0_LOSC	**CMU0 Loss of Clock** CMU0 loss of clock
9 CMU1_LOSR	**CMU1 Loss of Reference Clock** CMU1 loss of reference clock
8 CMU0_LOSR	**CMU0 Loss of Reference Clock** CMU0 loss of reference clock
1 CMU1_OUTRNG	**CMU1 Clock Out of Range** CMU1 clock out of range
0 CMU0_OUTRNG	**CMU0 Clock Out of Range** CMU0 clock out of range

Yuntu Microelectronics

12.2.1.15 SCU CMUCMP_HIGH Register

System Clock Unit Clock Monitor High Comparison Value Register

Offset: 90h + n * 8h

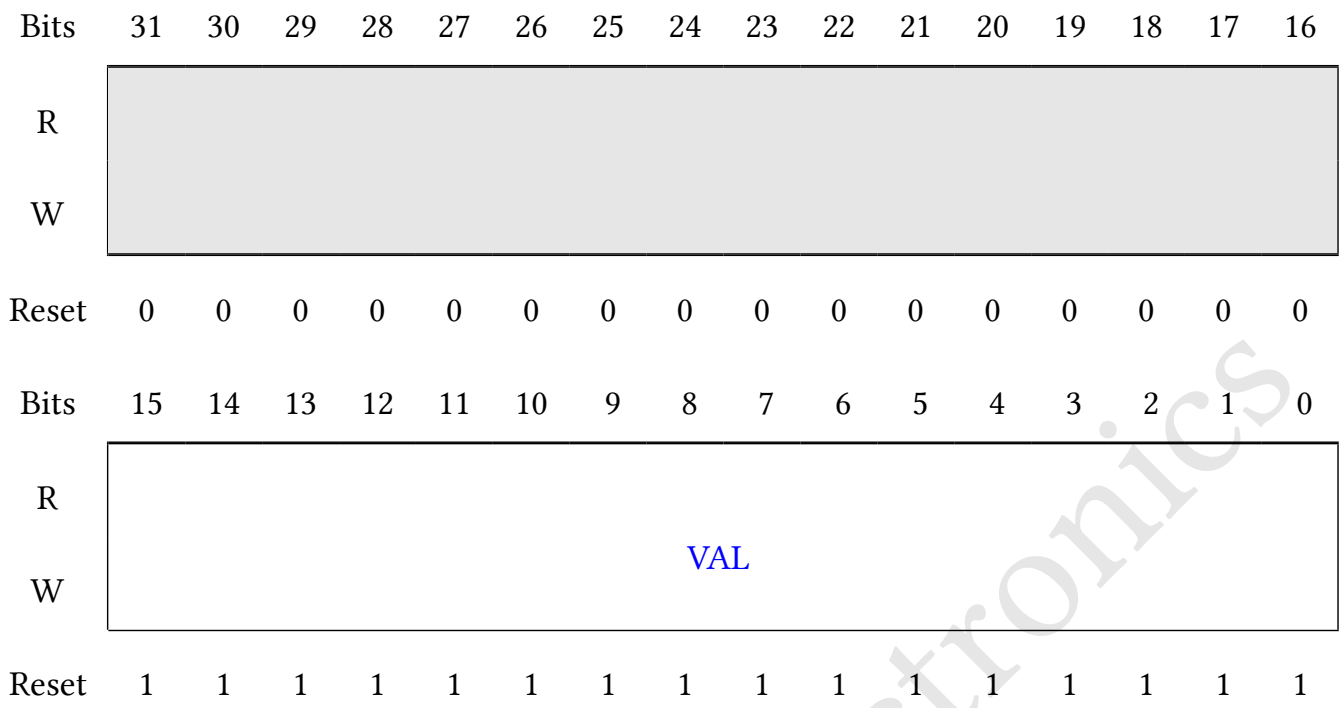


Table 12.16: SCU CMUCMP_HIGH Register Description

Field	Function
15 - 0 VAL	**High Comparison Value** High comparison value is greater than $\frac{F_{monitor_clk}}{F_{ref_clk}} * 128$. $F_{monitor_clk}$ is the frequency in MHz of monitored clock, such as FIRC, FXOSC. F_{ref_clk} is the frequency in MHz of reference clock, such as SIRC.

Yuntu Microelectronics

12.2.1.16 SCU CMUCMP_LOW Register

System Clock Unit Clock Monitor Low Comparison Value Register

Offset: 94h + n * 8h

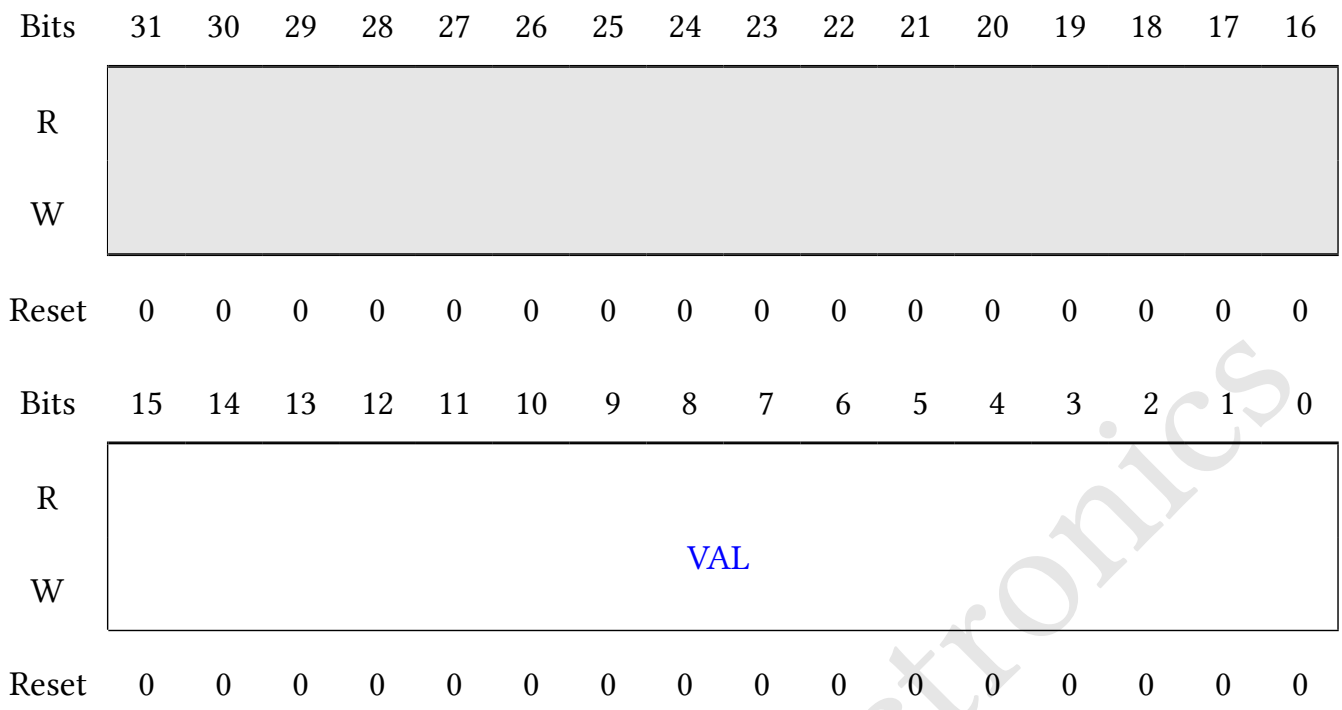


Table 12.17: SCU CMUCMP_LOW Register Description

Field	Function
15 - 0 VAL	**Low Comparison Value** High comparison value is less than $\frac{F_{monitor_clk}}{F_{ref_clk}} * 128$. $F_{monitor_clk}$ is the frequency in MHz of monitored clock, such as FIRC, FXOSC. F_{ref_clk} is the frequency in MHz of reference clock, such as SIRC.

12.3 Functional Description

12.3.1 Clock Switcher

The system clock unit(SCU) contains a glitch free clock switcher, user can switch the clock source when last clock switching is completed. User can configure the SCU CLKS[CLKSRCSEL] bit to change the system clock source, but the actual switch to the newly selected clock only happens when SCU STS[CLKST] change to be same as SCU CLKS[CLKSRCSEL]. If the newly selected clock source is not available the previous clock source will remain selected and SCU STS[CLKST] will not change.

12.3.2 System Clock Divider

The system clock unit(SCU) contains 3 dividers to generate core, fast bus and slow bus clock from system clock source selected by clock switcher. The core clock is divided from system clock source directly and fast bus clock is divided from core clock, the slow bus clock is divided from fast bus clock. The 3 dividers support divide-value from 1 to 16 by configure DIV[CPDIVS], DIV[FBDIVS] and DIV[SBDIVS], when clock dividers work with newly divider values the DIVSTS[CPDIVST], DIVSTS[FBDIVST] and DIVSTS[SBDIVST] reflected the divide ratios.

The setting of clock divider and system clock source selected by clock switcher should ensure the max

Yuntu Microelectronics

frequency of core/platform clock and fast bus clock lower than 48MHz, the max frequency of slow bus clock should be lower than 24MHz.

12.3.3 Clock Monitor Unit (CMU)

The SCU CMU block provides functions to monitor the abnormalities in the monitored clocks(checked clocks) and reference clocks, including loss of clock of checked clock, loss of clock of reference clock and out of frequency of checked clock. CMU supports to select SIRC as reference clock.

CMU is used to monitor various checked clocks

- CMU0: check FIRC Clock
- CMU1: check FXOSC Clock.

12.3.4 Clocks after Reset

When out of any of Reset, the FIRC and SIRC are enabled. FXOSC and SXOSC are disabled.

FIRC, FXOSC and SXOSC can be disabled. SIRC is always on unless it is forced to be disable in standby mode.

12.3.5 Operation Modes

- Active: SCU is full function.
- Sleep: SCU is full function.
- Deepsleep: Core clock, plat clock, fast bus clock and slow bus clock are gated, but all clock sources can be configured active.
- Standby: Only SIRC and SXOSC can be configured as active.

Yuntu Microelectronics

Chapter 13

Power Control Unit (PCU)

13.1 Introduction

The PCU is the Power Control Unit for the YTM32B1LE0 Family microcontrollers. It provides multiple power options to allow the user to optimize power consumption for the level of functionality needed. It includes the internal voltage regulators, power-on reset (POR) and the integrated low voltage detect system with reset (brownout) capability. The voltage regulator requires a 3.3V or 5V input to generate all the required secondary supplies.

13.1.1 Block Diagram

The following figure shows the PCU block diagram.

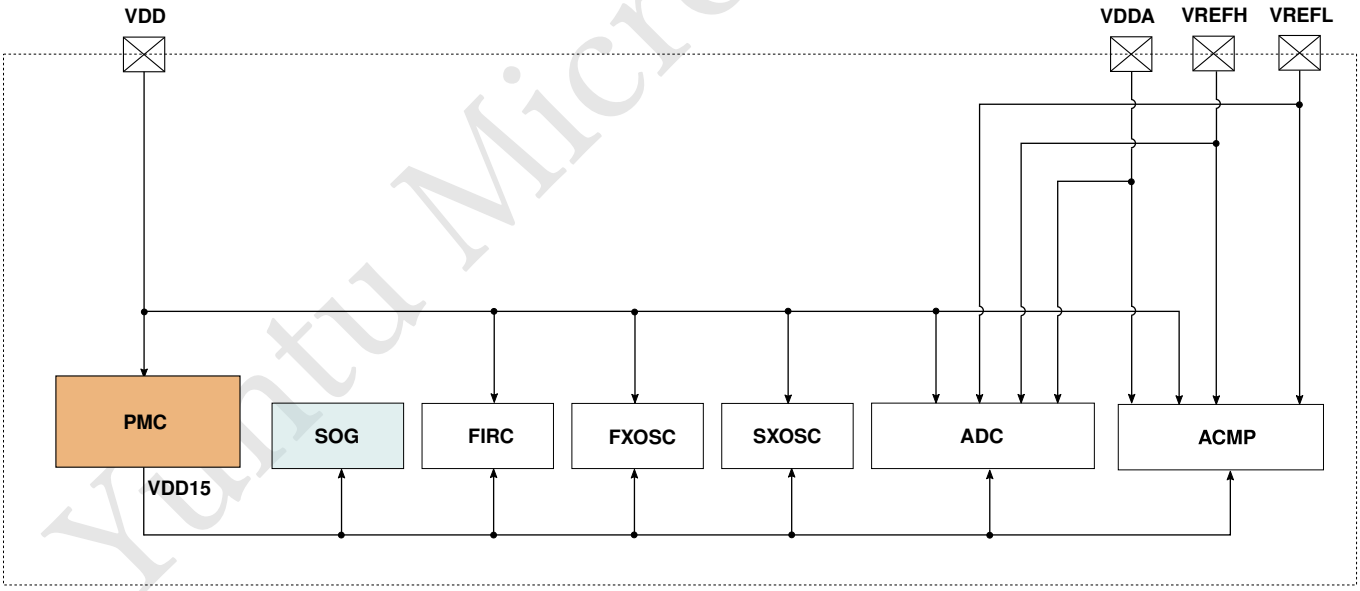


Figure 13.1: PCU Block Diagram

13.1.2 Features

- Combination of internal and external voltage regulator options, offering a variety of power modes
- Active POR providing brown-out detect
- Low voltage reset for all system relevant power domains (LVR)
- Low voltage detect (LVD) as an indication for software.

Yuntu Microelectronics

13.2 Register Definition

This section includes the PCU module memory map and detailed descriptions of all registers.

13.2.1 PCU Memory Map

PCU base address 4007d000h

Table 13.1: PCU Memory Map

Offset	Register	Width	Access	Reset Value
0h	BOR Control Register (BORCR)	32	RW	00000001h
4h	Low Voltage Detection Control Register (LVDCR)	32	RW	00100000h
8h	Control and Status Register (CSR)	32	RW	00200000h

13.2.1.1 PCU BORCR Register

Offset: 0h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R															BORLPMD	BOREN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 13.2: PCU BORCR Register Description

Field	Function
1 BORLPMD	**BOR Low Power Mode** BORLPMD Enable BOR Low Power Mode. That means BOR is enabled intermittently to saving power. The Power Saving efficiency is defined by ESCFG. BOREN must be set when BOR enters low power mode. **NOTE**. This funtion is reserved for internal testing only. The customer must strictly avoid setting this bit. 0b - BOR is not in Low Power Mode. 1b - BOR is in Low Power Mode(Energy Saving Mode).

Yuntu Microelectronics

Field	Function
0 BOREN	**BOR Enable** BOREN Enable the BOR module. When BOR is enabled, a bor reset occurs when the chip power supply voltage falls below BOR threshold voltage(about 2.2V). **NOTE**. This funtion is reserved for internal testing only. The customer must strictly avoid clearing this bit. 0b - BOR is disabled. 1b - BOR is enabled.

13.2.1.2 PCU LVDCR Register

Offset: 4h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R											LVD5VHYS		LVD5VCFG			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LVD5VOUT	LVD5VFLAG	LVD5VIE	LVD5VRE								FLTCFG		LVD5VFLTEN	LVD5VLPMD	LVD5VEN
W		W1C														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13.3: PCU LVDCR Register Description

Field	Function
21 - 20 LVD5VHYS	**LVD Hysteresis Configuration** This field configure the LVD Hysteresis. 00b - 0mV 01b - + 2.0% 10b - + 4.2% 11b - + 8.4%
19 - 16	**LVD5V Threshold Configure**

Yuntu Microelectronics

Field	Function
LVD5VCFG	This field configures the LVD5V Threshold Voltage. 0000b - 1.860V 0001b - 2.120V 0010b - 2.370V 0011b - 2.630V 0100b - 2.870V 0101b - 3.120V 0110b - 3.390V 0111b - 3.650V 1000b - 3.910V 1001b - 4.160V 1010b - 4.360V 1011b - 4.530V 1100b - 4.720V 1101b - 4.910V 1110b - Reserved 1111b - Reserved
15 LVD5VOUT	**LVD5V Output** This read only bit reflects the LVD5V detection result. If LVD5V glitch filter is enabled, this bit shows the filtered result. 0b - The supply voltage 5v is above LVD threshold. 1b - The supply voltage 5v is below LVD threshold.
14 LVD5VFLAG	**LVD5V Flag** Write 1 to clear this flag. 0b - The supply voltage 5v is above LVD threshold. 1b - The supply voltage 5v is below LVD threshold.
13 LVD5VIE	**LVD5V Interrupt Enable** Enable/Disable LVD5V interrupt. 0b - Don't assert interrupt when LVD5VFLAG is set. 1b - Assert a PCU interrupt when LVD5VFLAG is set.
12 LVD5VRE	**LVD5V Reset Enable** Enable/Disable LVD5V reset. 0b - Don't generate reset when the supply voltage 5v is below LVD threshold. 1b - Generate reset when the supply voltage 5v is below LVD threshold.
4 - 3 FLTCFG	**Config LVD5VOUT Glitch Filter Width** There' s an optional digital glitch filter for LVD5VOUT. The Filter Clock is SIRC divided by 64 (31.25KHz) and the glitch filter width can be configured by FLTCFG. The glitches that width is shorter than (FLTCFG+1)*(the period of SIRC multiplied by 64) can be filtered.
2	**LVD5VOUT Glitch Filter Enable**

Yuntu Microelectronics

Field	Function
LVD5VFLTEN	When this bit is set, LVD5V glitch filter is enabled. The filter is also enabled if LVD5VLPMD is set no matter LVD5VFLTEN is set or not. 0b - LVD5VOUT is not filtered by Glitch Filter if LVD5VLPMD is not set. 1b - LVD5VOUT is filtered by Glitch Filter.
1 LVD5VLPMD	**LVD5V Lower Power Mode Enable** When this bit is set, LVD5V is turned on intermittently for saving power. The Power Saving efficiency is defined by ESCFG. LVD5VEN must be set when LVD5V enters low power mode. 0b - LVD is not in Low Power Mode. 1b - LVD is in Low Power Mode(Energy Saving Mode).
0 LVD5VEN	**LVD5V Enable** LVD5VEN enable the low-voltage detect of VDD5v. After enabling LVD function, need wait for more than 3us settling time until LVD circuit is stable. 0b - LVD is disabled. 1b - LVD is enabled.

13.2.1.3 PCU CSR Register

Offset: 8h

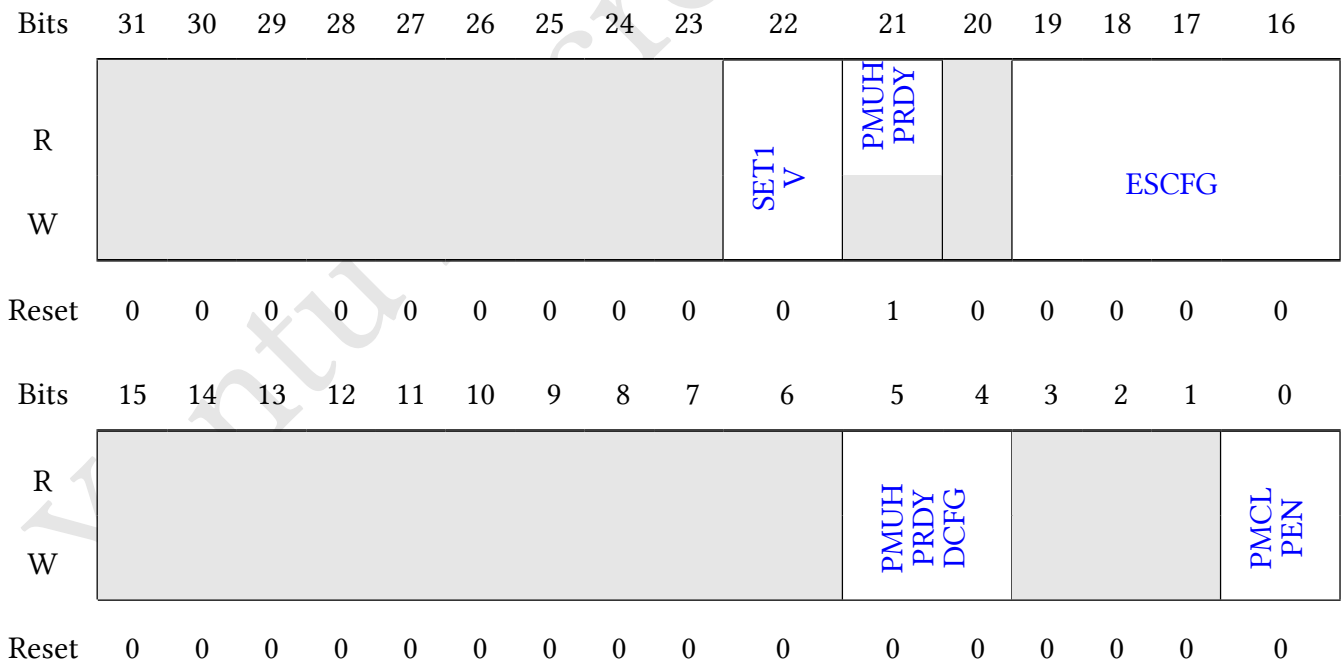


Table 13.4: PCU CSR Register Description

Field	Function
22	**Set Core Supply to 1v when PMU Is in Low Power Mode**

Yuntu Microelectronics

Field	Function
SET1V	When set, the core supply will change to 1v if PMU is in low power mode. 0b - Core supply is 1.5v 1b - Set core supply to 1v
21 PMUHPRDY	**PMU High Power Mode Ready** When set, it means PMU is in high power mode. FIRC must work in high power mode. So, please make sure this bit is 1 before enable FIRC or switch some clock to FIRC. 0b - PMU is in low power mode 1b - PMU high power mode is ready
19 - 16 ESCFG	**Energy Saving Configure** This field configures the Energy Saving efficiency. All values except those in the range described below are reserved. **NOTE**. This funtion is reserved for internal testing only. The customer must strictly avoid modifying these bits. 0000b - 0.10% of normal mode, Sampling period=512ms 0001b - 0.20% of normal mode, Sampling period=256ms 0010b - 0.40% of normal mode, Sampling period=128ms 0011b - 0.80% of normal mode, Sampling period=64ms 0100b - 1.56% of normal mode, Sampling period=32ms 0101b - 3.10% of normal mode, Sampling period=16ms 0110b - 6.25% of normal mode, Sampling period=8ms 0111b - 12.5% of normal mode, Sampling period=4ms 1000b - 25.0% of normal mode, Sampling period=2ms 1001b - 50.0% of normal mode, Sampling period=1ms
5 - 4 PMUHPRDY DCFG	**PMU High Power Mode Ready Delay Configure** When PMUHPRDY is in posedge state, it will delay for some time to be 1. The delay length is determined by the configured value. 00b - The delay length is 80us 01b - The delay length is 40us 10b - The delay length is 20us 11b - Reserved
0 PMCLPEN	**PMC Low Power Mode Enable** This bit is to control PMC to enter low power or not when chip enters low power mode. 0b - PMC remains at full power mode when chip enters low power mode. 1b - PMC will enter low power mode when chip enters low power mode.

13.3 Functional Description

The following sections describe functional details of the PCU.

Yuntu Microelectronics

13.3.1 Brown-out

PCU provides brown-out reset(BOR) to reset system when supply voltage is below BOR threshold.
In low power mode, the BORCR[BORLPMD] could be set to save power. BOR is no longer an immediate trigger, but a periodic detection. In this situation, the frequency of BOR detection is controlled by CSR[ESCFG].

NOTE. The BOR low power funtion is reserved for internal testing only. The customer must strictly avoid using this function.

13.3.2 Low Voltage Detect

PCU provides 5V low voltage detection(LVD). For LVD5V, PCU incorporates a glitch filter to provide a more stable output.
In low power mode, the LVDCR[LVD5VLPMD] could be set to save power. LVD5V is no longer an immediate trigger, but a periodic detection. In this situation, the frequency of LVD5V detection is controlled by CSR[ESCFG].

13.3.3 Active Mode

Allow maximum performance of the chip and use all functions of the chip. It is the default mode when out of reset or power on reset. The on-chip regulators are working at high power mode to supply powers for internal modules. In this mode the chip can work at max frequency 48MHz.

13.3.4 Sleep Mode

Part of the chip is in static state in Sleep mode, and the CPU Score is stopped, but all the other modules are working same as they are in active mode. All the registers and memory contents are retained. LVD protection is maintained. NVIC remains active to interrupts in Sleep mode.

13.3.5 Deepsleep Mode

The chip is in static state but the high power on chip regulators keep working, all registers and memories retain values, NVIC inactive, peripheries can work with asynchronous clock sources other than bus clocks, asynchronous interrupts from peripheries can wake up chip from Deepsleep mode.

13.3.6 Standby Mode

The chip is in static state, high power regulators are inactive, the chip is powered by low power on chip regulators, all registers and memories retain values, NVIC inactive, peripheries can work with slow asynchronous clocks other than bus clocks, asynchronous interrupts from peripheries can wake up chip from Standby mode.

Yuntu Microelectronics

Chapter 14

IP Clock Control (IPC)

14.1 Introduction

The IP Clock Control (IPC) module provides clock control and configuration for on-chip peripherals. Each peripheral has its clock control and configuration register.

The peripheral's operational (bus interface) clock must be enabled in the IPC register before the peripheral can be used.

A peripheral may also require an asynchronous (functional) clock, so the IPC also provides some clock inputs options for a peripheral which are software selectable.

Any peripheral can choose its asynchronous clock by setting the Clock Source Select (SRCSEL) bits.

Optionally, any peripheral can also have a clock divider for its asynchronous clock, selected by setting the Clock Divider (DIV) bits.

14.1.1 Features

- Peripheral Bus clock enable
- IPC clock source selection as follow
  - FIRC 48 MHz
  - SIRC 2 MHz
  - FXOSC 4~40 MHz
  - SXOSC 32.768 KHz
  - LPO 750 Hz
  - Core clock
- IPC clock divide values from 1 to 16
- Peripheral software reset

14.1.2 Block Diagram

The following figure shows the IPC block diagram.

Yuntu Microelectronics

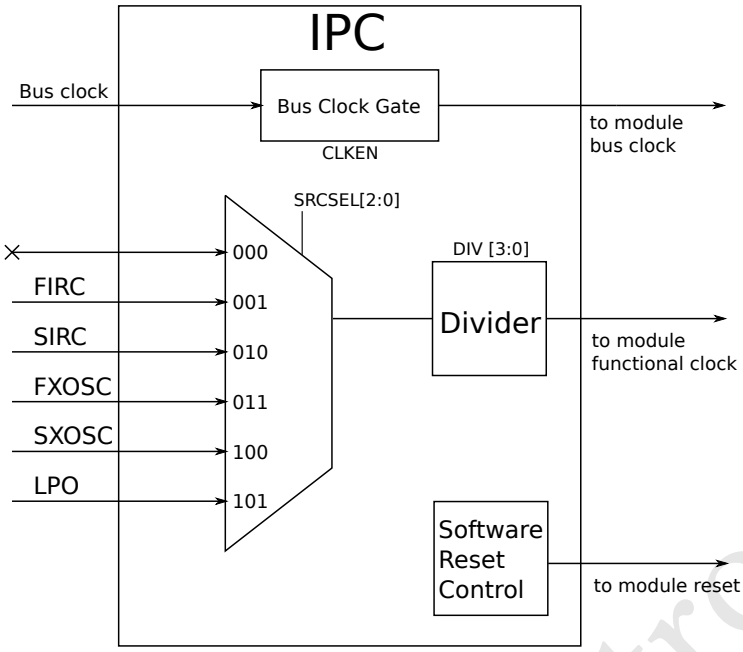


Figure 14.1: IPC Block Diagram

14.2 Peripheral Clock Table

Table 14.1: IP Clock Control Table

No.	Module Name	Bus Clock	Bus Clock Gate	Peripheral Functional Clock	Additional Clock	Software Reset
8	DMA	SYS_CLK	YES[1]	-	-	YES
14	DIVSQRT	-	-	-	-	-
15	GPIO	SYS_CLK	YES	YES	-	YES
24	EMU	SYS_CLK	YES	-	-	YES
32	EFM	SYS_CLK	YES[2]	-	-	-
36	CAN0	FAST_BUS_CLK	YES	-	-	YES
44	SPI0	SLOW_BUS_CLK	YES	YES	-	YES
45	SPI1	SLOW_BUS_CLK	YES	YES	-	YES
46	SPI2	SLOW_BUS_CLK	YES	YES	-	YES
50	CRC	SYS_CLK	YES	-	-	YES
55	pTMR0	SLOW_BUS_CLK	YES	-	-	YES
56	eTMR0	FAST_BUS_CLK	YES	-	-	YES
57	eTMR1	FAST_BUS_CLK	YES	-	-	YES
58	eTMR2	FAST_BUS_CLK	YES	-	-	YES
59	ADC0	FAST_BUS_CLK	YES	YES	-	YES
61	RTC	SLOW_BUS_CLK	YES	-	-	YES
64	lpTMR0	SLOW_BUS_CLK	YES	YES	-	YES

[1]Clock gate reset value is CLKEN=1
[2]Clock gate reset value is CLKEN=1

Yuntu Microelectronics

Table 14.1 continued from previous page

No.	Module Name	Bus Clock	Bus Clock Gate	Peripheral Functional Clock	Additional Clock	Software Reset
72	CIM	SLOW_BUS_CLK	YES	-	-	YES
73	PCTRLA	SLOW_BUS_CLK	-	-	-	YES
74	PCTRLB	SLOW_BUS_CLK	-	-	-	YES
75	PCTRLC	SLOW_BUS_CLK	-	-	-	YES
76	PCTRLD	SLOW_BUS_CLK	-	-	-	YES
77	PCTRLE	SLOW_BUS_CLK	-	-	-	YES
82	WDG	SLOW_BUS_CLK	YES[3]	-	-	YES
99	TMU	SLOW_BUS_CLK	-	-	-	YES
100	SCU	SLOW_BUS_CLK	YES	-	-	YES
102	I2C0	SLOW_BUS_CLK	YES	YES	-	YES
103	I2C1	SLOW_BUS_CLK	YES	YES	-	YES
106	UART0	SLOW_BUS_CLK	YES	YES	-	YES
107	UART1	SLOW_BUS_CLK	YES	YES	-	YES
108	UART2	SLOW_BUS_CLK	YES	YES	-	YES
115	ACMP0	SLOW_BUS_CLK	YES	YES	-	YES
125	PCU	SLOW_BUS_CLK	YES[4]	-	-	-
127	RCU	SLOW_BUS_CLK	YES[5]	-	-	-

Note:

1. Bus clock is used to operate registers of IPs, if Bus clock gate of corresponding IP is off, registers will not be required to operate. If operates, hard fault will generate.
2. Peripheral Functional Clock “YES” means its clock source and clock divider can be configured in the IPC register. “-” means can’t.
3. FlexCAN module clock source can be selected as FAST_BUS_CLK or FXOSC_CLK, this is set in register of FlexCAN IP.
4. RCU module digital filter function needs SIRC or LPO_CLK.

14.3 Register Definition

14.3.1 IPC Memory Map

IPC base address 40065000h

Table 14.2: IPC Memory Map

Offset	Register	Width	Access	Reset Value
0h	Control Register (CTRL0)	32	RW	00000000h
4h	Control Register (CTRL1)	32	RW	00000000h
8h	Control Register (CTRL2)	32	RW	00000000h
ch	Control Register (CTRL3)	32	RW	00000000h
10h	Control Register (CTRL4)	32	RW	00000000h

[3]Clock gate reset value is CLKEN=1
[4]Clock gate reset value is CLKEN=1
[5]Clock gate reset value is CLKEN=1

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
14h	Control Register (CTRL5)	32	RW	00000000h
18h	Control Register (CTRL6)	32	RW	00000000h
1ch	Control Register (CTRL7)	32	RW	00000000h
20h	Control Register (CTRL8)	32	RW	00000000h
24h	Control Register (CTRL9)	32	RW	00000000h
28h	Control Register (CTRL10)	32	RW	00000000h
2ch	Control Register (CTRL11)	32	RW	00000000h
30h	Control Register (CTRL12)	32	RW	00000000h
34h	Control Register (CTRL13)	32	RW	00000000h
38h	Control Register (CTRL14)	32	RW	00000000h
3ch	Control Register (CTRL15)	32	RW	00000000h
40h	Control Register (CTRL16)	32	RW	00000000h
44h	Control Register (CTRL17)	32	RW	00000000h
48h	Control Register (CTRL18)	32	RW	00000000h
4ch	Control Register (CTRL19)	32	RW	00000000h
50h	Control Register (CTRL20)	32	RW	00000000h
54h	Control Register (CTRL21)	32	RW	00000000h
58h	Control Register (CTRL22)	32	RW	00000000h
5ch	Control Register (CTRL23)	32	RW	00000000h
60h	Control Register (CTRL24)	32	RW	00000000h
64h	Control Register (CTRL25)	32	RW	00000000h
68h	Control Register (CTRL26)	32	RW	00000000h
6ch	Control Register (CTRL27)	32	RW	00000000h
70h	Control Register (CTRL28)	32	RW	00000000h
74h	Control Register (CTRL29)	32	RW	00000000h
78h	Control Register (CTRL30)	32	RW	00000000h
7ch	Control Register (CTRL31)	32	RW	00000000h
80h	Control Register (CTRL32)	32	RW	00000000h
84h	Control Register (CTRL33)	32	RW	00000000h
88h	Control Register (CTRL34)	32	RW	00000000h
8ch	Control Register (CTRL35)	32	RW	00000000h
90h	Control Register (CTRL36)	32	RW	00000000h
94h	Control Register (CTRL37)	32	RW	00000000h
98h	Control Register (CTRL38)	32	RW	00000000h
9ch	Control Register (CTRL39)	32	RW	00000000h
a0h	Control Register (CTRL40)	32	RW	00000000h
a4h	Control Register (CTRL41)	32	RW	00000000h
a8h	Control Register (CTRL42)	32	RW	00000000h
ach	Control Register (CTRL43)	32	RW	00000000h
b0h	Control Register (CTRL44)	32	RW	00000000h
b4h	Control Register (CTRL45)	32	RW	00000000h
b8h	Control Register (CTRL46)	32	RW	00000000h
bch	Control Register (CTRL47)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
c0h	Control Register (CTRL48)	32	RW	00000000h
c4h	Control Register (CTRL49)	32	RW	00000000h
c8h	Control Register (CTRL50)	32	RW	00000000h
cch	Control Register (CTRL51)	32	RW	00000000h
d0h	Control Register (CTRL52)	32	RW	00000000h
d4h	Control Register (CTRL53)	32	RW	00000000h
d8h	Control Register (CTRL54)	32	RW	00000000h
dch	Control Register (CTRL55)	32	RW	00000000h
e0h	Control Register (CTRL56)	32	RW	00000000h
e4h	Control Register (CTRL57)	32	RW	00000000h
e8h	Control Register (CTRL58)	32	RW	00000000h
ech	Control Register (CTRL59)	32	RW	00000000h
f0h	Control Register (CTRL60)	32	RW	00000000h
f4h	Control Register (CTRL61)	32	RW	00000000h
f8h	Control Register (CTRL62)	32	RW	00000000h
fch	Control Register (CTRL63)	32	RW	00000000h
100h	Control Register (CTRL64)	32	RW	00000000h
104h	Control Register (CTRL65)	32	RW	00000000h
108h	Control Register (CTRL66)	32	RW	00000000h
10ch	Control Register (CTRL67)	32	RW	00000000h
110h	Control Register (CTRL68)	32	RW	00000000h
114h	Control Register (CTRL69)	32	RW	00000000h
118h	Control Register (CTRL70)	32	RW	00000000h
11ch	Control Register (CTRL71)	32	RW	00000000h
120h	Control Register (CTRL72)	32	RW	00000000h
124h	Control Register (CTRL73)	32	RW	00000000h
128h	Control Register (CTRL74)	32	RW	00000000h
12ch	Control Register (CTRL75)	32	RW	00000000h
130h	Control Register (CTRL76)	32	RW	00000000h
134h	Control Register (CTRL77)	32	RW	00000000h
138h	Control Register (CTRL78)	32	RW	00000000h
13ch	Control Register (CTRL79)	32	RW	00000000h
140h	Control Register (CTRL80)	32	RW	00000000h
144h	Control Register (CTRL81)	32	RW	00000000h
148h	Control Register (CTRL82)	32	RW	00000000h
14ch	Control Register (CTRL83)	32	RW	00000000h
150h	Control Register (CTRL84)	32	RW	00000000h
154h	Control Register (CTRL85)	32	RW	00000000h
158h	Control Register (CTRL86)	32	RW	00000000h
15ch	Control Register (CTRL87)	32	RW	00000000h
160h	Control Register (CTRL88)	32	RW	00000000h
164h	Control Register (CTRL89)	32	RW	00000000h
168h	Control Register (CTRL90)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
16ch	Control Register (CTRL91)	32	RW	00000000h
170h	Control Register (CTRL92)	32	RW	00000000h
174h	Control Register (CTRL93)	32	RW	00000000h
178h	Control Register (CTRL94)	32	RW	00000000h
17ch	Control Register (CTRL95)	32	RW	00000000h
180h	Control Register (CTRL96)	32	RW	00000000h
184h	Control Register (CTRL97)	32	RW	00000000h
188h	Control Register (CTRL98)	32	RW	00000000h
18ch	Control Register (CTRL99)	32	RW	00000000h
190h	Control Register (CTRL100)	32	RW	00000000h
194h	Control Register (CTRL101)	32	RW	00000000h
198h	Control Register (CTRL102)	32	RW	00000000h
19ch	Control Register (CTRL103)	32	RW	00000000h
1a0h	Control Register (CTRL104)	32	RW	00000000h
1a4h	Control Register (CTRL105)	32	RW	00000000h
1a8h	Control Register (CTRL106)	32	RW	00000000h
1ach	Control Register (CTRL107)	32	RW	00000000h
1b0h	Control Register (CTRL108)	32	RW	00000000h
1b4h	Control Register (CTRL109)	32	RW	00000000h
1b8h	Control Register (CTRL110)	32	RW	00000000h
1bch	Control Register (CTRL111)	32	RW	00000000h
1c0h	Control Register (CTRL112)	32	RW	00000000h
1c4h	Control Register (CTRL113)	32	RW	00000000h
1c8h	Control Register (CTRL114)	32	RW	00000000h
1cch	Control Register (CTRL115)	32	RW	00000000h
1d0h	Control Register (CTRL116)	32	RW	00000000h
1d4h	Control Register (CTRL117)	32	RW	00000000h
1d8h	Control Register (CTRL118)	32	RW	00000000h
1dch	Control Register (CTRL119)	32	RW	00000000h
1e0h	Control Register (CTRL120)	32	RW	00000000h
1e4h	Control Register (CTRL121)	32	RW	00000000h
1e8h	Control Register (CTRL122)	32	RW	00000000h
1ech	Control Register (CTRL123)	32	RW	00000000h
1f0h	Control Register (CTRL124)	32	RW	00000000h
1f4h	Control Register (CTRL125)	32	RW	00000000h
1f8h	Control Register (CTRL126)	32	RW	00000000h
1fch	Control Register (CTRL127)	32	RW	00000000h

Yuntu Microelectronics

14.3.1.1 IPC CTRL Register

Offset: 0h + n * 4h

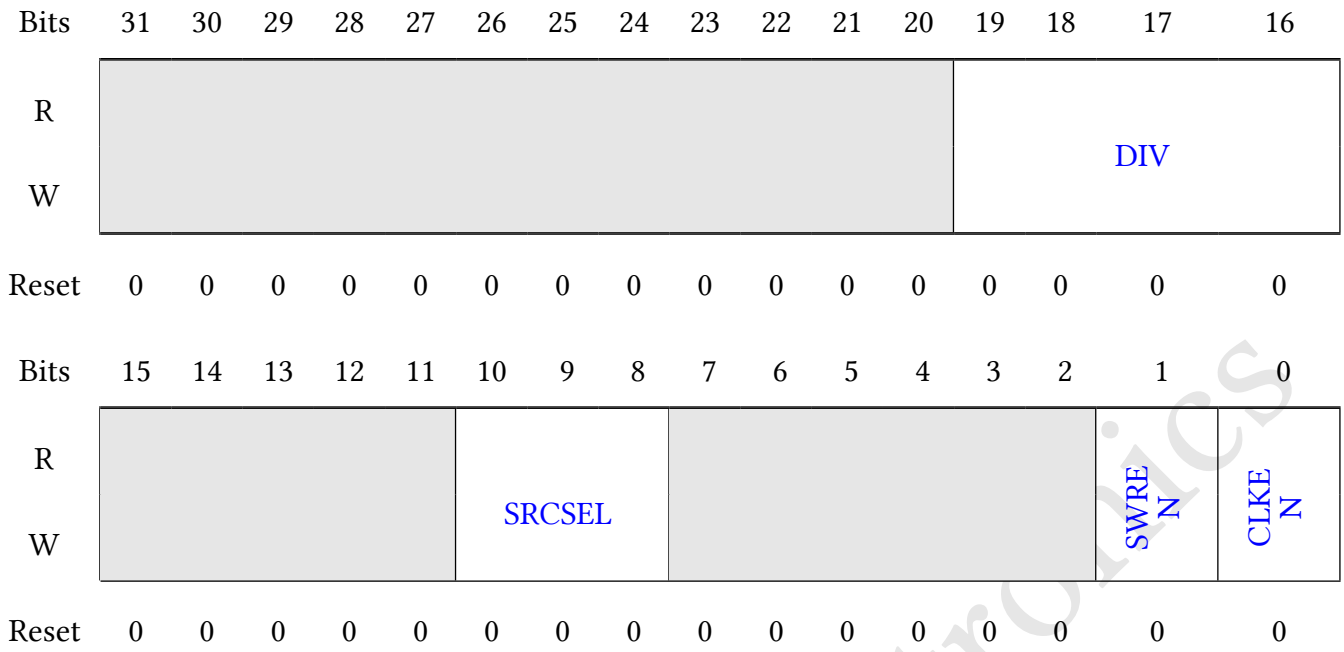


Table 14.3: IPC CTRL Register Description

Field	Function
19 - 16 DIV	**Clock Divider** The divide value = register value + 1 0000b - Divided by 1 0001b - Divided by 2 0010b - Divided by 3 0011b - Divided by 4 0100b - Divided by 5 0101b - Divided by 6 0110b - Divided by 7 0111b - Divided by 8 1000b - Divided by 9 1001b - Divided by 10 1010b - Divided by 11 1011b - Divided by 12 1100b - Divided by 13 1101b - Divided by 14 1110b - Divided by 15 1111b - Divided by 16
10 - 8	**Clock Source Selection**

Yuntu Microelectronics

Field	Function
SRCSEL	000b - Reserved 001b - Select FIRC 010b - Select SIRC 011b - Select FXOSC 100b - Select SXOSC 101b - Select LPO 110b - Reserved 111b - Reserved
1 SWREN	**Software Reset** 0b - Software reset is de-asserted 1b - Software reset is asserted
0 CLKEN	**Clock Enable** 0b - IP clock is disabled 1b - IP clock is enabled

14.4 Functional Description

The IPC module provides dedicated IPC registers for clock gating and configuration options for on-chip peripherals (modules).

Each module's IPC register contains a clock enable control bit (CLKEN) for its interface clock.

Before a module can be used, its interface clock must be enabled (CLKEN = 1) in the module's IPC register.

If a module has a functional clock, its IPC register may provide the clock source options, selected by programming the Clock Source Selection (SRCSEL) field.

Optionally, a module may also have a clock divider, selected by programming the Clock Divider (DIV) field.

NOTE: Please disable the corresponding configuration (CLKEN) while change a peripheral clock source (SRCSEL) and clock divider (DIV). Before the clock configuration changes, please also ensure the module is disabled.

Yuntu Microelectronics

Chapter 15

Reset Control Unit (RCU)

15.1 Introduction

Reset Control Unit implements the reset functions for the chip.

15.1.1 Features

- Record the reset sources of most recent resets.
- Configurable filter for reset pin.
- Reset pin filter can work at both Active and Sleep mode.

15.2 Register Definition

This subsection provides a description of the memory map and registers.

15.2.1 RCU Memory Map

RCU base address 4007f000h

Table 15.1: RCU Memory Map

Offset	Register	Width	Access	Reset Value
0h	Reset Source Status Register (RSSR)	32	RW	00000001h
4h	Reset Pin Filter Register (RPFR)	32	RW	0000FF00h

15.2.1.1 RCU RSSR Register

This register includes read-only status flags to indicate the source of the most recent reset. Multiple flags will be set if multiple reset events occur at the same time.

Offset: 0h

Yuntu Microelectronics

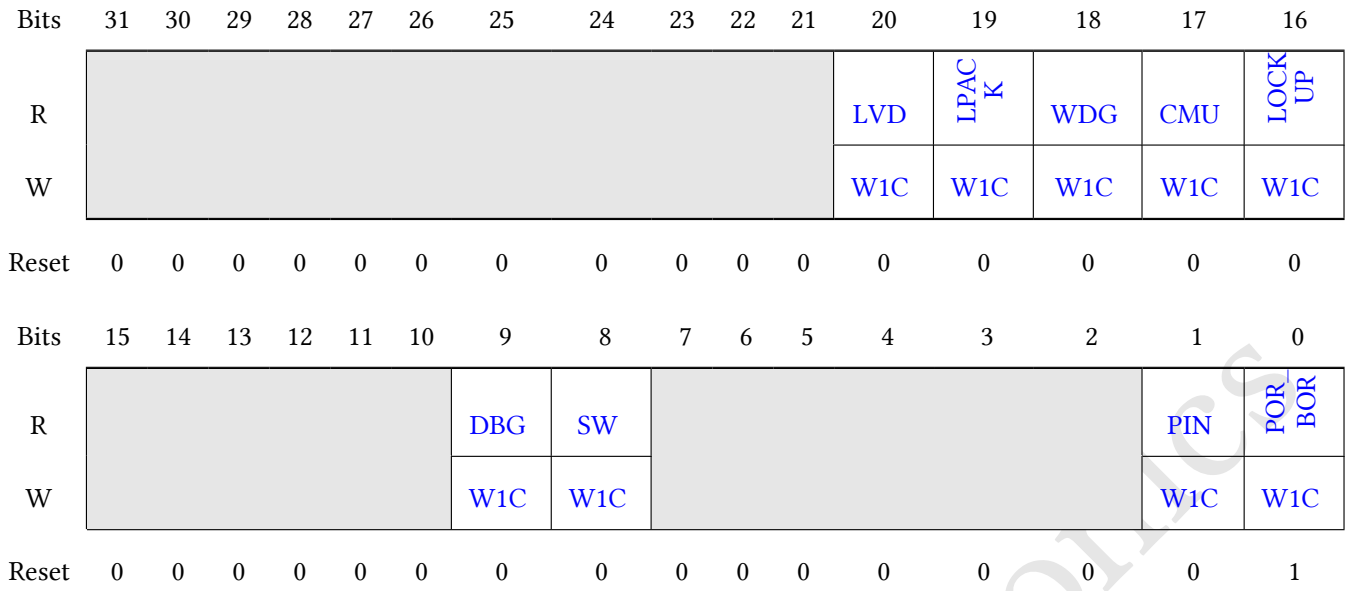


Table 15.2: RCU RSSR Register Description

Field	Function
20 LVD	**Low Voltage Detect** System reset caused by Low Voltage Detect. 0b - System reset not caused by Low Voltage Detect. 1b - System reset caused by Low Voltage Detect.
19 LPACK	**Low Power mode lack of ACK Reset** System reset caused by Deepsleep/Standby acknowledge timeout. Indicates that after an attempt to enter Deepsleep/Standby mode, a reset has been caused by a failure of one or more peripherals to acknowledge. 0b - System reset not caused by Deepsleep/Standby acknowledge timeout. 1b - System reset caused by Deepsleep/Standby acknowledge timeout.
18 WDG	**Watch Dog Reset** System reset caused by Clock Monitor Watchdog. 0b - System reset not caused by WDG. 1b - System reset caused by WDG.
17 CMU	**Clock Monitor Reset** System reset caused by Clock Monitor if CMURE is set. 0b - System reset not caused by CMU. 1b - System reset caused by CMU.
16 LOCKUP	**ARM Core Lockup Reset** System reset caused by ARM Core Lockup. 0b - System reset not caused by Lockup. 1b - System reset caused by Lockup.
9	**Debug Reset**

Yuntu Microelectronics

Field	Function
DBG	System reset caused by Debug. 0b - System reset not caused by debug. 1b - System reset caused by debug.
8 SW	**Software Reset** System reset caused by software to set SYSRESETREQ of ARM Core. 0b - System reset not caused by SW. 1b - System reset caused by SW.
1 PIN	**Pin Reset** System reset caused by RESET_b pin. 0b - System reset not caused by PIN. 1b - System reset caused by PIN.
0 POR_BOR	**Poweron or Brownout Reset** System reset caused by POR or BOR. 0b - System reset not caused by BOR or LVD. 1b - System reset caused by BOR or LVD.

15.2.1.2 RCU RPFR Register

This register controls the reset pin filter functions.

Offset: 4h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W																RSTPIN_OBEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FLTVAL													FLTCLKSEL	FLTSTEN	FLTEN
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Table 15.3: RCU RPFR Register Description

Field	Function
16	**Reset Pin Output Enable**

Yuntu Microelectronics

Field	Function
RSTPIN_OBEN	Enable bit of Reset pin output 0b - Not output low on reset pin when chip reset occurs 1b - Output low on reset pin when chip reset occurs
15 - 8 FLTVAL	**Reset Pin Filter Counter Value** FLTVAL control the filter counter value of SIRC/LPO clock cycles. 00000000b - Filter counter value is 0 00000001b - Filter counter value is 1 ... 11111111b - Filter counter value is 255
2 FLTCLKSEL	**Reset Pin Filter Clock Select** Select LPO_CLK and SIRC_CLK as Reset pin filter clock source 0b - Select LPO_CLK as Reset pin filter clock 1b - Select SIRC_CLK as Reset pin filter clock
1 FLTSTEN	**Reset Pin Filter Enable at Deepsleep and Standby Mode** Enable bit of Reset pin filter in deepsleep and standby mode, this bit should be set if filter need to be used in deepsleep and standby mode. This bit has no effect of reset pin filter in RUN mode 0b - Disable reset pin filter in deepsleep and standby mode 1b - Enable reset pin filter in deepsleep and standby mode if FLTEN is set
0 FLTEN	**Reset Pin filter enable** Enable bit of Reset pin filter, this bit should be set if filter need to be used. 0b - Disable reset pin filter 1b - Enable reset pin filter

Yuntu Microelectronics

Chapter 16

Power Mode

16.1 Introduction

This chapter describes the various power modes supported by the chip.

16.1.1 Active Mode

Allow maximum performance of the chip and use all functions of the chip. It is the default mode when out of reset or power on reset. The on-chip regulators are working at high power mode to supply powers for internal modules. In this mode the chip can work at max frequency 80MHz.

16.1.2 Sleep Mode

Part of the chip is in static state, in Sleep mode, the ARM Cortex-M33 core is stopped, but all the other modules are working same as they are in active mode. All the registers and memory contents are retained. LVD protection is maintained. NVIC remains active to interrupts in Sleep mode.

16.1.3 Deepsleep Mode

The chip is in static state but the high power on chip regulators keep working, all registers and memories retain values, NVIC inactive, peripheries can work with asynchronous clock sources other than bus clocks, asynchronous interrupts from peripheries can wake up chip from Deepsleep mode.

16.1.4 Standby Mode

The chip is in static state, high power regulators are inactive, the chip is powered by low power on chip regulators, all registers and memories retain values, NVIC inactive, peripheries can work with slow asynchronous clocks other than bus clocks, asynchronous interrupts from peripheries can wake up chip from Standby mode.

16.2 Power Modes Table

Table 16.1: Power Mode

Module	Active	Sleep	Deepsleep	Standby
Core module				

Yuntu Microelectronics

Table 16.1 continued from previous page

Module	Active	Sleep	Deepsleep	Standby
NVIC	FF[1]	FF	Static	Static
CPU	FF	Static	Static	Static
SYSTICK	FF	FF	Static	Static
DMA	FF	FF	Static	Static
Power module				
LDO15	FF	FF	FF	FF
Safety and security module				
LVD	FF	FF	FF	FF
WDG	FF	FF	FF	FF
CMU	FF	FF	FF	FF
EMU	FF	FF	Static	Static
CRC	FF	FF	Static	Static
Clocks				
FIRC	Optional FF (FIRC_EN)	Optional FF (FIRC_EN)	Optional FF (FIRC_DSEN)	OFF
FXOSC	Optional FF (FXOSC_EN)	Optional FF (FXOSC_EN)	Optional FF (FXOSC_DSEN)	OFF
SXOSC	Optional FF (SXOSC_EN)	Optional FF (SXOSC_EN)	Optional FF (SXOSC_DSEN)	OFF
SIRC	FF	FF	Optional FF (SIRC_DSEN)	Optional FF (SIRC_STBEN)
LPO	FF	FF	FF	FF
Core clock	FF	OFF	OFF	OFF
Platform clock	FF	FF	OFF	OFF
Fast bus clock	FF	FF	OFF	OFF
Slow bus clock	FF	FF	OFF	OFF
Memory and memory interface				
Flash	FF	FF	Optional FF (DPDEN)[2]	Optional FF (DPDEN)
OCRAM	FF	FF	FF	FF
EFM	FF	FF	Optional FF[3]	Optional FF[4]
Communication interfaces				
FlexCAN	FF	FF	Static	Static
SPI	FF	FF	FF	FF
I2C	FF	FF	FF	FF
UART	FF	FF	FF	FF
Timers				
pTMR	FF	FF	Static	Static
lpTMR	FF	FF	FF	FF
eTMR	FF	FF	Static	Static
TMU	FF	FF	FF	FF
Analog				
ADC	FF	FF	FF	Static
ACMP	FF	FF	FF	Static
Human-Machine interfaces				
IO	FF	FF	FF	FF

1. FF: Full Function
2. When DPDEN=1, Flash enters its deep Powerdown mode
3. Deep Standby Mode if EFM LPEN=1
4. Deep Standby Mode if EFM LPEN=1

Yuntu Microelectronics

16.3 Wake-up Sources

Table 16.2: Wakeup Sources

Wake-up resources	Sleep	Deepsleep (PMCLPEN=0)	Standby (PMCLPEN=1)	Comment
Reset	Y [1]	Y	Y	
NMI	Y	Y	Y	
SysTick	Y			
DMA0	Y			
DMA1	Y			
DMA2	Y			
DMA3	Y			
DMA_Error	Y			
EFM	Y			
EMU	Y	Y	Y	
PCU	Y	Y	Y	
GPIOA/B/C/D/E	Y	Y	Y	
CAN0_ORed	Y			
CAN0_ORed_0_15_MB	Y			
eTMR0_Ch_ORed	Y			
eTMR0_Fault	Y			
eTMR0_Ovf	Y			
eTMR1	Y			
eTMR2	Y			
CAN0_ORed_16_31_MB	Y			
pTMR	Y			
lpTMR0	Y	Y	Y	
RTC	Y	Y	Y	
WDG	Y	Y	Y	
I2C0	Y	Y	Y	
I2C1	Y	Y	Y	
SPI0	Y	Y	Y	
SPI1	Y	Y	Y	
SPI2	Y	Y	Y	
UART0	Y	Y	Y	
UART1	Y	Y	Y	
UART2	Y	Y	Y	
ADC0	Y	Y	Y	
ACMP0	Y	Y	Y	

1. "Y" means specified IRQ can wake up MCU in corresponding power mode.

Yuntu Microelectronics

Part VI

Communication Interfaces

Yuntu Microelectronics

Chapter 17

Flexible Controller Area Network (FlexCAN)

17.1 Chip-specific FlexCAN Information

17.1.1 Instantiation Information

This chip contains one instance of FlexCAN, main configuration is as below:

- support 32 MBs
- support CAN 2.0B and CAN FD
- support External Tick Time
- support RX FIFO Filtering
- support IRMQ
- support DMA
- not support ERF(Enhanced RX FIFO)
- not support PNET
- not support ECC
- not support HR Time stamp
- not support CIA

NOTE: See interrupt table for the vector numbers for the CAN instances and their corresponding interrupt sources.

17.2 Introduction

The FlexCAN module is a communication controller implementing the CAN protocol according to the ISO 11898-1 standard and CAN Specification 2.0 Part B. Figure: “Block diagram of CAN” shows a general block diagram of the FlexCAN, which includes one associated memory for storing message buffers (MBs), Receive Global Mask registers, Receive Individual Mask registers, Receive (Rx) FIFO filters, and Rx FIFO ID filters.

NOTE: The Legacy Rx FIFOs cannot be used in CAN FD mode. The Enhanced Rx FIFO, if implemented, can be used in CAN FD mode.

17.2.1 Block Diagram

The following figure shows the FlexCAN block diagram.

Yuntu Microelectronics

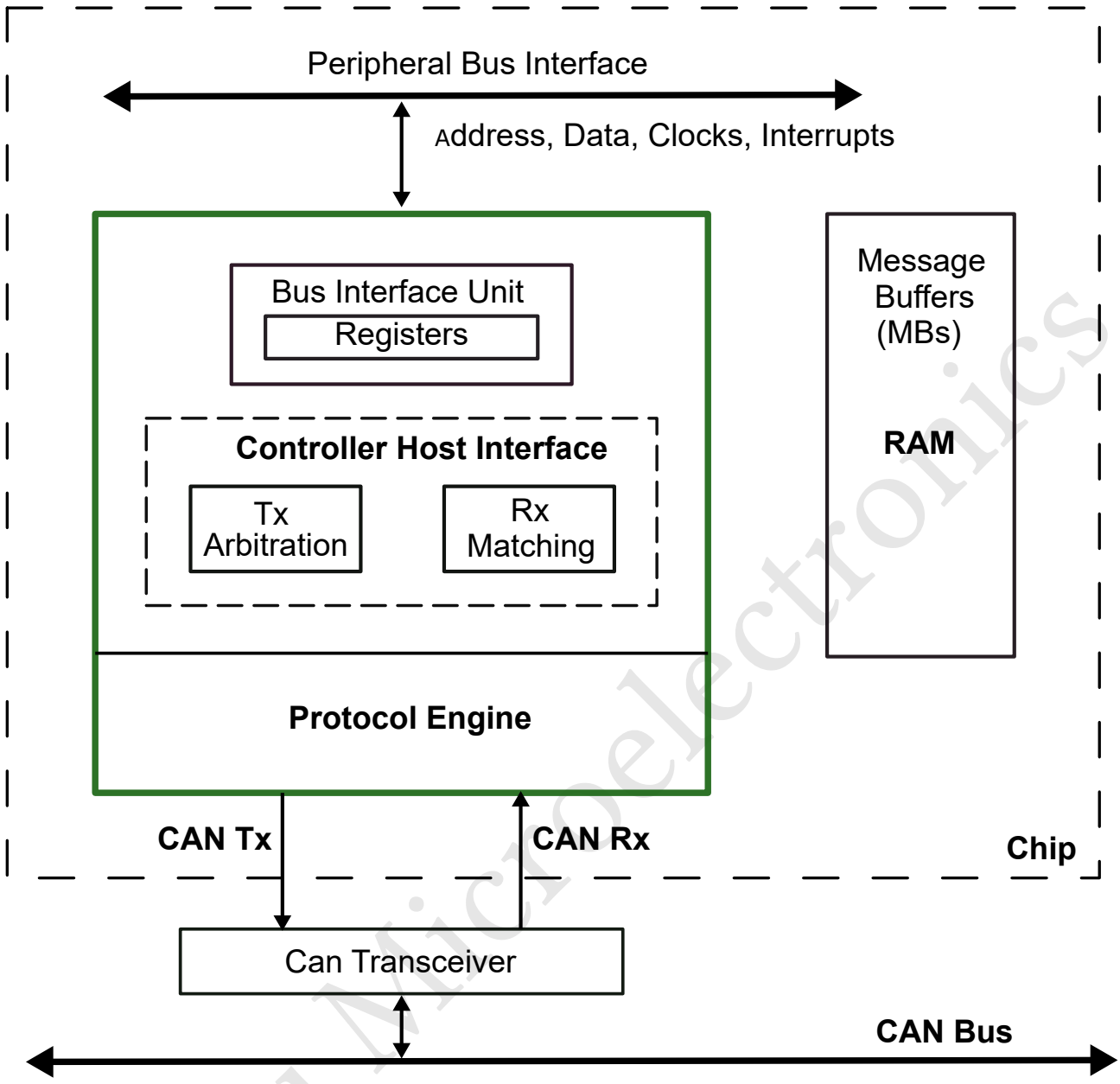


Figure 17.1: Block Diagram of CAN

17.2.2 Overview

The CAN protocol was primarily designed to be used as a vehicle serial data bus, meeting the specific requirements of this field, including:

- Real-time processing
- Reliable operation in the EMI environment of a vehicle
- Cost-effectiveness
- Required bandwidth

The FlexCAN module is a full implementation of the CAN protocol specification, the CAN with Flexible Data rate (CAN FD) protocol, and the CAN 2.0 Part B protocol, supporting both standard and extended message frames and long payloads up to 64 bytes, transferred at rates up to 8 Mbps. The message buffers are stored in an embedded RAM dedicated to the FlexCAN module.

Yuntu Microelectronics

The Protocol Engine (PE) submodule manages the serial communication on the CAN bus:

- Requesting RAM access for receiving and transmitting message frames
- Validating received messages
- Performing error handling
- Detecting CAN FD messages

The Controller Host Interface (CHI) submodule handles message buffer selection for reception and transmission, including arbitration and ID matching algorithms for both CAN FD and non-CAN FD message formats.

The Bus Interface Unit (BIU) submodule controls the access to and from the internal interface bus to establish connection to the CPU and to other blocks. Clocks, address and data buses, interrupt outputs, DMA, and test signals are accessed through the BIU.

17.2.3 Features

Features of the FlexCAN module include:

- Full implementation of the CAN FD protocol and CAN Specification 2.0, Part B
  - Standard data frames
  - Extended data frames
  - Zero to sixty-four bytes data length
  - Programmable bit rate
  - Content-related addressing
- Compliant with the ISO 11898-1 standard
- Silicon-proven implementation passing ISO 16845-1:2016 CAN conformance tests
- Flexible mailboxes configurable to store 0 to 8, 16, 32, or 64 bytes data length
- Each mailbox configurable as receive or transmit, all supporting standard and extended messages
- Individual Rx Mask registers per mailbox
- Full-featured Legacy Rx FIFO with storage capacity for up to 6 CAN frames and automatic internal pointer handling with DMA support
- Full-featured Enhanced Rx FIFO with storage capacity for up to 32 CAN FD frames and automatic internal pointer handling with DMA support
- Transmission abort capability
- Flexible message buffers, totaling 128 message buffers of 8 bytes data length each, configurable as Rx or Tx
- Programmable clock source to the CAN Protocol Engine, either peripheral clock or oscillator clock
- RAM not used by reception or transmission structures can be used as general purpose RAM space
- Listen-Only mode capability
- Programmable Loop-Back mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number, or highest priority
- Time stamp based on 16-bit free-running timer, with an optional external time tick or high-resolution 32-bit on-chip timer

Yuntu Microelectronics

- Global network time, synchronized by a specific message
- Maskable interrupts
- Independence from the transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages
- Low-Power modes, with programmable Wake-Up on bus activity or matching with received frames (Pretended Networking)
- Transceiver Delay Compensation feature when transmitting CAN FD messages at faster data rates
- Remote request frames may be managed automatically or by software
- CAN bit time settings and configuration bits can only be written in Freeze mode
- Tx mailbox status (lowest priority buffer or empty buffer)
- Identifier Acceptance Filter Hit Indicator (IDHIT) register for received frames
- SYNCH bit available in Error in Status 1 register to indicate that the FlexCAN is synchronous with CAN bus
- CRC status for transmitted message
- Legacy Rx FIFO Global Mask register
- Selectable priority between mailboxes and Rx FIFO during matching process
- Powerful Legacy Rx FIFO ID filtering, capable of matching incoming IDs against either 128 extended, 256 standard, or 512 partial (8 bit) IDs, with up to 32 ID Filter Table elements
- Powerful Enhanced Rx FIFO ID filtering, capable of matching incoming IDs against either 64 extended or 128 standard ID filter elements with three filtering schemes: mask + filter, range, and two filters without mask
- 100% backward compatibility with previous FlexCAN version
- Supports detection and correction of errors in memory read accesses. Each byte of FlexCAN memory is associated to 5 parity bits. The error correction mechanism ensures that in this 13-bit word, errors in one bit can be corrected (correctable errors) and errors in 2 bits can be detected but not corrected (non-correctable errors).
- Supports Pretended Networking functionality in Low-Power modes: Doze mode, Deepsleep mode

17.2.4 Mode of Operation

The FlexCAN module has the following functional modes:

- Normal mode (User or Supervisor):

  In Normal mode, the FlexCAN operates receiving and/or transmitting message frames, errors are managed normally, and all CAN protocol functions are enabled.

  User and Supervisor modes differ in the access to some restricted control registers.

- Freeze mode:

  Freeze mode is enabled when the CAN_MCR.FRZ bit is set. If enabled, Freeze mode is entered when CAN_MCR.HALT is set or when Debug mode is requested at chip level and

Yuntu Microelectronics

CAN_MCR.FRZACK is set by the FlexCAN module. In Freeze mode, no transmission or reception of frames is done and synchronicity to the CAN bus is lost.

See "Freeze Mode" for more information.

- Loop-Back mode:

  The FlexCAN enters Loop-Back mode when CAN_CTRL1.LPB is asserted. In LoopBack mode, the FlexCAN performs an internal loop back that can be used for self-test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic '1'). The FlexCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In Loop-Back mode, the FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.

- Listen-Only mode:

  The FlexCAN enters Listen-Only mode when CAN_CTRL1.LOM is asserted. In Listen-Only mode, transmission is disabled, all error counters are frozen, and the FlexCAN operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station are received. If the FlexCAN detects a message that has not been acknowledged, it flags a BIT0 error (without changing the receive error counter), as if it were trying to acknowledge the message.

- CAN FD Active mode:

  In CAN FD Active mode, the FlexCAN is capable of transmitting and receiving all messages formatted according to the CAN FD Protocol and CAN 2.0 Protocol 2.0 in an interleaved fashion. The CPU can set the FlexCAN into CAN FD Active mode by setting the MCR.FDEN when the FlexCAN is in Freeze Mode.

  Below table summarizes the differences in FlexCAN feature availability in CAN FD Active mode (CAN_MCR.FDEN = 1) and Classical CAN mode (CAN_MCR.FDEN = 0).

Table 17.1: CAN Feature Availability in Classic CAN and CAN FD Modes

Feature	Classical CAN	CAN FD
Legacy RX FIFO	YES	NO
Legacy RX FIFO DMA	YES	NO
Enhanced RX FIFO	YES	YES
Enhanced RX FIFO DMA	YES	YES
Pretended Networking	YES	NO

For Low-Power operation, the FlexCAN module has:

- Module Disable mode:

  Module Disable mode is entered when CAN_MCR.MDIS is asserted by the CPU and CAN_MCR.LPMACK is asserted by the FlexCAN. When disabled, the FlexCAN issues a request to disable the clocks to the PE and CHI submodules. Exit from Module Disable mode is done by negating CAN_MCR.MDIS.

  See "Module Disable Mode" for more information.

- Doze mode:

Yuntu Microelectronics

Doze mode is entered when CAN_MCR.DOZE is asserted, Doze mode is requested at the chip level, and CAN_MCR.LPMACK is asserted by the FlexCAN. When in Doze mode, the FlexCAN issues a request to disable the clocks to the PE and the CHI submodules. Exit from Doze mode occurs when CAN_MCR.DOZE is negated, when the chip is removed from Doze mode, or when activity is detected on the CAN bus and the Self Wake Up mechanism is enabled.

See ”Doze Mode” for more information.

- Deepsleep mode:

  Deepsleep mode is entered when Deepsleep mode is requested at chip level and CAN_MCR.LPMACK is asserted by the FlexCAN. When in Deepsleep Mode, the FlexCAN puts itself in an inactive state, then informs the CPU that the clocks can be shut down globally. Exit from Deepsleep mode occurs when the Deepsleep mode request is removed, or when activity is detected on the CAN bus and the Self Wake Up mechanism is enabled.

  See ”Deepsleep Mode” for more information.

- Pretended Networking mode:

  Pretended Networking can be selected to operate together with Doze mode or Deepsleep mode. Before entering Doze mode or Deepsleep mode, CAN_MCR.PNET_EN must be asserted. Once in Low-Power mode, the CHI submodule clocks are shut down and the PE submodule is kept clocked, so that the receive process is still active to filter incoming messages (see ”Receive Process under Pretended Networking Mode”) as defined by the configuration registers (see ”Pretended Networking Control 1 Register (CAN_CTRL1_PN)”. Upon detecting a Wake-Up event, a Wake Up interrupt is issued to the system. When CAN_MCR.PNET_EN is asserted, the CPU must disable the Self Wake Up feature by negating CAN_MCR.SLFWAK.

17.3 Register Definition

The address space occupied by the FlexCAN has 128 bytes for registers starting at the module base address, followed by embedded RAM starting at address offset 0x0080.

Each register is identified by its complete name and the corresponding mnemonic. The access type can be Supervisor (S) or Unrestricted (U). Most of the registers can be configured to have either Supervisor or Unrestricted access by programming the SUPV field in the CAN_MCR register. These registers are identified as S/U in the Access column of Table: ”CAN register access and reset information” .

NOTE: An invalid register access results in a bus error. This includes reading a write-only register, writing a read-only register, or accessing an invalid address.

17.3.1 Register Access and Reset Information

Table 17.2: CAN Register Access and Reset Information

Register	Access Type	Hard Reset	Soft Reset
Module Configuration Register (CAN_MCR)	S	Yes	Yes
Control 1 Register (CAN_CTRL1)	S/U	Yes	No
Free Running Timer Register (CAN_TIMER)	S/U	Yes	Yes
Rx Mailboxes Global Mask Register (CAN_RXMGMASK)	S/U	No	No
Rx 14 Mask Register (CAN_RX14MASK)	S/U	No	No

Yuntu Microelectronics

Table 17.2 continued from previous page

Register	Access Type	Hard Reset	Soft Reset
Rx 15 Mask Register (CAN_RX15MASK)	S/U	No	No
Error Counter Register (CAN_ECR)	S/U	Yes	Yes
Error and Status 1 Register (CAN_ESR1)	S/U	Yes	Yes
Interrupt Masks 2 Register (CAN_IMASK2)	S/U	Yes	Yes
Interrupt Masks 1 Register (CAN_IMASK1)	S/U	Yes	Yes
Interrupt Flags 2 Register (CAN_IFLAG2)	S/U	Yes	Yes
Interrupt Flags 1 Register (CAN_IFLAG1)	S/U	Yes	Yes
Control 2 Register (CAN_CTRL2)	S/U	Yes	No
Error and Status 2 Register (CAN_ESR2)	S/U	Yes	Yes
CRC Register (CAN_CRCR)	S/U	Yes	Yes
Legacy Rx FIFO Global Mask Register (CAN_RXFGMASK)	S/U	No	No
Legacy Rx FIFO Information Register (CAN_RXFIR)	S/U	No	No
CAN Bit Timing Register (CAN_CBT)	S/U	Yes	No
Interrupt Masks 4 Register (CAN_IMASK4)	S/U	Yes	Yes
Interrupt Masks 3 Register (CAN_IMASK3)	S/U	Yes	Yes
Interrupt Flags 4 Register (CAN_IFLAG4)	S/U	Yes	Yes
Interrupt Flags 3 Register (CAN_IFLAG3)	S/U	Yes	Yes
Message buffers	S/U	No	No
Rx Individual Mask Registers (CAN_RXIMRn)	S/U	No	No
Memory Error Control Register (CAN_MECR)	S/U	Yes	Yes
Error Injection Address Register (CAN_ERRIAR)	S/U	Yes	Yes
Error Injection Data Pattern Register (CAN_ERRIDPR)	S/U	Yes	Yes
Error Injection Parity Pattern Register (CAN_ERRIPPR)	S/U	Yes	Yes
Error Report Address Register (CAN_RERRAR)	S/U	Yes	Yes
Error Report Data Register (CAN_RERRDR)	S/U	Yes	Yes
Error Report Syndrome Register (CAN_RERRSYNR)	S/U	Yes	Yes
Error Status Register (CAN_ERRSR)	S/U	Yes	Yes
Pretended Networking Control 1 Register (CAN_CTRL1_PN)	S/U	Yes	Yes
Pretended Networking Control 2 Register (CAN_CTRL2_PN)	S/U	Yes	Yes
Pretended Networking Wake Up Match Register (CAN_WU_MTC)	S/U	Yes	Yes
Pretended Networking ID Filter 1 Register (CAN_FLT_ID1)	S/U	Yes	Yes
Pretended Networking DLC Filter Register (CAN_FLT_DLC)	S/U	Yes	Yes
Pretended Networking Payload Low Filter 1 Register (CAN_PL1_LO)	S/U	Yes	Yes

Yuntu Microelectronics

Table 17.2 continued from previous page

Register	Access Type	Hard Reset	Soft Reset
Pretended Networking Payload High Filter 1 Register (CAN_PL1_HI)	S/U	Yes	Yes
Pretended Networking ID Filter 2 Register / ID Mask Register(CAN_FLT_ID2_IDMASK)	S/U	Yes	Yes
Pretended Networking Payload Low Filter 2 Register / Payload Low Mask Register (CAN_PL2_PLMASK_LO)	S/U	Yes	Yes
Pretended Networking Payload High Filter 2 High Order Bits / Payload High Mask Register (CAN_PL2_PLMASK_HI)	S/U	Yes	Yes
Pretended Networking Wake Up Message Buffer 0 registers (CAN_WMB0_*)	S/U	Yes	No
Pretended Networking Wake Up Message Buffer 1 registers (CAN_WMB1_*)	S/U	Yes	No
Pretended Networking Wake Up Message Buffer 2 registers (CAN_WMB2_*)	S/U	Yes	No
Pretended Networking Wake Up Message Buffer 3 registers (CAN_WMB3_*)	S/U	Yes	No
Enhanced CAN Bit Timing Prescalers Register (CAN_EPRS)	S/U	Yes	No
Enhanced Nominal CAN Bit Timing Register (CAN_ENCBT)	S/U	Yes	No
Enhanced Data Phase CAN Bit Timing Register (CAN_EDCBT)	S/U	Yes	No
Enhanced Transceiver Delay Compensation Register (CAN_ETDC)	S/U	Yes	No
CAN FD Control Register (CAN_FDCTRL)	S/U	Yes	No
CAN FD Bit Timing Register (CAN_FDCBT)	S/U	Yes	No
CAN FD CRC Register (CAN_FDCRC)	S/U	Yes	Yes
Enhanced Rx FIFO Control Register (CAN_ERFCR)	S/U	Yes	Yes
Enhanced Rx FIFO Interrupt Enable Register (CAN_ERFIER)	S/U	Yes	Yes
Enhanced Rx FIFO Status Register (CAN_ERFSR)	S/U	Yes	Yes
High Resolution Time Stamp Registers (CAN_HR_TIME_STAMPn)	S/U	No	No
Enhanced Rx FIFO Filter Element Registers (CAN_ERFFELn)	S/U	No	No

Yuntu Microelectronics

17.3.2 Memory Mapping

The FlexCAN module can store CAN messages for transmission and reception using message buffers and Rx FIFO structures.

The address range from offset 0x80 to 0x87F allocates the one-hundred-twenty-eight 128-bit message buffers (MBs).

The address range from offset 0x1000 to 0x17FF allocates an extra message buffer space that can be used

for CAN FD operation.

The message buffer memory map is described in "FlexCAN Message Buffer Memory Map".

The address range from offset 0x2000 to 0x204C allocates the Enhanced Rx FIFO output. The address range from offset 0x2050 to 0x29FC allocates the rest of Enhanced RX FIFO 31 elements. The memory map for the Enhanced Rx FIFO is described in "Enhanced Rx FIFO Structure".

17.3.3 CAN Memory Map

CAN0 base address 40024000h

Table 17.3: CAN Memory Map

Offset	Register	Width	Access	Reset Value
0h	Module Configuration Register (MCR)	32	RW	D890000Fh
4h	Module Configuration Register 1 (CTRL1)	32	RW	00000000h
8h	Free Running Timer Register (TIMER)	32	RW	00000000h
10h	Rx Mailboxes Global Mask Register (RXMGMASK)	32	RW	00000000h
14h	Rx 14 Mask Register (RX14MASK)	32	RW	00000000h
18h	Rx 15 Mask Register (RX15MASK)	32	RW	00000000h
1ch	Error Counter Register (ECR)	32	RW	00000000h
20h	Error and Status Register (ESR1)	32	RW	00000000h
24h	Interrupt Mask 2 Register (IMASK2)	32	RW	00000000h
28h	Interrupt Mask 1 Register (IMASK1)	32	RW	00000000h
2ch	Interrupt Flag 2 Register (IFLAG2)	32	RW	00000000h
30h	Interrupt Flag 1 Register (IFLAG1)	32	RW	00000000h
34h	Control 2 Register (CTRL2)	32	RW	00000000h
38h	Error and Status 2 Register (ESR2)	32	RW	00000000h
44h	CRC Register (CRCR)	32	RW	00000000h
48h	Legacy Rx FIFO Global Mask Register (RXFGMASK)	32	RW	00000000h
4ch	Legacy Rx FIFO Information Register (RXFIR)	32	RW	00000000h
50h	Bit Timing Register (CBT)	32	RW	00000000h
68h	Interrupt Mask 4 Register (IMASK4)	32	RW	00000000h
6ch	Interrupt Mask 3 Register (IMASK3)	32	RW	00000000h
70h	Interrupt Flags 4 Register (IFLAG4)	32	RW	00000000h
74h	Interrupt Flags 3 Register (IFLAG3)	32	RW	00000000h
80h	Embeded RAM (RAM0)	32	RW	00000000h
84h	Embeded RAM (RAM1)	32	RW	00000000h
88h	Embeded RAM (RAM2)	32	RW	00000000h
8ch	Embeded RAM (RAM3)	32	RW	00000000h
90h	Embeded RAM (RAM4)	32	RW	00000000h
94h	Embeded RAM (RAM5)	32	RW	00000000h
98h	Embeded RAM (RAM6)	32	RW	00000000h
9ch	Embeded RAM (RAM7)	32	RW	00000000h
a0h	Embeded RAM (RAM8)	32	RW	00000000h
a4h	Embeded RAM (RAM9)	32	RW	00000000h
a8h	Embeded RAM (RAM10)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
ach	Embeded RAM (RAM11)	32	RW	00000000h
b0h	Embeded RAM (RAM12)	32	RW	00000000h
b4h	Embeded RAM (RAM13)	32	RW	00000000h
b8h	Embeded RAM (RAM14)	32	RW	00000000h
bch	Embeded RAM (RAM15)	32	RW	00000000h
c0h	Embeded RAM (RAM16)	32	RW	00000000h
c4h	Embeded RAM (RAM17)	32	RW	00000000h
c8h	Embeded RAM (RAM18)	32	RW	00000000h
cch	Embeded RAM (RAM19)	32	RW	00000000h
d0h	Embeded RAM (RAM20)	32	RW	00000000h
d4h	Embeded RAM (RAM21)	32	RW	00000000h
d8h	Embeded RAM (RAM22)	32	RW	00000000h
dch	Embeded RAM (RAM23)	32	RW	00000000h
e0h	Embeded RAM (RAM24)	32	RW	00000000h
e4h	Embeded RAM (RAM25)	32	RW	00000000h
e8h	Embeded RAM (RAM26)	32	RW	00000000h
ech	Embeded RAM (RAM27)	32	RW	00000000h
f0h	Embeded RAM (RAM28)	32	RW	00000000h
f4h	Embeded RAM (RAM29)	32	RW	00000000h
f8h	Embeded RAM (RAM30)	32	RW	00000000h
fch	Embeded RAM (RAM31)	32	RW	00000000h
100h	Embeded RAM (RAM32)	32	RW	00000000h
104h	Embeded RAM (RAM33)	32	RW	00000000h
108h	Embeded RAM (RAM34)	32	RW	00000000h
10ch	Embeded RAM (RAM35)	32	RW	00000000h
110h	Embeded RAM (RAM36)	32	RW	00000000h
114h	Embeded RAM (RAM37)	32	RW	00000000h
118h	Embeded RAM (RAM38)	32	RW	00000000h
11ch	Embeded RAM (RAM39)	32	RW	00000000h
120h	Embeded RAM (RAM40)	32	RW	00000000h
124h	Embeded RAM (RAM41)	32	RW	00000000h
128h	Embeded RAM (RAM42)	32	RW	00000000h
12ch	Embeded RAM (RAM43)	32	RW	00000000h
130h	Embeded RAM (RAM44)	32	RW	00000000h
134h	Embeded RAM (RAM45)	32	RW	00000000h
138h	Embeded RAM (RAM46)	32	RW	00000000h
13ch	Embeded RAM (RAM47)	32	RW	00000000h
140h	Embeded RAM (RAM48)	32	RW	00000000h
144h	Embeded RAM (RAM49)	32	RW	00000000h
148h	Embeded RAM (RAM50)	32	RW	00000000h
14ch	Embeded RAM (RAM51)	32	RW	00000000h
150h	Embeded RAM (RAM52)	32	RW	00000000h
154h	Embeded RAM (RAM53)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
158h	Embeded RAM (RAM54)	32	RW	00000000h
15ch	Embeded RAM (RAM55)	32	RW	00000000h
160h	Embeded RAM (RAM56)	32	RW	00000000h
164h	Embeded RAM (RAM57)	32	RW	00000000h
168h	Embeded RAM (RAM58)	32	RW	00000000h
16ch	Embeded RAM (RAM59)	32	RW	00000000h
170h	Embeded RAM (RAM60)	32	RW	00000000h
174h	Embeded RAM (RAM61)	32	RW	00000000h
178h	Embeded RAM (RAM62)	32	RW	00000000h
17ch	Embeded RAM (RAM63)	32	RW	00000000h
180h	Embeded RAM (RAM64)	32	RW	00000000h
184h	Embeded RAM (RAM65)	32	RW	00000000h
188h	Embeded RAM (RAM66)	32	RW	00000000h
18ch	Embeded RAM (RAM67)	32	RW	00000000h
190h	Embeded RAM (RAM68)	32	RW	00000000h
194h	Embeded RAM (RAM69)	32	RW	00000000h
198h	Embeded RAM (RAM70)	32	RW	00000000h
19ch	Embeded RAM (RAM71)	32	RW	00000000h
1a0h	Embeded RAM (RAM72)	32	RW	00000000h
1a4h	Embeded RAM (RAM73)	32	RW	00000000h
1a8h	Embeded RAM (RAM74)	32	RW	00000000h
1ach	Embeded RAM (RAM75)	32	RW	00000000h
1b0h	Embeded RAM (RAM76)	32	RW	00000000h
1b4h	Embeded RAM (RAM77)	32	RW	00000000h
1b8h	Embeded RAM (RAM78)	32	RW	00000000h
1bch	Embeded RAM (RAM79)	32	RW	00000000h
1c0h	Embeded RAM (RAM80)	32	RW	00000000h
1c4h	Embeded RAM (RAM81)	32	RW	00000000h
1c8h	Embeded RAM (RAM82)	32	RW	00000000h
1cch	Embeded RAM (RAM83)	32	RW	00000000h
1d0h	Embeded RAM (RAM84)	32	RW	00000000h
1d4h	Embeded RAM (RAM85)	32	RW	00000000h
1d8h	Embeded RAM (RAM86)	32	RW	00000000h
1dch	Embeded RAM (RAM87)	32	RW	00000000h
1e0h	Embeded RAM (RAM88)	32	RW	00000000h
1e4h	Embeded RAM (RAM89)	32	RW	00000000h
1e8h	Embeded RAM (RAM90)	32	RW	00000000h
1ech	Embeded RAM (RAM91)	32	RW	00000000h
1f0h	Embeded RAM (RAM92)	32	RW	00000000h
1f4h	Embeded RAM (RAM93)	32	RW	00000000h
1f8h	Embeded RAM (RAM94)	32	RW	00000000h
1fch	Embeded RAM (RAM95)	32	RW	00000000h
200h	Embeded RAM (RAM96)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
204h	Embeded RAM (RAM97)	32	RW	00000000h
208h	Embeded RAM (RAM98)	32	RW	00000000h
20ch	Embeded RAM (RAM99)	32	RW	00000000h
210h	Embeded RAM (RAM100)	32	RW	00000000h
214h	Embeded RAM (RAM101)	32	RW	00000000h
218h	Embeded RAM (RAM102)	32	RW	00000000h
21ch	Embeded RAM (RAM103)	32	RW	00000000h
220h	Embeded RAM (RAM104)	32	RW	00000000h
224h	Embeded RAM (RAM105)	32	RW	00000000h
228h	Embeded RAM (RAM106)	32	RW	00000000h
22ch	Embeded RAM (RAM107)	32	RW	00000000h
230h	Embeded RAM (RAM108)	32	RW	00000000h
234h	Embeded RAM (RAM109)	32	RW	00000000h
238h	Embeded RAM (RAM110)	32	RW	00000000h
23ch	Embeded RAM (RAM111)	32	RW	00000000h
240h	Embeded RAM (RAM112)	32	RW	00000000h
244h	Embeded RAM (RAM113)	32	RW	00000000h
248h	Embeded RAM (RAM114)	32	RW	00000000h
24ch	Embeded RAM (RAM115)	32	RW	00000000h
250h	Embeded RAM (RAM116)	32	RW	00000000h
254h	Embeded RAM (RAM117)	32	RW	00000000h
258h	Embeded RAM (RAM118)	32	RW	00000000h
25ch	Embeded RAM (RAM119)	32	RW	00000000h
260h	Embeded RAM (RAM120)	32	RW	00000000h
264h	Embeded RAM (RAM121)	32	RW	00000000h
268h	Embeded RAM (RAM122)	32	RW	00000000h
26ch	Embeded RAM (RAM123)	32	RW	00000000h
270h	Embeded RAM (RAM124)	32	RW	00000000h
274h	Embeded RAM (RAM125)	32	RW	00000000h
278h	Embeded RAM (RAM126)	32	RW	00000000h
27ch	Embeded RAM (RAM127)	32	RW	00000000h
280h	Embeded RAM (RAM128)	32	RW	00000000h
284h	Embeded RAM (RAM129)	32	RW	00000000h
288h	Embeded RAM (RAM130)	32	RW	00000000h
28ch	Embeded RAM (RAM131)	32	RW	00000000h
290h	Embeded RAM (RAM132)	32	RW	00000000h
294h	Embeded RAM (RAM133)	32	RW	00000000h
298h	Embeded RAM (RAM134)	32	RW	00000000h
29ch	Embeded RAM (RAM135)	32	RW	00000000h
2a0h	Embeded RAM (RAM136)	32	RW	00000000h
2a4h	Embeded RAM (RAM137)	32	RW	00000000h
2a8h	Embeded RAM (RAM138)	32	RW	00000000h
2ach	Embeded RAM (RAM139)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
2b0h	Embeded RAM (RAM140)	32	RW	00000000h
2b4h	Embeded RAM (RAM141)	32	RW	00000000h
2b8h	Embeded RAM (RAM142)	32	RW	00000000h
2bch	Embeded RAM (RAM143)	32	RW	00000000h
2c0h	Embeded RAM (RAM144)	32	RW	00000000h
2c4h	Embeded RAM (RAM145)	32	RW	00000000h
2c8h	Embeded RAM (RAM146)	32	RW	00000000h
2cch	Embeded RAM (RAM147)	32	RW	00000000h
2d0h	Embeded RAM (RAM148)	32	RW	00000000h
2d4h	Embeded RAM (RAM149)	32	RW	00000000h
2d8h	Embeded RAM (RAM150)	32	RW	00000000h
2dch	Embeded RAM (RAM151)	32	RW	00000000h
2e0h	Embeded RAM (RAM152)	32	RW	00000000h
2e4h	Embeded RAM (RAM153)	32	RW	00000000h
2e8h	Embeded RAM (RAM154)	32	RW	00000000h
2ech	Embeded RAM (RAM155)	32	RW	00000000h
2f0h	Embeded RAM (RAM156)	32	RW	00000000h
2f4h	Embeded RAM (RAM157)	32	RW	00000000h
2f8h	Embeded RAM (RAM158)	32	RW	00000000h
2fch	Embeded RAM (RAM159)	32	RW	00000000h
300h	Embeded RAM (RAM160)	32	RW	00000000h
304h	Embeded RAM (RAM161)	32	RW	00000000h
308h	Embeded RAM (RAM162)	32	RW	00000000h
30ch	Embeded RAM (RAM163)	32	RW	00000000h
310h	Embeded RAM (RAM164)	32	RW	00000000h
314h	Embeded RAM (RAM165)	32	RW	00000000h
318h	Embeded RAM (RAM166)	32	RW	00000000h
31ch	Embeded RAM (RAM167)	32	RW	00000000h
320h	Embeded RAM (RAM168)	32	RW	00000000h
324h	Embeded RAM (RAM169)	32	RW	00000000h
328h	Embeded RAM (RAM170)	32	RW	00000000h
32ch	Embeded RAM (RAM171)	32	RW	00000000h
330h	Embeded RAM (RAM172)	32	RW	00000000h
334h	Embeded RAM (RAM173)	32	RW	00000000h
338h	Embeded RAM (RAM174)	32	RW	00000000h
33ch	Embeded RAM (RAM175)	32	RW	00000000h
340h	Embeded RAM (RAM176)	32	RW	00000000h
344h	Embeded RAM (RAM177)	32	RW	00000000h
348h	Embeded RAM (RAM178)	32	RW	00000000h
34ch	Embeded RAM (RAM179)	32	RW	00000000h
350h	Embeded RAM (RAM180)	32	RW	00000000h
354h	Embeded RAM (RAM181)	32	RW	00000000h
358h	Embeded RAM (RAM182)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
35ch	Embeded RAM (RAM183)	32	RW	00000000h
360h	Embeded RAM (RAM184)	32	RW	00000000h
364h	Embeded RAM (RAM185)	32	RW	00000000h
368h	Embeded RAM (RAM186)	32	RW	00000000h
36ch	Embeded RAM (RAM187)	32	RW	00000000h
370h	Embeded RAM (RAM188)	32	RW	00000000h
374h	Embeded RAM (RAM189)	32	RW	00000000h
378h	Embeded RAM (RAM190)	32	RW	00000000h
37ch	Embeded RAM (RAM191)	32	RW	00000000h
380h	Embeded RAM (RAM192)	32	RW	00000000h
384h	Embeded RAM (RAM193)	32	RW	00000000h
388h	Embeded RAM (RAM194)	32	RW	00000000h
38ch	Embeded RAM (RAM195)	32	RW	00000000h
390h	Embeded RAM (RAM196)	32	RW	00000000h
394h	Embeded RAM (RAM197)	32	RW	00000000h
398h	Embeded RAM (RAM198)	32	RW	00000000h
39ch	Embeded RAM (RAM199)	32	RW	00000000h
3a0h	Embeded RAM (RAM200)	32	RW	00000000h
3a4h	Embeded RAM (RAM201)	32	RW	00000000h
3a8h	Embeded RAM (RAM202)	32	RW	00000000h
3ach	Embeded RAM (RAM203)	32	RW	00000000h
3b0h	Embeded RAM (RAM204)	32	RW	00000000h
3b4h	Embeded RAM (RAM205)	32	RW	00000000h
3b8h	Embeded RAM (RAM206)	32	RW	00000000h
3bch	Embeded RAM (RAM207)	32	RW	00000000h
3c0h	Embeded RAM (RAM208)	32	RW	00000000h
3c4h	Embeded RAM (RAM209)	32	RW	00000000h
3c8h	Embeded RAM (RAM210)	32	RW	00000000h
3cch	Embeded RAM (RAM211)	32	RW	00000000h
3d0h	Embeded RAM (RAM212)	32	RW	00000000h
3d4h	Embeded RAM (RAM213)	32	RW	00000000h
3d8h	Embeded RAM (RAM214)	32	RW	00000000h
3dch	Embeded RAM (RAM215)	32	RW	00000000h
3e0h	Embeded RAM (RAM216)	32	RW	00000000h
3e4h	Embeded RAM (RAM217)	32	RW	00000000h
3e8h	Embeded RAM (RAM218)	32	RW	00000000h
3ech	Embeded RAM (RAM219)	32	RW	00000000h
3f0h	Embeded RAM (RAM220)	32	RW	00000000h
3f4h	Embeded RAM (RAM221)	32	RW	00000000h
3f8h	Embeded RAM (RAM222)	32	RW	00000000h
3fch	Embeded RAM (RAM223)	32	RW	00000000h
400h	Embeded RAM (RAM224)	32	RW	00000000h
404h	Embeded RAM (RAM225)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
408h	Embeded RAM (RAM226)	32	RW	00000000h
40ch	Embeded RAM (RAM227)	32	RW	00000000h
410h	Embeded RAM (RAM228)	32	RW	00000000h
414h	Embeded RAM (RAM229)	32	RW	00000000h
418h	Embeded RAM (RAM230)	32	RW	00000000h
41ch	Embeded RAM (RAM231)	32	RW	00000000h
420h	Embeded RAM (RAM232)	32	RW	00000000h
424h	Embeded RAM (RAM233)	32	RW	00000000h
428h	Embeded RAM (RAM234)	32	RW	00000000h
42ch	Embeded RAM (RAM235)	32	RW	00000000h
430h	Embeded RAM (RAM236)	32	RW	00000000h
434h	Embeded RAM (RAM237)	32	RW	00000000h
438h	Embeded RAM (RAM238)	32	RW	00000000h
43ch	Embeded RAM (RAM239)	32	RW	00000000h
440h	Embeded RAM (RAM240)	32	RW	00000000h
444h	Embeded RAM (RAM241)	32	RW	00000000h
448h	Embeded RAM (RAM242)	32	RW	00000000h
44ch	Embeded RAM (RAM243)	32	RW	00000000h
450h	Embeded RAM (RAM244)	32	RW	00000000h
454h	Embeded RAM (RAM245)	32	RW	00000000h
458h	Embeded RAM (RAM246)	32	RW	00000000h
45ch	Embeded RAM (RAM247)	32	RW	00000000h
460h	Embeded RAM (RAM248)	32	RW	00000000h
464h	Embeded RAM (RAM249)	32	RW	00000000h
468h	Embeded RAM (RAM250)	32	RW	00000000h
46ch	Embeded RAM (RAM251)	32	RW	00000000h
470h	Embeded RAM (RAM252)	32	RW	00000000h
474h	Embeded RAM (RAM253)	32	RW	00000000h
478h	Embeded RAM (RAM254)	32	RW	00000000h
47ch	Embeded RAM (RAM255)	32	RW	00000000h
480h	Embeded RAM (RAM256)	32	RW	00000000h
484h	Embeded RAM (RAM257)	32	RW	00000000h
488h	Embeded RAM (RAM258)	32	RW	00000000h
48ch	Embeded RAM (RAM259)	32	RW	00000000h
490h	Embeded RAM (RAM260)	32	RW	00000000h
494h	Embeded RAM (RAM261)	32	RW	00000000h
498h	Embeded RAM (RAM262)	32	RW	00000000h
49ch	Embeded RAM (RAM263)	32	RW	00000000h
4a0h	Embeded RAM (RAM264)	32	RW	00000000h
4a4h	Embeded RAM (RAM265)	32	RW	00000000h
4a8h	Embeded RAM (RAM266)	32	RW	00000000h
4ach	Embeded RAM (RAM267)	32	RW	00000000h
4b0h	Embeded RAM (RAM268)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
4b4h	Embeded RAM (RAM269)	32	RW	00000000h
4b8h	Embeded RAM (RAM270)	32	RW	00000000h
4bch	Embeded RAM (RAM271)	32	RW	00000000h
4c0h	Embeded RAM (RAM272)	32	RW	00000000h
4c4h	Embeded RAM (RAM273)	32	RW	00000000h
4c8h	Embeded RAM (RAM274)	32	RW	00000000h
4cch	Embeded RAM (RAM275)	32	RW	00000000h
4d0h	Embeded RAM (RAM276)	32	RW	00000000h
4d4h	Embeded RAM (RAM277)	32	RW	00000000h
4d8h	Embeded RAM (RAM278)	32	RW	00000000h
4dch	Embeded RAM (RAM279)	32	RW	00000000h
4e0h	Embeded RAM (RAM280)	32	RW	00000000h
4e4h	Embeded RAM (RAM281)	32	RW	00000000h
4e8h	Embeded RAM (RAM282)	32	RW	00000000h
4ech	Embeded RAM (RAM283)	32	RW	00000000h
4f0h	Embeded RAM (RAM284)	32	RW	00000000h
4f4h	Embeded RAM (RAM285)	32	RW	00000000h
4f8h	Embeded RAM (RAM286)	32	RW	00000000h
4fch	Embeded RAM (RAM287)	32	RW	00000000h
500h	Embeded RAM (RAM288)	32	RW	00000000h
504h	Embeded RAM (RAM289)	32	RW	00000000h
508h	Embeded RAM (RAM290)	32	RW	00000000h
50ch	Embeded RAM (RAM291)	32	RW	00000000h
510h	Embeded RAM (RAM292)	32	RW	00000000h
514h	Embeded RAM (RAM293)	32	RW	00000000h
518h	Embeded RAM (RAM294)	32	RW	00000000h
51ch	Embeded RAM (RAM295)	32	RW	00000000h
520h	Embeded RAM (RAM296)	32	RW	00000000h
524h	Embeded RAM (RAM297)	32	RW	00000000h
528h	Embeded RAM (RAM298)	32	RW	00000000h
52ch	Embeded RAM (RAM299)	32	RW	00000000h
530h	Embeded RAM (RAM300)	32	RW	00000000h
534h	Embeded RAM (RAM301)	32	RW	00000000h
538h	Embeded RAM (RAM302)	32	RW	00000000h
53ch	Embeded RAM (RAM303)	32	RW	00000000h
540h	Embeded RAM (RAM304)	32	RW	00000000h
544h	Embeded RAM (RAM305)	32	RW	00000000h
548h	Embeded RAM (RAM306)	32	RW	00000000h
54ch	Embeded RAM (RAM307)	32	RW	00000000h
550h	Embeded RAM (RAM308)	32	RW	00000000h
554h	Embeded RAM (RAM309)	32	RW	00000000h
558h	Embeded RAM (RAM310)	32	RW	00000000h
55ch	Embeded RAM (RAM311)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
560h	Embeded RAM (RAM312)	32	RW	00000000h
564h	Embeded RAM (RAM313)	32	RW	00000000h
568h	Embeded RAM (RAM314)	32	RW	00000000h
56ch	Embeded RAM (RAM315)	32	RW	00000000h
570h	Embeded RAM (RAM316)	32	RW	00000000h
574h	Embeded RAM (RAM317)	32	RW	00000000h
578h	Embeded RAM (RAM318)	32	RW	00000000h
57ch	Embeded RAM (RAM319)	32	RW	00000000h
580h	Embeded RAM (RAM320)	32	RW	00000000h
584h	Embeded RAM (RAM321)	32	RW	00000000h
588h	Embeded RAM (RAM322)	32	RW	00000000h
58ch	Embeded RAM (RAM323)	32	RW	00000000h
590h	Embeded RAM (RAM324)	32	RW	00000000h
594h	Embeded RAM (RAM325)	32	RW	00000000h
598h	Embeded RAM (RAM326)	32	RW	00000000h
59ch	Embeded RAM (RAM327)	32	RW	00000000h
5a0h	Embeded RAM (RAM328)	32	RW	00000000h
5a4h	Embeded RAM (RAM329)	32	RW	00000000h
5a8h	Embeded RAM (RAM330)	32	RW	00000000h
5ach	Embeded RAM (RAM331)	32	RW	00000000h
5b0h	Embeded RAM (RAM332)	32	RW	00000000h
5b4h	Embeded RAM (RAM333)	32	RW	00000000h
5b8h	Embeded RAM (RAM334)	32	RW	00000000h
5bch	Embeded RAM (RAM335)	32	RW	00000000h
5c0h	Embeded RAM (RAM336)	32	RW	00000000h
5c4h	Embeded RAM (RAM337)	32	RW	00000000h
5c8h	Embeded RAM (RAM338)	32	RW	00000000h
5cch	Embeded RAM (RAM339)	32	RW	00000000h
5d0h	Embeded RAM (RAM340)	32	RW	00000000h
5d4h	Embeded RAM (RAM341)	32	RW	00000000h
5d8h	Embeded RAM (RAM342)	32	RW	00000000h
5dch	Embeded RAM (RAM343)	32	RW	00000000h
5e0h	Embeded RAM (RAM344)	32	RW	00000000h
5e4h	Embeded RAM (RAM345)	32	RW	00000000h
5e8h	Embeded RAM (RAM346)	32	RW	00000000h
5ech	Embeded RAM (RAM347)	32	RW	00000000h
5f0h	Embeded RAM (RAM348)	32	RW	00000000h
5f4h	Embeded RAM (RAM349)	32	RW	00000000h
5f8h	Embeded RAM (RAM350)	32	RW	00000000h
5fch	Embeded RAM (RAM351)	32	RW	00000000h
600h	Embeded RAM (RAM352)	32	RW	00000000h
604h	Embeded RAM (RAM353)	32	RW	00000000h
608h	Embeded RAM (RAM354)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
60ch	Embeded RAM (RAM355)	32	RW	00000000h
610h	Embeded RAM (RAM356)	32	RW	00000000h
614h	Embeded RAM (RAM357)	32	RW	00000000h
618h	Embeded RAM (RAM358)	32	RW	00000000h
61ch	Embeded RAM (RAM359)	32	RW	00000000h
620h	Embeded RAM (RAM360)	32	RW	00000000h
624h	Embeded RAM (RAM361)	32	RW	00000000h
628h	Embeded RAM (RAM362)	32	RW	00000000h
62ch	Embeded RAM (RAM363)	32	RW	00000000h
630h	Embeded RAM (RAM364)	32	RW	00000000h
634h	Embeded RAM (RAM365)	32	RW	00000000h
638h	Embeded RAM (RAM366)	32	RW	00000000h
63ch	Embeded RAM (RAM367)	32	RW	00000000h
640h	Embeded RAM (RAM368)	32	RW	00000000h
644h	Embeded RAM (RAM369)	32	RW	00000000h
648h	Embeded RAM (RAM370)	32	RW	00000000h
64ch	Embeded RAM (RAM371)	32	RW	00000000h
650h	Embeded RAM (RAM372)	32	RW	00000000h
654h	Embeded RAM (RAM373)	32	RW	00000000h
658h	Embeded RAM (RAM374)	32	RW	00000000h
65ch	Embeded RAM (RAM375)	32	RW	00000000h
660h	Embeded RAM (RAM376)	32	RW	00000000h
664h	Embeded RAM (RAM377)	32	RW	00000000h
668h	Embeded RAM (RAM378)	32	RW	00000000h
66ch	Embeded RAM (RAM379)	32	RW	00000000h
670h	Embeded RAM (RAM380)	32	RW	00000000h
674h	Embeded RAM (RAM381)	32	RW	00000000h
678h	Embeded RAM (RAM382)	32	RW	00000000h
67ch	Embeded RAM (RAM383)	32	RW	00000000h
680h	Embeded RAM (RAM384)	32	RW	00000000h
684h	Embeded RAM (RAM385)	32	RW	00000000h
688h	Embeded RAM (RAM386)	32	RW	00000000h
68ch	Embeded RAM (RAM387)	32	RW	00000000h
690h	Embeded RAM (RAM388)	32	RW	00000000h
694h	Embeded RAM (RAM389)	32	RW	00000000h
698h	Embeded RAM (RAM390)	32	RW	00000000h
69ch	Embeded RAM (RAM391)	32	RW	00000000h
6a0h	Embeded RAM (RAM392)	32	RW	00000000h
6a4h	Embeded RAM (RAM393)	32	RW	00000000h
6a8h	Embeded RAM (RAM394)	32	RW	00000000h
6ach	Embeded RAM (RAM395)	32	RW	00000000h
6b0h	Embeded RAM (RAM396)	32	RW	00000000h
6b4h	Embeded RAM (RAM397)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
6b8h	Embeded RAM (RAM398)	32	RW	00000000h
6bch	Embeded RAM (RAM399)	32	RW	00000000h
6c0h	Embeded RAM (RAM400)	32	RW	00000000h
6c4h	Embeded RAM (RAM401)	32	RW	00000000h
6c8h	Embeded RAM (RAM402)	32	RW	00000000h
6cch	Embeded RAM (RAM403)	32	RW	00000000h
6d0h	Embeded RAM (RAM404)	32	RW	00000000h
6d4h	Embeded RAM (RAM405)	32	RW	00000000h
6d8h	Embeded RAM (RAM406)	32	RW	00000000h
6dch	Embeded RAM (RAM407)	32	RW	00000000h
6e0h	Embeded RAM (RAM408)	32	RW	00000000h
6e4h	Embeded RAM (RAM409)	32	RW	00000000h
6e8h	Embeded RAM (RAM410)	32	RW	00000000h
6ech	Embeded RAM (RAM411)	32	RW	00000000h
6f0h	Embeded RAM (RAM412)	32	RW	00000000h
6f4h	Embeded RAM (RAM413)	32	RW	00000000h
6f8h	Embeded RAM (RAM414)	32	RW	00000000h
6fch	Embeded RAM (RAM415)	32	RW	00000000h
700h	Embeded RAM (RAM416)	32	RW	00000000h
704h	Embeded RAM (RAM417)	32	RW	00000000h
708h	Embeded RAM (RAM418)	32	RW	00000000h
70ch	Embeded RAM (RAM419)	32	RW	00000000h
710h	Embeded RAM (RAM420)	32	RW	00000000h
714h	Embeded RAM (RAM421)	32	RW	00000000h
718h	Embeded RAM (RAM422)	32	RW	00000000h
71ch	Embeded RAM (RAM423)	32	RW	00000000h
720h	Embeded RAM (RAM424)	32	RW	00000000h
724h	Embeded RAM (RAM425)	32	RW	00000000h
728h	Embeded RAM (RAM426)	32	RW	00000000h
72ch	Embeded RAM (RAM427)	32	RW	00000000h
730h	Embeded RAM (RAM428)	32	RW	00000000h
734h	Embeded RAM (RAM429)	32	RW	00000000h
738h	Embeded RAM (RAM430)	32	RW	00000000h
73ch	Embeded RAM (RAM431)	32	RW	00000000h
740h	Embeded RAM (RAM432)	32	RW	00000000h
744h	Embeded RAM (RAM433)	32	RW	00000000h
748h	Embeded RAM (RAM434)	32	RW	00000000h
74ch	Embeded RAM (RAM435)	32	RW	00000000h
750h	Embeded RAM (RAM436)	32	RW	00000000h
754h	Embeded RAM (RAM437)	32	RW	00000000h
758h	Embeded RAM (RAM438)	32	RW	00000000h
75ch	Embeded RAM (RAM439)	32	RW	00000000h
760h	Embeded RAM (RAM440)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
764h	Embeded RAM (RAM441)	32	RW	00000000h
768h	Embeded RAM (RAM442)	32	RW	00000000h
76ch	Embeded RAM (RAM443)	32	RW	00000000h
770h	Embeded RAM (RAM444)	32	RW	00000000h
774h	Embeded RAM (RAM445)	32	RW	00000000h
778h	Embeded RAM (RAM446)	32	RW	00000000h
77ch	Embeded RAM (RAM447)	32	RW	00000000h
780h	Embeded RAM (RAM448)	32	RW	00000000h
784h	Embeded RAM (RAM449)	32	RW	00000000h
788h	Embeded RAM (RAM450)	32	RW	00000000h
78ch	Embeded RAM (RAM451)	32	RW	00000000h
790h	Embeded RAM (RAM452)	32	RW	00000000h
794h	Embeded RAM (RAM453)	32	RW	00000000h
798h	Embeded RAM (RAM454)	32	RW	00000000h
79ch	Embeded RAM (RAM455)	32	RW	00000000h
7a0h	Embeded RAM (RAM456)	32	RW	00000000h
7a4h	Embeded RAM (RAM457)	32	RW	00000000h
7a8h	Embeded RAM (RAM458)	32	RW	00000000h
7ach	Embeded RAM (RAM459)	32	RW	00000000h
7b0h	Embeded RAM (RAM460)	32	RW	00000000h
7b4h	Embeded RAM (RAM461)	32	RW	00000000h
7b8h	Embeded RAM (RAM462)	32	RW	00000000h
7bch	Embeded RAM (RAM463)	32	RW	00000000h
7c0h	Embeded RAM (RAM464)	32	RW	00000000h
7c4h	Embeded RAM (RAM465)	32	RW	00000000h
7c8h	Embeded RAM (RAM466)	32	RW	00000000h
7cch	Embeded RAM (RAM467)	32	RW	00000000h
7d0h	Embeded RAM (RAM468)	32	RW	00000000h
7d4h	Embeded RAM (RAM469)	32	RW	00000000h
7d8h	Embeded RAM (RAM470)	32	RW	00000000h
7dch	Embeded RAM (RAM471)	32	RW	00000000h
7e0h	Embeded RAM (RAM472)	32	RW	00000000h
7e4h	Embeded RAM (RAM473)	32	RW	00000000h
7e8h	Embeded RAM (RAM474)	32	RW	00000000h
7ech	Embeded RAM (RAM475)	32	RW	00000000h
7f0h	Embeded RAM (RAM476)	32	RW	00000000h
7f4h	Embeded RAM (RAM477)	32	RW	00000000h
7f8h	Embeded RAM (RAM478)	32	RW	00000000h
7fch	Embeded RAM (RAM479)	32	RW	00000000h
800h	Embeded RAM (RAM480)	32	RW	00000000h
804h	Embeded RAM (RAM481)	32	RW	00000000h
808h	Embeded RAM (RAM482)	32	RW	00000000h
80ch	Embeded RAM (RAM483)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
810h	Embeded RAM (RAM484)	32	RW	00000000h
814h	Embeded RAM (RAM485)	32	RW	00000000h
818h	Embeded RAM (RAM486)	32	RW	00000000h
81ch	Embeded RAM (RAM487)	32	RW	00000000h
820h	Embeded RAM (RAM488)	32	RW	00000000h
824h	Embeded RAM (RAM489)	32	RW	00000000h
828h	Embeded RAM (RAM490)	32	RW	00000000h
82ch	Embeded RAM (RAM491)	32	RW	00000000h
830h	Embeded RAM (RAM492)	32	RW	00000000h
834h	Embeded RAM (RAM493)	32	RW	00000000h
838h	Embeded RAM (RAM494)	32	RW	00000000h
83ch	Embeded RAM (RAM495)	32	RW	00000000h
840h	Embeded RAM (RAM496)	32	RW	00000000h
844h	Embeded RAM (RAM497)	32	RW	00000000h
848h	Embeded RAM (RAM498)	32	RW	00000000h
84ch	Embeded RAM (RAM499)	32	RW	00000000h
850h	Embeded RAM (RAM500)	32	RW	00000000h
854h	Embeded RAM (RAM501)	32	RW	00000000h
858h	Embeded RAM (RAM502)	32	RW	00000000h
85ch	Embeded RAM (RAM503)	32	RW	00000000h
860h	Embeded RAM (RAM504)	32	RW	00000000h
864h	Embeded RAM (RAM505)	32	RW	00000000h
868h	Embeded RAM (RAM506)	32	RW	00000000h
86ch	Embeded RAM (RAM507)	32	RW	00000000h
870h	Embeded RAM (RAM508)	32	RW	00000000h
874h	Embeded RAM (RAM509)	32	RW	00000000h
878h	Embeded RAM (RAM510)	32	RW	00000000h
87ch	Embeded RAM (RAM511)	32	RW	00000000h
880h	RX Individual Mask Register (RXIMR0)	32	RW	00000000h
884h	RX Individual Mask Register (RXIMR1)	32	RW	00000000h
888h	RX Individual Mask Register (RXIMR2)	32	RW	00000000h
88ch	RX Individual Mask Register (RXIMR3)	32	RW	00000000h
890h	RX Individual Mask Register (RXIMR4)	32	RW	00000000h
894h	RX Individual Mask Register (RXIMR5)	32	RW	00000000h
898h	RX Individual Mask Register (RXIMR6)	32	RW	00000000h
89ch	RX Individual Mask Register (RXIMR7)	32	RW	00000000h
8a0h	RX Individual Mask Register (RXIMR8)	32	RW	00000000h
8a4h	RX Individual Mask Register (RXIMR9)	32	RW	00000000h
8a8h	RX Individual Mask Register (RXIMR10)	32	RW	00000000h
8ach	RX Individual Mask Register (RXIMR11)	32	RW	00000000h
8b0h	RX Individual Mask Register (RXIMR12)	32	RW	00000000h
8b4h	RX Individual Mask Register (RXIMR13)	32	RW	00000000h
8b8h	RX Individual Mask Register (RXIMR14)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
8bch	RX Individual Mask Register (RXIMR15)	32	RW	00000000h
8c0h	RX Individual Mask Register (RXIMR16)	32	RW	00000000h
8c4h	RX Individual Mask Register (RXIMR17)	32	RW	00000000h
8c8h	RX Individual Mask Register (RXIMR18)	32	RW	00000000h
8cch	RX Individual Mask Register (RXIMR19)	32	RW	00000000h
8d0h	RX Individual Mask Register (RXIMR20)	32	RW	00000000h
8d4h	RX Individual Mask Register (RXIMR21)	32	RW	00000000h
8d8h	RX Individual Mask Register (RXIMR22)	32	RW	00000000h
8dch	RX Individual Mask Register (RXIMR23)	32	RW	00000000h
8e0h	RX Individual Mask Register (RXIMR24)	32	RW	00000000h
8e4h	RX Individual Mask Register (RXIMR25)	32	RW	00000000h
8e8h	RX Individual Mask Register (RXIMR26)	32	RW	00000000h
8ech	RX Individual Mask Register (RXIMR27)	32	RW	00000000h
8f0h	RX Individual Mask Register (RXIMR28)	32	RW	00000000h
8f4h	RX Individual Mask Register (RXIMR29)	32	RW	00000000h
8f8h	RX Individual Mask Register (RXIMR30)	32	RW	00000000h
8fch	RX Individual Mask Register (RXIMR31)	32	RW	00000000h
900h	RX Individual Mask Register (RXIMR32)	32	RW	00000000h
904h	RX Individual Mask Register (RXIMR33)	32	RW	00000000h
908h	RX Individual Mask Register (RXIMR34)	32	RW	00000000h
90ch	RX Individual Mask Register (RXIMR35)	32	RW	00000000h
910h	RX Individual Mask Register (RXIMR36)	32	RW	00000000h
914h	RX Individual Mask Register (RXIMR37)	32	RW	00000000h
918h	RX Individual Mask Register (RXIMR38)	32	RW	00000000h
91ch	RX Individual Mask Register (RXIMR39)	32	RW	00000000h
920h	RX Individual Mask Register (RXIMR40)	32	RW	00000000h
924h	RX Individual Mask Register (RXIMR41)	32	RW	00000000h
928h	RX Individual Mask Register (RXIMR42)	32	RW	00000000h
92ch	RX Individual Mask Register (RXIMR43)	32	RW	00000000h
930h	RX Individual Mask Register (RXIMR44)	32	RW	00000000h
934h	RX Individual Mask Register (RXIMR45)	32	RW	00000000h
938h	RX Individual Mask Register (RXIMR46)	32	RW	00000000h
93ch	RX Individual Mask Register (RXIMR47)	32	RW	00000000h
940h	RX Individual Mask Register (RXIMR48)	32	RW	00000000h
944h	RX Individual Mask Register (RXIMR49)	32	RW	00000000h
948h	RX Individual Mask Register (RXIMR50)	32	RW	00000000h
94ch	RX Individual Mask Register (RXIMR51)	32	RW	00000000h
950h	RX Individual Mask Register (RXIMR52)	32	RW	00000000h
954h	RX Individual Mask Register (RXIMR53)	32	RW	00000000h
958h	RX Individual Mask Register (RXIMR54)	32	RW	00000000h
95ch	RX Individual Mask Register (RXIMR55)	32	RW	00000000h
960h	RX Individual Mask Register (RXIMR56)	32	RW	00000000h
964h	RX Individual Mask Register (RXIMR57)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
968h	RX Individual Mask Register (RXIMR58)	32	RW	00000000h
96ch	RX Individual Mask Register (RXIMR59)	32	RW	00000000h
970h	RX Individual Mask Register (RXIMR60)	32	RW	00000000h
974h	RX Individual Mask Register (RXIMR61)	32	RW	00000000h
978h	RX Individual Mask Register (RXIMR62)	32	RW	00000000h
97ch	RX Individual Mask Register (RXIMR63)	32	RW	00000000h
980h	RX Individual Mask Register (RXIMR64)	32	RW	00000000h
984h	RX Individual Mask Register (RXIMR65)	32	RW	00000000h
988h	RX Individual Mask Register (RXIMR66)	32	RW	00000000h
98ch	RX Individual Mask Register (RXIMR67)	32	RW	00000000h
990h	RX Individual Mask Register (RXIMR68)	32	RW	00000000h
994h	RX Individual Mask Register (RXIMR69)	32	RW	00000000h
998h	RX Individual Mask Register (RXIMR70)	32	RW	00000000h
99ch	RX Individual Mask Register (RXIMR71)	32	RW	00000000h
9a0h	RX Individual Mask Register (RXIMR72)	32	RW	00000000h
9a4h	RX Individual Mask Register (RXIMR73)	32	RW	00000000h
9a8h	RX Individual Mask Register (RXIMR74)	32	RW	00000000h
9ach	RX Individual Mask Register (RXIMR75)	32	RW	00000000h
9b0h	RX Individual Mask Register (RXIMR76)	32	RW	00000000h
9b4h	RX Individual Mask Register (RXIMR77)	32	RW	00000000h
9b8h	RX Individual Mask Register (RXIMR78)	32	RW	00000000h
9bch	RX Individual Mask Register (RXIMR79)	32	RW	00000000h
9c0h	RX Individual Mask Register (RXIMR80)	32	RW	00000000h
9c4h	RX Individual Mask Register (RXIMR81)	32	RW	00000000h
9c8h	RX Individual Mask Register (RXIMR82)	32	RW	00000000h
9cch	RX Individual Mask Register (RXIMR83)	32	RW	00000000h
9d0h	RX Individual Mask Register (RXIMR84)	32	RW	00000000h
9d4h	RX Individual Mask Register (RXIMR85)	32	RW	00000000h
9d8h	RX Individual Mask Register (RXIMR86)	32	RW	00000000h
9dch	RX Individual Mask Register (RXIMR87)	32	RW	00000000h
9e0h	RX Individual Mask Register (RXIMR88)	32	RW	00000000h
9e4h	RX Individual Mask Register (RXIMR89)	32	RW	00000000h
9e8h	RX Individual Mask Register (RXIMR90)	32	RW	00000000h
9ech	RX Individual Mask Register (RXIMR91)	32	RW	00000000h
9f0h	RX Individual Mask Register (RXIMR92)	32	RW	00000000h
9f4h	RX Individual Mask Register (RXIMR93)	32	RW	00000000h
9f8h	RX Individual Mask Register (RXIMR94)	32	RW	00000000h
9fch	RX Individual Mask Register (RXIMR95)	32	RW	00000000h
a00h	RX Individual Mask Register (RXIMR96)	32	RW	00000000h
a04h	RX Individual Mask Register (RXIMR97)	32	RW	00000000h
a08h	RX Individual Mask Register (RXIMR98)	32	RW	00000000h
a0ch	RX Individual Mask Register (RXIMR99)	32	RW	00000000h
a10h	RX Individual Mask Register (RXIMR100)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
a14h	RX Individual Mask Register (RXIMR101)	32	RW	00000000h
a18h	RX Individual Mask Register (RXIMR102)	32	RW	00000000h
a1ch	RX Individual Mask Register (RXIMR103)	32	RW	00000000h
a20h	RX Individual Mask Register (RXIMR104)	32	RW	00000000h
a24h	RX Individual Mask Register (RXIMR105)	32	RW	00000000h
a28h	RX Individual Mask Register (RXIMR106)	32	RW	00000000h
a2ch	RX Individual Mask Register (RXIMR107)	32	RW	00000000h
a30h	RX Individual Mask Register (RXIMR108)	32	RW	00000000h
a34h	RX Individual Mask Register (RXIMR109)	32	RW	00000000h
a38h	RX Individual Mask Register (RXIMR110)	32	RW	00000000h
a3ch	RX Individual Mask Register (RXIMR111)	32	RW	00000000h
a40h	RX Individual Mask Register (RXIMR112)	32	RW	00000000h
a44h	RX Individual Mask Register (RXIMR113)	32	RW	00000000h
a48h	RX Individual Mask Register (RXIMR114)	32	RW	00000000h
a4ch	RX Individual Mask Register (RXIMR115)	32	RW	00000000h
a50h	RX Individual Mask Register (RXIMR116)	32	RW	00000000h
a54h	RX Individual Mask Register (RXIMR117)	32	RW	00000000h
a58h	RX Individual Mask Register (RXIMR118)	32	RW	00000000h
a5ch	RX Individual Mask Register (RXIMR119)	32	RW	00000000h
a60h	RX Individual Mask Register (RXIMR120)	32	RW	00000000h
a64h	RX Individual Mask Register (RXIMR121)	32	RW	00000000h
a68h	RX Individual Mask Register (RXIMR122)	32	RW	00000000h
a6ch	RX Individual Mask Register (RXIMR123)	32	RW	00000000h
a70h	RX Individual Mask Register (RXIMR124)	32	RW	00000000h
a74h	RX Individual Mask Register (RXIMR125)	32	RW	00000000h
a78h	RX Individual Mask Register (RXIMR126)	32	RW	00000000h
a7ch	RX Individual Mask Register (RXIMR127)	32	RW	00000000h
ae0h	Memory Error Control Register (MECR)	32	RW	00000000h
ae4h	Error Injection Address Register (ERRIAR)	32	RW	00000000h
ae8h	Error Injection Data Pattern Register (ERRIDPR)	32	RW	00000000h
aech	Error Injection Parity Pattern Register (ERRIPPR)	32	RW	00000000h
af0h	Error Report Address Register (RERRAR)	32	RW	00000000h
af4h	Error Report Data Register (RERRDR)	32	RW	00000000h
af8h	Error Report Syndrome Register (RERRSYNR)	32	RW	00000000h
afch	Error Status Register (ERRSR)	32	RW	00000000h
b00h	Pretended Networking Control 1 Register (CTRL1_PN)	32	RW	00000000h
b04h	Pretended Networking Control 2 Register (CTRL2_PN)	32	RW	00000000h
b08h	Pretended Networking Wake Up Match Register (WU_MTC)	32	RW	00000000h
b0ch	Pretended Networking ID Filter 1 Register (FLT_ID1)	32	RW	00000000h
b10h	Pretended Networking DLC Filter Register (FLT_DLC)	32	RW	00000000h
b14h	Pretended Networking Payload Low Filter 1 Register (PL1_LO)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
b18h	Pretended Networking Payload High Filter 1 Register (PL1_HI)	32	RW	00000000h
b1ch	Pretended Networking ID Filter 2 Register (FLT_ID2_IDMASK)	32	RW	00000000h
b20h	Pretended Networking Payload Low Filter 2 Register (PL2_PLMASK_LO)	32	RW	00000000h
b24h	Pretended Networking Payload High Filter 2 Register (PL2_PLMASK_HI)	32	RW	00000000h
1680h	(WMB0_WMBn_CS)	32	RW	00000000h
1684h	(WMB0_WMBn_ID)	32	RW	00000000h
1688h	(WMB0_WMBn_D03)	32	RW	00000000h
168ch	(WMB0_WMBn_D47)	32	RW	00000000h
1690h	(WMB1_WMBn_CS)	32	RW	00000000h
1694h	(WMB1_WMBn_ID)	32	RW	00000000h
1698h	(WMB1_WMBn_D03)	32	RW	00000000h
169ch	(WMB1_WMBn_D47)	32	RW	00000000h
16a0h	(WMB2_WMBn_CS)	32	RW	00000000h
16a4h	(WMB2_WMBn_ID)	32	RW	00000000h
16a8h	(WMB2_WMBn_D03)	32	RW	00000000h
16ach	(WMB2_WMBn_D47)	32	RW	00000000h
16b0h	(WMB3_WMBn_CS)	32	RW	00000000h
16b4h	(WMB3_WMBn_ID)	32	RW	00000000h
16b8h	(WMB3_WMBn_D03)	32	RW	00000000h
16bch	(WMB3_WMBn_D47)	32	RW	00000000h
bf0h	Enhanced CAN Bit Timing Prescalers Register (EPRS)	32	RW	00000000h
bf4h	Enhanced Nominal CAN Bit Timing Register (ENCBT)	32	RW	00000000h
bf8h	Enhanced Data Phase CAN Bit Timing Register (EDCBT)	32	RW	00000000h
bfch	Enhanced Transceiver Delay Compensation Register (ETDC)	32	RW	00000000h
c00h	FD Control Register (FDCTRL)	32	RW	00000000h
c04h	FD Bit Timing Register (FDCBT)	32	RW	00000000h
c08h	FD CRC Register (FDCRC)	32	RW	00000000h
c0ch	Enhanced Rx FIFO Control Register (ERFCR)	32	RW	00000000h
c10h	Enhanced Rx FIFO Interrupt Enable Register (ERFIER)	32	RW	00000000h
c14h	Enhanced Rx FIFO Status Register (ERFSR)	32	RW	00000000h
c30h	High Resolution Time Stamp Registers (HR_TIME_STAMPn0)	32	RW	00000000h
c34h	High Resolution Time Stamp Registers (HR_TIME_STAMPn1)	32	RW	00000000h
c38h	High Resolution Time Stamp Registers (HR_TIME_STAMPn2)	32	RW	00000000h
c3ch	High Resolution Time Stamp Registers (HR_TIME_STAMPn3)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
c40h	High Resolution Time Stamp Registers (HR_TIME_STAMPn4)	32	RW	00000000h
c44h	High Resolution Time Stamp Registers (HR_TIME_STAMPn5)	32	RW	00000000h
c48h	High Resolution Time Stamp Registers (HR_TIME_STAMPn6)	32	RW	00000000h
c4ch	High Resolution Time Stamp Registers (HR_TIME_STAMPn7)	32	RW	00000000h
c50h	High Resolution Time Stamp Registers (HR_TIME_STAMPn8)	32	RW	00000000h
c54h	High Resolution Time Stamp Registers (HR_TIME_STAMPn9)	32	RW	00000000h
c58h	High Resolution Time Stamp Registers (HR_TIME_STAMPn10)	32	RW	00000000h
c5ch	High Resolution Time Stamp Registers (HR_TIME_STAMPn11)	32	RW	00000000h
c60h	High Resolution Time Stamp Registers (HR_TIME_STAMPn12)	32	RW	00000000h
c64h	High Resolution Time Stamp Registers (HR_TIME_STAMPn13)	32	RW	00000000h
c68h	High Resolution Time Stamp Registers (HR_TIME_STAMPn14)	32	RW	00000000h
c6ch	High Resolution Time Stamp Registers (HR_TIME_STAMPn15)	32	RW	00000000h
c70h	High Resolution Time Stamp Registers (HR_TIME_STAMPn16)	32	RW	00000000h
c74h	High Resolution Time Stamp Registers (HR_TIME_STAMPn17)	32	RW	00000000h
c78h	High Resolution Time Stamp Registers (HR_TIME_STAMPn18)	32	RW	00000000h
c7ch	High Resolution Time Stamp Registers (HR_TIME_STAMPn19)	32	RW	00000000h
c80h	High Resolution Time Stamp Registers (HR_TIME_STAMPn20)	32	RW	00000000h
c84h	High Resolution Time Stamp Registers (HR_TIME_STAMPn21)	32	RW	00000000h
c88h	High Resolution Time Stamp Registers (HR_TIME_STAMPn22)	32	RW	00000000h
c8ch	High Resolution Time Stamp Registers (HR_TIME_STAMPn23)	32	RW	00000000h
c90h	High Resolution Time Stamp Registers (HR_TIME_STAMPn24)	32	RW	00000000h
c94h	High Resolution Time Stamp Registers (HR_TIME_STAMPn25)	32	RW	00000000h
c98h	High Resolution Time Stamp Registers (HR_TIME_STAMPn26)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
c9ch	High Resolution Time Stamp Registers (HR_TIME_STAMPn27)	32	RW	00000000h
ca0h	High Resolution Time Stamp Registers (HR_TIME_STAMPn28)	32	RW	00000000h
ca4h	High Resolution Time Stamp Registers (HR_TIME_STAMPn29)	32	RW	00000000h
ca8h	High Resolution Time Stamp Registers (HR_TIME_STAMPn30)	32	RW	00000000h
cach	High Resolution Time Stamp Registers (HR_TIME_STAMPn31)	32	RW	00000000h
cb0h	High Resolution Time Stamp Registers (HR_TIME_STAMPn32)	32	RW	00000000h
cb4h	High Resolution Time Stamp Registers (HR_TIME_STAMPn33)	32	RW	00000000h
cb8h	High Resolution Time Stamp Registers (HR_TIME_STAMPn34)	32	RW	00000000h
cbch	High Resolution Time Stamp Registers (HR_TIME_STAMPn35)	32	RW	00000000h
cc0h	High Resolution Time Stamp Registers (HR_TIME_STAMPn36)	32	RW	00000000h
cc4h	High Resolution Time Stamp Registers (HR_TIME_STAMPn37)	32	RW	00000000h
cc8h	High Resolution Time Stamp Registers (HR_TIME_STAMPn38)	32	RW	00000000h
ccch	High Resolution Time Stamp Registers (HR_TIME_STAMPn39)	32	RW	00000000h
cd0h	High Resolution Time Stamp Registers (HR_TIME_STAMPn40)	32	RW	00000000h
cd4h	High Resolution Time Stamp Registers (HR_TIME_STAMPn41)	32	RW	00000000h
cd8h	High Resolution Time Stamp Registers (HR_TIME_STAMPn42)	32	RW	00000000h
cdch	High Resolution Time Stamp Registers (HR_TIME_STAMPn43)	32	RW	00000000h
ce0h	High Resolution Time Stamp Registers (HR_TIME_STAMPn44)	32	RW	00000000h
ce4h	High Resolution Time Stamp Registers (HR_TIME_STAMPn45)	32	RW	00000000h
ce8h	High Resolution Time Stamp Registers (HR_TIME_STAMPn46)	32	RW	00000000h
cech	High Resolution Time Stamp Registers (HR_TIME_STAMPn47)	32	RW	00000000h
cf0h	High Resolution Time Stamp Registers (HR_TIME_STAMPn48)	32	RW	00000000h
cf4h	High Resolution Time Stamp Registers (HR_TIME_STAMPn49)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
cf8h	High Resolution Time Stamp Registers (HR_TIME_STAMPn50)	32	RW	00000000h
cfch	High Resolution Time Stamp Registers (HR_TIME_STAMPn51)	32	RW	00000000h
d00h	High Resolution Time Stamp Registers (HR_TIME_STAMPn52)	32	RW	00000000h
d04h	High Resolution Time Stamp Registers (HR_TIME_STAMPn53)	32	RW	00000000h
d08h	High Resolution Time Stamp Registers (HR_TIME_STAMPn54)	32	RW	00000000h
d0ch	High Resolution Time Stamp Registers (HR_TIME_STAMPn55)	32	RW	00000000h
d10h	High Resolution Time Stamp Registers (HR_TIME_STAMPn56)	32	RW	00000000h
d14h	High Resolution Time Stamp Registers (HR_TIME_STAMPn57)	32	RW	00000000h
d18h	High Resolution Time Stamp Registers (HR_TIME_STAMPn58)	32	RW	00000000h
d1ch	High Resolution Time Stamp Registers (HR_TIME_STAMPn59)	32	RW	00000000h
d20h	High Resolution Time Stamp Registers (HR_TIME_STAMPn60)	32	RW	00000000h
d24h	High Resolution Time Stamp Registers (HR_TIME_STAMPn61)	32	RW	00000000h
d28h	High Resolution Time Stamp Registers (HR_TIME_STAMPn62)	32	RW	00000000h
d2ch	High Resolution Time Stamp Registers (HR_TIME_STAMPn63)	32	RW	00000000h
d30h	High Resolution Time Stamp Registers (HR_TIME_STAMPn64)	32	RW	00000000h
d34h	High Resolution Time Stamp Registers (HR_TIME_STAMPn65)	32	RW	00000000h
d38h	High Resolution Time Stamp Registers (HR_TIME_STAMPn66)	32	RW	00000000h
d3ch	High Resolution Time Stamp Registers (HR_TIME_STAMPn67)	32	RW	00000000h
d40h	High Resolution Time Stamp Registers (HR_TIME_STAMPn68)	32	RW	00000000h
d44h	High Resolution Time Stamp Registers (HR_TIME_STAMPn69)	32	RW	00000000h
d48h	High Resolution Time Stamp Registers (HR_TIME_STAMPn70)	32	RW	00000000h
d4ch	High Resolution Time Stamp Registers (HR_TIME_STAMPn71)	32	RW	00000000h
d50h	High Resolution Time Stamp Registers (HR_TIME_STAMPn72)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
d54h	High Resolution Time Stamp Registers (HR_TIME_STAMPn73)	32	RW	00000000h
d58h	High Resolution Time Stamp Registers (HR_TIME_STAMPn74)	32	RW	00000000h
d5ch	High Resolution Time Stamp Registers (HR_TIME_STAMPn75)	32	RW	00000000h
d60h	High Resolution Time Stamp Registers (HR_TIME_STAMPn76)	32	RW	00000000h
d64h	High Resolution Time Stamp Registers (HR_TIME_STAMPn77)	32	RW	00000000h
d68h	High Resolution Time Stamp Registers (HR_TIME_STAMPn78)	32	RW	00000000h
d6ch	High Resolution Time Stamp Registers (HR_TIME_STAMPn79)	32	RW	00000000h
d70h	High Resolution Time Stamp Registers (HR_TIME_STAMPn80)	32	RW	00000000h
d74h	High Resolution Time Stamp Registers (HR_TIME_STAMPn81)	32	RW	00000000h
d78h	High Resolution Time Stamp Registers (HR_TIME_STAMPn82)	32	RW	00000000h
d7ch	High Resolution Time Stamp Registers (HR_TIME_STAMPn83)	32	RW	00000000h
d80h	High Resolution Time Stamp Registers (HR_TIME_STAMPn84)	32	RW	00000000h
d84h	High Resolution Time Stamp Registers (HR_TIME_STAMPn85)	32	RW	00000000h
d88h	High Resolution Time Stamp Registers (HR_TIME_STAMPn86)	32	RW	00000000h
d8ch	High Resolution Time Stamp Registers (HR_TIME_STAMPn87)	32	RW	00000000h
d90h	High Resolution Time Stamp Registers (HR_TIME_STAMPn88)	32	RW	00000000h
d94h	High Resolution Time Stamp Registers (HR_TIME_STAMPn89)	32	RW	00000000h
d98h	High Resolution Time Stamp Registers (HR_TIME_STAMPn90)	32	RW	00000000h
d9ch	High Resolution Time Stamp Registers (HR_TIME_STAMPn91)	32	RW	00000000h
da0h	High Resolution Time Stamp Registers (HR_TIME_STAMPn92)	32	RW	00000000h
da4h	High Resolution Time Stamp Registers (HR_TIME_STAMPn93)	32	RW	00000000h
da8h	High Resolution Time Stamp Registers (HR_TIME_STAMPn94)	32	RW	00000000h
dach	High Resolution Time Stamp Registers (HR_TIME_STAMPn95)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
db0h	High Resolution Time Stamp Registers (HR_TIME_STAMPn96)	32	RW	00000000h
db4h	High Resolution Time Stamp Registers (HR_TIME_STAMPn97)	32	RW	00000000h
db8h	High Resolution Time Stamp Registers (HR_TIME_STAMPn98)	32	RW	00000000h
dbch	High Resolution Time Stamp Registers (HR_TIME_STAMPn99)	32	RW	00000000h
dc0h	High Resolution Time Stamp Registers (HR_TIME_STAMPn100)	32	RW	00000000h
dc4h	High Resolution Time Stamp Registers (HR_TIME_STAMPn101)	32	RW	00000000h
dc8h	High Resolution Time Stamp Registers (HR_TIME_STAMPn102)	32	RW	00000000h
dcch	High Resolution Time Stamp Registers (HR_TIME_STAMPn103)	32	RW	00000000h
dd0h	High Resolution Time Stamp Registers (HR_TIME_STAMPn104)	32	RW	00000000h
dd4h	High Resolution Time Stamp Registers (HR_TIME_STAMPn105)	32	RW	00000000h
dd8h	High Resolution Time Stamp Registers (HR_TIME_STAMPn106)	32	RW	00000000h
ddch	High Resolution Time Stamp Registers (HR_TIME_STAMPn107)	32	RW	00000000h
de0h	High Resolution Time Stamp Registers (HR_TIME_STAMPn108)	32	RW	00000000h
de4h	High Resolution Time Stamp Registers (HR_TIME_STAMPn109)	32	RW	00000000h
de8h	High Resolution Time Stamp Registers (HR_TIME_STAMPn110)	32	RW	00000000h
dech	High Resolution Time Stamp Registers (HR_TIME_STAMPn111)	32	RW	00000000h
df0h	High Resolution Time Stamp Registers (HR_TIME_STAMPn112)	32	RW	00000000h
df4h	High Resolution Time Stamp Registers (HR_TIME_STAMPn113)	32	RW	00000000h
df8h	High Resolution Time Stamp Registers (HR_TIME_STAMPn114)	32	RW	00000000h
dfch	High Resolution Time Stamp Registers (HR_TIME_STAMPn115)	32	RW	00000000h
e00h	High Resolution Time Stamp Registers (HR_TIME_STAMPn116)	32	RW	00000000h
e04h	High Resolution Time Stamp Registers (HR_TIME_STAMPn117)	32	RW	00000000h
e08h	High Resolution Time Stamp Registers (HR_TIME_STAMPn118)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
e0ch	High Resolution Time Stamp Registers (HR_TIME_STAMPn119)	32	RW	00000000h
e10h	High Resolution Time Stamp Registers (HR_TIME_STAMPn120)	32	RW	00000000h
e14h	High Resolution Time Stamp Registers (HR_TIME_STAMPn121)	32	RW	00000000h
e18h	High Resolution Time Stamp Registers (HR_TIME_STAMPn122)	32	RW	00000000h
e1ch	High Resolution Time Stamp Registers (HR_TIME_STAMPn123)	32	RW	00000000h
e20h	High Resolution Time Stamp Registers (HR_TIME_STAMPn124)	32	RW	00000000h
e24h	High Resolution Time Stamp Registers (HR_TIME_STAMPn125)	32	RW	00000000h
e28h	High Resolution Time Stamp Registers (HR_TIME_STAMPn126)	32	RW	00000000h
e2ch	High Resolution Time Stamp Registers (HR_TIME_STAMPn127)	32	RW	00000000h
3000h	Enhanced Rx FIFO Filter Element Registers (ERFFELn0)	32	RW	00000000h
3004h	Enhanced Rx FIFO Filter Element Registers (ERFFELn1)	32	RW	00000000h
3008h	Enhanced Rx FIFO Filter Element Registers (ERFFELn2)	32	RW	00000000h
300ch	Enhanced Rx FIFO Filter Element Registers (ERFFELn3)	32	RW	00000000h
3010h	Enhanced Rx FIFO Filter Element Registers (ERFFELn4)	32	RW	00000000h
3014h	Enhanced Rx FIFO Filter Element Registers (ERFFELn5)	32	RW	00000000h
3018h	Enhanced Rx FIFO Filter Element Registers (ERFFELn6)	32	RW	00000000h
301ch	Enhanced Rx FIFO Filter Element Registers (ERFFELn7)	32	RW	00000000h
3020h	Enhanced Rx FIFO Filter Element Registers (ERFFELn8)	32	RW	00000000h
3024h	Enhanced Rx FIFO Filter Element Registers (ERFFELn9)	32	RW	00000000h
3028h	Enhanced Rx FIFO Filter Element Registers (ERFFELn10)	32	RW	00000000h
302ch	Enhanced Rx FIFO Filter Element Registers (ERFFELn11)	32	RW	00000000h
3030h	Enhanced Rx FIFO Filter Element Registers (ERFFELn12)	32	RW	00000000h
3034h	Enhanced Rx FIFO Filter Element Registers (ERFFELn13)	32	RW	00000000h
3038h	Enhanced Rx FIFO Filter Element Registers (ERFFELn14)	32	RW	00000000h
303ch	Enhanced Rx FIFO Filter Element Registers (ERFFELn15)	32	RW	00000000h
3040h	Enhanced Rx FIFO Filter Element Registers (ERFFELn16)	32	RW	00000000h
3044h	Enhanced Rx FIFO Filter Element Registers (ERFFELn17)	32	RW	00000000h
3048h	Enhanced Rx FIFO Filter Element Registers (ERFFELn18)	32	RW	00000000h
304ch	Enhanced Rx FIFO Filter Element Registers (ERFFELn19)	32	RW	00000000h
3050h	Enhanced Rx FIFO Filter Element Registers (ERFFELn20)	32	RW	00000000h
3054h	Enhanced Rx FIFO Filter Element Registers (ERFFELn21)	32	RW	00000000h
3058h	Enhanced Rx FIFO Filter Element Registers (ERFFELn22)	32	RW	00000000h
305ch	Enhanced Rx FIFO Filter Element Registers (ERFFELn23)	32	RW	00000000h
3060h	Enhanced Rx FIFO Filter Element Registers (ERFFELn24)	32	RW	00000000h
3064h	Enhanced Rx FIFO Filter Element Registers (ERFFELn25)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
3068h	Enhanced Rx FIFO Filter Element Registers (ERFFELn26)	32	RW	00000000h
306ch	Enhanced Rx FIFO Filter Element Registers (ERFFELn27)	32	RW	00000000h
3070h	Enhanced Rx FIFO Filter Element Registers (ERFFELn28)	32	RW	00000000h
3074h	Enhanced Rx FIFO Filter Element Registers (ERFFELn29)	32	RW	00000000h
3078h	Enhanced Rx FIFO Filter Element Registers (ERFFELn30)	32	RW	00000000h
307ch	Enhanced Rx FIFO Filter Element Registers (ERFFELn31)	32	RW	00000000h
3080h	Enhanced Rx FIFO Filter Element Registers (ERFFELn32)	32	RW	00000000h
3084h	Enhanced Rx FIFO Filter Element Registers (ERFFELn33)	32	RW	00000000h
3088h	Enhanced Rx FIFO Filter Element Registers (ERFFELn34)	32	RW	00000000h
308ch	Enhanced Rx FIFO Filter Element Registers (ERFFELn35)	32	RW	00000000h
3090h	Enhanced Rx FIFO Filter Element Registers (ERFFELn36)	32	RW	00000000h
3094h	Enhanced Rx FIFO Filter Element Registers (ERFFELn37)	32	RW	00000000h
3098h	Enhanced Rx FIFO Filter Element Registers (ERFFELn38)	32	RW	00000000h
309ch	Enhanced Rx FIFO Filter Element Registers (ERFFELn39)	32	RW	00000000h
30a0h	Enhanced Rx FIFO Filter Element Registers (ERFFELn40)	32	RW	00000000h
30a4h	Enhanced Rx FIFO Filter Element Registers (ERFFELn41)	32	RW	00000000h
30a8h	Enhanced Rx FIFO Filter Element Registers (ERFFELn42)	32	RW	00000000h
30ach	Enhanced Rx FIFO Filter Element Registers (ERFFELn43)	32	RW	00000000h
30b0h	Enhanced Rx FIFO Filter Element Registers (ERFFELn44)	32	RW	00000000h
30b4h	Enhanced Rx FIFO Filter Element Registers (ERFFELn45)	32	RW	00000000h
30b8h	Enhanced Rx FIFO Filter Element Registers (ERFFELn46)	32	RW	00000000h
30bch	Enhanced Rx FIFO Filter Element Registers (ERFFELn47)	32	RW	00000000h
30c0h	Enhanced Rx FIFO Filter Element Registers (ERFFELn48)	32	RW	00000000h
30c4h	Enhanced Rx FIFO Filter Element Registers (ERFFELn49)	32	RW	00000000h
30c8h	Enhanced Rx FIFO Filter Element Registers (ERFFELn50)	32	RW	00000000h
30cch	Enhanced Rx FIFO Filter Element Registers (ERFFELn51)	32	RW	00000000h
30d0h	Enhanced Rx FIFO Filter Element Registers (ERFFELn52)	32	RW	00000000h
30d4h	Enhanced Rx FIFO Filter Element Registers (ERFFELn53)	32	RW	00000000h
30d8h	Enhanced Rx FIFO Filter Element Registers (ERFFELn54)	32	RW	00000000h
30dch	Enhanced Rx FIFO Filter Element Registers (ERFFELn55)	32	RW	00000000h
30e0h	Enhanced Rx FIFO Filter Element Registers (ERFFELn56)	32	RW	00000000h
30e4h	Enhanced Rx FIFO Filter Element Registers (ERFFELn57)	32	RW	00000000h
30e8h	Enhanced Rx FIFO Filter Element Registers (ERFFELn58)	32	RW	00000000h
30ech	Enhanced Rx FIFO Filter Element Registers (ERFFELn59)	32	RW	00000000h
30f0h	Enhanced Rx FIFO Filter Element Registers (ERFFELn60)	32	RW	00000000h
30f4h	Enhanced Rx FIFO Filter Element Registers (ERFFELn61)	32	RW	00000000h
30f8h	Enhanced Rx FIFO Filter Element Registers (ERFFELn62)	32	RW	00000000h
30fch	Enhanced Rx FIFO Filter Element Registers (ERFFELn63)	32	RW	00000000h
3100h	Enhanced Rx FIFO Filter Element Registers (ERFFELn64)	32	RW	00000000h
3104h	Enhanced Rx FIFO Filter Element Registers (ERFFELn65)	32	RW	00000000h
3108h	Enhanced Rx FIFO Filter Element Registers (ERFFELn66)	32	RW	00000000h
310ch	Enhanced Rx FIFO Filter Element Registers (ERFFELn67)	32	RW	00000000h
3110h	Enhanced Rx FIFO Filter Element Registers (ERFFELn68)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
3114h	Enhanced Rx FIFO Filter Element Registers (ERFFELn69)	32	RW	00000000h
3118h	Enhanced Rx FIFO Filter Element Registers (ERFFELn70)	32	RW	00000000h
311ch	Enhanced Rx FIFO Filter Element Registers (ERFFELn71)	32	RW	00000000h
3120h	Enhanced Rx FIFO Filter Element Registers (ERFFELn72)	32	RW	00000000h
3124h	Enhanced Rx FIFO Filter Element Registers (ERFFELn73)	32	RW	00000000h
3128h	Enhanced Rx FIFO Filter Element Registers (ERFFELn74)	32	RW	00000000h
312ch	Enhanced Rx FIFO Filter Element Registers (ERFFELn75)	32	RW	00000000h
3130h	Enhanced Rx FIFO Filter Element Registers (ERFFELn76)	32	RW	00000000h
3134h	Enhanced Rx FIFO Filter Element Registers (ERFFELn77)	32	RW	00000000h
3138h	Enhanced Rx FIFO Filter Element Registers (ERFFELn78)	32	RW	00000000h
313ch	Enhanced Rx FIFO Filter Element Registers (ERFFELn79)	32	RW	00000000h
3140h	Enhanced Rx FIFO Filter Element Registers (ERFFELn80)	32	RW	00000000h
3144h	Enhanced Rx FIFO Filter Element Registers (ERFFELn81)	32	RW	00000000h
3148h	Enhanced Rx FIFO Filter Element Registers (ERFFELn82)	32	RW	00000000h
314ch	Enhanced Rx FIFO Filter Element Registers (ERFFELn83)	32	RW	00000000h
3150h	Enhanced Rx FIFO Filter Element Registers (ERFFELn84)	32	RW	00000000h
3154h	Enhanced Rx FIFO Filter Element Registers (ERFFELn85)	32	RW	00000000h
3158h	Enhanced Rx FIFO Filter Element Registers (ERFFELn86)	32	RW	00000000h
315ch	Enhanced Rx FIFO Filter Element Registers (ERFFELn87)	32	RW	00000000h
3160h	Enhanced Rx FIFO Filter Element Registers (ERFFELn88)	32	RW	00000000h
3164h	Enhanced Rx FIFO Filter Element Registers (ERFFELn89)	32	RW	00000000h
3168h	Enhanced Rx FIFO Filter Element Registers (ERFFELn90)	32	RW	00000000h
316ch	Enhanced Rx FIFO Filter Element Registers (ERFFELn91)	32	RW	00000000h
3170h	Enhanced Rx FIFO Filter Element Registers (ERFFELn92)	32	RW	00000000h
3174h	Enhanced Rx FIFO Filter Element Registers (ERFFELn93)	32	RW	00000000h
3178h	Enhanced Rx FIFO Filter Element Registers (ERFFELn94)	32	RW	00000000h
317ch	Enhanced Rx FIFO Filter Element Registers (ERFFELn95)	32	RW	00000000h
3180h	Enhanced Rx FIFO Filter Element Registers (ERFFELn96)	32	RW	00000000h
3184h	Enhanced Rx FIFO Filter Element Registers (ERFFELn97)	32	RW	00000000h
3188h	Enhanced Rx FIFO Filter Element Registers (ERFFELn98)	32	RW	00000000h
318ch	Enhanced Rx FIFO Filter Element Registers (ERFFELn99)	32	RW	00000000h
3190h	Enhanced Rx FIFO Filter Element Registers (ERFFELn100)	32	RW	00000000h
3194h	Enhanced Rx FIFO Filter Element Registers (ERFFELn101)	32	RW	00000000h
3198h	Enhanced Rx FIFO Filter Element Registers (ERFFELn102)	32	RW	00000000h
319ch	Enhanced Rx FIFO Filter Element Registers (ERFFELn103)	32	RW	00000000h
31a0h	Enhanced Rx FIFO Filter Element Registers (ERFFELn104)	32	RW	00000000h
31a4h	Enhanced Rx FIFO Filter Element Registers (ERFFELn105)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
31a8h	Enhanced Rx FIFO Filter Element Registers (ERFFELn106)	32	RW	00000000h
31ach	Enhanced Rx FIFO Filter Element Registers (ERFFELn107)	32	RW	00000000h
31b0h	Enhanced Rx FIFO Filter Element Registers (ERFFELn108)	32	RW	00000000h
31b4h	Enhanced Rx FIFO Filter Element Registers (ERFFELn109)	32	RW	00000000h
31b8h	Enhanced Rx FIFO Filter Element Registers (ERFFELn110)	32	RW	00000000h
31bch	Enhanced Rx FIFO Filter Element Registers (ERFFELn111)	32	RW	00000000h
31c0h	Enhanced Rx FIFO Filter Element Registers (ERFFELn112)	32	RW	00000000h
31c4h	Enhanced Rx FIFO Filter Element Registers (ERFFELn113)	32	RW	00000000h
31c8h	Enhanced Rx FIFO Filter Element Registers (ERFFELn114)	32	RW	00000000h
31cch	Enhanced Rx FIFO Filter Element Registers (ERFFELn115)	32	RW	00000000h
31d0h	Enhanced Rx FIFO Filter Element Registers (ERFFELn116)	32	RW	00000000h
31d4h	Enhanced Rx FIFO Filter Element Registers (ERFFELn117)	32	RW	00000000h
31d8h	Enhanced Rx FIFO Filter Element Registers (ERFFELn118)	32	RW	00000000h
31dch	Enhanced Rx FIFO Filter Element Registers (ERFFELn119)	32	RW	00000000h
31e0h	Enhanced Rx FIFO Filter Element Registers (ERFFELn120)	32	RW	00000000h
31e4h	Enhanced Rx FIFO Filter Element Registers (ERFFELn121)	32	RW	00000000h
31e8h	Enhanced Rx FIFO Filter Element Registers (ERFFELn122)	32	RW	00000000h
31ech	Enhanced Rx FIFO Filter Element Registers (ERFFELn123)	32	RW	00000000h
31f0h	Enhanced Rx FIFO Filter Element Registers (ERFFELn124)	32	RW	00000000h
31f4h	Enhanced Rx FIFO Filter Element Registers (ERFFELn125)	32	RW	00000000h
31f8h	Enhanced Rx FIFO Filter Element Registers (ERFFELn126)	32	RW	00000000h
31fch	Enhanced Rx FIFO Filter Element Registers (ERFFELn127)	32	RW	00000000h

Yuntu Microelectronics

17.3.3.1 CAN MCR Register

CAN_MCR defines global system configurations, such as the module operation modes and the maximum message buffer configuration.

Offset: 0h

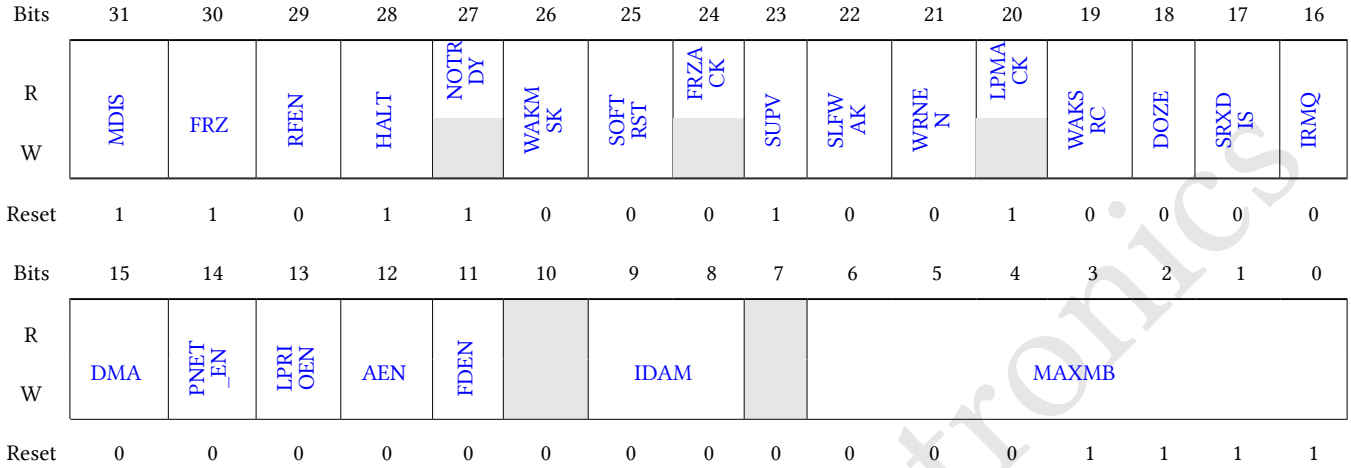


Table 17.4: CAN MCR Register Description

Field	Function
31 MDIS	**Module Disable - Disables or enables the FlexCAN module.** When disabled, the FlexCAN disables the clocks to the PE and CHI submodules. MDIS is not affected by soft reset. 0b - Enable the FlexCAN module 1b - Disable the FlexCAN module
30 FRZ	**Freeze Enable** The FRZ bit specifies the FlexCAN behavior when CAN_MCR.HALT isset or when Debug mode is requested at chip level. When FRZ is asserted, the FlexCAN is enabled to enter Freeze mode. Negation of FRZ causes the FlexCAN to exit from Freeze mode. FRZ is set by hardware when a non-correctable error is detected and CAN_MECR.NCEFAFRZ is asserted. 0b - Not enabled to enter freeze mode 1b - Enabled to enter freeze mode
29 RFEN	**legacy Rx FIFO Enable - Enables or disables the legacy Rx FIFO feature** When RFEN is set, MBs 0-5 cannot be used for normal reception and transmission because the corresponding memory region (0x80-0xDC) is used by the FIFO engine and additional MBs (up to 32, depending on CAN_CTRL2.RFFN setting), which are used as Legacy Rx FIFO ID Filter Table elements. RFEN also impacts the minimum number of peripheral clocks per CAN bit. RFEN can be written in Freeze mode only because it is blocked by hardware in other modes. NOTE: RFEN cannot be set when CAN FD operation is enabled (see FDEN bit). 0b - Disable Rx FIFO 1b - Enable Rx FIFO.

Yuntu Microelectronics

Field	Function
28 HALT	**Halt FlexCAN** Assertion of HALT puts the FlexCAN module into Freeze mode. The CPU should clear HALT after initializing the message buffers and the control registers CAN_CTRL1 and CAN_CTRL2. No reception or transmission is performed by the FlexCAN before HALT is cleared. Freeze mode cannot be entered when the FlexCAN is in a Low-Power mode. The HALT bit is set by hardware when a non-correctable error is detected and CAN_MECR.NCEFAFRZ is asserted. 0b - No Freeze mode request 1b - Enter Freeze mode if the FRZ bit is asserted
27 NOTRDY	**FlexCAN not ready** NOTRDY is a read-only bit that indicates that the FlexCAN is either in Module Disable, Doze, Deepsleep, or Freeze mode. NOTRDY is negated when the FlexCAN has exited these modes. NOTRDY bit is not affected by soft reset 0b - FlexCAN is either in Normal, Listen-Only or Loop-Back mode 1b - FlexCAN is either in Module Disable, Doze, Deepsleep or Freeze mode
26 WAKMSK	**FlexCAN Wake-Up Interrupt Mask** Enables Wake Up interrupt generation under the Self Wake Up mechanism: 0b - Disable Wake Up interrupt 1b - Enable Wake Up interrupt
25 SOFTRST	**FlexCAN Software Reset** When SOFTRST is asserted, the FlexCAN resets its internal state machines and some of the memory-mapped registers. SOFTRST can be asserted directly by the CPU by writing to CAN_MCR. SOFTRST is also asserted when global soft reset is requested at chip level. Because soft reset is synchronous and has to follow a request/acknowledge procedure across clock domains, it may take some time to fully propagate its effect. The SOFTRST bit remains asserted while reset is pending, and is automatically negated when reset completes. Therefore, software can poll SOFTRST to know when the soft reset has completed. Soft reset cannot be applied when clocks are shut down in a Low-Power mode. The FlexCAN should be first removed from Low-Power mode, then soft reset can be applied. SOFTRST is not affected by soft reset 0b - No reset request 1b - Reset the registers affected by software reset
24	**Freeze Mode Acknowledge**

Yuntu Microelectronics

Field	Function
FRZACK	FRZACK is a read-only bit that indicates that the FlexCAN is in Freeze mode and its prescaler is stopped. A Freeze mode request cannot be granted until current transmission or reception processes have finished. Therefore, software can poll FRZACK to know when the FlexCAN has actually entered Freeze mode. If the Freeze mode request is negated, FRZACK is negated after the FlexCAN prescaler is running again. If Freeze mode is requested when the FlexCAN is in a Low-Power mode, FRZACK will be set only when the Low-Power mode is exited. See "Freeze Mode". FRZACK is not affected by soft reset. NOTE: FRZACK is asserted within 178 CAN bits from the Freeze mode request by the CPU, and negated within 2 CAN bits after the Freeze mode request removal (see "Protocol Timing"). 0b - FlexCAN not in Freeze Mode, prescaler running 1b - FlexCAN in Freeze Mode, prescaler stopped
23 SUPV	**Supervisor Mode - Configures the FlexCAN to be either in Supervisor or User mode.** The registers affected by SUPV are marked as S/U in the Access Type column of Table: "CAN register access and reset information". The SUPV reset value is 1, so the affected registers start with Supervisor access only. SUPV can be written only in Freeze mode because it is blocked by hardware in other modes. 1b - FlexCAN is in Supervisor Mode, Affected registers allow only Supervisor access. Unrestricted access behaves as though the access was done to an unimplemented register location. 0b - FlexCAN is in User mode, Affected registers allow both Supervisor and Unrestricted accesses
22 SLFWAK	**Self Wake-Up - Enables the Self Wake Up feature when the FlexCAN is in a Low-Power mode other than Module Disable.** When the Self Wake Up feature is enabled, the FlexCAN module monitors the bus for Wake-Up event (a recessive-to-dominant transition). If a Wake-Up event is detected during Deepsleep mode, the FlexCAN, if enabled to do so, generates a Wake Up interrupt to the CPU so that it can exit Deepsleep mode globally and the FlexCAN can request to resume the clocks. When FlexCAN is in a Low-Power mode other than Module Disable, SLFWAK cannot be written as it is blocked by hardware. When Pretended Networking mode is set, Self Wake-Up must be disabled. 1b - Enable FlexCAN self Wake-Up feature. 0b - Disable FlexCAN self Wake-Up feature.
21	**Warning Interrupt Enable**

Yuntu Microelectronics

Field	Function
WRNEN	Assertion of WRNEN enables the generation of the TWRNINT and RWRNINT flags in the Error and Status Register 1 (CAN_ESR1). If WRNEN is negated, the TWRNINT and RWRNINT flags will always be zero, independent of the values of the error counters, and no warning interrupt will ever be generated. WRNEN can be written only in Freeze mode because it is blocked by hardware in other modes. 1b - TWRNINT and RWRNINT bits are set when the respective error counter transitions from less than 96 to greater than or equal to 96 0b - TWRING and RWRNINT bits are zero, independent of the values in the error counters
20 LPMACK	**Low-Power Mode Acknowledge** LPMACK is a read-only bit that indicates that the FlexCAN is in a Low-Power mode (Module Disable, Doze or Deepsleep). A Low-Power mode cannot be entered until all current transmission or reception processes have finished. Therefore, the CPU can poll LPMACK to know when the FlexCAN has actually entered Low-Power mode. LPMACK is not affected by soft reset. NOTE: LPMACK is asserted within 180 CAN bits from the Low-Power mode request by the CPU, and negated within 2 CAN bits after the Low-Power mode request removal (see "Protocol Timing"). When the FlexCAN is in Pretended Networking mode, LPMACK is negated within 180 CAN bits after the low-power mode request removal. 1b - FlexCAN is in the Low-Power mode 0b - FlexCAN is not in the Low-Power mode
19 WAKSRC	**Wake Up Source - Controls whether the integrated low-pass filter is applied to protect the Rx CAN input from spurious Wake-Up.** WAKSRC can be written only in Freeze mode because it is blocked by hardware in other modes. 1b - FlexCAN uses the filtered RX input to detect recessive to dominant edges on the can bus 0b - FlexCAN uses the unfiltered RX input to detect recessive to dominant edges on the can bus
18 DOZE	**Doze Mode Enable - Controls whether the FlexCAN is allowed to enter Low-Power mode when Doze mode is requested at chip level.** DOZE is automatically reset when the FlexCAN wakes up from Doze mode upon detecting activity on the CAN bus (Self Wake Up enabled). 1b - FlexCAN is enabled to enter Low-Power mode when Doze mode is requested. 0b - FlexCAN is not enabled to enter Low-Power mode when Doze mode is requested.
17	**Self Reception Disable**

Yuntu Microelectronics

Field	Function
SRXDIS	SRXDIS controls whether the FlexCAN is allowed to receive frames transmitted by itself. If SRXDIS is asserted, frames transmitted by the FlexCAN are not stored in any MB, regardless of whether the MB is programmed with an ID that matches the transmitted frame, and no interrupt flag or interrupt signal is generated due to the frame reception. SRXDIS can be written only in Freeze mode because it is blocked by hardware in other modes 1b - Disable self-reception. 0b - Enable self-reception.
16 IRMQ	**Individual Rx Masking And Queue Enable** IRMQ controls whether Rx matching process is based on individual masking and queue or on masking scheme with CAN_RXMGMASK, CAN_RX14MASK, CAN_RX15MASK, and CAN_RXFGMASK. IRMQ can be written in Freeze mode only because it is blocked by hardware in other modes. 1b - Enable individual Rx masking and queue feature. 0b - Disable individual Rx masking and queue feature. For backward compatibility with legacy applications, the reading of C/S word locks the MB even if it is EMPTY
15 DMA	**DMA Enable** The DMA bit enables or disables the DMA feature. The DMA feature can only be used with the Legacy Rx FIFO or the Enhanced Rx FIFO; consequently, either CAN_MCR.RFEN or CAN_ERFCR.ERFEN must be asserted. When DMA and RFEN are both set, CAN_IFLAG1.BUF5I generates the DMA request and no Rx FIFO interrupt is generated. The DMA bit can be written in Freeze mode only as it is blocked by hardware in other modes. The DMA bit is not available(reserved, write 0) if parameter DMA_EN is 0 1b - Enable DMA feature for Rx FIFO. 0b - Disable DMA feature for Rx FIFO.
14 PNET_EN	**Pretended Networking Enable** When Pretended Networking is enabled, the PE submodule is kept operational in Doze mode and Deepsleep mode, so that it is able to process Rx message filtering as defined by the Pretended Networking configuration registers. See ”Receive Process under Pretended Networking Mode”. PNET_EN can be written in Freeze mode only. The PNET_EN bit is not available (reserved, write 0) if parameter PNET_EN is 0. 1b - Enable Pretended Networking mode. 0b - Disable Pretended Networking mode.
13	**Local Priority Enable**

Yuntu Microelectronics

Field	Function
LPRIOEN	The LPRIOEN bit is provided for backward compatibility with legacy applications. LPRIOEN controls whether the Local Priority feature is enabled or not. Local Priority is used to expand the ID used during the arbitration process. With this expanded ID concept, the arbitration process is done based on the full 32-bit word, but the actual transmitted ID still has 11 bits for standard frames and 29 bits for extended frames. LPRIOEN can be written only in Freeze mode because it is blocked by hardware in other modes.
12 AEN	**Abort Enable - Enables the Tx abort mechanism.** The Tx abort mechanism ensures a safe procedure for aborting a pending transmission, so that no frame is sent on the CAN bus without notification. See "Transmission Abort Mechanism". AEN can be written only in Freeze mode because it is blocked by hardware in other modes. NOTE: When AEN is asserted, the abort mechanism must only be used for updating mailboxes configured for transmission. CAUTION: Writing the Abort code into Rx mailboxes can cause unpredictable results when AEN is asserted. 1b - Enable abort 0b - Disable abort.
11 FDEN	**CAN FD Operation Enable –Enables CAN with Flexible Data rate (CAN FD) operation.** FDEN can be written in Freeze mode only. The FEDN bit is not available (reserved, write 0) if parameter FD_EN is 0. NOTE: The Rx FIFO Enable (RFEN) bit cannot be set if FDEN is asserted. 1b - CAN FD is enabled. The FlexCAN is able to receive and transmit messages in both CAN FD and CAN 2.0 formats. 0b - CAN FD is disabled. The FlexCAN is able to receive and transmit messages in CAN 2.0 format.
9 - 8 IDAM	**ID Acceptance Mode –Identifies the format of the Rx FIFO ID Filter Table elements.** All elements of the table are configured at the same time by IDAM (they are all the same format). See "Legacy Rx FIFO Structure". IDAM can be written only in Freeze mode because it is blocked by hardware in other modes. 00b - Format A –One full ID (standard and extended) per ID Filter Table element 01b - Format B –Two full standard IDs or two partial 14-bit (standard and extended) IDs per ID Filter Table element 10b - Format C –Four partial 8-bit standard IDs per ID Filter Table element 11b - Format D –All frames rejected.
6 - 0	**Number of the Last Message Buffer –Defines the number of the last message buffer that will take part in the matching and arbitration processes.**

Yuntu Microelectronics

Field	Function
MAXMB	The reset value (0x0F) is equivalent to a 16 message buffer configuration. MAXMB can be written only in Freeze mode because it is blocked by hardware in other modes. Number of the last MB = MAXMB NOTE: MAXMB must be programmed with a value smaller than or equal to the number of available message buffers, as described in "FlexCAN Memory Partition for CAN FD". Additionally, the definition of MAXMB value must take into account the region of MBs occupied by Rx FIFO and its ID filters table space defined by CAN_CTRL2.RFFN. MAXMB also impacts the minimum number of peripheral clocks per CAN bit as described in the FlexCAN Integration Guide.

17.3.3.2 CAN CTRL1 Register

The CAN_CTRL1 register is defined for specific FlexCAN control features related to the CAN bus, such as bit-rate, programmable sampling point within an Rx bit, Loop-Back mode, Listen-Only mode, Bus Off recovery behavior, and interrupt enabling (Bus Off, Error, Warning). It also determines the division factor for the clock prescaler.

The CAN bit timing variables (PRESDIV, PROPSEG, PSEG1, PSEG2, and RJW) can also be configured in CAN_CBT register, which extends the range of all these variables. If CAN_CBT.BTF is set, the PRESDIV, PROPSEG, PSEG1, PSEG2, and RJW fields of CAN_CTRL1 become read-only.

When CAN_CTRL2.BTE = 1, the PRESDIV, PROPSEG, PSEG1, PSEG2, and RJW fields in CAN_CTRL1 are not used. They are read as zero and a write operation to them has no effect.

NOTE: When the CAN FD feature is enabled, do not use the PRESDIV, RJW, PSEG1, PSEG2, and PROPSEG fields of the CAN_CTRL1 register for CAN bit timing. Instead, use the CAN_CBT register's EPRESDIV, ERJW, EPSEG1, EPSEG2, and EPROPSEG fields.

The contents of the CAN_CTRL1 register are not affected by soft reset.

NOTE: The CAN bit variables in CAN_CTRL1 and in CAN_CBT are stored in the same register.

Offset: 4h

Yuntu Microelectronics

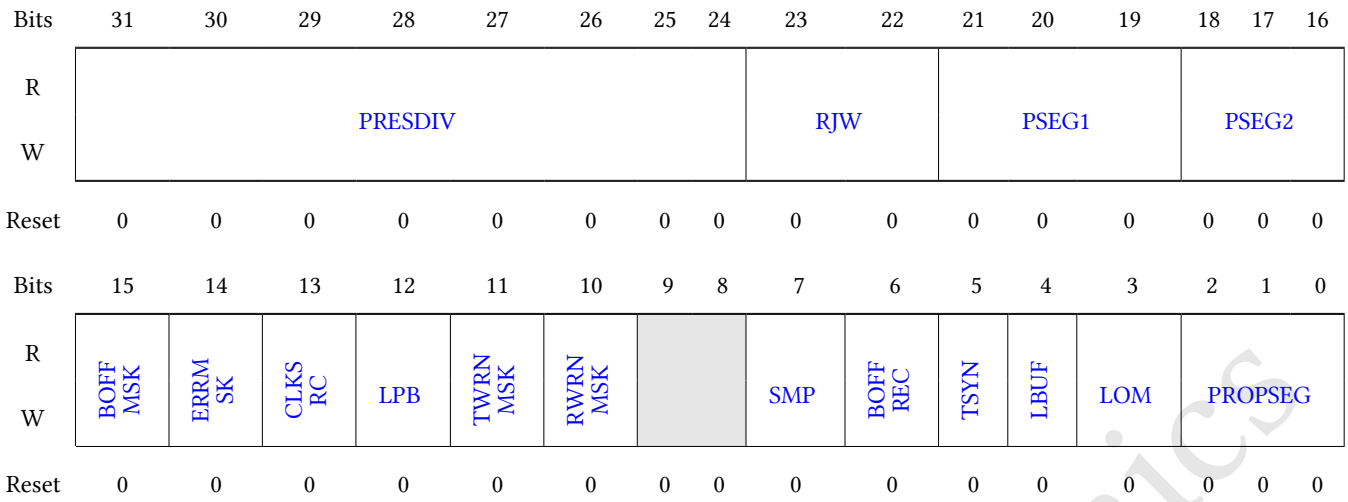


Table 17.5: CAN CTRL1 Register Description

Field	Function
31 - 24 PRESDIV	**Prescaler Division Factor –Defines the ratio between the PE clock frequency and the serial clock (Sclock) frequency.** The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency. The maximum value of PRESDIV is 0xFF, which gives a minimum Sclock frequency equal to the PE clock frequency divided by 256. See section "Protocol Timing". PRESDIV can be written only in Freeze mode because it is blocked by hardware in other modes. Sclock frequency = PE clock frequency / (PRESDIV + 1)
23 - 22 RJW	**Resync Jump Width –Defines the maximum number of time quanta that a bit time can be changed by one re-synchronization.** One time quantum is equal to the Sclock period. The valid programmable values for RJW are 0–3. RJW can be written only in Freeze mode because it is blocked by hardware in other modes. Resync Jump Width = RJW + 1
21 - 19 PSEG1	**Phase Segment 1 –Defines the length of phase segment 1 in the bit time.** The valid programmable values are 0–7. PSEG1 can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 1 = (PSEG1 + 1) × Time-Quanta
18 - 16 PSEG2	**Phase Segment 2 –Defines the length of phase segment 2 in the bit time.** The valid programmable values are 1–7. PSEG2 can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 2 = (PSEG2 + 1) × Time-Quanta
15	**Bus Off Interrupt Mask –Provides a mask for the Bus Off interrupt (CAN_ESR1.BOFFINT).**

Yuntu Microelectronics

Field	Function
BOFFMSK	0b - Bus Off interrupt disabled. 1b - Bus Off interrupt enabled.
14 ERRMSK	**Error Interrupt Mask –Provides a mask for the Error interrupt (CAN_ESR1.ERRINT).** 0b - Error interrupt disabled. 1b - Error interrupt enabled.
13 CLKSRC	**CAN Engine Clock Source –Selects the clock source to the PE submodule to be either the peripheral clock or the oscillator clock.** The selected clock is fed to the prescaler to generate the serial clock (Sclock). To ensure reliable operation, CLKSRC can be written only in Disable mode. In other modes, writing to CLKSRC is blocked by hardware. See "Protocol Timing". 0b - The PE clock source is the oscillator clock. Under this condition, the oscillator clock frequency must be lower than the peripheral clock frequency. 1b - The PE clock source is the peripheral clock.
12 LPB	**Loop-Back Mode –Configures the FlexCAN to operate in Loop-Back mode.** In Loop-Back mode, the FlexCAN performs an internal loop back that can be used for self-test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic 1). The FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In Loop-Back mode, the FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. LPB can be written only in Freeze mode because it is blocked by hardware in other modes. NOTE: In Loop-Back mode, CAN_MCR.SRXDIS cannot be asserted because it would impede the self-reception of a transmitted message. NOTE: CAN_FDCTRL.TDCEN must be 0 (Transceiver Delay Compensation feature disabled) when LPB is asserted. 0b - Loop-Back disabled. 1b - Loop-Back enabled.
11 TWRNMSK	**Tx Warning Interrupt Mask –Provides a mask for the Tx Warning interrupt (CAN_ESR1.TWRNINT).** TWRNMSK is read as zero when CAN_MCR.WRNEN is negated. TWRNMSK can be written only if CAN_MCR.WRNEN is asserted 0b - Tx Warning interrupt disabled. 1b - Tx Warning interrupt enabled.
10	**Rx Warning Interrupt Mask –Provides a mask for the Rx Warning interrupt (CAN_ESR1.RWRNINT).**

Yuntu Microelectronics

Field	Function
RWRNMSK	RWRNMSK is read as zero when CAN_MCR.WRNEN is negated. RWRNMSK can be written only if CAN_MCR.WRNEN is asserted. 0b - Rx Warning interrupt disabled. 1b - Rx Warning interrupt enabled.
7 SMP	**CAN Bit Sampling –Defines the sampling mode of CAN bits at the Rx input.** SMP can be written only in Freeze mode because it is blocked by hardware in other modes. NOTE: For proper operation, to assert SMP it is necessary to guarantee a minimum value of two time quanta in CAN_CTRL1.PSEG1 (or CAN_CBT.EPSEG1). SMP cannot be asserted when CAN FD is enabled (CAN_MCR.FDEN = 1). 0b - One sample is used to determine the bit value. 1b - Three samples are used to determine the value of the received bit, the regular sample (sample point) and two preceding samples; a majority rule is used.
6 BOFFREC	**Bus Off Recovery –Defines how the FlexCAN recovers from Bus Off state.** If BOFFREC is negated, automatic recovering from Bus Off state occurs according to the CAN Specification 2.0B. If BOFFREC is asserted, automatic recovering from Bus Off is disabled and the FlexCAN module remains in Bus Off state until BOFFREC is negated by software. If the negation occurs before 128 sequences of 11 recessive bits are detected on the CAN bus, then Bus Off recovery happens as if the BOFFREC bit had never been asserted. If the negation occurs after 128 sequences of 11 recessive bits occurred, then the FlexCAN will re-synchronize to the bus by waiting for 11 recessive bits before joining the bus. After negation, BOFFREC can be re-asserted again during Bus Off, but it will be effective only the next time the FlexCAN module enters Bus Off. If BOFFREC was negated when the FlexCAN module entered Bus Off, asserting BOFFREC during Bus Off will not be effective for the current Bus Off recovery. 0b - Automatic recovering from Bus Off state enabled. 1b - Automatic recovering from Bus Off state disabled.
5 TSYN	**Timer Sync –Enables a mechanism that resets the free-running timer each time a message is received in message buffer 0.** The Timer Sync feature provides a means to synchronize multiple FlexCAN stations with a special ”SYNC” message (global network time). If CAN_MCR.RFEN is set (Legacy Rx FIFO enabled), the first available mailbox, according to the CAN_CTRL2.RFFN setting, is used for timer synchronization instead of MB0. TSYN can be written in Freeze mode only because it is blocked by hardware in other modes. 0b - Timer Sync feature disabled. 1b - Timer Sync feature enabled.
4	**Lowest Buffer Transmitted First –Defines the ordering mechanism for message buffer transmission.**

Yuntu Microelectronics

Field	Function
LBUF	When LBUF is asserted, the CAN_MCR.LPRIOEN bit does not affect the priority arbitration. LBUF can be written only in Freeze mode because it is blocked by hardware in other modes. 0b - Buffer with highest priority is transmitted first. 1b - Lowest number buffer is transmitted first.
3 LOM	**Listen-Only Mode –Configures the FlexCAN to operate in Listen-Only mode.** In Listen-Only Mode mode, transmission is disabled, all error counters described in CAN_ECR are frozen and the FlexCAN module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If the FlexCAN detects a message that has not been acknowledged, it flags a BIT0 error without changing the receive error counter (RXERRCNT) in CAN_ECR, as if it were trying to acknowledge the message. Listen-Only mode is acknowledged by the state of the CAN_ESR1.FLTCONF field indicating Passive Error. There can be some delay between the Listen-Only mode request and acknowledge. LOM can be written in Freeze mode only because it is blocked by hardware in other modes. 0b - Listen-Only mode is deactivated. 1b - FlexCAN module operates in Listen-Only mode.
2 - 0 PROPSEG	**Propagation Segment –Defines the length of the propagation segment in the bit time.** The valid programmable values are 0–7. PROPSEG can be written only in Freeze mode because it is blocked by hardware in other modes. Propagation segment time = (PROPSEG + 1) × time-quanta. Time-quantum = one Sclock period.

17.3.3.3 CAN TIMER Register

The CAN_TIMER register represents a 16-bit free-running counter that can be read and written by the CPU. The timer starts from 0x0 after reset, counts linearly to 0xFFFF, and wraps around.
When CAN_CTRL2.TIMER_SRC is asserted, the timer is continuously incremented by an external time tick. The time tick must be synchronous to the peripheral clock, with a minimum pulse width of one ipg_clk cycle.
When CAN_CTRL2.TIMER_SRC is negated, the timer is incremented by the CAN bit clock, which defines the baud rate on the CAN bus. During a message transmission/reception, the timer increments by one for each bit that is received or transmitted. When there is no message on the bus, the timer counts using the previously programmed baud rate. The timer is not incremented during Module Disable, Doze, Deepsleep, and Freeze modes.
The timer value is captured when the second bit of the identifier field of any frame is on the CAN bus. This captured value is written into the Time Stamp entry in a message buffer after a successful reception or transmission of a message.
If CAN_CTRL1.TSYN is asserted, the timer is reset whenever a message is received in the first available mailbox, according to the CAN_CTRL2.RFFN setting.
The CPU can write to this register anytime. However, if the write occurs at the same time that the timer is being reset by a reception in the first mailbox, then the write value is discarded.
Reading this register affects the mailbox unlocking procedure, see ”Mailbox Lock Mechanism”.

Yuntu Microelectronics

Offset: 8h

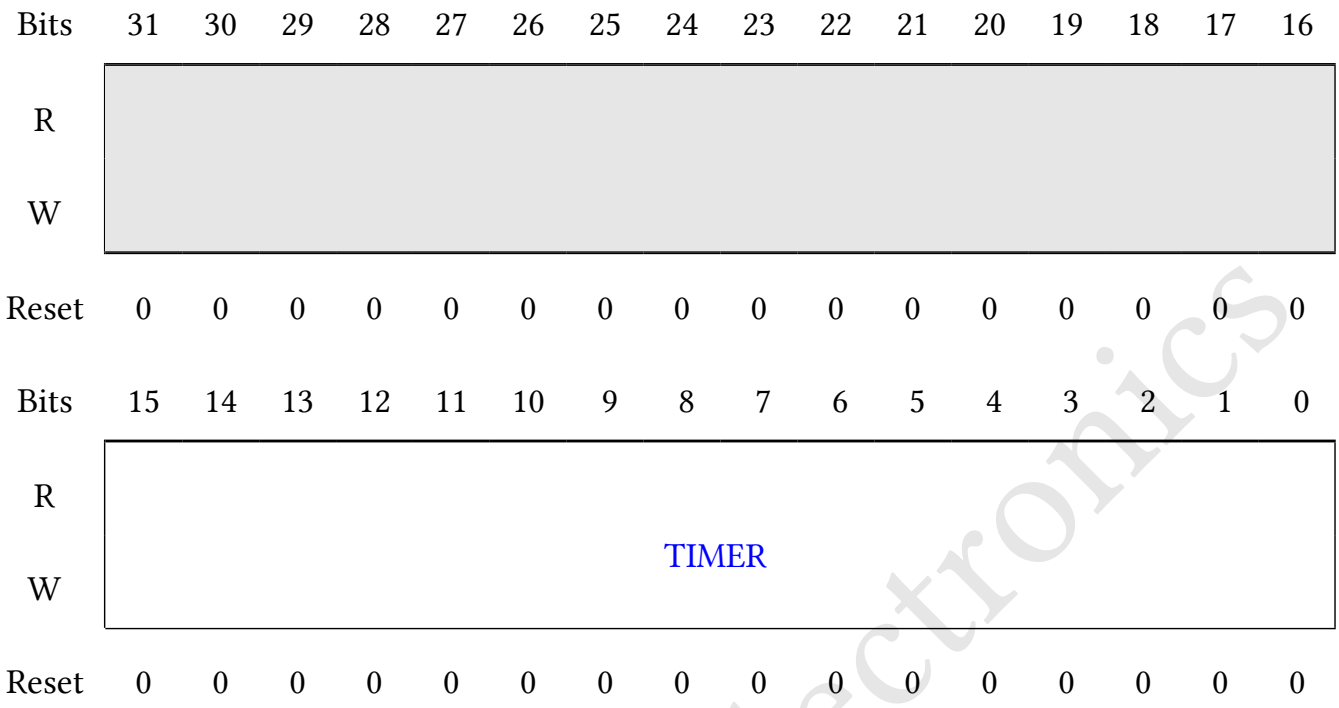


Table 17.6: CAN TIMER Register Description

Field	Function
15 - 0 TIMER	**Timer Value –Contains the free-running counter value.**

17.3.3.4 CAN RXMGMASK Register

The CAN_RXMGMASK register is located in RAM.

CAN_RXMGMASK is provided for legacy application support:

- When the CAN_MCR.IRMQ bit is negated, CAN_RXMGMASK is always in effect (the bits in the MG field will mask the mailbox filter bits).
- When the CAN_MCR.IRMQ bit is asserted, CAN_RXMGMASK has no effect (the bits in the MG field will not mask the mailbox filter bits).

CAN_RXMGMASK is used to mask the filter fields of all Rx MBs, excluding MBs 14–15, which have individual mask registers.

CAN_RXMGMASK can only be written in Freeze mode as it is blocked by hardware in other modes.

Yuntu Microelectronics

Table 17.7: CAN which MG Bits Mask Each Mailbox Filter Field

SMB.RTR[1]	CAN_CTRL2.RRS	CAN_CTRL2.RACEN	Mailbox Filter Feilds			
			MB.RTR	MB.IDE	MB.ID	Reserved
0	-	0	not [2]	note [3]	MG[28:0]	MG[31:29]
0	-	1	MG[31]	MG[30]	MG[28:0]	MG[29]
1	0	-	-	-	-	MG[31:0]
1	1	0	-	-	MG[28:0]	MG[31:29]
1	1	1	MG[31]	MG[30]	MG[28:0]	MG[29]

1. RTR bit of the incoming frame. It is saved into an auxiliary MB called Rx Serial Message Buffer (Rx SMB).
2. If CAN_CTRL2.EACEN is negated, the RTR bit of the mailbox is never compared with the RTR bit of the incoming frame.
3. If CAN_CTRL2.EACEN is negated, the IDE bit of the mailbox is always compared with the IDE bit of the incoming frame.

Offset: 10h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MG															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MG															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.8: CAN RXMGMASK Register Description

Field	Function
31 - 0	**Rx Mailboxes Global Mask Register**

Yuntu Microelectronics

Field	Function
MG	The MG bits mask the mailbox filter bits. Note that the alignment with the ID word of the mailbox is not perfect as the two most significant MG bits affect the fields RTR and IDE, which are located in the Control and Status (C/S) word of the mailbox. Details see table CAN which MG bits mask each mailbox filter field 0b - The corresponding bit in the filter is ”don’t care”. 1b - The corresponding bit in the filter is checked.

17.3.3.5 CAN RX14MASK Register

The CAN_RX14MASK register is located in RAM.

CAN_RX14MASK is provided for legacy application support. When the CAN_MCR.IRMQ bit is asserted, CAN_RX14MASK has no effect.

CAN_RX14MASK is used to mask the filter fields of message buffer 14.

CAN_RX14MASK can only be programmed when the FlexCAN is in Freeze mode as it is blocked by hardware in other modes.

Offset: 14h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RX14M															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RX14M															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.9: CAN RX14MASK Register Description

Field	Function
31 - 0	**Rx 14 Mask Register**

Yuntu Microelectronics

Field	Function
RX14M	Each mask bit masks the corresponding mailbox 14 filter field in the same way that CAN_RXMGMASK masks other mailboxes' filters. See the description of the CAN_RXMGMASK register. 0b - The corresponding bit in the filter is "don't care". 1b - The corresponding bit in the filter is checked.

17.3.3.6 CAN RX15MASK Register

The CAN_RX15MASK register is located in RAM.

CAN_RX15MASK is provided for legacy application support. When the CAN_MCR.IRMQ bit is asserted, CAN_RX15MASK has no effect.

CAN_RX15MASK is used to mask the filter fields of message buffer 15.

CAN_RX15MASK can be programmed only when the FlexCAN is in Freeze mode because it is blocked by hardware in other modes.

Offset: 18h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RX15M															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RX15M															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.10: CAN RX15MASK Register Description

Field	Function
31 - 0	**Rx Buffer 15 Mask Bits**

Yuntu Microelectronics

Field	Function
RX15M	Each mask bit masks the corresponding mailbox 15 filter field in the same way that CAN_RXMGMASK masks other mailboxes' filters. See the description of the CAN_RXMGMASK register. 0b - The corresponding bit in the filter is "don't care". 1b - The corresponding bit in the filter is checked.

17.3.3.7 CAN ECR Register

The CAN_ECR register has four 8-bit fields reflecting the value of the FlexCAN error counters:

- Transmit Error Counter (TXERRCNT field)
- Receive Error Counter (RXERRCNT field)
- Transmit Error Counter for errors detected in the Data Phase of CAN FD messages with the BRS bit set (TXERRCNT_FAST field)
- Receive Error Counter for errors detected in the Data Phase of CAN FD messages with the BRS bit set (RXERRCNT_FAST field)

The TXERRCNT and RXERRCNT counters take into account all errors in both CAN FD and non-FD message formats. TXERRCNT_FAST and RXERRCNT_FAST are dedicated to count only the errors that occur in the Data Phase of CAN FD frames with the BRS bit set.

The Fault Confinement State (FLTCONF field in CAN_ESR1) is updated based on the TXERRCNT and RXERRCNT counters only. The TXERRCNT and RXERRCNT counters can be written in Freeze mode only. The TXERRCNT_FAST and RXERRCNT_FAST counters are read-only except in Freeze mode, in which the CPU can write value zero. The rules for incrementing and decrementing these counters are described in the CAN protocol and are completely implemented in the FlexCAN module.

The basic rules for FlexCAN bus state transitions are:

- If the value of TXERRCNT or RXERRCNT increases to be greater than or equal to 128, the CAN_ESR1.FLTCONF field is updated to reflect the Error Passive state.
- If the FlexCAN state is Error Passive, and either TXERRCNT or RXERRCNT decrements to a value less than or equal to 127 while the other already satisfies this condition, the CAN_ESR1.FLTCONF field is updated to reflect the Error Active state.
- If the value of TXERRCNT increases to be greater than 255, the CAN_ESR1.FLTCONF field is updated to reflect the Bus Off state and an interrupt may be issued. The value of TXERRCNT is then reset to zero.
- If the FlexCAN is in the Bus Off state, then TXERRCNT is cascaded together with another internal counter to count the 128 occurrences of 11 consecutive recessive bits on the bus. Hence, TXERRCNT is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing TXERRCNT. When TXERRCNT reaches the value of 128, the CAN_ESR1.FLTCONF field is updated to be Error Active and both error counters are reset to zero. At any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the TXERRCNT value. The TXERRCNT_FAST counter is frozen during Bus Off.
- If during system startup, only one node is operating, then its TXERRCNT increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the

Yuntu Microelectronics

CAN_ESR1.ACKERR bit). After the transition to the Error Passive state, TXERRCNT is no longer incremented by acknowledge errors. Therefore, the device never goes to the Bus Off state.

- If RXERRCNT increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to return to the Error Active state.
- The TXERRCNT_FAST and RXERRCNT_FAST error counter values increment and decrement based on errors detected only in the Data Phase of CAN FD frames with the BRS bit set, following the same increment and decrement rules as the TXERRCNT and RXERRCNT counters. These counters do not wrap around and get stuck at their maximum value (255). They stop counting and keep their values frozen when the FlexCAN is in the Bus Off state. They are reset when the FlexCAN leaves the Bus Off state and restart counting after the FlexCAN returns to the Error Active state.
- When the FlexCAN is in Pretended Networking mode, RXERRCNT and RXERRCNT_FAST keep counting errors and error flags are stored, TXERRCNT and TXERRCNT_FAST preserve their values and do not change since no transmission occurs under Pretended Networking mode are updated in CAN_ECR adn CAN_ESR1 when the FlexCAN returns to Normal mode. The FAST error flags in CAN_ESR1 are not set if the FlexCAN is in Pretended Networking mode.

Offset: 1ch

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RXERRCNT_FAST								TXERRCNT_FAST							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RXERRCNT								TXERRCNT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.11: CAN ECR Register Description

Field	Function
31 - 24	**Receive Error Counter for Fast Bits**

Yuntu Microelectronics

Field	Function
RXERRCNT _FAST	Receive Error Counter for Fast Bits –Receive error counter for errors detected in the Data Phase of received CAN FD messages with the BRS bit set. The RXERRCNT_FAST counter is read-only except in Freeze mode, in which the CPU can write an 8-bit zero value only. RXERRCNT_FAST is not available (reserved) when configuration parameter FD_EN is 0.
23 - 16 TXERRCNT _FAST	**Transmit Error Counter for Fast Bits** Transmit Error Counter for Fast Bits –Transmit error counter for errors detected in the Data Phase of transmitted CAN FD messages with the BRS bit set. The TXERRCNT_FAST counter is read-only except in Freeze mode, in which the CPU can write a 8-bit zero value only. TXERRCNT_FAST is not available (reserved) when configuration parameter FD_EN is 0.
15 - 8 RXERRCNT	**Receive Error Counter** Receive Error Counter –Receive error counter for all errors detected in received messages. The RXERRCNT counter is read-only except in Freeze mode, in which it can be written by the CPU.
7 - 0 TXERRCNT	**Transmit Error Counter** Transmit Error Counter –Transmit error counter for all errors detected in transmitted messages. The TXERRCNT counter is read-only except in Freeze mode, in which it can be written by the CPU.

17.3.3.8 CAN ESR1 Register

The CAN_ESR1 register reports various error conditions detected in the reception and transmission of a CAN frame, some general status of the device, and is the source of some interrupts to the CPU.

The reported error conditions are:

- BIT1ERR, BIT0ERR, ACKKERR, CRCERR, FRMERR, and STFERR, for errors detected in CAN frames of any format.
- BIT1ERR_FAST, BIT0ERR_FAST, CRCERR_FAST, FRMERR_FAST, and STFERR_FAST for errors detected in the Data Phase of CAN FD frames with the BRS bit set only.

An error detected in a single CAN frame may be reported by one or more error flags. Also, error reporting is cumulative, in case additional error events occur in subsequent frames before the CPU reads CAN_ESR1.

TXWRN, RXWRN, IDLE, TX, FLTCONF, RX and SYNCH are status bits.

BOFFINT, BOFFDONEINT, ERRINT, ERRINT_FAST, WAKINT, TWRNINT, and RWRNINT are interrupt bits. It is recommended that the CPU use the following procedure when servicing interrupt requests generated by these bits:

Yuntu Microelectronics

1. Read CAN_ESR1 to capture all error condition and status bits. This action clears the respective bits that were set since the last read access.
2. Write 1 to clear the interrupt bit that has triggered the interrupt request.
3. Write 1 to clear the ERR_OVR bit if it is set.

Starting from all error flags cleared, a first error event sets either the ERRINT or the ERRINT_FAST (provided the corresponding mask bit is asserted). If other error events in subsequent frames occur before the CPU serves the interrupt request, the ERR_OVR bit is set to indicate that errors from different frames have accumulated.

Table 17.12: CAN State Related Bit Encodings

SYNCH	IDLE	TX	RX	FlexCAN State
0	0	0	0	Not synchronized to CAN bus
1	1	x	x	Idle
1	0	1	0	Transmitting
1	0	0	1	Receiving

Offset: 20h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BIT1ERR_FAST	BIT0ERR_FAST		CRCERR_FAST	FRMERR_FAST	STFERR_FAST					ERROVR	ERRINT_FAST	BOFFDONEINT	SYNCH	TWRNINT	RWRNINT
W											W1C	W1C	W1C		W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BIT1ERR	BIT0ERR	ACKERR	CRCERR	FRMERR	STFERR	TXWRN	RXWRN	IDLE	TX	FLTCONF		RX	BOFFINT	ERRINT	WAKINT
W														W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.13: CAN ESR1 Register Description

Field	Function
31 BIT1ERR_FAST	**Bit 1 Error in the Data Phase of CAN FD Frames with BRS Bit Set** BIT1ERR_FAST indicates when an inconsistency occurs between the transmitted and the received bit in the Data Phase of CAN FD frames with the BRS bit set. BIT1ERR_FAST is not available (reserved) when configuration parameter FD_EN is 0. 0b - No such occurrence. 1b - At least one bit sent as recessive is received as dominant.
30	**Bit 0 Error in the Data Phase of CAN FD Frames with BRS Bit Set**

Yuntu Microelectronics

Field	Function
BIT0ERR_ FAST	BIT0ERR_FAST indicates when an inconsistency occurs between the transmitted and the received bit in the Data Phase of CAN FD frames with the BRS bit set. BIT0ERR_FAST is not available (reserved) when configuration parameter FD_EN is 0. 0b - No such occurrence. 1b - At least one bit sent as recessive is received as dominant.
28 CRCERR_F AST	**CRC Error in the Data Phase of CAN FD Frames with BRS Bit Sets** CRCERR_FAST indicates that a CRC error has been detected by the receiver node in the CRC field of CAN FD frames with the BRS bit set; that is, the calculated CRC is different from the received CRC. CRCERR_FAST is not available (reserved) when configuration parameter FD_EN is 0. 0b - No such occurrence. 1b - A CRC error occurred since last read of this register.
27 FRMERR_F AST	**Form Error in the Data Phase of CAN FD Frames with BRS Bit Set** FRMERR_FAST indicates that a form error has been detected by the receiver node in the Data Phase of CAN FD frames with the BRS bit set; that is, a fixed-form bit field contains at least one illegal bit. FRMERR_FAST is not available (reserved) when configuration parameter FD_EN is 0. 0b - No such occurrence. 1b - A form error occurred since last read of this register.
26 STFERR_F AST	**Stuffing Error in the Data Phase of CAN FD Frames with BRS Bit Set** STFERR_FAST indicates that a stuffing error has been detected in the Data Phase of CAN FD frames with the BRS bit set. STFERR_FAST is not available (reserved) when configuration parameter FD_EN is 0. 0b - No such occurrence. 1b - A stuffing error occurred since last read of this register.
21 ERROVR	**Error Overrun –Indicates that an error condition occurred when any error flag is already set.** ERROVR is cleared by writing 1 to it. 0b - Overrun has not occurred. 1b - Overrun has occurred.
20	**Error Interrupt for Errors Detected in the Data Phase of CAN FD Frames with the BRS Bit Set**

Yuntu Microelectronics

Field	Function
ERRINT_FAST	ERRINT_FAST indicates that at least one of the error bits detected in the Data Phase of CAN FD frames with the BRS bit set (BIT1ERR_FAST, BIT0ERR_FAST, CRCERR_FAST, FRMERR_FAST, or STFERR_FAST) is set. If the corresponding mask bit (CAN_CTRL2.ERRMSK_FAST) is set, an interrupt is generated to the CPU. ERRINT_FAST is cleared by writing 1 to it. Writing 0 has no effect. ERRINT_FAST is not available (reserved) when configuration parameter FD_EN is 0. 0b - No such occurrence. 1b - Indicates setting of any error bit detected in the Data Phase of CAN FD frames with the BRS bit set.
19 BOFFDONEINT	**Bus Off Done Interrupt** BOFFDONEINT is set when the Tx Error Counter (TXERRCNT) has finished counting 128 occurrences of 11 consecutive recessive bits on the CAN bus and is ready to leave Bus Off. If the corresponding mask bit (CAN_CTRL2.BOFFDONEMSK) is set, an interrupt is generated to the CPU. BOFFDONEINT is cleared by writing 1 to it. Writing 0 has no effect. 0b - No such occurrence. 1b - FlexCAN module has completed Bus Off process.
18 SYNCH	**CAN Synchronous Status** SYNCH is a read-only flag that indicates whether the FlexCAN is synchronized to the CAN bus and able to participate in the communication process. It is set and cleared by the FlexCAN. See CAN state related bit encodings 0b - FlexCAN is not synchronized to the CAN bus. 1b - FlexCAN is synchronized to the CAN bus.
17 TWRNINT	**TX Warning Interrupt Flag** If CAN_MCR.WRNEN is asserted, the TWRNINT bit is set when the TXWRN flag transitions from 0 to 1, meaning that the Tx error counter reached 96. If the corresponding mask bit (CAN_CTRL1.TWRNMSK) is set, an interrupt is generated to the CPU. TWRNINT is cleared by writing 1 to it. Writing 0 has no effect. When CAN_MCR.WRNEN is negated, the TWRNINT flag is masked. The CPU must clear the TWRNINT flag before negating CAN_MCR.WRNEN. Otherwise, TWRNINT will be set when CAN_MCR.WRNEN is set again. TWRNINT is not generated during the Bus Off state and is not updated during Freeze mode. When the FlexCAN returns to Normal mode from Pretended Networking mode, TWRNINT is not updated. See ”Receive Process under Pretended Networking Mode”. 0b - No such occurrence. 1b - The Tx error counter transitioned from less than 96 to greater than or equal to 96.
16	**RX Warning Interrupt Flag**

Yuntu Microelectronics

Field	Function
RWRNINT	If CAN_MCR.WRNEN is asserted, the RWRNINT bit is set when the RXWRN flag transitions from 0 to 1, meaning that the Rx error counters reached 96. If the corresponding mask bit (CAN_CTRL1.RWRNMSK) is set, an interrupt is generated to the CPU. RWRNINT is cleared by writing it to 1 to it. Writing 0 has no effect. When CAN_MCR.WRNEN is negated, the RWRNINT flag is masked. The CPU must clear the RWRNINT flag before negating CAN_MCR.WRNEN. Otherwise, RWRNINT will be set when CAN_MCR.WRNEN is set again. RWRNINT is not updated during Freeze mode. When the FlexCAN returns to Normal mode from Pretended Networking mode, RWRNINT is updated to reflect the Rx error counter state. See "Receive Process under Pretended Networking Mode" 0b - No such occurrence. 1b - The Rx error counter transitioned from less than 96 to greater than or equal to 96.
15 BIT1ERR	**Bit 1 Error –Indicates when an inconsistency occurs between the transmitted and the received bit in a non-CAN FD message or in the arbitration or Data Phase of a CAN FD message.** NOTE: BIT1ERR is not set by a transmitter in case of arbitration field or ACK slot, or in case of a node sending a passive error flag that detects dominant bits. BIT1ERR is updated when the FlexCAN returns to Normal mode from Pretended Networking mode.
14 BIT0ERR	**Bit 0 Error–Indicates when an inconsistency occurs between the transmitted and the received bit in a non-CAN FD message or in the arbitration or Data Phase of a CAN FD message.** BIT0ERR is updated when the FlexCAN returns to Normal mode from Pretended Networking mode. 0b - No such occurrence. 1b - At least one bit sent as dominant is received as recessive.
13 ACKERR	**Acknowledge Error –Indicates that an Acknowledge error was detected by the transmitter node (that is, a dominant bit was detected during the ACK SLOT).** ACKERR is updated when the FlexCAN returns to Normal mode from Pretended Networking mode. 0b - No such occurrence. 1b - An ACK error occurred since last read of this register.
12	**Cyclic Redundancy Check Error –Indicates that a CRC error was detected by the receiver node either in a non-FD message or in the arbitration or Data Phase of a frame in CAN FD format (that is, the calculated CRC is different from the received CRC).**

Yuntu Microelectronics

Field	Function
CRCERR	CRCERR is updated when the FlexCAN returns to Normal mode from Pretended Networking mode. 0b - No such occurrence. 1b - A CRC error occurred since last read of this register.
11 FRMERR	**Form Error –Indicates that a form error was detected by the receiver node in a non-FD message or in a CAN FD message arbitration or Data Phase (that is, a fixed-form bit field contains at least one illegal bit).** FRMERR is updated when the FlexCAN returns to Normal mode from Pretended Networking mode. 0b - No such occurrence. 1b - A form error occurred since last read of this register.
10 STFERR	**Stuffing Error –Indicates that a stuffing error was detected by the receiver node in a non-FD message or in a CAN FD message arbitration or Data Phase.** STFERR is updated when the FlexCAN returns to Normal mode from Pretended Networking mode. 0b - No such occurrence. 1b - A stuffing error occurred since last read of this register.
9 TXWRN	**TX Error Warning –Indicates when repetitive errors are occurring during message transmission.** TXWRN is affected by the value of CAN_ECR.TXERRCNT only. TXWRN is not updated during Freeze mode. TXWRN is updated when the FlexCAN returns to Normal mode from Pretended Networking mode. 0b - No such occurrence. 1b - TXERRCNT is greater than or equal to 96.
8 RXWRN	**RX Error Warning** RXWRN is affected by the value of CAN_ECR.RXERRCNT only. RXWRN is not updated during Freeze mode. RXWRN is updated when the FlexCAN returns to Normal mode from Pretended Networking mode. 0b - No such occurrence. 1b - RXERRCNT is greater than or equal to 96.
7 IDLE	**IDLE** Indicates when the CAN bus is in the IDLE state. See CAN state related bit encodings 0b - No such occurrence. 1b - CAN bus is now IDLE.
6	**FlexCAN in Transmission**

Yuntu Microelectronics

Field	Function
TX	Indicates if the FlexCAN is transmitting a message. See CAN state related bit encodings 0b - FlexCAN is not transmitting a message. 1b - FlexCAN is transmitting a message.
5 - 4 FLTCONF	**FlexCAN Fault Confinement State –Indicates the fault confinement state of the FlexCAN module.** If CAN_CTRL1.LOM is asserted, after a delay that depends on the CAN bit timing, the FLTCONF field will indicate Error Passive. The same delay affects how FLTCONF reflects an update to the CAN_ECR register by the CPU. It may take up to one CAN bit time to get them coherent again. FLTCONF is affected by soft reset. However, if CAN_CTRL1.LOM is asserted, the FLTCONF reset value lasts just one CAN bit. After this time, FLTCONF indicates Error Passive. 00b - Error Active 01b - Error Passive 10b - Bus Off 11b - Bus Off
3 RX	**FlexCAN in Reception** Indicates if the FlexCAN is receiving a message. See CAN state related bit encodings 0b - FlexCAN is not receiving a message. 1b - FlexCAN is receiving a message.
2 BOFFINT	**Bus Off Interrupt** BOFFINT is set when the FlexCAN enters the Bus Off state. If the corresponding mask bit (CAN_CTRL1.BOFFMSK) is set, an interrupt is generated to the CPU. BOFFINT is cleared by writing 1 to it. Writing 0 has no effect. 0b - No such occurrence. 1b - FlexCAN module entered Bus Off state.
1 ERRINT	**Error Interrupt** ERRINT indicates that at least one of the error bits (BIT1ERR, BIT0ERR, ACKERR, CRCERR, FRMERR, or STFERR) is set. If the corresponding mask bit (CAN_CTRL1.ERRMSK) is set, an interrupt is generated to the CPU. ERRINT is cleared by writing 1 to it. Writing 0 has no effect. 0b - No such occurrence. 1b - Indicates setting of any error bit in CAN_ESR1.
0	**Wake-Up Interrupt –Applies when the FlexCAN is in Low-Power mode under Self Wake Up mechanism, either Doze mode or Deepsleep mode.**

Yuntu Microelectronics

Field	Function
WAKINT	When a recessive-to-dominant transition is detected on the CAN bus and the CAN_MCR.WAKMSK bit is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. When CAN_MCR.SLFWAK is negated, the WAKINT flag is masked. The CPU must clear the WAKINT flag before negating the CAN_MCR.SLFWAK bit. Otherwise, WAKINT will be set when CAN_MCR.SLFWAK is set again. Writing 0 has no effect. WAKINT is cleared by writing 1 to it. Writing 0 has no effect. 0b - No such occurrence. 1b - Indicates a recessive-to-dominant transition was received on the CAN bus.

17.3.3.9 CAN IMASK2 Register

The CAN_IMASK2 register allows any number of the 32 message buffer interrupts to be enabled or disabled for MB63 to MB32. CAN_IMASK2 contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception (as indicated by the corresponding CAN_IFLAG2 bit).
The CAN_IMASK2 register is not available if parameter NUMBER_OF_MB is less than 64.

Offset: 24h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BUF63TO32M															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUF63TO32M															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.14: CAN IMASK2 Register Description

Field	Function
31 - 0	**Buffer MB i Mask**

Yuntu Microelectronics

Field	Function
BUF63TO32M	Each bit enables or disables the corresponding FlexCAN Message Buffer Interrupt for MB63 to MB32. NOTE: Setting or clearing a bit in CAN_IMASK2 can assert or negate an interrupt request if the corresponding CAN_IFLAG2 bit is set. 0b - The corresponding buffer interrupt is disabled. 1b - The corresponding buffer interrupt is enabled.

17.3.3.10 CAN IMASK1 Register

The CAN_IMASK1 register allows any number of the 32 message buffer interrupts to be enabled or disabled for MB31 to MB. CAN_IMASK1 contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception (as indicated by the corresponding CAN_IFLAG1 bit).

Offset: 28h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BUF31TO0M															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUF31TO0M															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.15: CAN IMASK1 Register Description

Field	Function
31 - 0 BUF31TO0M	**Buffer MB i Mask** Each bit enables or disables the corresponding FlexCAN Message Buffer Interrupt for MB31 to MB0. NOTE: Setting or clearing a bit in CAN_IMASK1 can assert or negate an interrupt request if the corresponding CAN_IFLAG1 bit is set. 0b - The corresponding buffer interrupt is disabled. 1b - The corresponding buffer interrupt is enabled.

Yuntu Microelectronics

17.3.3.11 CAN IFLAG2 Register

The CAN_IFLAG2 register defines the flags for the 32 message buffer interrupts for MB63 to MB32. CAN_IFLAG2 contains one interrupt flag bit per buffer. Each successful transmission or reception sets the respective CAN_IFLAG2 bit. If the corresponding CAN_IMASK2 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.
Before updating the CAN_MCR.MAXMB field, the CPU must service the CAN_IFLAG2 bits whose MB value is greater than the CAN_MCR.MAXMB to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.
The CAN_IFLAG2 register is not available if parameter NUMBER_OF_MB is less than 64.

Offset: 2ch

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BUF63TO32I															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BUF63TO32I															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.16: CAN IFLAG2 Register Description

Field	Function
31 - 0 BUF63TO32I	**Buffer MB i Interrupt** Each bit flags the corresponding FlexCAN Message Buffer Interrupt for MB63 to MB32. 0b - The corresponding buffer has no occurrence of successfully completed transmission or reception. 1b - The corresponding buffer has successfully completed transmission or reception.

17.3.3.12 CAN IFLAG1 Register

The CAN_IFLAG1 register defines the flags for the 32 message buffer interrupts for MB31 to MB0. CAN_IFLAG1 contains one interrupt flag bit per buffer. Each successful transmission or reception sets

Yuntu Microelectronics

the corresponding CAN_IFLAG1 bit. If the corresponding CAN_IMASK1 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.

Before updating the CAN_MCR.MAXMB field, the CPU must service the CAN_IFLAG1 bits whose MB value is greater than the CAN_MCR.MAXMB to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

There is an exception when DMA for Legacy Rx FIFO is enabled, as described below.

Legacy Rx FIFO

The BUF7I to BUF5I flags are also used to represent Legacy FIFO interrupts when the Legacy Rx FIFO is enabled. When CAN_MCR.RFEN is set and CAN_MCR.DMA is negated, the function of the 8 least significant interrupt flags changes:

- BUF7I, BUF6I, and BUF5I indicate operating conditions of the Legacy FIFO.
- BUF0I is used to empty the Legacy FIFO.
- BUF4I to BUF1I are reserved.

Before setting CAN_MCR.RFEN, the CPU must service the IFLAG bits asserted in the Legacy Rx FIFO region (see ”Legacy Rx FIFO”). Otherwise, these IFLAG bits will mistakenly show the related MBs now belonging to FIFO as having contents to be serviced. When CAN_MCR.RFEN is negated, the Legacy FIFO flags must be cleared. The same care must be taken when a CAN_CTRL2.RFFN value is selected extending Legacy Rx FIFO filters beyond MB7. For example, when RFFN is 0x8, the MB0–23 range is occupied by Legacy Rx FIFO filters and related IFLAG bits must be cleared.

The Legacy Rx FIFO must be disabled when CAN_MCR.FDEN is set.

Legacy Rx FIFO with DMA

When both CAN_MCR.RFEN and CAN_MCR.DMA are asserted (DMA feature for Legacy Rx FIFO enabled), the function of the 8 least significant interrupt flags (BUF7I–BUF0I) are changed to support the DMA operation:

- BUF7I, BUF6I, and BUF4I–BUF1I are not used.
- BUF5I indicates the operating condition of the Legacy Rx FIFO.
- BUF5I does not generate a CPU interrupt, but generates a DMA request.
- BUF0I is used to empty the Legacy FIFO.

IMASK1 bits in the Legacy Rx FIFO region are not considered when CAN_MCR.DMA is asserted. In addition, the CPU must not clear the flag BUF5I when DMA is enabled. Before setting CAN_MCR.DMA, the CPU must service the IFLAGs asserted in the Rx FIFO region.

When CAN_MCR.DMA is negated, the Legacy Rx FIFO must be empty.

Offset: 30h

Yuntu Microelectronics

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BUF31TO8I															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BUF31TO8I								BUF7I	BUF6I	BUF5I	BUF4TO1I				BUF0I
W	W1C								W1C	W1C	W1C	W1C				W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.17: CAN IFLAG1 Register Description

Field	Function
31 - 8 BUF31TO8I	**Buffer MBi Interrupt** Each bit flags the corresponding FlexCAN Message Buffer Interrupt for MB31 to MB8. Only the lower 8 bits are available if parameter NUMBER_OF_MB is 16. 0b - The corresponding buffer has no occurrence of successfully completed transmission or reception. 1b - The corresponding buffer has successfully completed transmission or reception.
7 BUF7I	**Buffer MB7 Interrupt or Legacy Rx FIFO Overflow** When CAN_MCR.RFEN is cleared (Legacy Rx FIFO disabled), BUF71 flags the interrupt for MB7. When CAN_MCR.RFEN is set (Legacy Rx FIFO enabled), BUF7I represents Legacy Rx FIFO Overflow. In this case, BUF7I indicates that a message was lost because the Legacy Rx FIFO is full. Note that BUF7I is not asserted when the Legacy Rx FIFO is full and the message was captured by a mailbox. NOTE: BUF7I is cleared by the FlexCAN whenever CAN_MCR.RFEN is changed by a CPU write. 0b - No occurrence of MB7 completing transmission/reception when CAN_MCR.RFEN=0, or of Legacy Rx FIFO overflow when CAN_MCR.RFEN=1. 1b - MB7 completed transmission/reception when CAN_MCR.RFEN=0, or Legacy Rx FIFO overflow when CAN_MCR.RFEN=1.
6	**Buffer MB6 Interrupt or Legacy Rx FIFO Warning**

Yuntu Microelectronics

Field	Function
BUF6I	When CAN_MCR.RFEN is cleared (Legacy Rx FIFO disabled), BUF6I flags the interrupt for MB6. When CAN_MCR.RFEN is set (Legacy Rx FIFO enabled), BUF6I represents Legacy Rx FIFO Warning. In this case, BUF6I indicates when the number of unread messages in the Legacy Rx FIFO increases from 4 to 5 due to the reception of a new message, meaning that the Legacy Rx FIFO is almost full. Note that if BUF6I is cleared when the number of unread messages is greater than 4, BUF6I is not asserted again until the number of unread messages in the Legacy Rx FIFO decreases to 4 or less. NOTE: BUF6I is cleared by the FlexCAN whenever CAN_MCR.RFEN is changed by a CPU write. 0b - No occurrence of MB6 completing transmission/reception when CAN_MCR.RFEN=0, or of Legacy Rx FIFO almost full when CAN_MCR.RFEN=1. 1b - MB6 completed transmission/reception when CAN_MCR.RFEN=0, or Legacy Rx FIFO almost full when CAN_MCR.RFEN=1.
5 BUF5I	**Buffer MB5 Interrupt or Frames Available in Legacy Rx FIFO** When CAN_MCR.RFEN is cleared (Legacy Rx FIFO disabled), BUF5I flags the interrupt for MB5. When CAN_MCR.RFEN is set (Legacy Rx FIFO enabled), BUF5I represents Frames Available in Legacy Rx FIFO and indicates that at least one frame is available to be read from the Legacy Rx FIFO. When CAN_MCR.DMA is set, BUF5I generates a DMA request and the CPU must not clear BUF5I by writing 1 to it. NOTE: BUF5I is cleared by the FlexCAN whenever CAN_MCR.RFEN is changed by a CPU write. 0b - No occurrence of MB5 completing transmission/reception when CAN_MCR.RFEN=0, or of frame(s) available in the FIFO, when CAN_MCR.RFEN=1. 1b - MB5 completed transmission/reception when CAN_MCR.RFEN=0, or frame(s) available in the Legacy Rx FIFO when CAN_MCR.RFEN=1. BUF5I generates a DMA request when both CAN_MCR.RFEN and CAN_MCR.DMA are set.
4 - 1 BUF4TO1I	**Buffer MB Interrupt or Reserved** When CAN_MCR.RFEN is cleared (Legacy Rx FIFO disabled), BUF4TO1I flag the interrupts for MB4 to MB1. When CAN_MCR.RFEN is set (Legacy Rx FIFO enabled), BUF4TO1I flags are reserved. NOTE: BUF4TO1I are cleared by the FlexCAN whenever CAN_MCR.RFEN is changed by a CPU write. 0000b - The corresponding buffer has no occurrence of successfully completed transmission or reception when CAN_MCR.RFEN=0. 0001b - The corresponding buffer has successfully completed transmission or reception when CAN_MCR.RFEN=0.
0	**Buffer MB0 Interrupt or Clear FIFO Bit**

Yuntu Microelectronics

Field	Function
BUF0I	When CAN_MCR.RFEN is set (Legacy Rx FIFO enabled), BUF0I is used to trigger the clear FIFO operation. This operation empties Legacy Rx FIFO contents. Before performing this operation, the CPU must service all FIFO related IFLAGs. When CAN_MCR.DMA is set (DMA is enabled), this operation also clears the BUF5I flag and consequently aborts the DMA request. The clear FIFO operation occurs when the CPU writes 1 to BUF0I, which is only allowed in Freeze mode and is blocked by hardware in other conditions. When CAN_MCR.RFEN is cleared (Legacy Rx FIFO disabled), BUF0I flags the interrupt for MB0: 0b - MB0 has no occurrence of successfully completed transmission or reception when CAN_MCR.RFEN=0. 1b - MB0 has successfully completed transmission or reception when CAN_MCR.RFEN=0.

17.3.3.13 CAN CTRL2 Register

| The CAN_CTRL2 register complements CAN_CTRL1, providing control bits for memory write access in Freeze mode, for extending FIFO filter quantity, and for adjusting the operation of internal FlexCAN processes like matching and arbitration.

The contents of CAN_CTRL2 are not affected by soft reset.

Table 17.18: CAN TASD Reset Value

Number of MBs	TASD Reset Value(FD_EN=0)	TASD Reset Value(FD_EN=1)
16	22	Invalid configuration
32	20	20
64	16	16
96	12	12
128	8	2

Table 17.19: CAN Number of Legacy of RX FIFO Filters

RFFN[3:0]	Number of Rx FIFO Filter Elements	Message Buffers Occupied by Rx FIFO and ID Filter Table	Remaining Available Mailboxes	Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks	RX FIFO ID Filter Table Elements Affected by Rx FIFO Global Mask
0x0	8	MB 0-7	MB 8-N	Elements 0-7	None
0x1	16	MB 0-9	MB 10-N	Elements 0-9	Elements 10-15
0x2	24	MB 0-11	MB 12-N	Elements 0-11	Elements 12-23
0x3	32	MB 0-13	MB 14-N	Elements 0-13	Elements 14-31
0x4	40	MB 0-15	MB 16-N	Elements 0-15	Elements 16-39
0x5	48	MB 0-17	MB 18-N	Elements 0-17	Elements 18-47
0x6	56	MB 0-19	MB 20-N	Elements 0-19	Elements 20-55
0x7	64	MB 0-21	MB 22-N	Elements 0-21	Elements 22-63
0x8	72	MB 0-23	MB 24-N	Elements 0-23	Elements 24-71
0x9	80	MB 0-25	MB 26-N	Elements 0-25	Elements 26-79
0xA	88	MB 0-27	MB 28-N	Elements 0-27	Elements 28-87
0xB	96	MB 0-29	MB 30-N	Elements 0-29	Elements 30-95
0xC	104	MB 0-31	MB 32-N	Elements 0-31	Elements 32-103
0xD	112	MB 0-33	MB 34-N	Elements 0-31	Elements 32-111
0xE	120	MB 0-35	MB 36-N	Elements 0-31	Elements 32-119
0xF	128	MB 0-37	MB 38-N	Elements 0-31	Elements 32-127

Offset: 34h

Yuntu Microelectronics

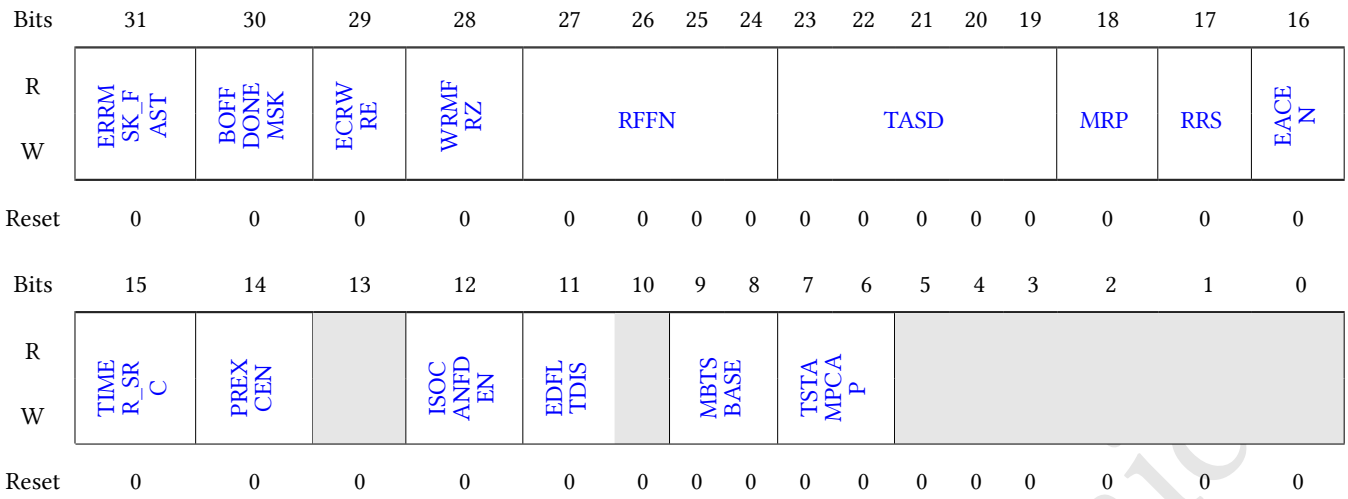


Table 17.20: CAN CTRL2 Register Description

Field	Function
31 ERRMSK_FAST	**Error Interrupt Mask for errors detected in the Data Phase of fast CAN FD frames** ERRMSK_FAST provides a mask for the ERRINT_FAST Interrupt in CAN_ESR1. ERRMASK_FAST is not available (reserved, write 0) if parameter FD_EN is 0. 0b - ERRINT_FAST error interrupt disabled. 1b - ERRINT_FAST error interrupt enabled.
30 BOFFDONEMSK	**Bus Off Done Interrupt Mask** Provides a mask for the Bus Off done interrupt in CAN_ESR1. 0b - Bus Off done interrupt disabled 1b - Bus Off done interrupt enabled.
29 ECRWRE	**Error-Correction Configuration Register Write Enable –Enables the CAN_MECR register to be updated.** ECRWRE is automatically set to 0 if the protocol described in section "Detection and Correction of Memory Errors" is not followed. ECRWRE is not available (reserved, write 0) if parameter ECC_EN is 0. 0b - Disable update. 1b - Enable update.
28 WRMFRZ	**Write-Access To Memory In Freeze Mode –Enable unrestricted write access to FlexCAN memory in Freeze mode.** WRMFRZ can only be written in Freeze mode and has no effect out of Freeze mode. CAN_MCR.RFEN bit must not be set during FlexCAN memory initialization. WRMFRZ is not available (reserved, write 0) if parameter ECC_EN is 0. 0b - Maintain the write access restrictions. 1b - Enable unrestricted write access to FlexCAN memory.
27 - 24	

Yuntu Microelectronics

Field	Function
RFFN	Number of Legacy Rx FIFO Filters –Defines the number of Legacy Rx FIFO filters. The maximum selectable number of filters is a function of configuration parameter NUMBER_OF_MB. RFFN must not be programmed with values that cause the number of message buffers occupied by Legacy Rx FIFO and ID Filter to exceed the number of mailboxes present, defined by CAN_MCR.MAXMB. Each group of eight filters occupies a memory space equivalent to two message buffers, which means that as more filters are implemented fewer mailboxes are available. Considering that the Legacy Rx FIFO occupies the memory space originally reserved for MB0–5, RFFN should be programmed with a value corresponding to a number of filters not greater than the number of available memory words, which can be calculated as follows: (SETUP_MB - 6) × 4 where SETUP_MB is the lower value between parameter NUMBER_OF_MB and register field CAN_MCR.MAXMB. The number of remaining mailboxes available will be: (SETUP_MB - 8) - (RFFN × 2) If the number of Legacy Rx FIFO filters programmed through RFFN exceeds the SETUP_MB value (memory space available), the exceeding ones will not be functional. NOTE: The number of the last remaining available mailbox is defined by the least value between (NUMBER_OF_MB - 1) and the CAN_MCR.MAXMB field. If Rx Individual Mask registers are not enabled, all Legacy Rx FIFO filters are affected by the Legacy Rx FIFO Global Mask. RFFN can only be written in Freeze mode as it is blocked by hardware in other modes.
23 - 19 TASD	**Tx Arbitration Start Delay –Indicates how many CAN bits the Tx arbitration process start point can be delayed from the first bit of CRC field on CAN bus.** TASD can be written only in Freeze mode because it is blocked by hardware in other modes. See ”Tx Arbitration Start Delay” for more details.
18 MRP	**Mailboxes Reception Priority** If MRP is set, the matching process starts from the mailboxes and if no match occurs the matching continues on the Legacy Rx FIFO. MRP can be written only in Freeze mode because it is blocked by hardware in other modes. 0b - Matching starts from Legacy Rx FIFO and continues on mailboxes. 1b - Matching starts from mailboxes and continues on Legacy Rx FIFO.
17	**Remote Request Storing**

Yuntu Microelectronics

Field	Function
RRS	If RRS is asserted, a Remote Request Frame is submitted to a matching process and stored in the corresponding message buffer in the same fashion as a Data Frame. No automatic Remote Response Frame will be generated. If RRS is negated, the Remote Request Frame is submitted to a matching process and an automatic Remote Response Frame is generated if a message buffer with CODE=0b1010 is found with the same ID. RRS can be written only in Freeze mode because it is blocked by hardware in other modes. 0b - Remote Response Frame is generated. 1b - Remote Request Frame is stored.
16 EACEN	**Entire Frame Arbitration Field Comparison Enable For Rx Mailboxes** EACEN controls the comparison of IDE and RTR bits within Rx mailboxes filters with their corresponding bits in the incoming frame by the matching process. EACEN does not affect matching for Legacy Rx FIFO. EACEN can be written only in Freeze mode because it is blocked by hardware in other modes. 0b - Rx mailbox filter’ s IDE bit is always compared and RTR is never compared, regardless of mask bits. 1b - Enables the comparison of both the IDE and RTR bits of an Rx mailbox filter with the corresponding bits of the incoming frame. Mask bits do apply.
15 TIMER_SRC	**Timer Source –Selects the time tick source used for incrementing the free-running timer counter.** TIMER_SRC can be written in Freeze mode only. 0b - The free-running timer is clocked by the CAN bit clock, which defines the baud rate on the CAN bus. 1b - The free-running timer is clocked by an external time tick. The period can be either adjusted to be equal to the baud rate on the CAN bus or a different value as required.
14 PREXCEN	**Protocol Exception Enable –Enables the protocol exception feature.** PREXCEN is writable only in Freeze mode. PREXCEN is not available (reserved, write 0) if parameter FE_EN is 0. 0b - Protocol exception is disabled. 1b - Protocol exception is enabled.
12	**ISO CAN FD Enable**

Yuntu Microelectronics

Field	Function
ISOCANFD EN	ISOCANFDEN enables ISO CAN FD compliant operation by enabling the following features, which are part of the ISO 11898 standard and not included in the original (Bosch) CAN FD protocol specification: • The count of variable stuff bits inserted from the Start of Frame bit to the last bit of Data field. Also, the modulo 8 count of variable stuff bits plus the respective parity bit (even parity calculated over the 3-bit modulo 8 count) are combined as the 4-bit Stuff Count field and inserted before the CRC Sequence field. • CRC calculation extends beyond the end of Data field and takes the Stuff Count field bits into account. ISOCANFDEN is writable only in Freeze mode. ISOCANFDEN is not available (reserved, write 0) if parameter FD_EN is 0. 0b - Disable ISO CAN FD specific features (non-ISO CAN FD operation). 1b - Enable ISO CAN FD compliant operation.
11 EDFLTDIS	**Edge Filter Disable –Disables the edge filter used during the bus integration state.** When the edge filter is enabled, two consecutive nominal time quanta with dominant bus state are required to detect an edge that causes synchronization. When synchronization occurs, the counting of the sequence of eleven consecutive recessive bits is restarted. The edge filter prevents the dominant pulses that are shorter than a nominal bit time (present during the data phase of an FD Frame) from being mistaken for an idle condition. EDFLTDIS is writable only in Freeze mode. EDFLTDIS is not available (reserved, write 0) if parameter FD_EN is 0. 0b - Edge filter is enabled. 1b - Edge filter is disabled.
9 - 8 MBTSBASE	**Message Buffer Time Stamp Base** MBTSBASE selects which time base is used for capturing the 16-bit TIME STAMP field of the message buffer. MBTSBASE is writable only in Freeze mode. MBTSBASE is not available (reserved, write 0) if parameter TIME_STAMP_EN is 0. 00b - Message buffer time stamp base is CAN_TIMER. 01b - Message buffer time stamp base is lower 16 bits of highresolution timer. 10b - Message buffer time stamp base is upper 16 bits of high-resolution timer. 11b - Reserved.
7 - 6	**Time Stamp Capture Point**

Yuntu Microelectronics

Field	Function
TSTAMPCAP	TSTAMPCAP configures the point in time at which a 32-bit time base is captured during a CAN frame and stored in the highresolution time stamp register (CAN_HR_TIME_STAMPn). For Classical CAN frames, capture points can be the start of frame (SOF) bit or the point in time when a CAN frame is considered valid, which is the 7th bit of end of frame for transmission and the 6th bit of the end of frame for reception. For CAN FD frames, the high-resolution time stamp can be captured at SOF, the point in time when a CAN FD frame is considered valid, or the res bit. TSTAMPCAP is writable only in Freeze mode. TSTAMPCAP is not available (reserved, write 0) if parameter TIME_STAMP_EN is 0. 00b - The high-resolution time stamp capture is disabled. 01b - The high-resolution time stamp is captured at the end of the CAN frame. 10b - The high-resolution time stamp is captured at the start of the CAN frame. 11b - The high-resolution time stamp is captured at the start of frame for classical CAN frames and at the res bit for CAN FD frames.

17.3.3.14 CAN ESR2 Register

The CAN_ESR2 register reports some general status information

Offset: 38h

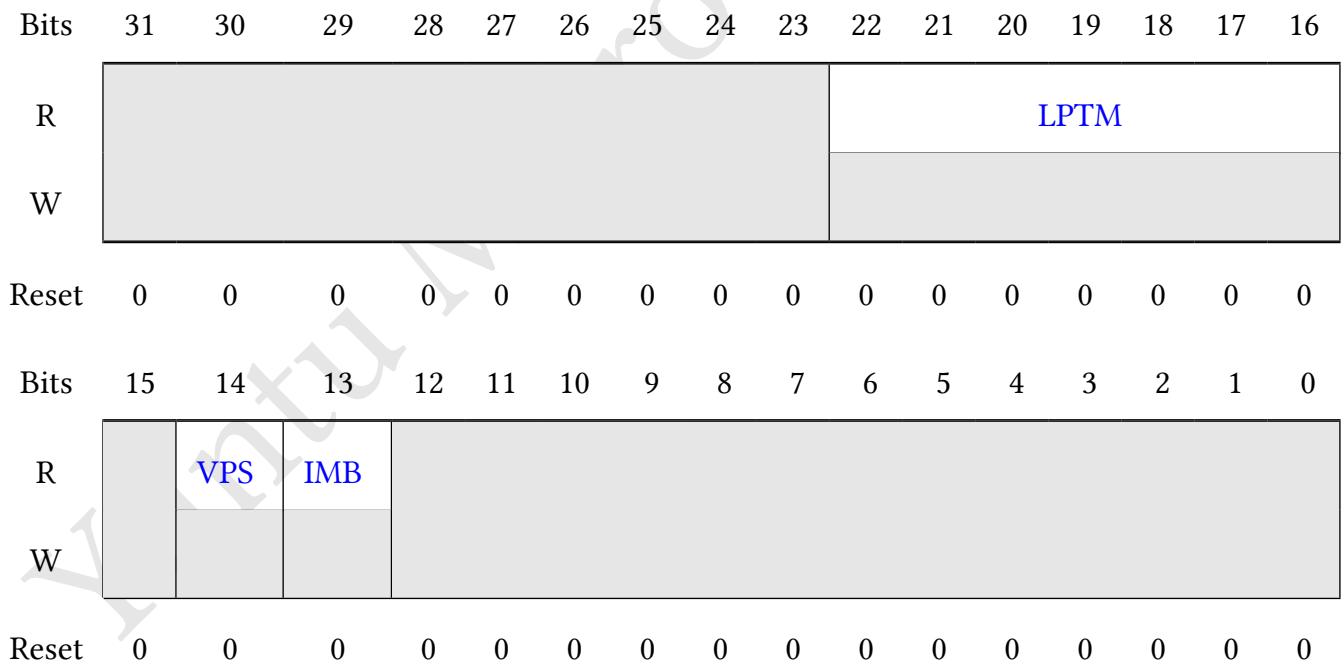


Table 17.21: CAN ESR2 Register Description

Field	Function
22 - 16	**Lowest Priority Tx Mailbox**

Field	Function
LPTM	If CAN_ESR2.VPS is asserted, LPTM indicates the lowest number inactive mailbox (see the CAN_ESR2.IMB bit description). If there is no inactive mailbox, then the mailbox indicated depends on the CAN_CTRL1.LBUF bit value: • If CAN_CTRL1.LBUF is negated, the mailbox indicated is the one that has the greatest arbitration value (see "Highest-Priority Mailbox First"). • If CAN_CTRL1.LBUF is asserted, the mailbox indicated is the highest number active Tx mailbox. If a Tx mailbox is being transmitted, it is not considered in LPTM calculation. If CAN_ESR2.IMB is not asserted and a frame is transmitted successfully, LPTM is updated with its mailbox number.
14 VPS	**Valid Priority Status –Indicates whether CAN_ESR2.IMB and CAN_ESR2.LPTM contents are currently valid or not.** VPS is asserted upon every complete Tx arbitration process unless the CPU writes to the C/S word of a mailbox that has already been scanned (is behind Tx Arbitration Pointer during the Tx arbitration process). If there is no inactive mailbox and only one Tx mailbox that is being transmitted, then VPS is not asserted. VPS is negated upon the start of every Tx arbitration process or upon a write to the C/S word of any mailbox. NOTE: VPS is not affected by any CPU write into the C/S of an MB that is blocked by the abort mechanism. When CAN_MCR.AEN is asserted, the abort code write into the C/S of an MB that is being transmitted (pending abort), or any write attempt into a Tx MB with CAN_IFLAG set is blocked. 0b - Contents of IMB and LPTM are invalid. 1b - Contents of IMB and LPTM are valid.
13 IMB	**Inactive Mailbox** If CAN_ESR2.VPS is asserted, IMB indicates whether there is any inactive mailbox (CODE field is either 0b1000 or 0b0000). IMB is asserted in the following cases: • During arbitration, if a CAN_ESR2.LPTM is found and it is inactive. • If CAN_ESR2.MB is not asserted and a frame is transmitted successfully. IMB is always cleared at start of arbitration (See "Arbitration Process"). NOTE: If an MB is successfully transmitted and CAN_ESR2.IMB=0 (no inactive mailbox), then CAN_ESR2.VPS and CAN_ESR2.IMB are asserted and the index related to the MB just transmitted is loaded into CAN_ESR2.LPTM. 0b - If CAN_ESR2.VPS is asserted, CAN_ESR2.LPTM is not an inactive mailbox. 1b - If CAN_ESR2.VPS is asserted, there is at least one inactive mailbox. CAN_ESR2.LPTM content is the number of the first one.

17.3.3.15 CAN CRCR Register

The CAN_CRCR register provides information about the CRC of transmitted messages for non-FD messages. CAN_CRCR only reports the 15 low order bits of CRC calculations for messages in CAN FD format that require either 17 or 21 bits. For CAN FD format frames, the CAN_FDCRC register must be used. CAN_CRCR is updated at the same time the Tx interrupt flag is asserted. The CAN_FDCRC register is not available when parameter FD_EN is 0.

Yuntu Microelectronics

Offset: 44h

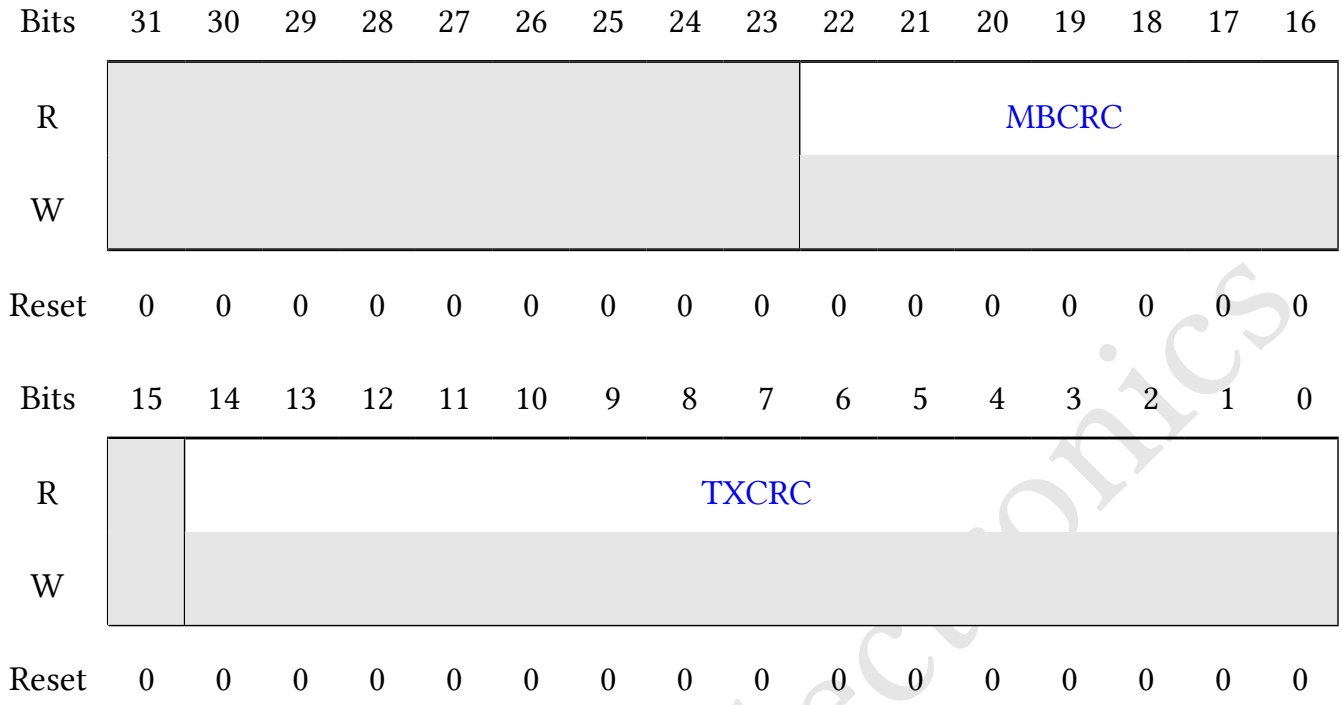


Table 17.22: CAN CRCR Register Description

Field	Function
22 - 16 MBCRC	**CRC Mailbox** Indicates the number of the mailbox corresponding to the value in the CAN_CRCR.TXCRC field.
14 - 0 TXCRC	**Transmitted CRC Value** Indicates the CRC value of the last transmitted message for non-FD frames. For CAN FD frames, the CRC value is reported in CAN_FDCRC.

17.3.3.16 CAN RXFGMASK Register

| The CAN_RXFGMASK register is located in RAM.

If the Rx FIFO is enabled, CAN_RXFGMASK is used to mask the Rx FIFO ID Filter Table elements that do not have a corresponding CAN_RXIMR according to CAN_CTRL2.RFFN field setting.

CAN_RXFGMASK can only be written in Freeze mode as it is blocked by hardware in other modes.

Yuntu Microelectronics

Table 17.23: CAN Correspondence of Legacy RX FIFO Global Mask Bits to IDF Fields

Legacy Rx FIFO ID Filter Table Elements Format (CAN_MCR.IDAM)	Identifier Acceptance Filter Fields					
	RTR	IDE	RXIDA	RXIDB[1]	RXIDC[2]	Reserved
A	FGM[31]	FGM[30]	FGM[29:1]	-		FGM[0]
B	FGM[31], FGM[15]	FGM[30], FGM[14]		FGM[29:16], FGM[13:0]	-	
C	-	-	-	-	FGM[31:24], FGM[23:16], FGM[15:8], FGM[7:0]	-

1. If CAN_MCR.IDAM is equivalent to format B, only the fourteen most significant bits of the Identifier of the incoming frame are compared with the Legacy Rx FIFO filter.
2. If CAN_MCR.IDAM field is equivalent to format C, only the eight most significant bits of the Identifier of the incoming frame are compared with the Legacy Rx FIFO filter.

Offset: 48h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FGM															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FGM															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.24: CAN RXFGMASK Register Description

Field	Function
31 - 0	**Legacy Rx FIFO Global Mask Bits**

Yuntu Microelectronics

Field	Function
FGM	These bits mask the ID Filter Table elements bits in a perfect alignment. 0b - The corresponding bit in the filter is ”don’ t care”. 1b - The corresponding bit in the filter is checked.

17.3.3.17 CAN RXFIR Register

The CAN_RXFIR register provides information on the Legacy Rx FIFO.
CAN_RXFIR is the port through which the CPU accesses the output of the RXFIR FIFO located in RAM. The RXFIR FIFO is written by the FlexCAN whenever a new message is moved into the Legacy Rx FIFO and its output is updated whenever the output of the Legacy Rx FIFO is updated with the next message. See ”Legacy Rx FIFO” for instructions on reading CAN_RXFIR.

Offset: 4ch

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R								IDHIT								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.25: CAN RXFIR Register Description

Field	Function
8 - 0 IDHIT	**Identifier Acceptance Filter Hit Indicator** Indicates which Identifier Acceptance Filter was hit by the received message that is in the output of the Legacy Rx FIFO. If multiple filters match the incoming message ID, the first matching IDAF found (lowest number) by the matching process is indicated. IDHIT is valid only when CAN_IFLAG1.BUF5I is asserted.

17.3.3.18 CAN CBT Register

The CAN_CBT register is an alternative way to store the CAN bit timing variables described in the CAN_CTRL1 register. EPRESDIV, EPROPSEG, EPSEG1, EPSEG2, and ERJW are extended versions of CAN_CTRL1 register PRESDIV, PROPSEG, PSEG1, PSEG2, and RJW fields respectively.
The BTF bit selects the use of the timing variables defined in CAN_CBT.
The contents of CAN_CBT are not affected by soft reset.
When CAN_CTRL2.BTE = 1, the EPRESDIV, ERJW, EPROPSEG, EPSEG1, and EPSEG2 fields in CAN_CBT are read as zero and a write operation to them has no effect.

NOTE: The CAN bit variables in CAN_CTRL1 and in CAN_CBT are stored in the same register.
NOTE: When the CAN FD feature is enabled (CAN_MCR.FDEN is set), always set $
$ CAN_CBT.BTF.

Offset: 50h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BTF	EPRESDIV										ERJW				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EPROPSEG						EPSEG1					EPSEG2				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.26: CAN CBT Register Description

Field	Function
31 BTF	**Bit Timing Format Enable –Enables the use of extended CAN bit timing fields EPRESDIV, EPROPSEG, EPSEG1, EPSEG2, and ERJW, replacing the CAN bit timing variables defined in CAN_CTRL1.** BTF can be written in Freeze mode only. 0b - Extended bit time definitions disabled. 1b - Extended bit time definitions enabled.
30 - 21	**Extended Prescaler Division Factor**

Yuntu Microelectronics

Field	Function
EPRESDIV	When CAN_CBT.BTF is asserted, EPRESDIV defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency. EPRESDIV extends the CAN_CTRL1.PRESDIV value range. Sclock frequency = PE clock frequency / (EPRESDIV + 1) The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency (See "Protocol Timing"). When CAN_CBT.BTF is negated, EPRESDIV has no effect. EPRESDIV can be written only in Freeze mode because it is blocked by hardware in other modes.
20 - 16 ERJW	**Extended Resync Jump Width** When CAN_CBT.BTF is asserted, ERJW defines the maximum number of time quanta that a bit time can be changed by one resynchronization. ERJW extends the CAN_CTRL1.RJW value range. Resync Jump Width = ERJW + 1 One time quantum is equal to the Sclock period. When CAN_CBT.BTF is negated, ERJW has no effect. ERJW can be written only in Freeze mode because it is blocked by hardware in other modes.
15 - 10 EPROPSEG	**Extended Propagation Segment** When CAN_CBT.BTF is asserted, EPROPSEG defines the length of the Propagation Segment in the bit time. EPROPSEG extends the CAN_CTRL1.PROPSEG value range. Propagation Segment Time = (EPROPSEG + 1) × Time-Quanta Time-Quantum = one Sclock period When CAN_CBT.BTF is negated, EPROPSEG has no effect. EPROPSEG can be written only in Freeze mode because it is blocked by hardware in other modes.
9 - 5 EPSEG1	**Extended Phase Segment 1** When CAN_CBT.BTF is asserted, EPSEG1 defines the length of Phase Segment 1 in the bit time. EPSEG1 extends the CAN_CTRL1.PSEG1 value range. Phase Buffer Segment 1 = (EPSEG1 + 1) × Time-Quanta Time-Quantum = one Sclock period When CAN_CBT.BTF is negated, EPSEG1 has no effect. EPSEG1 can be written only in Freeze mode because it is blocked by hardware in other modes.
4 - 0	**Extended Phase Segment 2**

Yuntu Microelectronics

Field	Function
EPSEG2	When CAN_CBT.BTF is asserted, EPSEG2 defines the length of Phase Segment 2 in the bit time. EPSEG2 extends the CAN_CTRL1.PSEG2 value range. Phase Buffer Segment 1 = (EPSEG2 + 1) × Time-Quanta Time-Quantum = one Sclock period When CAN_CBT.BTF is negated, EPSEG2 has no effect. EPSEG2 can be written only in Freeze mode because it is blocked by hardware in other modes.

17.3.3.19 CAN IMASK4 Register

The CAN_IMASK4 register allows any number of the 32 message buffer interrupts to be enabled or disabled for MB127 to MB96. CAN_IMASK4 contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception (as indicated by the corresponding CAN_IFLAG4 bit).
The CAN_IMASK4 register is not available if parameter NUMBER_OF_MB is less than 128.

Offset: 68h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BUF127TO96M															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUF127TO96M															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.27: CAN IMASK4 Register Description

Field	Function
31 - 0	**Buffer MB i Mask**

Yuntu Microelectronics

Field	Function
BUF127TO 96M	Each bit enables or disables the corresponding FlexCAN Message Buffer Interrupt for MB127 to MB96. When CAN FD is enabled, the MB range is defined according to the MBDSRn bit fields of the CAN_FDCTRL register. NOTE: Setting or clearing a bit in CAN_IMASK4 can assert or negate an interrupt request if the corresponding CAN_IFLAG4 bit is set. 0b - The corresponding buffer interrupt is disabled. 1b - The corresponding buffer interrupt is enabled.

17.3.3.20 CAN IMASK3 Register

The CAN_IMASK3 register allows any number of the 32 message buffer interrupts to be enabled or disabled for MB95 to MB64. CAN_IMASK3 contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception (as indicated by the corresponding CAN_IFLAG3 bit).
The CAN_IMASK3 register is not available if parameter NUMBER_OF_MB is less than 96.

Offset: 6ch

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BUF95TO64M															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUF95TO64M															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.28: CAN IMASK3 Register Description

Field	Function
31 - 0	**Buffer MB i Mask**

Yuntu Microelectronics

Field	Function
BUF95TO64M	Each bit enables or disables the corresponding FlexCAN Message Buffer Interrupt for MB95 to MB64. When CAN FD is enabled, the MB range is defined according to the MBDSRn bit fields of the CAN_FDCTRL register. NOTE: Setting or clearing a bit in CAN_IMASK3 can assert or negate an interrupt request if the corresponding CAN_IFLAG3 bit is set. 0b - The corresponding buffer interrupt is disabled. 1b - The corresponding buffer interrupt is enabled.

17.3.3.21 CAN IFLAG4 Register

The CAN_IFLAG4 register defines the flags for the 32 message buffer interrupts for MB127 to MB96. CAN_IFLAG4 contains one interrupt flag bit per buffer. Each successful transmission or reception sets the respective CAN_IFLAG4 bit. If the corresponding CAN_IMASK4 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.
Before updating the CAN_MCR.MAXMB field, the CPU must service the CAN_IFLAG4 bits whose MB value is greater than the CAN_MCR.MAXMB to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.
The CAN_IFLAG4 register is not available if parameter NUMBER_OF_MB is less than 128.

Offset: 70h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BUF127TO96															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BUF127TO96															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.29: CAN IFLAG4 Register Description

Field	Function
31 - 0	**Buffer MB Interrupt**

Yuntu Microelectronics

Field	Function
BUF127TO96	Each bit flags the corresponding FlexCAN Message Buffer Interrupt for MB127 to MB96. When CAN FD is enabled, the MB range is defined according to the MBDSRn bit fields of the CAN_FDCTRL register. 0b - The corresponding buffer has no occurrence of successfully completed transmission or reception. 1b - The corresponding buffer has successfully completed transmission or reception.

17.3.3.22 CAN IFLAG3 Register

The CAN_IFLAG3 register defines the flags for the 32 Message Buffer interrupts for MB95 to MB64. CAN_IFLAG3 contains one interrupt flag bit per buffer. Each successful transmission or reception sets the respective CAN_IFLAG3 bit. If the corresponding CAN_IMASK3 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.
Before updating the CAN_MCR.MAXMB field, the CPU must service the CAN_IFLAG3 bits whose MB value is greater than the CAN_MCR.MAXMB to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.
The CAN_IFLAG3 register is not available if parameter NUMBER_OF_MB is less than 96.

Offset: 74h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BUF95TO64															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BUF95TO64															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.30: CAN IFLAG3 Register Description

Field	Function
31 - 0	**Buffer MB i Interrupt**

Yuntu Microelectronics

Field	Function
BUF95TO64	Each bit flags the corresponding FlexCAN Message Buffer Interrupt for MB95 to MB64. When CAN FD is enabled, the MB range is defined according to the MBDSRn bit fields of the CAN_FDCTRL register. 0b - The corresponding buffer has no occurrence of successfully completed transmission or reception. 1b - The corresponding buffer has successfully completed transmission or reception.

17.3.3.23 CAN RAM Register

Offset: 80h + n * 4h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RAM															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RAM															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.31: CAN RAM Register Description

Field	Function
31 - 0 RAM	

17.3.3.24 CAN RXIMR Register

The RX Individual Mask Registers are used to store the acceptance masks for ID filtering in Rx MBs and the Legacy Rx FIFO.
When the Legacy Rx FIFO is disabled (CAN_MCR.RFEN is negated), an individual mask is provided for each available Rx mailbox on a one-to-one correspondence. When the Legacy Rx FIFO is enabled

Yuntu Microelectronics

(CAN_MCR.RFEN is asserted), an individual mask is provided for each Legacy Rx FIFO ID Filter Table Element on a one-to-one correspondence depending on the setting of $
$ CAN_CTRL2.RFFN (see "Legacy Rx FIFO").
CAN_RXIMR0 stores the individual mask associated with either MB0 or ID Filter Table Element 0, CAN_RXIMR1 stores the individual mask associated with either MB1 or ID Filter Table Element 1, and so on.
CAN_RXIMRn registers can only be accessed by the CPU when the FlexCAN is in Freeze mode; otherwise, they are blocked by hardware. The CAN_RXIMRn registers are not affected by reset. They are located in RAM and must be explicitly initialized prior to any reception.
The CAN_RXIMRn registers are not available if parameter IRMQ_EN is 0.

Offset: 880h + n * 4h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MI															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MI															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.32: CAN RXIMR Register Description

Field	Function
31 - 0 MI	**Individual Mask Bits** Each Individual Mask Bit masks the corresponding bit in both the mailbox filter and Legacy Rx FIFO ID Filter Table element in distinct ways. For mailbox filters, see the RXMGMASK register description. For Legacy Rx FIFO ID Filter Table elements, see the RXFGMASK register description. 0b - The corresponding bit in the filter is "don't care". 1b - The corresponding bit in the filter is checked.

Yuntu Microelectronics

17.3.3.25 CAN MECR Register

The CAN_MECR register contains control bits for memory error detection and correction (ECC).

NOTE: When CAN_CTRL2.ECRWRE is 0, writes to CAN_MECR are blocked, with the exception of the ECRWRDIS bit.

Offset: ae0h

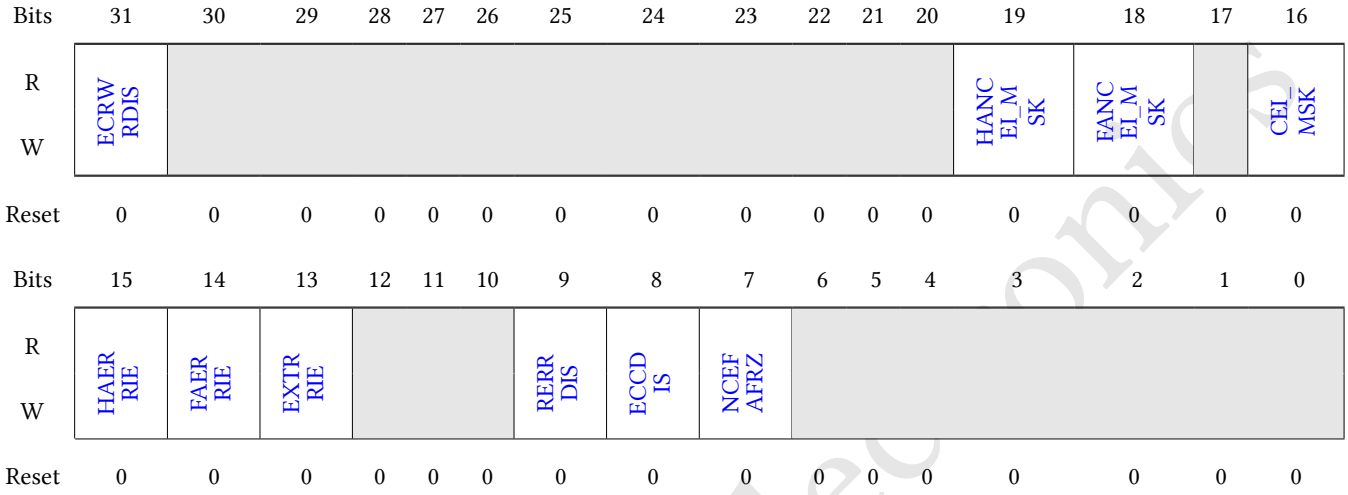


Table 17.33: CAN MECR Register Description

Field	Function
31 ECRWRDIS	**Error Configuration Register Write Disable –Disables writes to this register.** ECRWRDIS is automatically set to 1 (disabled) when CTRL2.ECRWRE is asserted. The protocol described in "Detection and Correction of Memory Errors" must be followed. 0b - Write is enabled. 1b - Write is disabled.
19 HANCEI_MSK	**Host Access With Non-Correctable Errors Interrupt Mask** Enables the interrupt in case of non-correctable errors detected in memory reads issued by the host (CPU). 0b - Interrupt is disabled. 1b - Interrupt is enabled.
18 FANCEI_MSK	**FlexCAN Access With Non-Correctable Errors Interrupt Mask** Enables the interrupt in case of non-correctable errors detected in memory reads issued by the FlexCAN internal processes. 0b - Interrupt is disabled. 1b - Interrupt is enabled.
16	**Correctable Errors Interrupt Mask**

Yuntu Microelectronics

Field	Function
CEI_MSK	Enables the interrupt in case of correctable errors detected in memory reads issued by the host or FlexCAN internal processes. 0b - Interrupt is disabled. 1b - Interrupt is enabled.
15 HAERRIE	**Host Access Error Injection Enable** Enables the injection of errors only in memory reads issued by the host (CPU). 0b - Interrupt is disabled. 1b - Interrupt is enabled.
14 FAERRIE	**FlexCAN Access Error Injection Enable** Enables the injection of errors only in memory reads issued by the FlexCAN internal processes. 0b - Interrupt is disabled. 1b - Interrupt is enabled.
13 EXTRRIE	**Extended Error Injection Enable** Memory accesses performed by internal FlexCAN processes are 64-bit accesses. EXTERRIE extends the error injection on 32-bit memory accesses to the complementary 32-bit word using the same 32-bit error injection data and parity words. See 'Error Injection Data Pattern Register (CAN_ERRIDPR)' and 'Error Injection Parity Pattern Register (CAN_ERRIPPR)'. 0b - Error injection is applied only to the 32-bit word. 1b - Error injection is applied to the 64-bit word.
9 RERRDIS	**Error Report Disable –Disables the update of the error report registers.** NOTE: When reading the report registers, RERRDIS must be set to ensure coherence on the consecutive register reads. 0b - Enable updates of the error report registers. 1b - Disable updates of the error report registers. The update of error-related flags and the generation of bus transfer errors are still active.
8 ECCDIS	**Error Correction Disable –Disables the memory detection and correction mechanism.** In addition to disabling the error report mechanism, ECCDIS = 1 also stops the update of the error-related flags and generation of bus transfer errors. The parity bits continue to be calculated and written into memory on write transactions. 0b - Enable memory error correction. 1b - Disable memory error correction.
7	**Non-Correctable Errors In FlexCAN Access Put Device In Freeze Mode**

Yuntu Microelectronics

Field	Function
NCEFAFRZ	Determines the response when a non-correctable error is detected in a memory read performed by FlexCAN internal processes. 0b - Keep normal operation. 1b - Put FlexCAN in Freeze mode (according to "Freeze Mode"). In this case, entering Freeze mode prevents corrupted data from being treated as valid by FlexCAN internal processes.

17.3.3.26 CAN ERRIAR Register

The CAN_ERRIAR register holds the address where a memory read error is to be injected, as described in Error Injection .

The size and usage of the FlexCAN RAM depends on the FlexCAN hardware configuration. below table shows how to convert from the memory map address to the location in the physical FlexCAN RAM for the following example FlexCAN configuration:

- NUMBER_OF_MB = 32
- IRMQ_EN = 1
- FD_En = 1
- ENHANCE_MB_MEM = 1
- TIME_STAMP_EN = 1
- ERX_FIFO_EN = 1, ERX_FIFO_SIZE = 32, ERX_FIFO_FLT_ELEM = 128

Where pairs of values are provided, the first is the address for CAN_MCR.FDEN negated, the second is for CAN_MCR.FDEN asserted.

Table 17.34: CAN Error Injection Address Mapping for Maximum Configuration

RAM Contents	Injection Address(1)	Memory Map
FlexCAN Registers	Not mapped	—
MBs	0x0000-0x07FC	0x0080-0x087C
RXIMRs	0x1000-0x11FC	0x0880-0x0A7C
RXFIR_0	0x1200	0x0A80
RXFIR_1	0x1204	0x0A84
RXFIR_2	0x1208	0x0A88
RXFIR_3	0x120C	0x0A8C
RXFIR_4	0x1210	0x0A90
RXFIR_5	0x1214	0x0A94
Reserved	—	0x0A98
RXMGMASK	0x1220	0x0AA0
RXFGMASK	0x1224	0x0AA4
RX14MASK	0x1228	0x0AA8
RX15MASK	0x122C	0x0AAC
Tx_SMB	0x1230	0x0AB0 / 0x0F28
Rx_SMB0	0x1240 / 0x1278	0x0AC0 / 0x0F70
Rx_SMB1	0x1250 / 0x12C0	0x0AD0 / 0x0FB8
Rx_SMB0_TIME_STAMP	0x1308	0x0C20
Rx_SMB1_TIME_STAMP	0x130C	0x0C24

Yuntu Microelectronics

Table 17.34 continued from previous page

RAM Contents	Injection Address(1)	Memory Map
HR_TIME_STAMP	0x1310-0x150C	0x0C30-0xE2C
Enhanced MB Memory	0x0800-0x0FFC	0x1000-0x17FC
Enhanced Rx FIFO	0x1510-0x1F0C	0x2000-0x29FC
Enhanced Rx FIFO Filter Elements	0x1F10-0x210C	0x3000-0x31FC

See "RAM Addresses for Error Injection and Reporting" for the RAM address mapping for all FlexCAN configurations.

The CAN_ERRIAR register is not available when parameter ECC_EN is 0.

Offset: ae4h

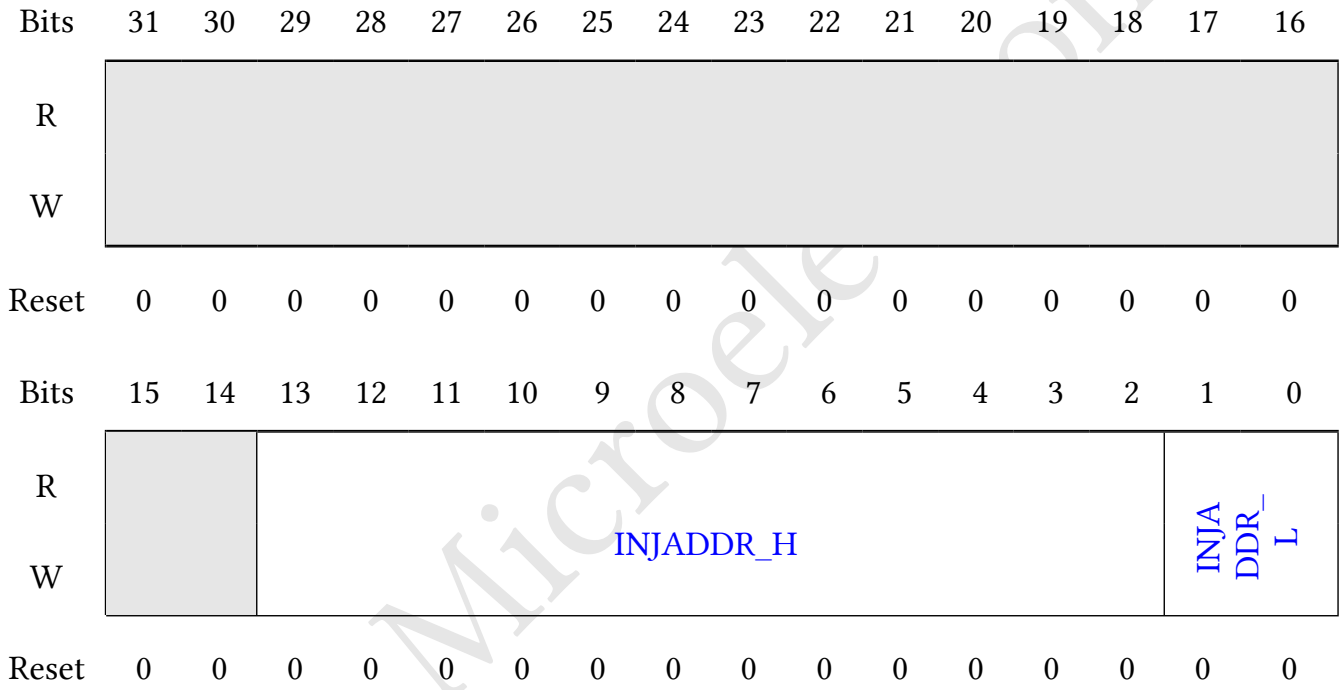


Table 17.35: CAN ERRIAR Register Description

Field	Function
13 - 2 INJADDR_H	**Error Injection Address High** Defines the twelve most significant bits of the physical RAM address where a memory read error is to be injected. See CAN error injection address mapping for maximum configuration
1 - 0 INJADDR_L	**Error Injection Address Low** Read-only field that defines the two least significant bits of the physical RAM address where a memory read error is to be injected. INJADDR_L[1:0] are always zero to ensure that the address is on a 32-bit boundary.

Yuntu Microelectronics

17.3.3.27 CAN ERRIDPR Register

The CAN_ERRIDPR register holds the error pattern to be injected in the data word read from memory.
The CAN_ERRIDPR register is not available when parameter ECC_EN is 0.

Offset: ae8h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DFLIP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DFLIP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.36: CAN ERRIDPR Register Description

Field	Function
31 - 0 DFLIP	**Data Flip Pattern** Bits set to 1 in the flip pattern cause the corresponding data bit in the word read from memory to invert.

17.3.3.28 CAN ERRIPPR Register

The CAN_ERRIPPR register holds the error pattern to be injected in parity bits read from memory along with data word.
Bits set to 1 in the flip pattern cause the corresponding parity bit in the word read from memory to invert.
The CAN_ERRIPPR register is not available when parameter ECC_EN is 0.

Offset: aech

Yuntu Microelectronics

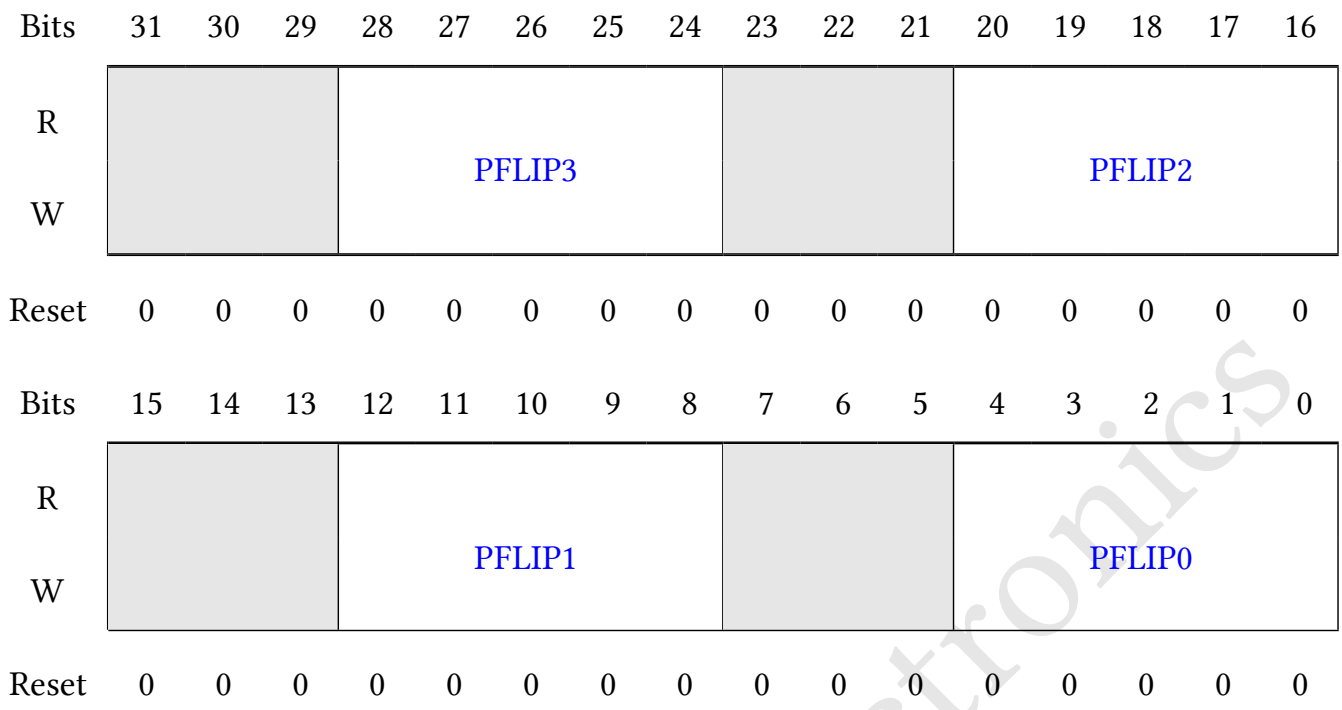


Table 17.37: CAN ERRIPPR Register Description

Field	Function
28 - 24 PFLIP3	**Parity Flip Pattern For Byte 3 (most significant)**
20 - 16 PFLIP2	**Parity Flip Pattern For Byte 2**
12 - 8 PFLIP1	**Parity Flip Pattern For Byte 1**
4 - 0 PFLIP0	**Parity Flip Pattern For Byte 0 (least significant)**

17.3.3.29 CAN RERRAR Register

The CAN_RERRAR register reports the address used for an access in which an error (correctable or non-correctable) was detected, and reports the identification of the source of that access.
The address is always reported using a 32-bit alignment. Non-aligned accesses (CAN_ERRADDR[1:0] are non-0) are reported with the address aligned and data is reported in CAN_RERRDR accordingly shifted. In case of errors detected in accesses larger than 32-bit (as performed by FlexCAN internal processes), the address of the 32-bit word in which the error was detected is reported. In case of errors detected in more than one 32-bit word, only the least significant address is reported.
The CAN_RERRAR register is not available when parameter ECC_EN is 0.

Offset: af0h

Yuntu Microelectronics

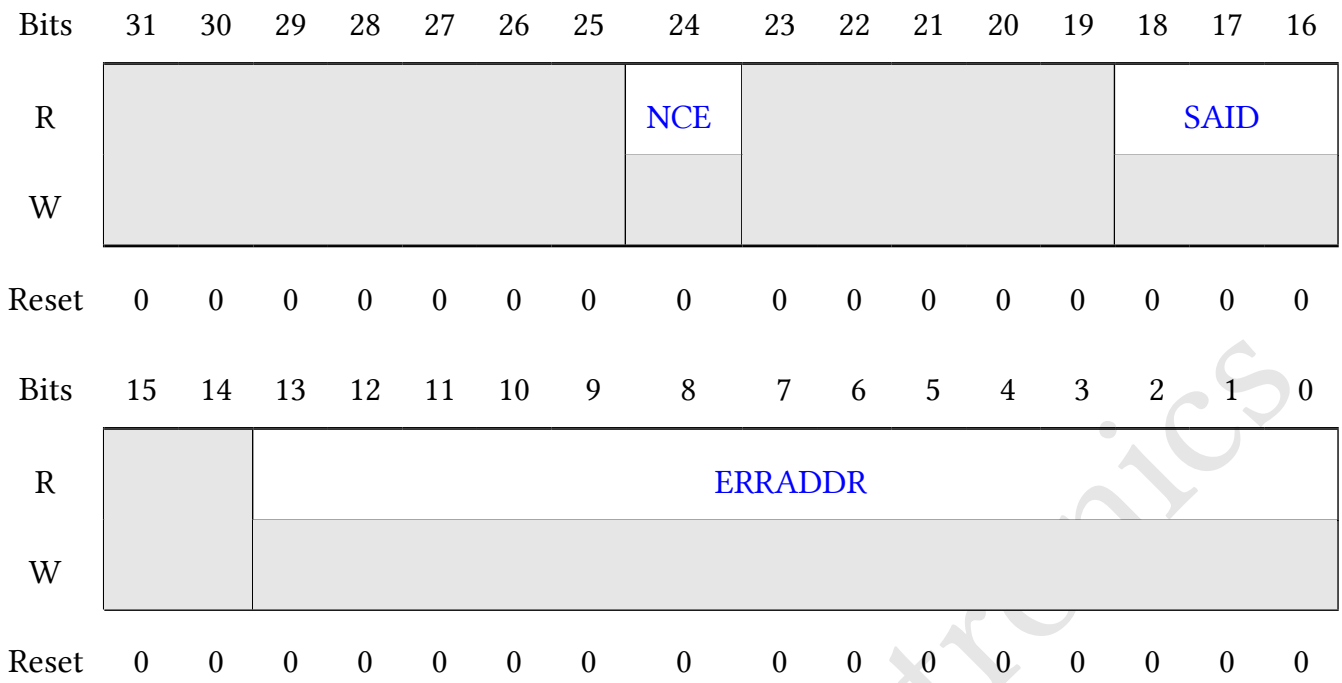


Table 17.38: CAN RERRAR Register Description

Field	Function
24 NCE	**Non-Correctable Error** Indicates that the report is due to a noncorrectable error. 0b - Reporting a correctable-error 1b - Reporting a non-correctable error
18 - 16 SAID	**SAID** SAID[2] - Identification of the requestor of the memory read request: • 0: Requested by FlexCAN internal processes • 1: Requested by host (CPU) SAID[1] - Details of FlexCAN operation: • 0: Move • 1: Scanning SAID[0] - Operation that requested the memory read: • 0: Transmission • 1: Reception
13 - 0 ERRADDR	**Address Where the Error was Detected** See the description of Error Injection Address Register (CAN_ERRIAR) .

17.3.3.30 CAN RERRDR Register

The CAN_RERRDR register reports the raw data (unmodified by the correction performed by ECC logic) read from memory with error. The value reported does not represent the transient values of the BUSY bit when reading a message buffer (See CAN message buffer code for RX buffers). The CAN_RERRDR register is not available when parameter ECC_EN is 0.

Yuntu Microelectronics

Offset: af4h

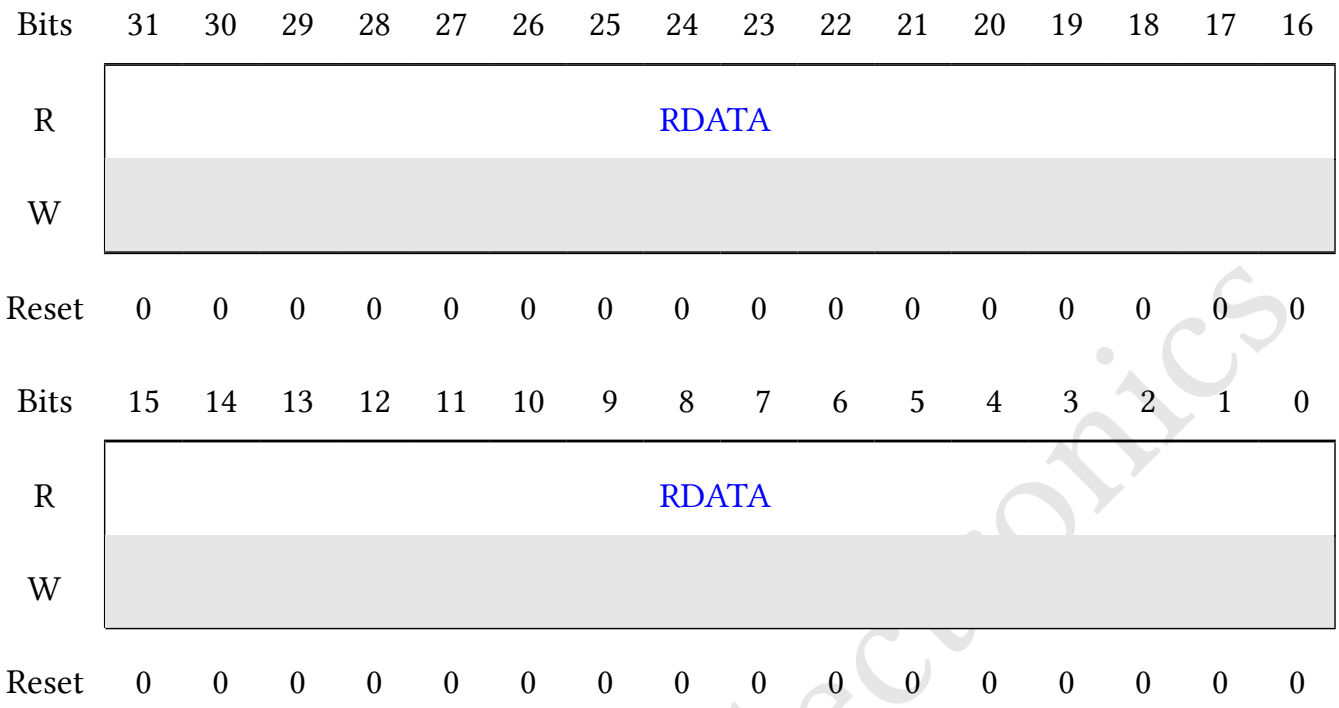


Table 17.39: CAN RERRDR Register Description

Field	Function
31 - 0 RDATA	**Raw data word read from memory with error**

17.3.3.31 CAN RERRSYNR Register

The CAN_RERRSYNR register holds the syndrome detected in a memory read with error. It also reports the bytes that were read in this 32-bit read transaction.

Each SYNDn field indicates the type of error and which bit in byte (n) is affected by the error, as defined in Table 38. (SYND3 corresponds to the most significant byte in the data word read from memory; SYND0 corresponds to the least significant.)

Each BEn field indicates which byte in the 32-bit word reported was effectively read. The syndrome bits are calculated for all bytes, even for the non-read ones. Errors detected in non-read bytes are indicated and reported. See ”Error Indication” and ”Error Reporting”.

Table 17.40: CAN Syndrome Definition

SYNDn(hex)	Type	Bit Affected
00	—	None (no error)
01	Code	0
02	Code	1
04	Code	2
07	Data	5

Yuntu Microelectronics

Table 17.40 continued from previous page

SYNDn(hex)	Type	Bit Affected
08	Code	3
0E	Data	7
10	Code	4
13	Data	2
15	Data	6
16	Data	1
19	Data	3
1A	Data	4
1C	Data	0
06	—	All-zeros non-correctable error
1F	—	All-ones non-correctable error
All others	—	Non-correctable error

Offset: af8h

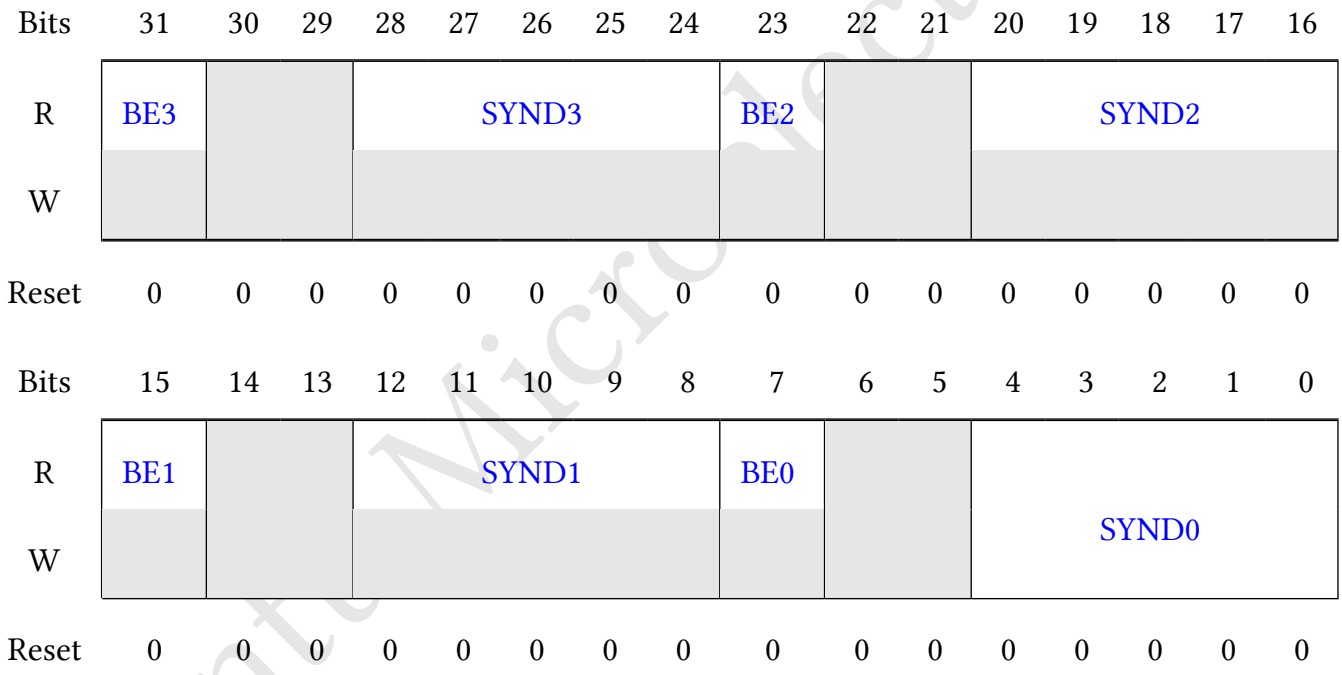


Table 17.41: CAN RERRSYNR Register Description

Field	Function
31 BE3	**Byte Enabled For Byte 3 (most significant)** 0b - The byte was not read. 1b - The byte was read.
28 - 24 SYND3	**Error Syndrome For Byte 3 (most significant)** See CAN syndrome definition.
23	**Byte Enabled For Byte 2**

Yuntu Microelectronics

Field	Function
BE2	0b - The byte was not read. 1b - The byte was read.
20 - 16 SYND2	**Error Syndrome For Byte 2** See CAN syndrome definition.
15 BE1	**Byte Enabled For Byte 1** 0b - The byte was not read. 1b - The byte was read.
12 - 8 SYND1	**Error Syndrome For Byte 1** See CAN syndrome definition.
7 BE0	**Byte Enabled For Byte 0 (least significant)** 0b - The byte was not read. 1b - The byte was read.
4 - 0 SYND0	**Error Syndrome For Byte 0 (least significant)** See CAN syndrome definition.

17.3.3.32 CAN ERRSR Register

The CAN_ERRSR register holds the status bits of the error correction and detection operations. These flags can be cleared by writing 1 to them. Writing 0 has no effect. The CAN_ERRSR register is not available when parameter ECC_EN is 0.

Offset: afch

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R													HANCEIF	FANCEIF		CEIF
W													W1C	W1C		W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R													HANCEIOF	FANCEIOF		CEIOF
W													W1C	W1C		W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Yuntu Microelectronics

Table 17.42: CAN ERRSR Register Description

Field	Function
19 HANCEIF	**Host Access With Non-Correctable Error Interrupt Flag** HANCEIF = 1 indicates that a non-correctable error was detected in a memory read initiated by the Hhost. A bus transfer error is asserted for that access. If CAN_MECR.HANCEI_MSK is set, the interrupt is asserted. 0b - No non-correctable errors were detected in host accesses so far. 1b - A non-correctable error was detected in a host access.
18 FANCEIF	**FlexCAN Access With Non-Correctable Error Interrupt Flag** FANCEIF = 1 indicates that a non-correctable error was detected in a memory read initiated by FlexCAN internal processes. If CAN_MECR.FANCEI_MSK is set, the interrupt is asserted. 0b - No non-correctable errors were detected in FlexCAN accesses so far. 1b - A non-correctable error was detected in a FlexCAN access.
16 CEIF	**Correctable Error Interrupt Flag** CEIF = 1 indicates that a correctable error was detected in a memory read. If CAN_MECR.CEI_MSK is set, the interrupt is asserted. 0b - No correctable errors were detected so far. 1b - A correctable error was detected.
3 HANCEIOF	**Host Access With Non-Correctable Error Interrupt Overrun Flag** HANCEIOF = 1 indicates that a non-correctable error was detected in a memory read initiated by the host when HANCEIF was set. No interrupt is associated with this flag. See ”Error Indication”. 0b - No overrun on non-correctable errors in host access. 1b - Overrun on non-correctable errors in host access.
2 FANCEIOF	**FlexCAN Access With Non-Correctable Error Interrupt Overrun Flag** FANCEIOF = 1 indicates that a non-correctable error was detected in a memory read initiated by FlexCAN internal processes when FANCEIF was set. No interrupt is associated with this flag. See ”Error Indication”. 0b - No overrun on non-correctable errors in FlexCAN access. 1b - Overrun on non-correctable errors in FlexCAN access.
0 CEIOF	**Correctable Error Interrupt Overrun Flag** CEIOF = 1 indicates that a correctable error was detected in a memory read when CEIF was set. No interrupt is associated with this flag. See ”Error Indication”. 0b - No overrun on correctable errors. 1b - Overrun on correctable errors.

17.3.3.33 CAN CTRL1_PN Register

The CAN_CTRL1_PN register contains control bits for Pretended Networking mode filtering selection. Configure CAN_CTRL1_PN with the filter criteria to be used to receive Wake-Up messages.

Yuntu Microelectronics

CAN_CTRL1_PN bits other than WTOF_MSK and WUMF_MSK can be written in Freeze mode only. The CAN_CTRL1_PN register is not available when parameter PNET_EN is 0.

Offset: b00h

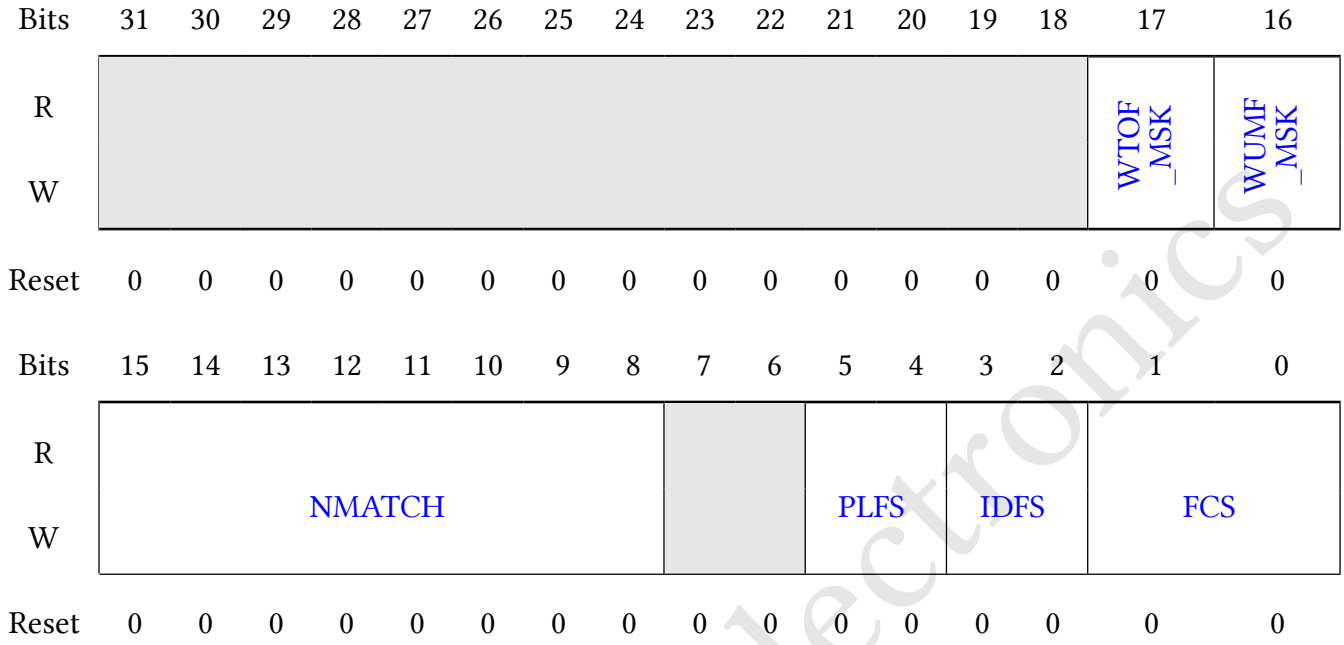


Table 17.43: CAN CTRL1_PN Register Description

Field	Function
17 WTOF_MSK	**Wake Up by Timeout Flag Mask Bit** Masks the generation of a Wake-Up event originated by a timeout 0b - Timeout Wake-Up event is disabled. 1b - Timeout Wake-Up event is enabled.
16 WUMF_MSK	**Wake Up by Match Flag Mask Bit** Masks the generation of a Wake-Up event originated by a successful filtered Rx message 0b - Wake-Up match event is disabled. 1b - Wake-Up match event is enabled.
15 - 8	

Yuntu Microelectronics

Field	Function
NMATCH	Number of Messages Matching the Same Filtering Criteria NMATCH defines the number of times a given message must match the predefined filtering criteria for ID and/or PL before generating a Wake-Up event. This quantity can be configured in the 1 to 255 range by using values from 0x01 to 0xFF, respectively 00000001b (0x01) - Received message must match the predefined filtering criteria for ID and/or PL once before generating a Wake-Up event. 00000010b (0x02) - Received message must match the predefined filtering criteria for ID and/or PL twice before generating a Wake-Up event. ..... 11111111b (0xff) - Received message must match the predefined filtering criteria for ID and/or PL 255 times before generating a Wake-Up event.
5 - 4 PLFS	**Payload Filtering Selection** Selects the level of payload filtering to be applied when FlexCAN is under Pretended Networking mode. Filtering does not accept remote messages (RTR=1) when payload filtering is active. 00b - Match upon a payload contents against an exact target value. 01b - Match upon a payload value greater than or equal to a specified target value. 1010b - Match upon a payload value smaller than or equal to a specified target value. 1011b - Match upon a payload value inside a range, greater than or equal to a specified lower limit, and smaller than or equal to a specified upper limit.
3 - 2 IDFS	**ID Filtering Selection –Selects the level of ID filtering to be applied when FlexCAN is under Pretended Networking mode.** In ID filtering, the IDE and RTR bits are also considered as part of the reception filter if the IDE_MSK and RTR_MSK bits in the CAN_FLT_ID2_IDMASK register are set. 00b - Match upon a ID contents against an exact target value. 01b - Match upon a ID value greater than or equal to a specified target value. 1010b - Match upon a ID value smaller than or equal to a specified target value. 1011b - Match upon a ID value inside a range, greater than or equal to a specified lower limit, and smaller than or equal to a specified upper limit.
1 - 0 FCS	**Filtering Combination Selection –Selects the filtering criteria to be applied when FlexCAN is under Pretended Networking mode.** See "Receive Process under Pretended Networking Mode") for additional details. 00b - Message ID filtering only. 01b - Message ID filtering and payload filtering. 1010b - Message ID filtering occurring a specified number of times. 1011b - Message ID filtering and payload filtering a specified number of times.

17.3.3.34 CAN CTRL2_PN Register

The CAN_CTRL2_PN register contains configuration bits for the timeout value under Pretended Networking mode. CAN_CTRL2_PN can be written in Freeze mode only.
The CAN_CTRL2_PN register is not available when parameter PNET_EN is 0.

Yuntu Microelectronics

Offset: b04h

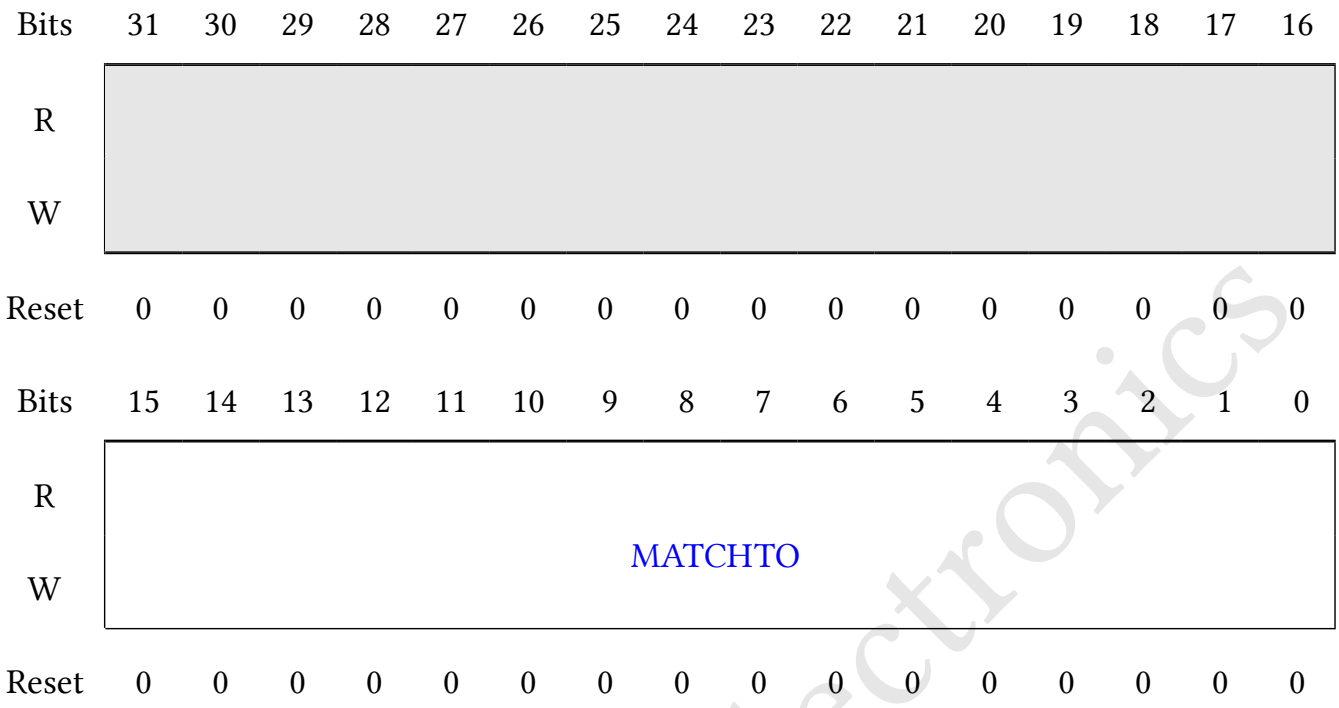


Table 17.44: CAN CTRL2_PN Register Description

Field	Function
15 - 0 MATCHTO	Timeout for No Message Matching the Filtering Criteria MATCHTO defines a timeout value that generates a Wake-Up event if CAN_MCR.PNET_EN is asserted. If the timeout counter reaches the target value when the FlexCAN is under Pretended Networking mode, a Wake-Up event is generated. The timeout limit can be configured from 1 to 65535 to control an internal 16-bit up-count timer to produce a trigger upon reaching this configured value. The internal timer is incremented based on periodic time ticks, of which the period is 64 times the CAN Bit Time unit. When MATCHTO is 0x0000, the timeout is disabled.

Yuntu Microelectronics

17.3.3.35 CAN WU_MTC Register

The CAN_WU_MTC register contains Wake-Up information related to the matching processes performed while the FlexCAN receives frames under Pretended Networking mode.
The CAN_WU_MTC register is not available when parameter PNET_EN is 0.

Offset: b08h

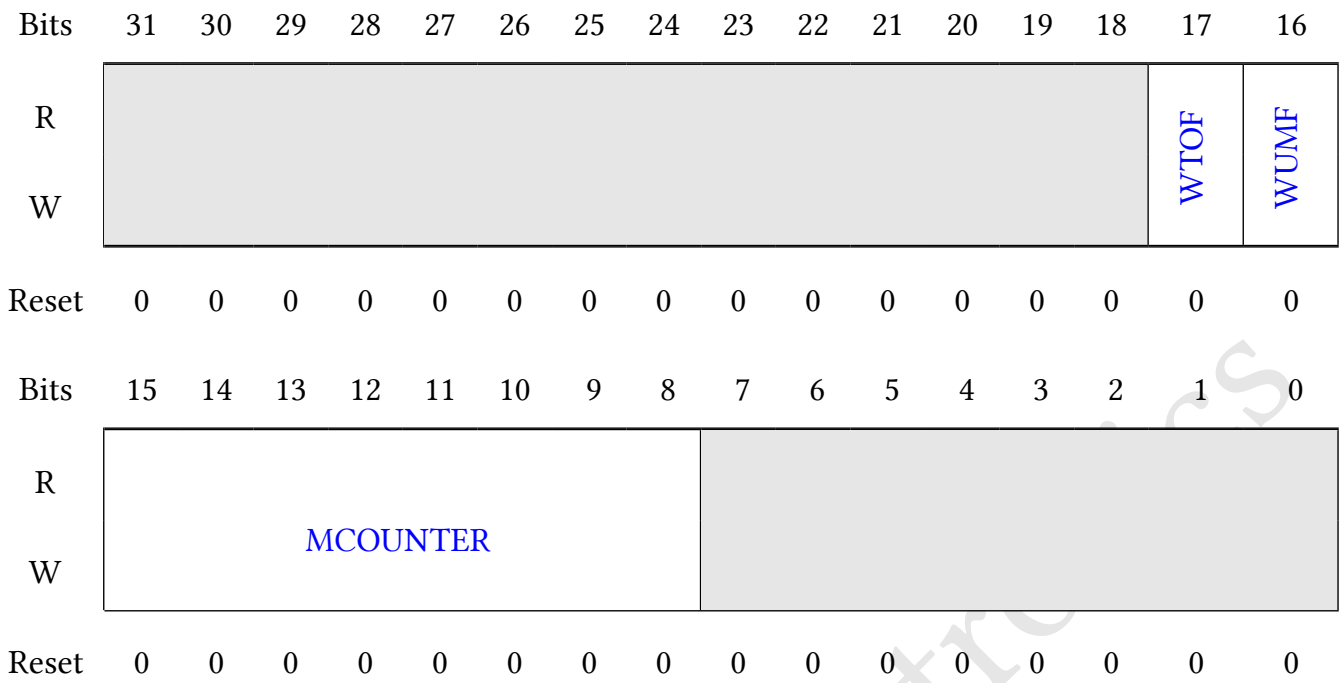


Table 17.45: CAN WU_MTC Register Description

Field	Function
17 WTOF	**Wake Up by Timeout Flag Bit –Indicates whether the FlexCAN has detected a timeout event during a time interval defined by CAN_CTRL2_PN.MATCHTO.** WTOF generates a Wake-Up event if CAN_CTRL1_PN.WTOF_MSK is asserted. 0b - No Wake-Up by timeout event detected. 1b - Wake-Up by timeout event detected.
16 WUMF	**Wake Up by Match Flag Bit** Indicates whether the FlexCAN has detected a matching Rx incoming message that passed the filtering criteria specified in CAN_CTRL1_PN. WUMF generates a Wake-Up event if CAN_CTRL1_PN.WUMF_MSK is asserted. 0b - No Wake-Up by match event detected. 1b - Wake-Up by match event detected.
15 - 8 MCOUNTER	Number of Matches while in Pretended Networking MCOUNTER reports the number of times a given message has matched the predefined filtering criteria for ID and/or PL before a Wake-Up event. MCOUNTER is reset by the FlexCAN when it enters in Pretended Networking mode, and is not affected by soft reset.

Yuntu Microelectronics

17.3.3.36 CAN FLT_ID1 Register

The CAN_FLT_ID1 register contains the FLT_ID1 target value and the IDE and RTR target values used to filter an incoming message ID. FLT_ID1 is used either for equal to, smaller than, greater than comparisons, or as the lower limit value in an ID range detection. CAN_FLT_ID1 can be written in Freeze mode only.
The CAN_FLT_ID1 register is not available when parameter PNET_EN is 0.

Offset: b0ch

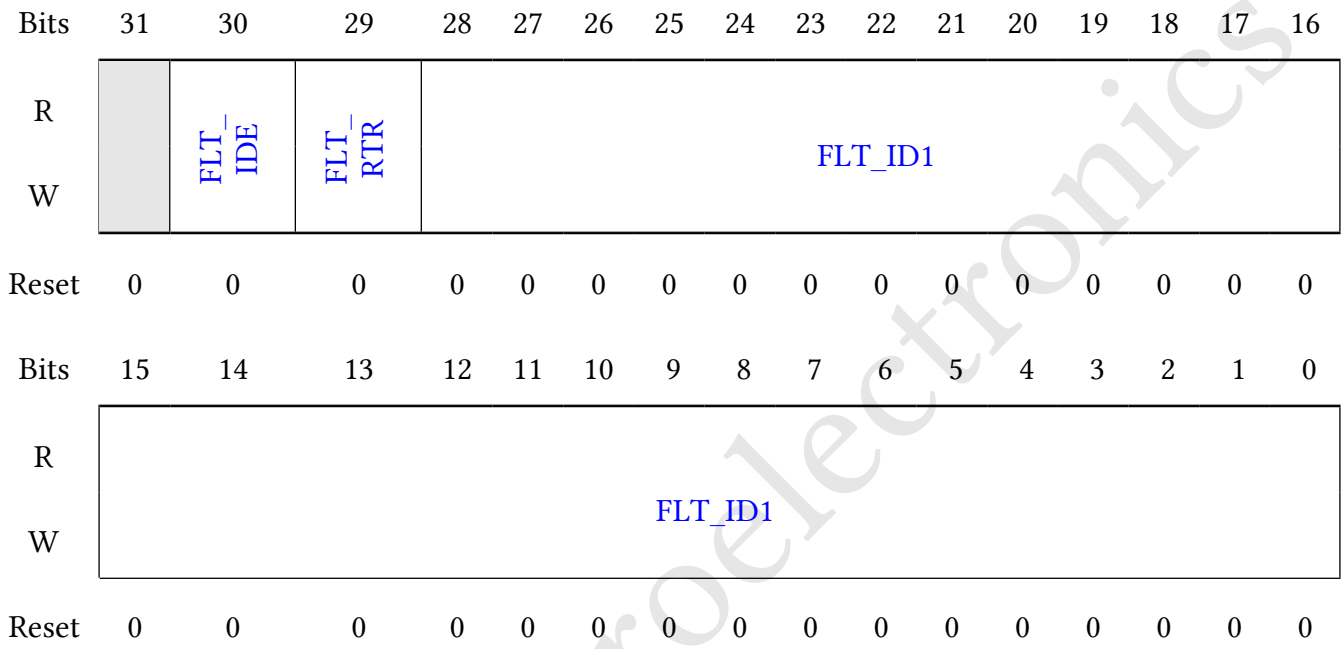


Table 17.46: CAN FLT_ID1 Register Description

Field	Function
30 FLT_IDE	**D Extended Filter –Identifies whether the frame format is standard or extended.** FLT_IDE is used as part of the ID reception filter. 0b - Accept standard frame format. 1b - Accept extended frame format.
29 FLT_RTR	**Remote Transmission Request Filter –Identifies whether the frame is remote or not.** FLT_RTR is used as part of the ID reception filter. 0b - Reject remote frame (accept data frame). 1b - Accept remote frame.
28 - 0 FLT_ID1	**ID Filter 1 for Pretended Networking Filtering** Defines either the 29 bits of an extended frame format, considering all bits, or the 11 bits of a standard frame format, considering just the 11 leftmost bits.

Yuntu Microelectronics

17.3.3.37 CAN FLT_DLC Register

The CAN_FLT_DLC register contains the DLC inside range target values (FLT_DLC_LO and FLT_DLC_HI) used to filter an incoming message. The DLC range is used only for payload filtering. CAN_FLT_DLC can be written in Freeze mode only.
NOTE: When a fixed quantity of data bytes is required, both FLT_DLC_LO and FLT_DLC_HI need to be configured with the same value; otherwise, a range of DLC is considered for filtering (see "Receive Process under Pretended Networking Mode").
The CAN_FLT_DLC register is not available when parameter PNET_EN is 0.

Offset: b10h

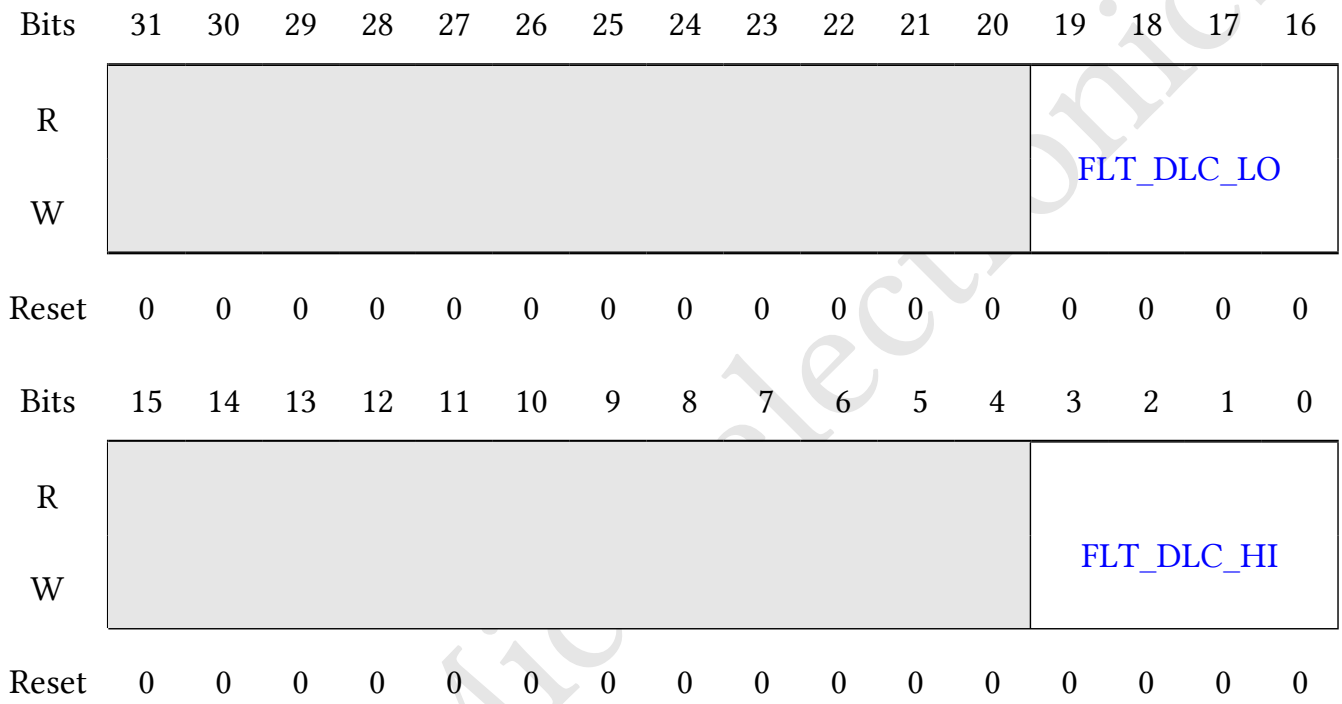


Table 17.47: CAN FLT_DLC Register Description

Field	Function
19 - 16 FLT_DLC_LO	**Lower Limit for Length of Data Bytes Filter** Specifies the lower limit for the number of data bytes considered valid for payload comparison. FLT_DLC_LO is used as part of payload reception filter.
3 - 0 FLT_DLC_HI	**Upper Limit for Length of Data Bytes Filter** Specifies the upper limit for the number of data bytes considered valid for payload comparison. FLT_DLC_HI is used as part of payload reception filter.

17.3.3.38 CAN PL1_LO Register

The CAN_PL1_LO register contains Payload Filter 1 low order bits of the target value used to filter incoming message payload. It is used either for "equal to", "smaller than or equal to", "greater than or equal to" comparisons, or as the lower limit value in a payload range detection. CAN_PL1_LO

can be written in Freeze mode only.
The CAN_PL1_LO register is not available when parameter PNET_EN is 0.

Offset: b14h

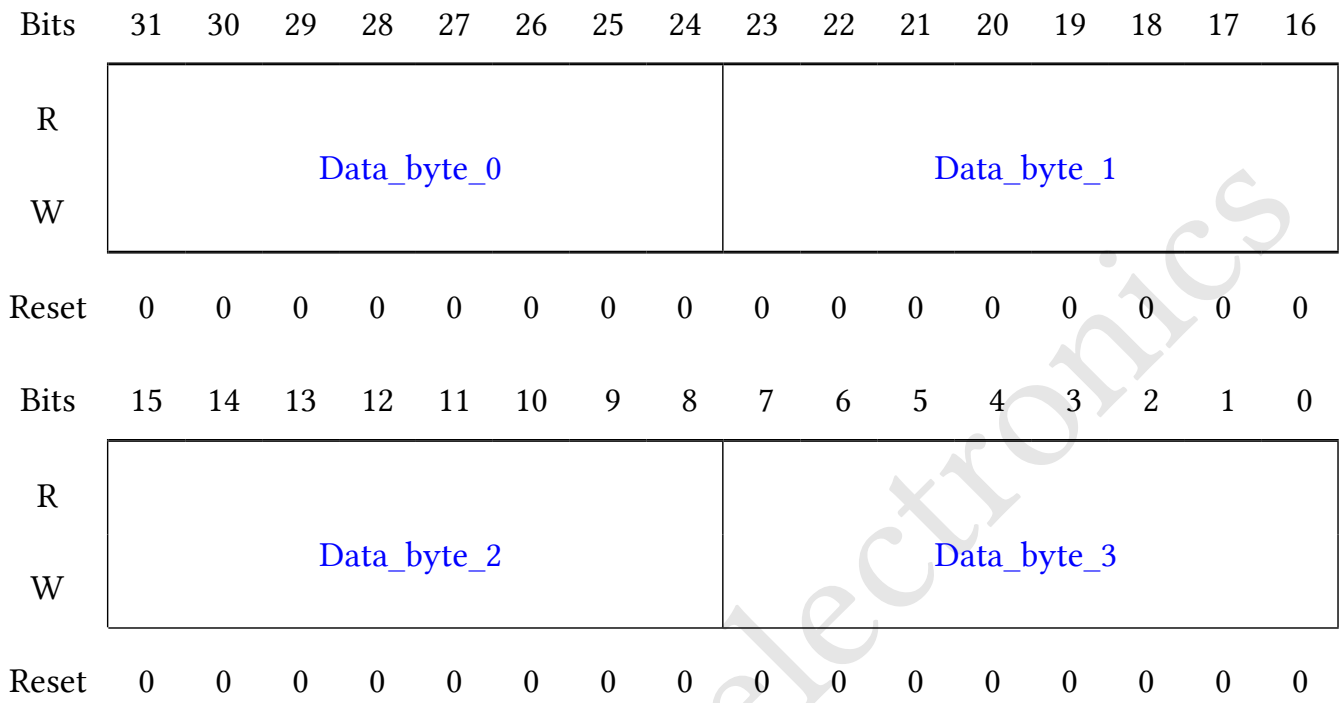


Table 17.48: CAN PL1_LO Register Description

Field	Function
31 - 24 Data_byte_0	Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to data byte 0.
23 - 16 Data_byte_1	Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to data byte 1.
15 - 8 Data_byte_2	Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to data byte 2.
7 - 0 Data_byte_3	Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to data byte 3.

Yuntu Microelectronics

17.3.3.39 CAN PL1_HI Register

The CAN_PL1_HI register contains Payload Filter 1 high order bits of the target value used to filter incoming message payload. It is used either for "equal to", "smaller than or equal to", "greater than or equal to" comparisons, or as the lower limit value in a payload range detection. CAN_PL1_HI can be written in Freeze mode only.
The CAN_PL1_HI register is not available when parameter PNET_EN is 0.

Offset: b18h

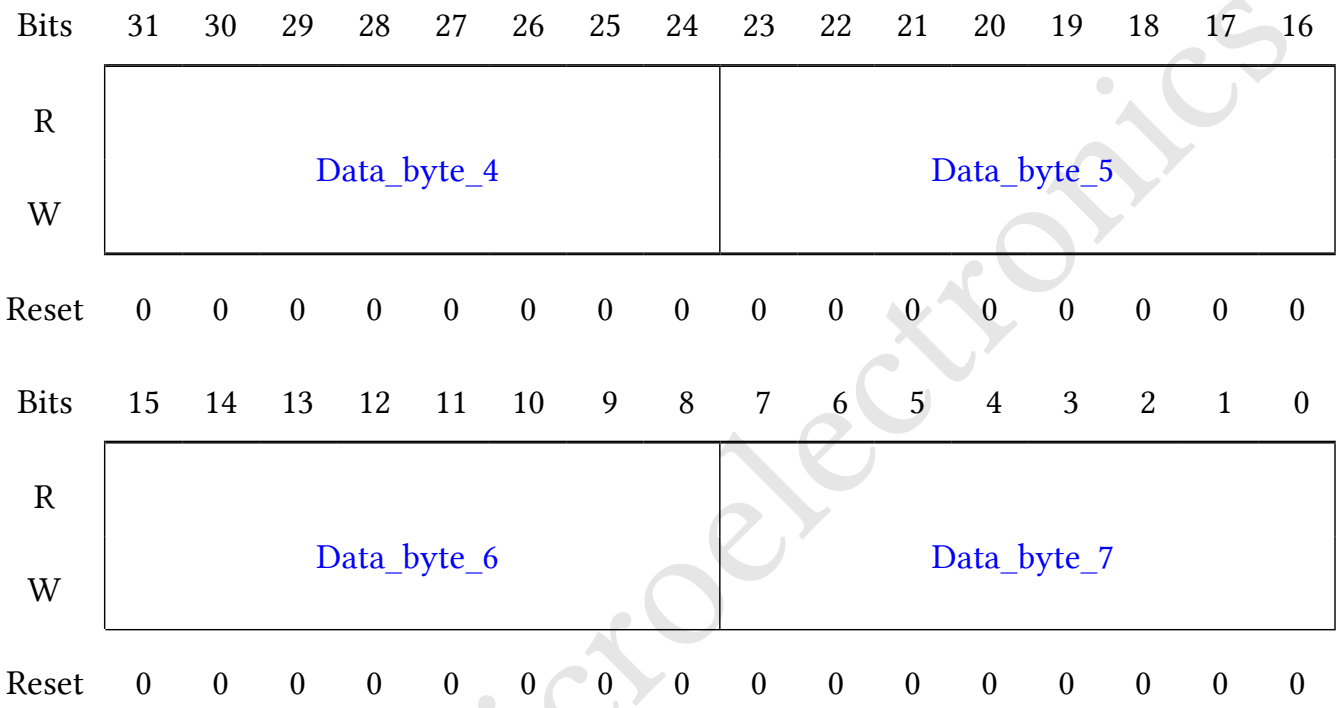


Table 17.49: CAN PL1_HI Register Description

Field	Function
31 - 24 Data_byte_4	Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to data byte 4.
23 - 16 Data_byte_5	Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to data byte 5.
15 - 8 Data_byte_6	Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to data byte 6.
7 - 0	

Yuntu Microelectronics

Field	Function
Data_byte_7	Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to data byte 7.

17.3.3.40 CAN FLT_ID2_IDMASK Register

The CAN_FLT_ID2_IDMASK register contains the FLT_ID2 target value used only as the upper limit value in ID range detection. Also, when exact ID filtering criteria is selected, CAN_FLT_ID2_IDMASK is used to store the ID mask. The IDE_MSK and RTR_MSK bits are used in both ID filtering (exact and range) to enable FLT_IDE and FLT_RTR respectively to be used as part of ID reception filter.
CAN_FLT_ID2_IDMASK can be written in Freeze mode only.
The CAN_FLT_ID2_IDMASK register is not available when parameter PNET_EN is 0.

Offset: b1ch

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		IDE_MSK	RTR_MSK	FLT_ID2_IDMASK												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FLT_ID2_IDMASK															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.50: CAN FLT_ID2_IDMASK Register Description

Field	Function
30 IDE_MSK	**ID Extended Mask Bit –Indicates whether the frame format (standard/extended) is used as part of the ID reception filter.** 0b - The corresponding bit in the filter is "don't care". 1b - The corresponding bit in the filter is checked.
29 RTR_MSK	Remote Transmission Request Mask Bit –Indicates whether the frame type (data/remote) is part of the ID reception filter. 0b - The corresponding bit in the filter is "don't care". 1b - The corresponding bit in the filter is checked.

Yuntu Microelectronics

Field	Function
28 - 0 FLT_ID2_ IDMASK	ID Filter 2 for Pretended Networking Filtering / ID Mask Bits for Pretended Networking ID Filtering FLT_ID2_IDMASK is used as FLT_ID2 in range of ID filtering to define the ID filter value in either: • Extended frame format (29 bits), considering the FLT_ID2[28:0], or • Standard frame format (11 bits), considering the FLT_ID2[28:18]. Other bits in the [17:0] range have no meaning. FLT_ID2_IDMASK can also be used as IDMASK in exact ID filtering to define the mask value for either: • Extended frame format (29 bits), considering the IDMASK[28:0], or • Standard frame format (11 bits), considering the IDMASK[28:18]. Other bits in the [17:0] range have no meaning.

17.3.3.41 CAN PL2_PLMASK_LO Register

The CAN_PL2_PLMASK_LO register has two functions:
• It contains the low order bits for the Payload Filter 2 used only as the upper limit value in a payload range detection.
• When exact payload filtering criteria is selected, CAN_PL2_PLMASK_LO is used as the payload mask. Otherwise, CAN_PL2_PLMASK_LO is unused. CAN_PL2_PLMASK_LO can be written in Freeze mode only.
The CAN_PL2_PLMASK_LO register is not available when parameter PNET_EN is 0.

Offset: b20h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Data_byte_0								Data_byte_1							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Data_byte_2								Data_byte_3							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Yuntu Microelectronics

Table 17.51: CAN PL2_PLMASK_LO Register Description

Field	Function
31 - 24 Data_byte_0	Payload Filter 2 low order bits / Payload Mask low order bits for Pretended Networking payload filtering corresponding to data byte 0.
23 - 16 Data_byte_1	Payload Filter 2 low order bits / Payload Mask low order bits for Pretended Networking payload filtering corresponding to data byte 1.
15 - 8 Data_byte_2	Payload Filter 2 low order bits / Payload Mask low order bits for Pretended Networking payload filtering corresponding to data byte 2.
7 - 0 Data_byte_3	Payload Filter 2 low order bits / Payload Mask low order bits for Pretended Networking payload filtering corresponding to data byte 3.

Yuntu Microelectronics

17.3.3.42 CAN PL2_PLMASK_HI Register

The CAN_PL2_PLMASK_HI register has two functions:
• It contains the high order bits for the Payload Filter 2 used only as the upper limit value in a payload range detection.
• When exact payload filtering criteria is selected, CAN_PL2_PLMASK_HI is used as the payload mask. Otherwise, CAN_PL2_PLMASK_HI is unused. CAN_PL2_PLMASK_HI can be written in Freeze mode only.
The CAN_PL2_PLMASK_HI register is not available when parameter PNET_EN is 0.

Offset: b24h

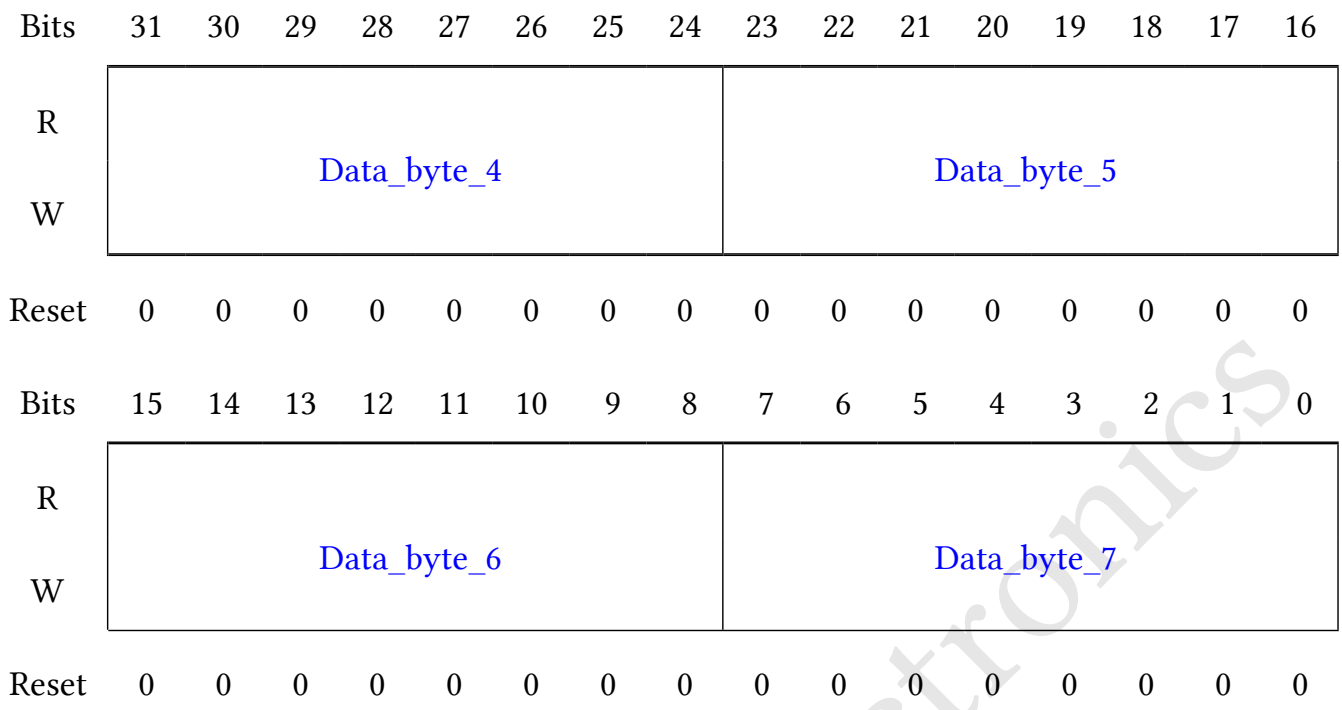


Table 17.52: CAN PL2_PLMASK_HI Register Description

Field	Function
31 - 24 Data_byte_4	Payload Filter 2 high order bits / Payload Mask high order bits for Pretended Networking payload filtering corresponding to data byte 4.
23 - 16 Data_byte_5	Payload Filter 2 high order bits / Payload Mask high order bits for Pretended Networking payload filtering corresponding to data byte 5.
15 - 8 Data_byte_6	Payload Filter 2 high order bits / Payload Mask high order bits for Pretended Networking payload filtering corresponding to data byte 6.
7 - 0 Data_byte_7	Payload Filter 2 high order bits / Payload Mask high order bits for Pretended Networking payload filtering corresponding to data byte 7.

17.3.3.43 CAN WMB_WMBn_CS Register

Each of the four WMBs contains a register to store the Control Status (C/S) information (IDE, RTR, and DLC fields) of an incoming Rx message.
NOTE: The C/S registers are located at 0xB40 for WMB0, 0xB50 for WMB1, 0xB60 for WMB2, and 0xB70 for WMB3.

Yuntu Microelectronics

The CAN_WMBn_CS registers are not available when parameter PNET_EN is 0.

Offset: 1680h + n * 10h

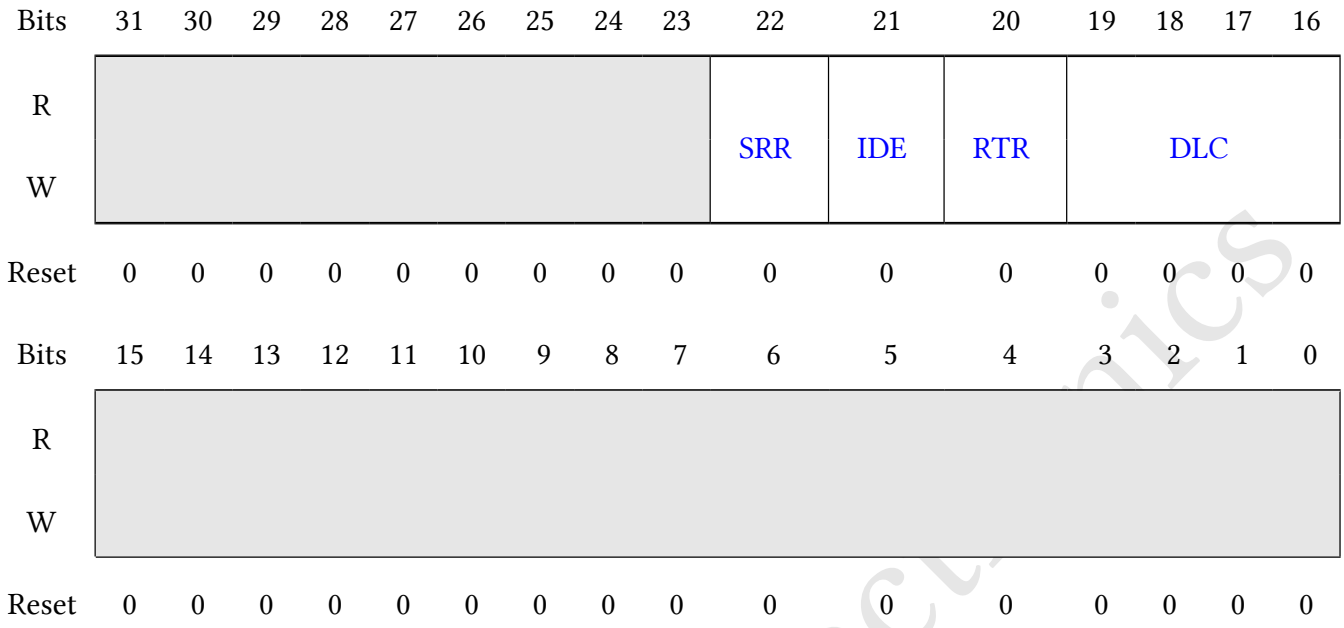


Table 17.53: CAN WMB_WMBn_CS Register Description

Field	Function
22 SRR	**Substitute Remote Request –Can be received either recessive or dominant.**
21 IDE	**ID Extended Bit –Identifies whether the frame format is standard or extended.** 0b - Frame format is standard. 1b - Frame format is extended.
20 RTR	**Remote Transmission Request Bit –Identifies whether the frame is remote or not.** 0b - Frame is data frame (not remote). 1b - Frame is a remote frame.
19 - 16 DLC	Length of Data in Bytes –Length (in bytes) of the Rx data received when FlexCAN is in Pretended Networking mode. DLC is written by the FlexCAN module, copied from the DLC (Data Length Code) field of the received frame. The DLC field indicates which data bytes are valid.

Yuntu Microelectronics

17.3.3.44 CAN WMB_WMBn_ID Register

Each of the four WMBs contains a register to store the ID information of an incoming Rx message.
NOTE: The ID registers are located at 0xB44 for WMB0, 0xB54 for WMB1, 0xB64 for WMB2, and 0xB74 for WMB3.
The CAN_WMBn_ID registers are not available when parameter PNET_EN is 0.

Offset: 1684h + n * 10h

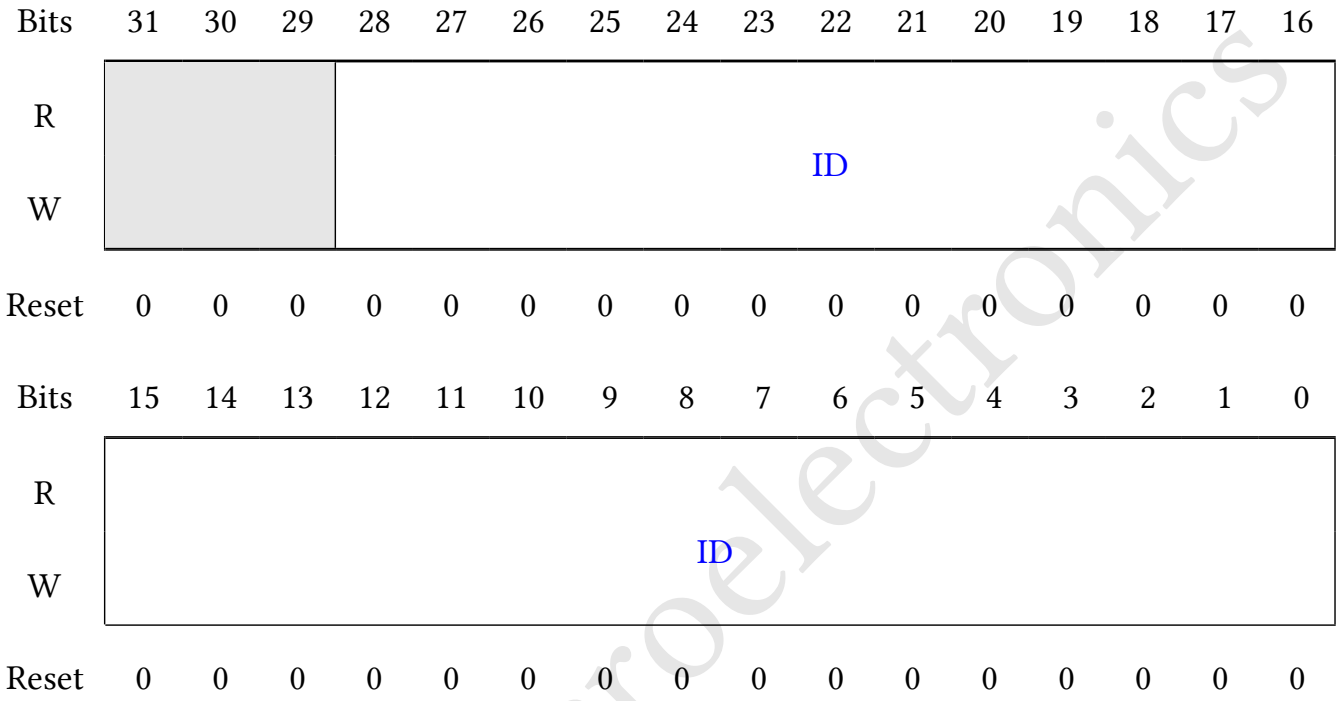


Table 17.54: CAN WMB_WMBn_ID Register Description

Field	Function
28 - 0 ID	Received ID under Pretended Networking Mode –Stores the received ID as either: • The 29 bits of the extended frame format (considering the ID[28:0] field), or • The 11 bits of the standard frame format (considering the ID[28:18] field only. The remaining bits in the ID[17:0] range have no meaning.

17.3.3.45 CAN WMB_WMBn_D03 Register

Each of the four WMBs contains a register to store the data bytes 0 to 3 of the payload information of an incoming Rx message. This register’s content is cleared when the incoming matched message is either a remote frame (RTR=1) or a data frame with DLC=0.
NOTE: The Data 0-3 registers are located at 0xB48 for WMB0, 0xB58 for WMB1, 0xB68 for WMB2, and 0xB78 for WMB3.
The CAN_WMBn_D03 registers are not available when parameter PNET_EN is 0.

Offset: 1688h + n * 10h

Yuntu Microelectronics

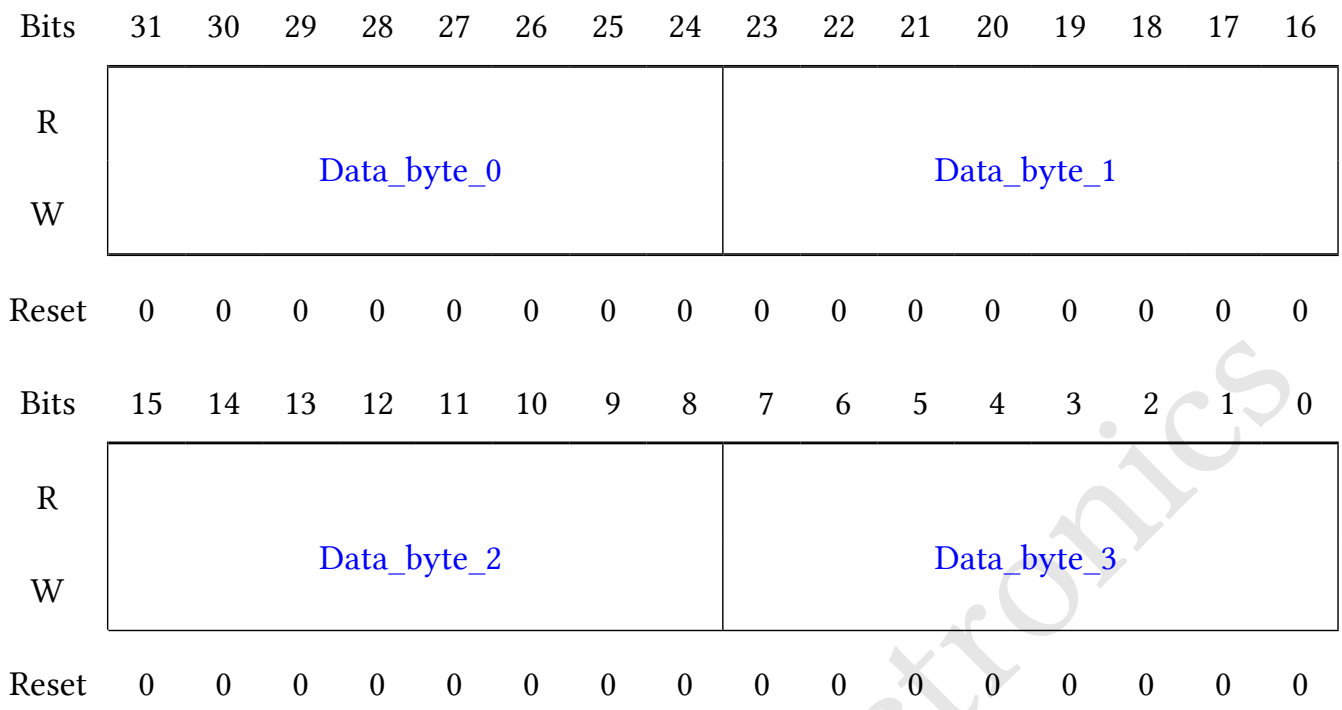


Table 17.55: CAN WMB_WMBn_D03 Register Description

Field	Function
31 - 24 Data_byte_0	Received payload corresponding to data byte 0 under Pretended Networking mode.
23 - 16 Data_byte_1	Received payload corresponding to data byte 1 under Pretended Networking mode.
15 - 8 Data_byte_2	Received payload corresponding to data byte 2 under Pretended Networking mode.
7 - 0 Data_byte_3	Received payload corresponding to data byte 3 under Pretended Networking mode.

17.3.3.46 CAN WMB_WMBn_D47 Register

Each of the four WMBs contains a register to store the data bytes 4 to 7 of the payload information of an incoming Rx message. This register's content is cleared when the incoming matched message is either a remote frame (RTR=1) or a data frame with DLC=0.
NOTE: The Data 4-7 registers are located at 0xB4C for WMB0, 0xB5C for WMB1, 0xB6C for WMB2, and

Yuntu Microelectronics

0xB7C for WMB3.
The CAN_WMBn_D47 registers are not available when parameter PNET_EN is 0.

Offset: 168ch + n * 10h

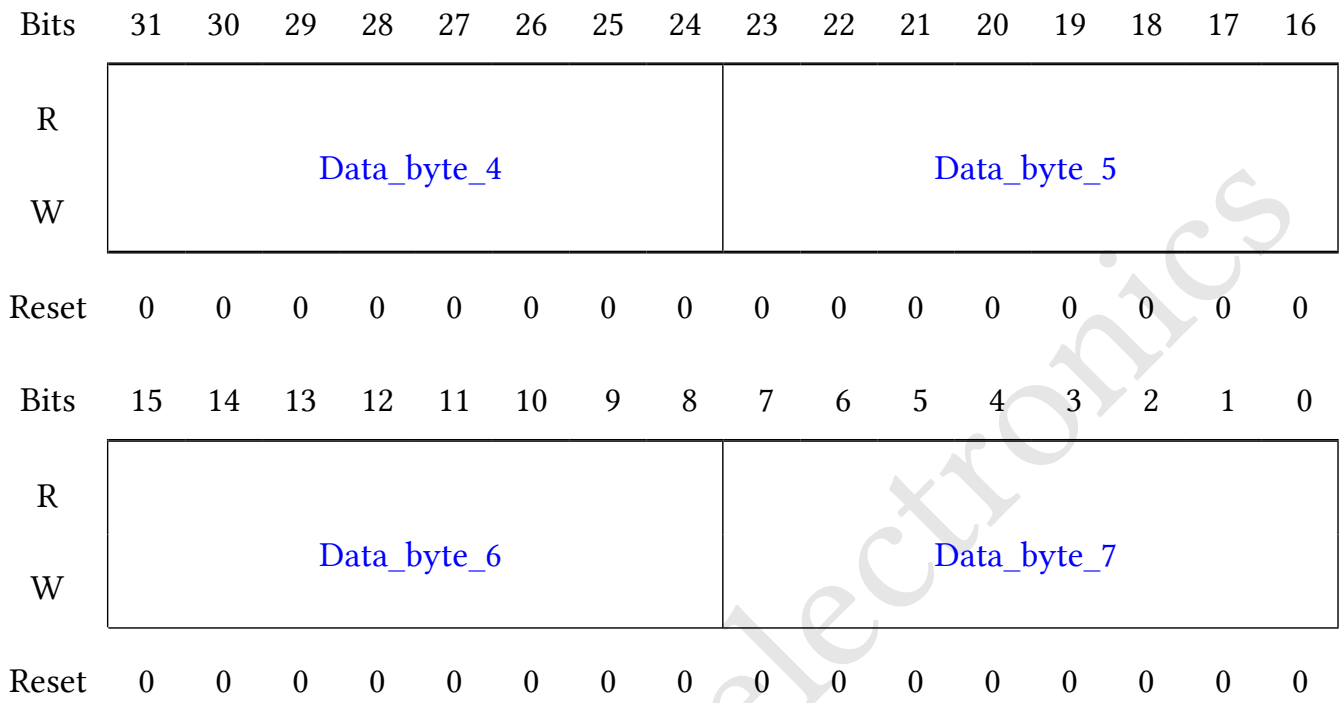


Table 17.56: CAN WMB_WMBn_D47 Register Description

Field	Function
31 - 24 Data_byte_4	Received payload corresponding to data byte 4 under Pretended Networking mode.
23 - 16 Data_byte_5	Received payload corresponding to data byte 5 under Pretended Networking mode.
15 - 8 Data_byte_6	Received payload corresponding to data byte 6 under Pretended Networking mode.
7 - 0 Data_byte_7	Received payload corresponding to data byte 7 under Pretended Networking mode.

Yuntu Microelectronics

17.3.3.47 CAN EPRS Register

The CAN_EPRS register defines the CAN bit timing prescalers for the nominal phase and data phase when CAN_CTRL2.BTE is set.
CAN_EPRS is used by the hardware only if CAN_CTRL2.BTE is set; otherwise a write operation has no effect and all fields are read as zero.
CAN_EPRS can be written only in Freeze mode because it is blocked by hardware in other modes.
The contents of CAN_EPRS are not affected by soft reset.
The CAN_EPRS register is not available when parameter CIA_BIT_TIMING is 0.

Offset: bf0h

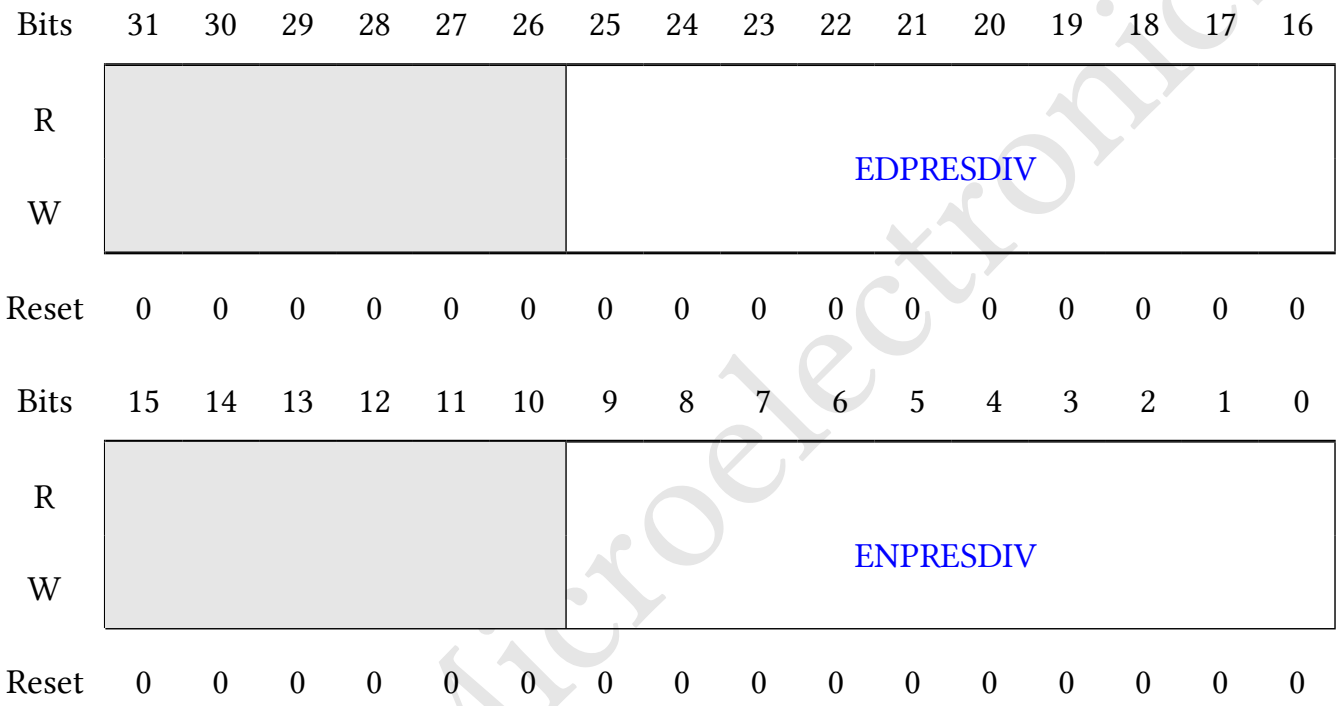


Table 17.57: CAN EPRS Register Description

Field	Function
25 - 16 EDPRESDIV	**Extended Data Phase Prescaler Division Factor** EDPRESDIV defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency in the data bit rate portion of a CAN FD message when CAN_CTRL2.BTE is set. The Sclock period defines the time quantum of the CAN FD protocol for the data bit rate. Sclock frequency = PE clock frequency / (EDPRESDIV + 1). NOTE: To minimize errors when processing FD frames, use the same value for ENPRESDIV and EDPRESDIV.
9 - 0	**Extended Nominal Phase Prescaler Division Factor**

Yuntu Microelectronics

Field	Function
ENPRESDIV	ENPRESDIV defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency for the nominal bit rate when CAN_CTRL2.BTE is set. The Sclock period defines the time quantum of the CAN protocol for the nominal phase. Sclock frequency = PE clock frequency / (ENPRESDIV + 1).

17.3.3.48 CAN ENCBT Register

The CAN_ENCBT register provides an alternative way to store the CAN bit timing variables described in the CAN_CTRL1 and CAN_CBT registers, to achieve higher CAN bit timing resolution.
CAN_ENCBT is used by the hardware only if CAN_CTRL2.BTE is set; otherwise a write operation has no effect and all fields are read as zero.
CAN_ENCBT can be written only in Freeze mode because it is blocked by hardware in other modes.
The contents of CAN_ENCBT are not affected by soft reset.
The CAN_ENCBT register is not available when parameter CIA_BIT_TIMING is 0.

Offset: bf4h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R				NRJW										NTSEG2		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NTSEG2								NTSEG1							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.58: CAN ENCBT Register Description

Field	Function
28 - 22	**Nominal Resynchronization Jump Width**

Yuntu Microelectronics

Field	Function
NRJW	NRJW defines the maximum number of time quanta that a nominal bit time can be changed by one resynchronization when CAN_CTRL2.BTE = 1. When CAN_CTRL2.BTE = 0, NRJW has no effect. One time quantum is equal to one Sclock period. Nominal Resynchronization Jump Width = NRJW + 1.
18 - 12 NTSEG2	**Nominal Time Segment 2** NTSEG2 defines the length of Time Segment 2 in the nominal bit time when CAN_CTRL2.BTE = 1. When CAN_CTRL2.BTE = 0, NTSEG2 has no effect. Nominal Time Segment 2 = (NTSEG2 + 1) × Time-Quanta Time-Quantum = one Sclock period
7 - 0 NTSEG1	**Nominal Time Segment 1** NTSEG1 defines the length of Time Segment 1 in the nominal bit time when CAN_CTRL2.BTE = 1. When CAN_CTRL2.BTE = 0, NTSEG1 has no effect. Nominal Time Segment 1 = (NTSEG1 + 1) × Time-Quanta Time-Quantum = one Sclock period

Yuntu Microelectronics

17.3.3.49 CAN EDCBT Register

The CAN_EDCBT register provides an alternative way to store the data phase CAN bit timing variables described in the CAN_FDCBT register, to achieve higher CAN bit timing resolution.
CAN_EDCBT is used by the hardware only if CAN_CTRL2.BTE is set; otherwise a write operation has no effect and all fields are read as zero.
CAN_EDCBT can be written only in Freeze mode because it is blocked by hardware in other modes.
The contents of CAN_EDCBT are not affected by soft reset.
The CAN_EDCBT register is not available when parameter CIA_BIT_TIMING is 0.
NOTE: CAN_EDCBT.DTSEG1 must be at least two time quanta.

Offset: bf8h

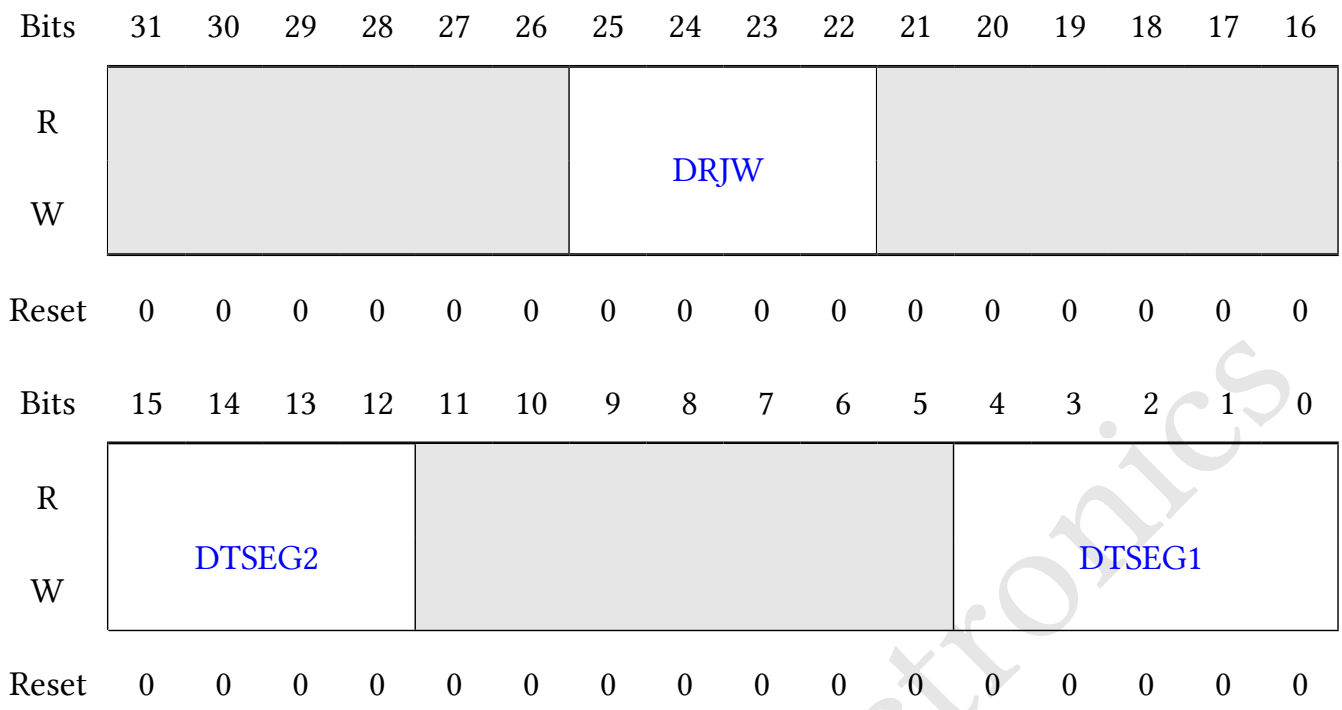


Table 17.59: CAN EDCBT Register Description

Field	Function
25 - 22 DRJW	**Data Phase Resynchronization Jump Width** DRJW defines the maximum number of time quanta that a data phase bit time can be changed by one resynchronization when CAN_CTRL2.BTE = 1. When CAN_CTRL2.BTE = 0, DRJW has no effect. One time quantum is equal to one Sclock period. Data Phase Resynchronization Jump Width = DRJW + 1.
15 - 12 DTSEG2	**Data Phase Time Segment 2** DTSEG2 defines the length of Time Segment 2 in the data phase bit time when CAN_CTRL2.BTE = 1. When CAN_CTRL2.BTE = 0, DTSEG2 has no effect. Data Phase Time Segment 2 = (DTSEG2 + 1) × Time-Quanta Time-Quantum = one Sclock period
4 - 0 DTSEG1	**Data Phase Time Segment 1** DTSEG1 defines the length of Time Segment 1 in the data phase bit time when CAN_CTRL2.BTE = 1. When CAN_CTRL2.BTE = 0, DTSEG1 has no effect. Data Phase Time Segment 1 = (DTSEG1 + 1) × Time-Quanta Time-Quantum = one Sclock period

Yuntu Microelectronics

17.3.3.50 CAN ETDC Register

The CAN_ETDC register contains extended versions of the fields CAN_FDCTRL.TDCOFF and CAN_FDCTRL.TDCVAL.
CAN_ETDC is used by the hardware only if CAN_CTRL2.BTE is set; otherwise a write operation has no effect and all fields are read as zero.
The CAN_ETDC register is not available when parameter CIA_BIT_TIMING is 0.
NOTE: See the transmitter delay compensation section of the CAN Protocol Standard (ISO 11898-1) for additional details.

Offset: bfch

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ETDCEN	TDMDIS								ETDCOFF						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ETDCFAIL								ETDCVAL							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.60: CAN ETDC Register Description

Field	Function
31 ETDCEN	**Transceiver Delay Compensation Enable - Enables or disables the TDC feature.** ETDCEN can be written in Freeze mode only. NOTE: TDC must be disabled when Loop-Back mode is enabled (see CAN_CTRL1.LPB). 0b - TDC is disabled. 1b - TDC is enabled.
30	**Transceiver Delay Measurement Disable –Disables or enables transceiver delay measurement.**

Yuntu Microelectronics

Field	Function
TDMDIS	When TDC measurement is disabled, the secondary sample point position is determined only by the enhanced TDC offset (ETDCOFF) field. Otherwise, if TCD measurement is enabled, then the secondary sample point position is determined by the sum of the transceiver delay measurement plus the enhanced TDC offset. TDMDIS is not affected by soft reset. TDMDIS can be enabled only if CAN_CTRL2.BTE is set. 0b - TDC measurement is enabled. 1b - TDC measurement is disabled.
22 - 16 ETDCOFF	**Enhanced Transceiver Delay Compensation Offset** ETDCOFF contains the offset value to be added to the measured transceiver’s loop delay in order to define the position of the delayed comparison point when bit rate switching is active. See “Transceiver Delay Compensation” on page 187 for additional details on how the loop delay measurement is performed. ETDCOFF can be written in Freeze mode only. Its value is defined in protocol engine (PE) clock periods and must be selected to be smaller than the CAN bit duration in the data phase bit rate for proper operation. ETDCOFF must not be configured as zero. NOTE: If CAN_CTRL2.BTE is set after a chip-level hard reset, ETCDOFF is read as 0x1.
15 ETDCFAIL	**Transceiver Delay Compensation Fail** ETDCFAIL indicates when the Transceiver Delay Compensation (TDC) mechanism is out of range, unable to compensate the transceiver’s loop delay and successfully compare the delayed received bits to the transmitted ones ETDCFAIL is set the first time the FlexCAN detects the out of range condition. To clear ETDCFAIL, write 1 to it. See ”Transceiver Delay Compensation”. 0b - Measured loop delay is in range. 1b - Measured loop delay is out of range.
7 - 0 ETDCVAL	**Enhanced Transceiver Delay Compensation Value** ETDCVAL contains the ETDCOFF field added to the measured value of the transceiver loop delay in the latest transmitted CAN FD frame with BRS equal to recessive. ETDCVAL is only updated by the hardware if the ETDCEN bit is set. ETDCVAL is affected by soft reset. NOTE: If TDMDIS is set, ETDCVAL stores ETDCOFF only.

Yuntu Microelectronics

17.3.3.51 CAN FDCTRL Register

The CAN_FDCTRL register contains control bits for CAN FD operation. CAN_FDCTRL also defines the data size of message buffers allocated in different partitions of RAM (memory blocks).

When an 8-byte payload is selected: - Block R0 allocates MB0 to MB31. - Block R1 allocates MB32 to MB63. - Block R2 allocates MB64 to MB95. - Block R3 allocates MB96 to MB127.

When a payload larger than 8 bytes is selected, the maximum number of MBs in a block is limited as

described in below table.

Table 17.61: CAN Number of Message Buffers

Payload Size	Maximum Number of Message Buffers per RAM Block
8 bytes	32
16 bytes	21
32 bytes	12
64 bytes	7

NOTE: One memory block fits exactly 32 MBs with 8-byte payload. For other payload sizes, empty memory may exist between the last MB in a block and the beginning of the next block. This empty memory corresponds to less than one MB, and must not be used.

The contents of CAN_FDCTRL are not affected by soft reset. The CAN_FDCTRL register is not available when parameter FD_EN is 0.

Offset: c00h

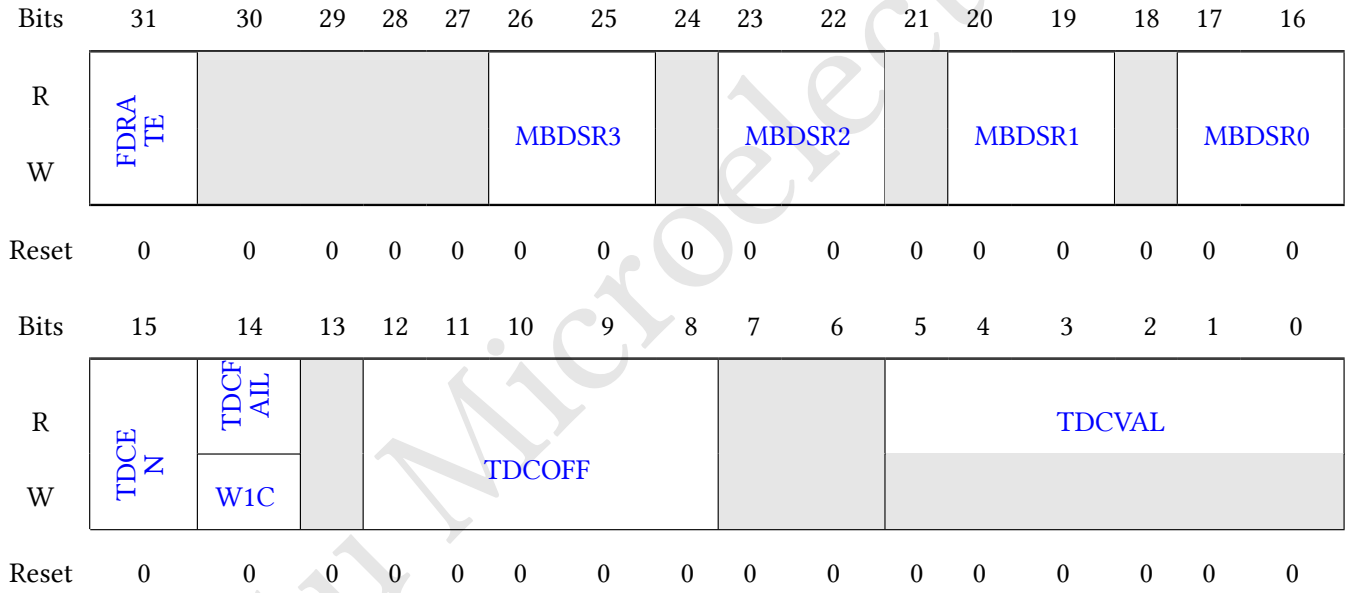


Table 17.62: CAN FDCTRL Register Description

Field	Function
31	**Bit Rate Switch Enable –Enables the effect of the Bit Rate Switch (BRS bit) during the data phase of Tx messages.**

Yuntu Microelectronics

Field	Function
FDRATE	The CPU can write the FDRATE bit any time. However, its effect becomes active only when the CAN bus is in the Wait for Bus Idle, Bus Idle, or Bus Off state, or when the current frame under reception or transmission reaches the interframe space. By negating FDRATE, the CPU can force all bits in CAN FD messages to be transmitted in nominal bit rate, regardless of the value of the BRS bit of the Tx MBs. 0b - Transmit a frame in nominal rate. The BRS bit in the Tx MB has no effect. 1b - Transmit a frame with bit rate switching if the BRS bit in the Tx MB is recessive.
26 - 25 MBDSR3	**Message Buffer Data Size for Region 3 –Selects the data size.** For region R3 of message buffers allocated in RAM: • 00: Selects 8 bytes per message buffer. • 01: Selects 16 bytes per message buffer. • 10: Selects 32 bytes per message buffer. • 11: Selects 64 bytes per message buffer. MBDSR3 can be written in Freeze mode only.
23 - 22 MBDSR2	**Message Buffer Data Size for Region 2 –Selects the data size.** For region R2 of message buffers allocated in RAM: • 00: Selects 8 bytes per message buffer. • 01: Selects 16 bytes per message buffer. • 10: Selects 32 bytes per message buffer. • 11: Selects 64 bytes per message buffer. MBDSR2 can be written in Freeze mode only.
20 - 19 MBDSR1	**Message Buffer Data Size for Region 1 –Selects the data size.** For region R1 of message buffers allocated in RAM: • 00: Selects 8 bytes per message buffer. • 01: Selects 16 bytes per message buffer. • 10: Selects 32 bytes per message buffer. • 11: Selects 64 bytes per message buffer. MBDSR1 can be written in Freeze mode only.
17 - 16 MBDSR0	**Message Buffer Data Size for Region 0 –Selects the data size.** For region R0 of message buffers allocated in RAM: • 00: Selects 8 bytes per message buffer. • 01: Selects 16 bytes per message buffer. • 10: Selects 32 bytes per message buffer. • 11: Selects 64 bytes per message buffer. MBDSR0 can be written in Freeze mode only.
15	**Transceiver Delay Compensation Enable –Enables or disables the TDC feature.**

Yuntu Microelectronics

Field	Function
TDCEN	TDCEN can be written in Freeze mode only. NOTE: TDC must be disabled when Loop-Back mode is enabled (see CAN_CTRL1.LPB). NOTE: If CAN_CTRL2.BTE =1, TDCEN is read as zero and a write operation has no effect. 0b - TDC is disabled. 1b - TDC is enabled.
14 TDCFAIL	**Transceiver Delay Compensation Fail** TDCFAIL indicates when the Transceiver Delay Compensation (TDC) mechanism is out of range, unable to compensate the transceiver’ s loop delay, and successfully compare the delayed received bits to the transmitted ones. TDCFAIL is set the first time the FlexCAN detects the out of range condition. To clear TDCFAIL, write 1 to it. See Transceiver Delay Compensation. NOTE: If CAN_CTRL2.BTE = 1, TDCFAIL is read as zero and a write operation has no effect. 0b - Measured loop delay is in range 1b - Measured loop delay is out of range
12 - 8 TDCOFF	**Transceiver Delay Compensation Offset** TDCOFF contains the offset value to be added to the measured transceiver’ s loop delay in order to define the position of the delayed comparison point when bit rate switching is active. See Transceiver Delay Compensation for more details on how the loop delay measurement is performed. TDCOFF can be written in Freeze mode only. Its value can be defined in PE clock periods and must be selected to be smaller than the CAN bit duration in the data bit rate for proper operation. NOTE: It is not recommended to use TDCOFF equal to zero. NOTE: If CAN_CTRL2.BTE = 1, TDCOFF is read as zero and a write operation has no effect.
5 - 0 TDCVAL	**Transceiver Delay Compensation Value** TDCVAL contains the value of the transceiver loop delay measured from the transmitted EDL to R0 transition edge to the respective received one added to the TDCOFF value. This value is an integer multiple of the PE clock period. See ”Protocol Timing” for more details on how the loop delay measurement is performed. NOTE: If CAN_CTRL2.BTE = 1, TDCVAL is read as zero.

Yuntu Microelectronics

17.3.3.52 CAN FDCBT Register

The CAN_FDCBT register stores the CAN bit timing variables used in the data phase of CAN FD messages when CAN_FDCTRL.FDRATE is set, compatible with CAN FD specification. FPRESDIV, FPROPSEG, FPSEG1, FPSEG2, and FRJW are used to define the time quantum duration, the number of time quanta per CAN bit, and the sample point position for the data bit rate portion of a CAN FD message with the BRS bit set.

The contents of CAN_FDCBT are not affected by soft reset.

NOTE: The sum of the Fast Propagation Segment (FPROPSEG) and Fast Phase Segment 1 (FPSEG1) must be at least two time quanta.

NOTE: If CAN_CTRL2.BTE = 1, the CAN_FDCBT register is read as zero and a write operation has no effect.

The CAN_FDCBT register is not available when parameter FD_EN is 0.

Offset: c04h

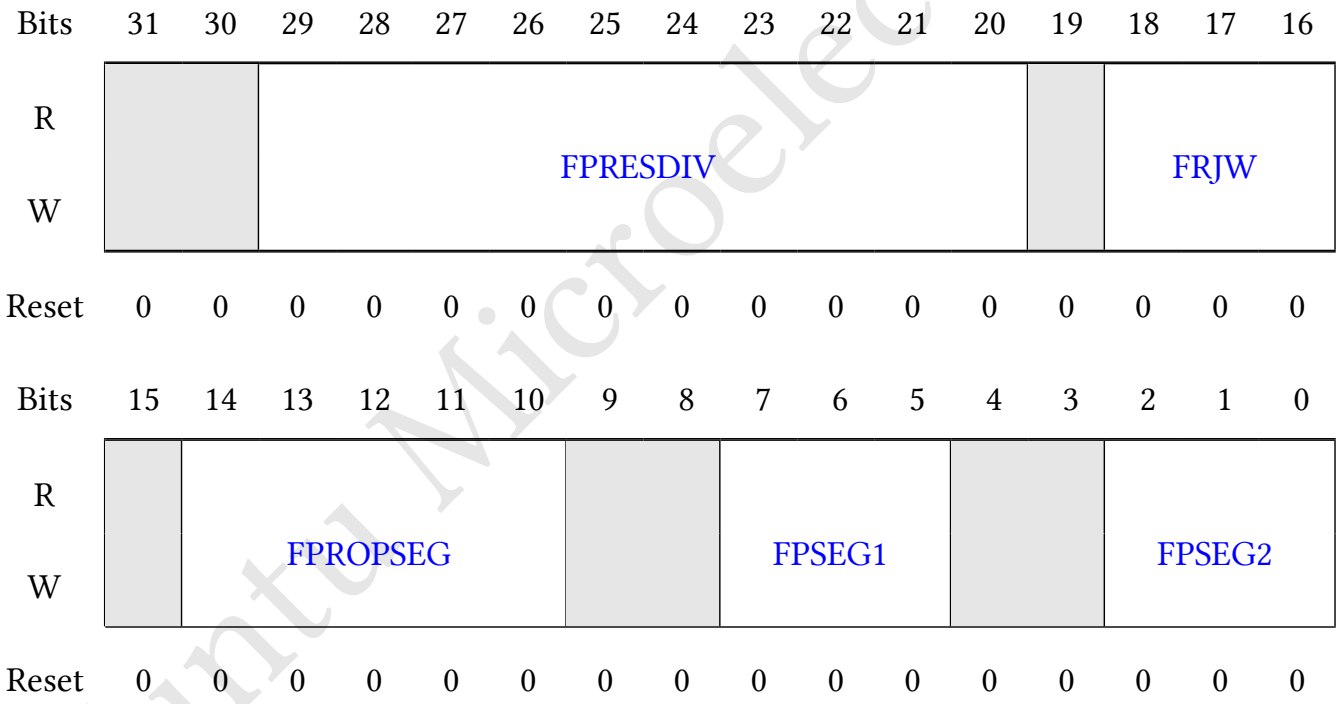


Table 17.63: CAN FDCBT Register Description

Field	Function
29 - 20	**Fast Prescaler Division Factor**

Yuntu Microelectronics

Field	Function
FPRESDIV	FPRESDIV defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency in the data bit rate portion of a CAN FD message with the BRS bit set. The Sclock period defines the time quantum of the CAN FD protocol for the data bit rate. Sclock frequency = PE clock frequency / (FPRESDIV + 1). NOTE: To minimize errors when processing FD frames, use the same value for FPRESDIV and PRESDIV (in CAN_CBT or CAN_CTRL1). For more details, see "CAN FD Frames". FPRESDIV can be written only in Freeze mode because it is blocked by hardware in other modes
18 - 16 FRJW	**Fast Resync Jump Width** FRJW defines the maximum number of time quanta that a bit time can be changed by one re-synchronization in the data bit rate portion of a CAN FD message with the BRS bit set. Resync Jump Width = FSJW + 1. One time quantum is equal to the Sclock period. FRJW can be written only in Freeze mode because it is blocked by hardware in other modes.
14 - 10 FPROPSEG	**Fast Propagation Segment** FPROPSEG defines the length of the Propagation Segment in the bit time in the data bit rate portion of a CAN FD message with the BRS bit set. Propagation Segment Time = FPROPSEG × Time-Quanta Time-Quantum = one Sclock period FPROPSEG can be written only in Freeze mode because it is blocked by hardware in other modes.
7 - 5 FPSEG1	**Fast Phase Segment 1** FPSEG1 defines the length of Phase Segment 1 in the bit time in the data bit rate portion of a CAN FD message with the BRS bit set. Phase Segment 1 = (FPSEG1 + 1) × Time-Quanta Time-Quantum = one Sclock period FPSEG1 can be written only in Freeze mode because it is blocked by hardware in other modes.
2 - 0 FPSEG2	**Fast Phase Segment 2** FPSEG2 defines the length of Phase Segment 2 in the data bit rate portion of a CAN FD message with the BRS bit set. Phase Segment 2 = (FPSEG2 + 1) × Time-Quanta Time-Quantum = one Sclock period FPSEG2 can be written only in Freeze mode because it is blocked by hardware in other modes.

Yuntu Microelectronics

17.3.3.53 CAN FDCRC Register

The CAN_FDCRC register provides information about the CRC of transmitted messages.

FlexCAN uses different CRC polynomials for different frame formats:

- The CRC_15 polynomial is used for all frames in CAN format.
- The CRC_17 polynomial is used for frames in CAN FD format with a data field up to sixteen bytes.
- The CRC_21 polynomial is used for frames in CAN FD format with a data field longer than sixteen bytes.

Each polynomial shown below results in a Hamming Distance of 6. CAN_FDCRC is updated at the same time the Tx Interrupt Flag is asserted.

- CRC_15 = 0xC599: $x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$
- CRC_17 = 0x3685B: $x^{17} + x^{16} + x^{14} + x^{13} + x^{11} + x^6 + x^4 + x^3 + x^1 + 1$
- CRC_21 = 0x302899: $x^{21} + x^{20} + x^{13} + x^{11} + x^7 + x^4 + x^3 + 1$

The CAN_FDCRC register is not available when parameter FD_EN is 0.

Offset: c08h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		FD_MBCRC										FD_TXCRC				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FD_TXCRC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17.64: CAN FDCRC Register Description

Field	Function
30 - 24 FD_MBCRC	**CRC Mailbox Number for FD_TXCRC** FD_MBCRC indicates the number of the mailbox corresponding to the value in the FD_TXCRC field, for both FD and non-FD frames. FD_MBCRC reports the same information as in the CAN_CRCR.MBCRC bit field.

Yuntu Microelectronics

Field	Function
20 - 0 FD_TXCRC	**Extended Transmitted CRC Value** FD_TXCRC contains the CRC value calculated over the most recent transmitted message. Different CRC polynomials are used for different frame formats. A 15-bit polynomial, CRC_15, is used for all frames in CAN format. The second 17-bit polynomial, CRC_17, is used for frames in CAN FD format with a data field up to sixteen bytes long. The third 21-bit polynomial, CRC_21, is used for frames in CAN FD format with a data field longer than sixteen bytes. For CRC_15 and CRC_17, the 6 most significant bits and the 4 most significant bits are reported as zeros, respectively. For CRC_15, this register has the same content as CAN_CRCR.

17.3.3.54 CAN ERFCR Register

The CAN_ERFCR register is not available when parameter ERX_FIFO_EN is 0. The CAN_ERFCR register is not available when parameter ERX_FIFO_EN is 0.

Table 17.65: CAN DMALW Value and Number of Enhanced RX FIFO Elements Transferred

DMALW	Number of 32-bit Words Transferred	Last FIFO Address
0	1	0x2000
1	2	0x2004
2	3	0x2008
3	4	0x200C
4	5	0x2010
5	6	0x2014
6	7	0x2018
7	8	0x201C
8	9	0x2020
9	10	0x2024
10	11	0x2028
11	12	0x202C
12	13	0x2030
13	14	0x2034
14	15	0x2038
15	16	0x203C
16	17	0x2040
17	18	0x2044
18	19	0x2048
19	20	0x204C

Yuntu Microelectronics

Table 17.66: CAN NEXIF Value and Number of Extended or Standard ID Filter Elements

NEXIF	NFE	Number of Extended ID Filter Elements	Number of Standard ID Filter Elements
0	63	0	128
1	63	1	126
2	63	2	124
3	63	3	122
4	63	4	120
5	63	5	118
6	63	6	116
7	63	7	114
8	63	8	112
9	63	9	110
10	63	10	108
11	63	11	106
12	63	12	104
13	63	13	102
14	63	14	100
15	63	15	98
16	63	16	96
17	63	17	94
18	63	18	92
19	63	19	90
20	63	20	88
21	63	21	86
22	63	22	84
23	63	23	82
24	63	24	80
25	63	25	78
26	63	26	76
27	63	27	74
28	63	28	72
29	63	29	70
30	63	30	68
31	63	31	66
32	63	32	64
33	63	33	62
34	63	34	60
35	63	35	58
36	63	36	56
37	63	37	54
38	63	38	52
39	63	39	50
40	63	40	48
41	63	41	46
42	63	42	44

Yuntu Microelectronics

Table 17.66 continued from previous page

NEXIF	NFE	Number of Extended ID Filter Elements	Number of Standard ID Filter Elements
43	63	43	42
44	63	44	40
45	63	45	38
46	63	46	36
47	63	47	34
48	63	48	32
49	63	49	30
50	63	50	28
51	63	51	26
52	63	52	24
53	63	53	22
54	63	54	20
55	63	55	18
56	63	56	16
57	63	57	14
58	63	58	12
59	63	59	10
60	63	60	8
61	63	61	6
62	63	62	4
63	63	63	2
64	63	64	0

Table 17.67: CAN NFE Value and Number of Enhanced RX FIFO Filter Elements

NFE	Maximum Number of Extended ID Filter Elements (NEXIF = NFE + 1)	Maximum Number of Standard ID Filter Elements (NEXIF = 0)
0	1	2
1	2	4
2	3	6
3	4	8
4	5	10
5	6	12
6	7	14
7	8	16
8	9	18
9	10	20
10	11	22
11	12	24
12	13	26
13	14	28
14	15	30
15	16	32
16	17	34

Yuntu Microelectronics

Table 17.67 continued from previous page

NFE	Maximum Number of Extended ID Filter Elements (NEXIF = NFE + 1)	Maximum Number of Standard ID Filter Elements (NEXIF = 0)
17	18	36
18	19	38
19	20	40
20	21	42
21	22	44
22	23	46
23	24	48
24	25	50
25	26	52
26	27	54
27	28	56
28	29	58
29	30	60
30	31	62
31	32	64
32	33	66
33	34	68
34	35	70
35	36	72
36	37	74
37	38	76
38	39	78
39	40	80
40	41	82
41	42	84
42	43	86
43	44	88
44	45	90
45	46	92
46	47	94
47	48	96
48	49	98
49	50	100
50	51	102
51	52	104
52	53	106
53	54	108
54	55	110
55	56	112
56	57	114
57	58	116
58	59	118
59	60	120

Yuntu Microelectronics

Table 17.67 continued from previous page

NFE	Maximum Number of Extended ID Filter Elements (NEXIF = NFE + 1)	Maximum Number of Standard ID Filter Elements (NEXIF = 0)
60	61	122
61	62	124
62	63	126
63	64	128

Offset: c0ch

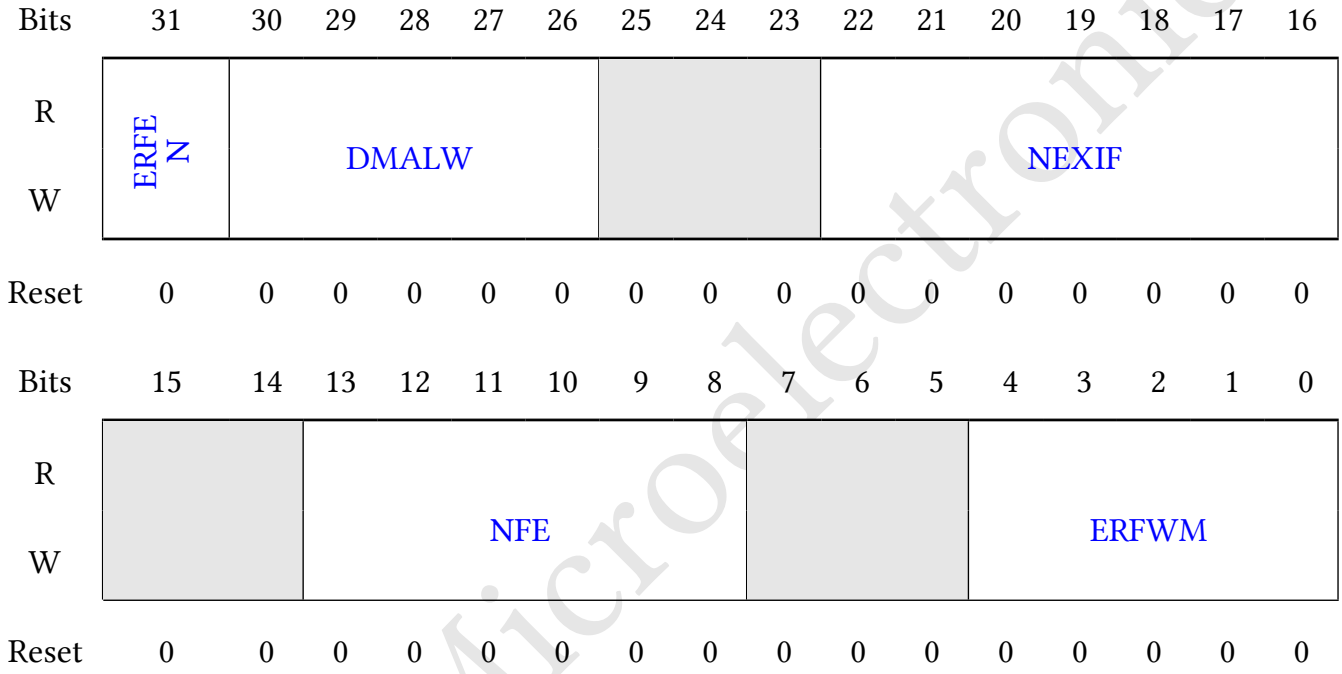


Table 17.68: CAN ERFCR Register Description

Field	Function
31 ERFEN	**Enhanced Rx FIFO Enable - Enables or disables the Enhanced Rx FIFO.** ERFEN can be written only in Freeze mode because it is blocked by hardware in other modes. NOTE: ERFEN must not be set if CAN_MCR.RFEN is set. 0b - Disable Enhanced Rx FIFO. 1b - Enable Enhanced Rx FIFO.
30 - 26	**DMA Last Word - Defines the last DMA address for each Enhanced Rx FIFO element.**

Yuntu Microelectronics

Field	Function
DMALW	Table ”CAN DMALW value and number of enhanced RX FIFO elements transferred” shows the number of elements and the last address for each Enhanced Rx FIFO element according to the value of DMALW. DMALW values not shown in Table ”CAN DMALW value and number of enhanced RX FIFO elements transferred” are reserved and must not be used. DMALW can be written only in Freeze mode because it is blocked by hardware in other modes.
22 - 16 NEXIF	**Number of Extended ID Filter Elements - Defines the number of extended ID filter elements used during the Enhanced Rx FIFO matching process.** NEXIF must be less than or equal to NFE + 1. The number of standard ID filter elements can be determined by the following equation: Number of Standard ID Filter Elements = 2 x (NFE - NEXIF +1) ”CAN NEXIF value and number of extended or standard ID filter elements” shows the number of extended ID filters and standard ID filters available for Enhanced Rx FIFO if all the filter elements are used. NEXIF can be written only in Freeze mode because it is blocked by hardware in other modes.
13 - 8 NFE	**Number of Enhanced Rx FIFO Filter Elements - Defines the total number of filter elements used during the Enhanced Rx FIFO matching process, as shown in Table 68.** NFE can be written only in Freeze mode because it is blocked by hardware in other modes.
4 - 0 ERFWM	**Enhanced Rx FIFO Watermark - Defines the minimum number of CAN messages stored in the Enhanced Rx FIFO.** Minimum number of CAN messages = ERFWM + 1. When ERFWM + 1 is reached, ERFSR.ERFWMI is set. ERFWM can be written only in Freeze mode because it is blocked by hardware in other modes.

Yuntu Microelectronics

17.3.3.55 CAN ERFIER Register

The CAN_ERFIER register is not available when parameter ERX_FIFO_EN is 0.

Offset: c10h

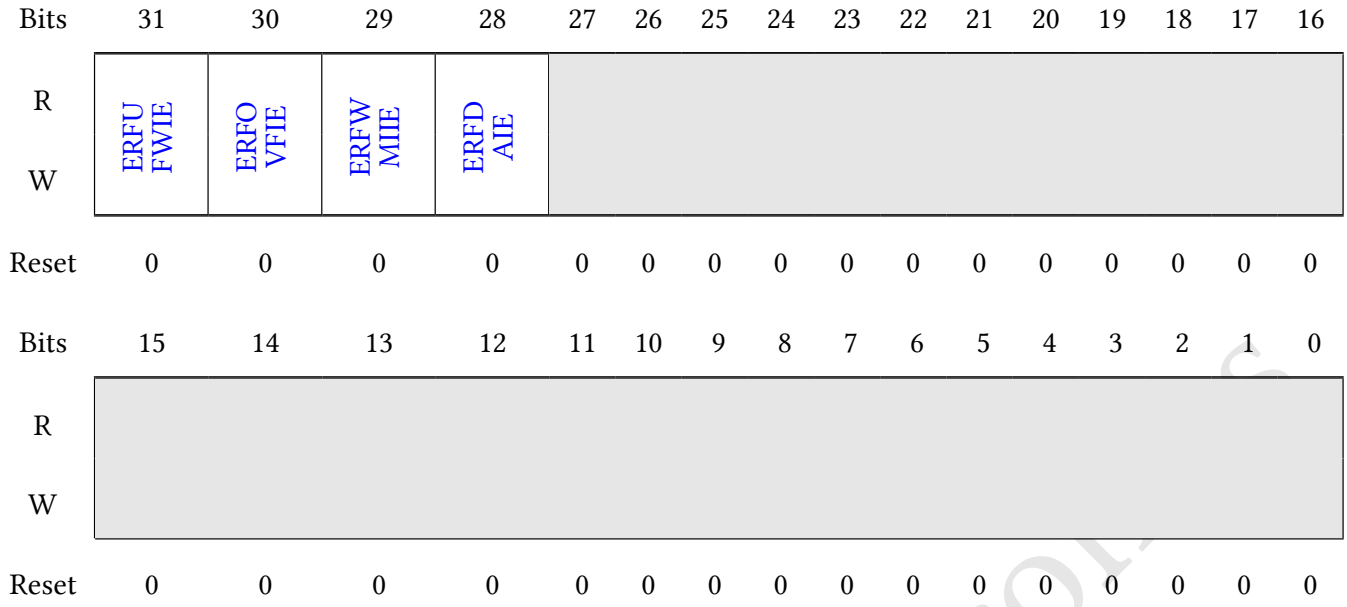


Table 17.69: CAN ERFIER Register Description

Field	Function
31 ERFUFWIE	**Enhanced Rx FIFO Underflow Interrupt Enable –Enables or disables the interrupt for Enhanced Rx FIFO underflow (ERFSR.ERFUFW).** 0b - Disable Enhanced Rx FIFO underflow interrupt. 1b - Enable Enhanced Rx FIFO underflow interrupt.
30 ERFOVFIE	**Enhanced Rx FIFO Overflow Interrupt Enable –Enables or disables the interrupt for Enhanced Rx FIFO overflow (ERFSR.ERFOVF).** 0b - Disable Enhanced Rx FIFO overflow interrupt. 1b - Enable Enhanced Rx FIFO overflow interrupt.
29 ERFWMIIE	**Enhanced Rx FIFO Watermark Indication Interrupt Enable –Enables or disables the interrupt for Enhanced Rx FIFO watermark (ERFSR.ERFWMI).** 0b - Disable Enhanced Rx FIFO watermark interrupt. 1b - Enable Enhanced Rx FIFO watermark interrupt.
28 ERFDAIE	**Enhanced Rx FIFO Data Available Interrupt Enable –Enables or disables the interrupt for Enhanced Rx FIFO data available (ERFSR.ERFDA).** 0b - Disable Enhanced Rx FIFO data available interrupt. 1b - Enable Enhanced Rx FIFO data available interrupt.

17.3.3.56 CAN ERFSR Register

The CAN_ERFSR register is not available when parameter ERX_FIFO_EN is 0.

Offset: c14h

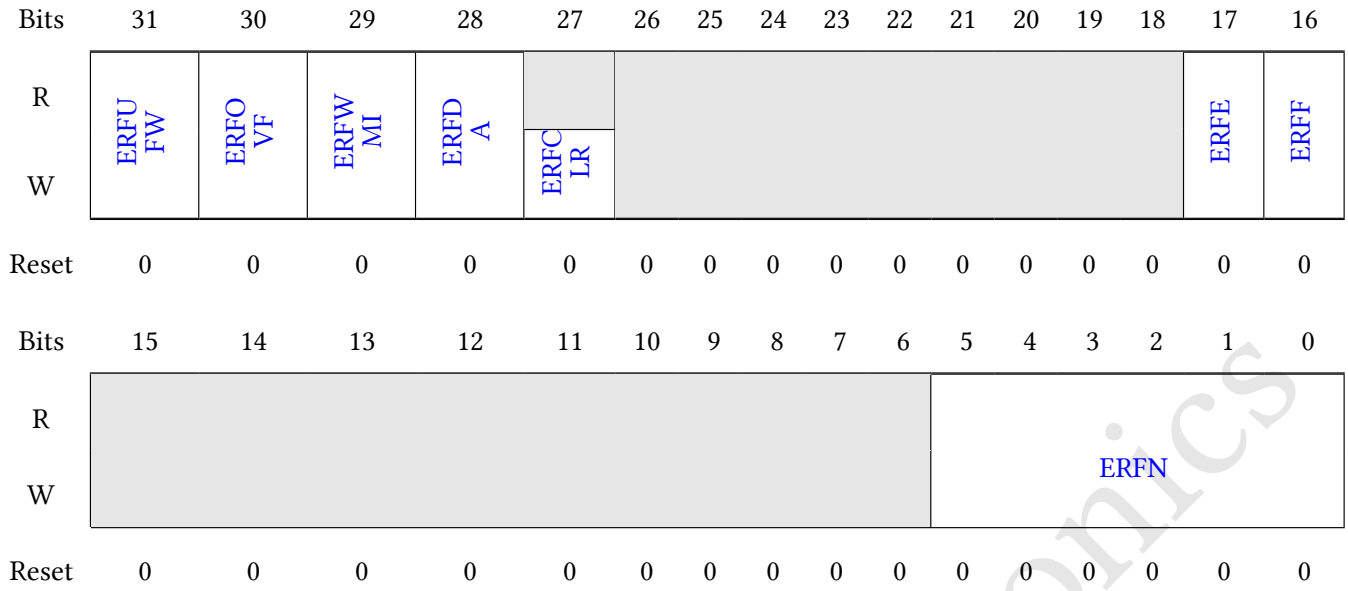


Table 17.70: CAN ERFSR Register Description

Field	Function
31 ERFUFW	**Enhanced Rx FIFO Underflow –Indicates that an underflow condition occurred in the Enhanced Rx FIFO** ERFUFW is cleared by writing 1 to it. 0b - Enhanced Rx FIFO underflow has not occurred. 1b - Enhanced Rx FIFO underflow has occurred. Interrupt ipi_int_erfufw is asserted if CAN_ERFIER.ERFUFWIE = 1.
30 ERFOVF	**Enhanced Rx FIFO Overflow –Indicates that an overflow condition occurred in the Enhanced Rx FIFO** ERFOVF is cleared by writing 1 to it. 0b - Enhanced Rx FIFO overflow has not occurred. 1b - Enhanced Rx FIFO overflow has occurred. Interrupt ipi_int_erfovf is asserted if CAN_ERFIER.ERFOVFIE = 1.
29 ERFWMI	**Enhanced Rx FIFO Watermark Indication –Set by hardware if the number of messages in the Enhanced Rx FIFO is greater than the watermark defined in CAN_ERFCR.ERFWM.** ERFWMI is cleared by writing 1 to it. 0b - The number of messages in the Enhanced Rx FIFO is not greater than the watermark. 1b - The number of messages in the Enhanced Rx FIFO is greater than the watermark. Interrupt ipi_int_erfwm is asserted if CAN_ERFIER.ERFWMIIE = 1.
28	**Enhanced Rx FIFO Data Available –Set by hardware if there is at least one message stored in the Enhanced Rx FIFO.**

Yuntu Microelectronics

Field	Function
ERFDA	ERFDA is cleared by writing 1 to it. 0b - There is no message stored in the Enhanced Rx FIFO. 1b - There is at least one message stored in the Enhanced Rx FIFO. Interrupt ipi_int_erfda is asserted if CAN_ERFIER.ERFDAIE = 1.
27 ERFCLR	**Enhanced Rx FIFO Clear** Writing 1 to ERFCLR when the FlexCAN is in Freeze mode clears the Enhanced Rx FIFO content. Writing 0 to ERFCLR or writing to ERFCLR when the FlexCAN is not in Freeze mode has no effect.
17 ERFE	**Enhanced Rx FIFO Empty** 0b - The Enhanced Rx FIFO is not empty. 1b - The Enhanced Rx FIFO is empty.
16 ERFF	**Enhanced Rx FIFO Full** 0b - The Enhanced Rx FIFO is not full. 1b - The Enhanced Rx FIFO is full.
5 - 0 ERFN	**Enhanced Rx FIFO Elements –Indicates the number of CAN messages stored in the Enhanced Rx FIFO.**

17.3.3.57 CAN HR_TIME_STAMPn Register

Each High Resolution Time Stamp register (CAN_HR_TIME_STAMPn) stores a copy of the 32-bit timer at the start or end of a CAN frame for the associated message buffer.

CAN_HR_TIME_STAMP0 stores the 32-bit time stamp associated with MB0;

CAN_HR_TIME_STAMP1 stores the 32-bit time stamp associated with MB1; and so on.

The CAN_HR_TIME_STAMPn registers are not affected by reset and must be written only in Freeze mode.

The CAN_HR_TIME_STAMPn registers are not available if parameter TIME_STAMP_EN is 0.

Table 17.71: CAN High Resolution Time Stamp Register Operation

TSTAMPCAP	Captured Time Base	Capture Point
00	None	None
01	32 bits of high-resolution on-chip timer	Seventh bit of the end of frame field for transmission and sixth bit of the end of frame field for reception.
10	32 bits of high-resolution on-chip timer	Start of frame.
11	32 bits of high-resolution on-chip timer	Start of frame for Classical CAN frame format and res bit for CAN FD frame format.

Offset: c30h + n * 4h

Yuntu Microelectronics

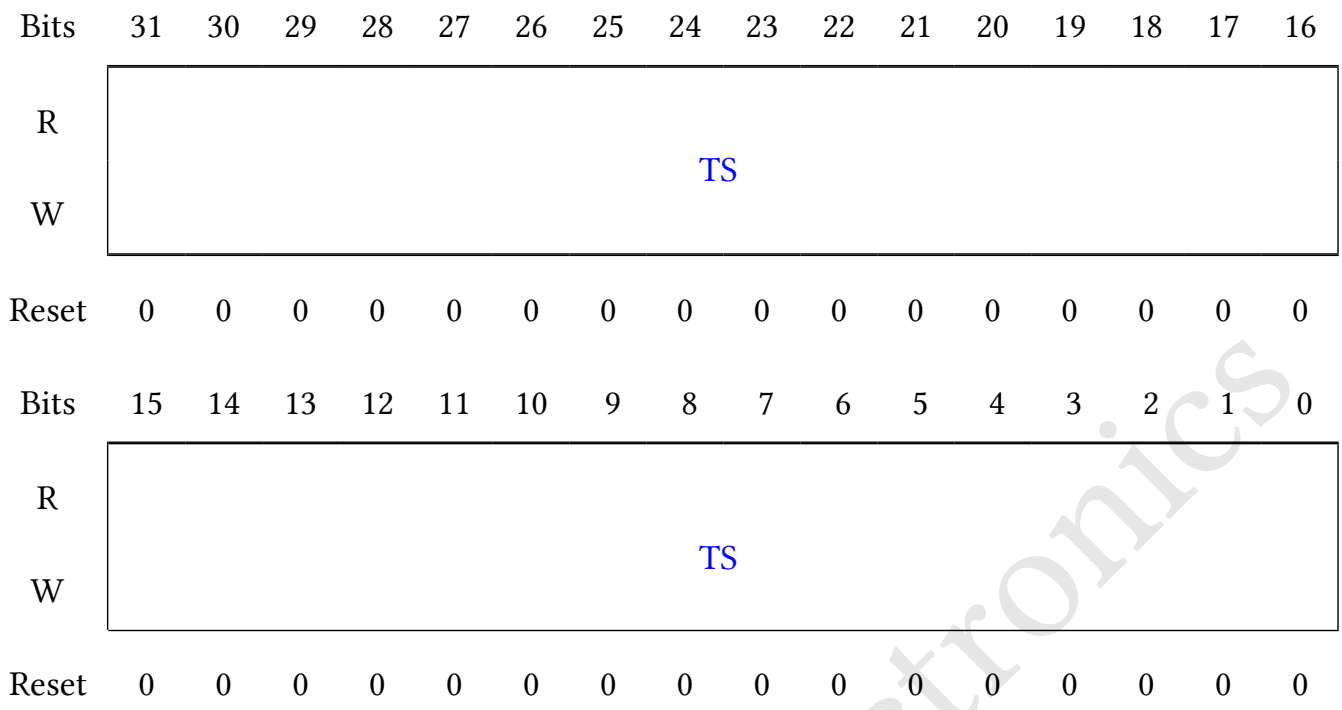


Table 17.72: CAN HR_TIME_STAMPn Register Description

Field	Function
31 - 0 TS	**High Resolution Time Stamp** Capture of 32-bit timer value for the corresponding message buffer. The capture point depends on the value of CAN_CTRL2.TSTAMPCAP, as shown in ”CAN high resolution time stamp register operation”.

Yuntu Microelectronics

17.3.3.58 CAN ERFFELn Register

The Enhanced Rx FIFO Filter Element Registers are used to store the filter elements of the Enhanced Rx FIFO.
For standard ID filtering, each CAN_ERFFELn register stores one filter element. For extended ID filtering, each pair of CAN_ERFFELn registers stores one filter element.
CAN_ERFFELn registers can only be accessed by the CPU when the FlexCAN is in Freeze mode; otherwise, they are blocked by hardware. The CAN_ERFFELn registers are not affected by reset. They are located in RAM and must be explicitly initialized prior to any reception.
The CAN_ERFFELn registers are not available if parameter ERX_FIFO_EN is 0.

Offset: 3000h + n * 4h

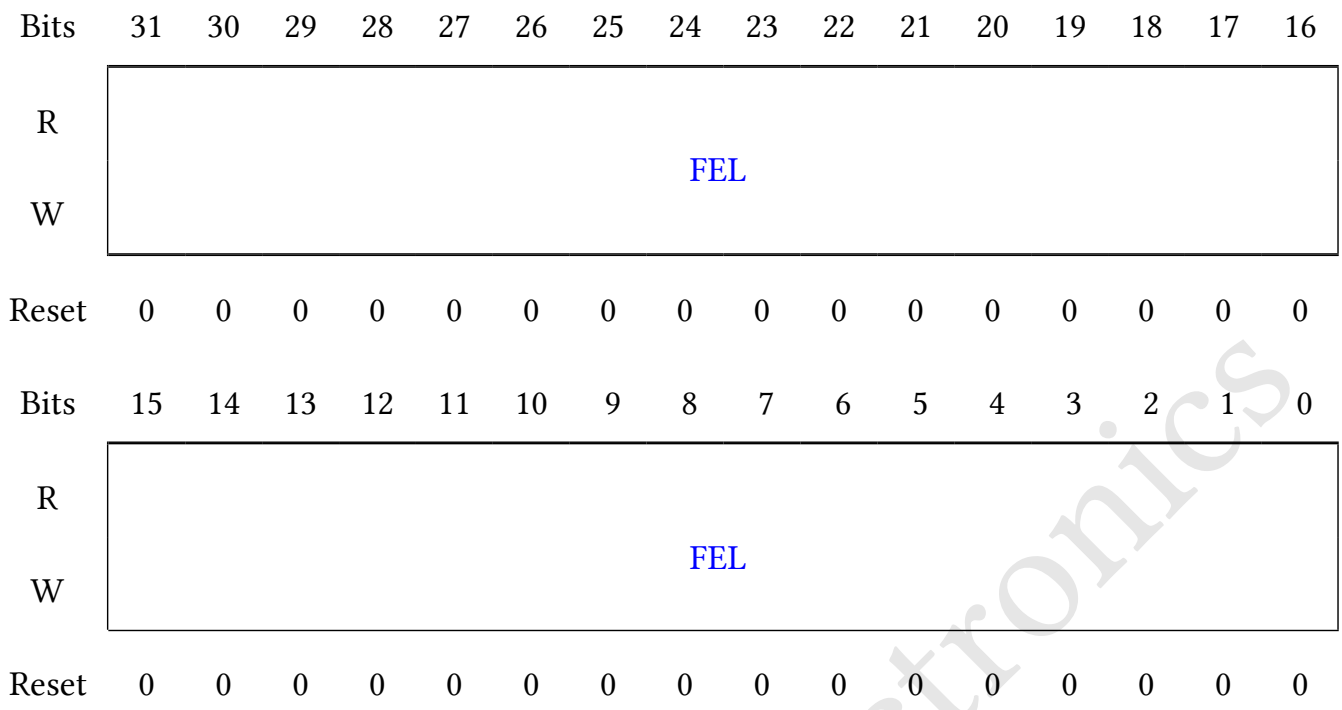


Table 17.73: CAN ERFFELn Register Description

Field	Function
31 - 0 FEL	**Filter Element Bits** Each filter element is used during the match process and, if the matching criteria are met, a message is stored in the Enhanced Rx FIFO.

17.3.4 Message Buffer Structure

Below figure shows the message buffer structure used by the FlexCAN module. Both Extended (29-bit identifier) and Standard (11-bit identifier) frames used in the CAN specification (Version 2.0 Part B) are represented. Each individual MB is formed by 16, 24, 40, or 72 bytes, depending on the quantity of data bytes allocated for the message payload: 8, 16, 32, or 64 data bytes, respectively.

The memory area from 0x80 to 0x87F is used by the mailboxes. When CAN FD is enabled, the exact address for each MB depends on the size of its payload. See “FlexCAN Memory Partition for CAN FD” for more detailed information.

When CAN FD is enabled, the memory area from 0x1000 to 0x17FF can be used for implementing up to 28 64-byte additional MBs. This additional MB region (enhanced MB memory) can be used partially or completely only if the total number of MBs configured by CAN_MCR.MAXMB is 128 or less (CAN_MCR.MAXMB = 127 or less). This means that the maximum number of MBs in the area from 0x80 to 0x87F and area from 0x1000 to 0x17FF is 128. See “FlexCAN Memory Partition for CAN FD” for more detailed information.

Yuntu Microelectronics

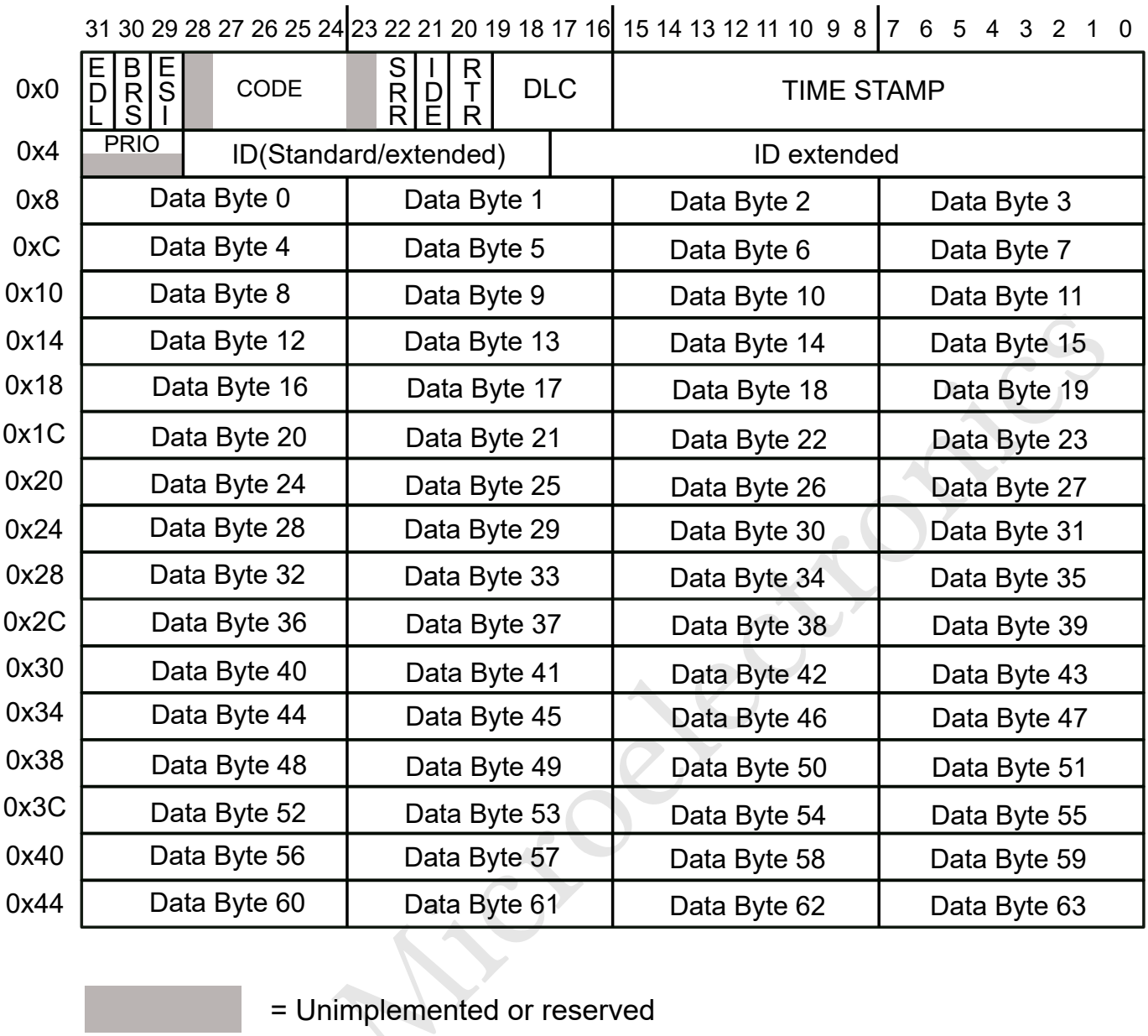


Figure 17.2: Message Buffer Structure Example with 64 Bytes Payload

EDL – Extended Data Length

- The EDL bit distinguishes between CAN format and CAN FD format frames. EDL must not be set for message buffers configured to RANSWER with code field 0b1010. See “CAN message buffer code for RX buffers”).

BRS – Bit Rate Switch

- The BRS bit defines whether the bit rate is switched inside a CAN FD format frame.

ESI – Error State Indicator

- The ESI bit indicates if the transmitting node is error active or error passive.

CODE – Message Buffer Code

- The CODE field can be accessed (read or write) by the CPU and by the FlexCAN module, as part of the message buffer matching and arbitration process. The encoding is shown in “CAN message

Yuntu Microelectronics

buffer code for RX buffers") and "CAN message buffer code for TX buffers"). See "Functional Description" for additional information.

Table 17.74: CAN Message Buffer Code for RX Buffers

CODE Description	Rx code BEFORE receive new frame	SRV[1]	Rx code AFTER successful reception[2]	RRS[3]	Comment
0000: INACTIVE - MB is not active	INACTIVE	–	–	–	MB does not participate in the matching process
0100: EMPTY - MB is active and empty	EMPTY	–	FULL	–	When a frame is received successfully (after the Move-in process), the CODE field is automatically updated to FULL.
0010: FULL - MB is full	FULL	Yes	FULL	–	The act of reading the C/S word followed by unlocking the MB (SRV) does not make the code return to EMPTY. It remains FULL. If a new frame is moved to the MB after the MB was serviced, the code still remains FULL. See "Matching Process" for matching details related to FULL code.
		No	OVERRUN	–	If the MB is FULL and a new frame is moved to this MB before the CPU services it, the CODE field is automatically updated to OVERRUN. See "Matching Process" for details about overrun behavior
0110: OVERRUN - MB is being overwritten into a full buffer.	OVERRUN	Yes	FULL	–	If the CODE field indicates OVERRUN and CPU has serviced the MB, when a new frame is moved to the MB, the code returns to FULL.
		No	OVERRUN	–	If the CODE field already indicates OVERRUN, and another new frame must be moved, the MB will be overwritten again, and the code will remain OVERRUN. See "Matching Process" for details about overrun behavior
1010: RANSWER[4] - A frame was configured to recognize a Remote Request Frame and transmit a Response Frame in return.[5]	RANSWER	–	TANSWER (1110)	0	A Remote Answer was configured to recognize a remote request frame received. After that, an MB is set to transmit a response frame. The code is automatically changed to TANSWER (1110). See "Matching Process" for details. If CAN_CTRL2.RRS is negated, transmit a response frame whenever a remote request frame with the same ID is received.

Yuntu Microelectronics

Table 17.74 continued from previous page

CODE Description	Rx code BEFORE receive new frame	SRV[1]	Rx code AFTER successful reception[2]	RRS[3]	Comment
		–	–	1	This code is ignored during matching and arbitration process. See "Matching Process" for details.
CODE[0]=1: BUSY - FlexCAN is updating the contents of the MB. The CPU must not access the MB	BUSY[6]	–	FULL	–	Indicates that the MB is being updated. It will be negated automatically and does not interfere with the next CODE.
		–	OVERRUN	–	

1. SRV: Serviced MB. MB was read and unlocked by reading CAN_TIMER or other MB.
2. A frame is considered a successful reception after the frame is moved to an MB(move-in process). See "Move-in" for details.
3. Remote Request Stored bit, see "Control 2 Register (CAN_CTRL2)" for details.
4. Code 1010 is not considered Tx and an MB with this code should not be aborted.
5. Code 1010 must be used in message buffers configured in CAN FD format, having the EDL bit set.
6. For Tx MBs, the BUSY bit should be ignored upon read, except when CAN_MCR.AEN is set. If this bit is asserted, the corresponding MB does not participate in the matching process.

Table 17.75: CAN Message Buffer Code for TX Buffers

CODE Description	Tx Code BEFORE Tx frame	MB RTR	Tx Code AFTER successful transmission	Comment
1000: INACTIVE - MB is not active.	INACTIVE	–	–	MB does not participate in arbitration process
1001: ABORT - MB is aborted	ABORT	–	–	MB does not participate in arbitration process
1100: DATA - MB is a Tx Data Frame (MB RTR must be 0)	DATA	0	INACTIVE	Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state.
1100: REMOTE – MB is a Tx Remote Request Frame (MB RTR must be 1).	REMOTE	1	EMPTY	Transmit remote request frame unconditionally once. After transmission, the MB automatically becomes an Rx Empty MB with the same ID.

Yuntu Microelectronics

Table 17.75 continued from previous page

CODE Description	Tx Code BEFORE Tx frame	MB RTR	Tx Code AFTER successful transmission	Comment
1110: TANSWER – MB is a Tx Response Frame from an incoming Remote Request Frame.	TANSWER	–	RANSWER	This is an intermediate code that is automatically written to the MB by the CHI as a result of a match to a remote request frame. The remote response frame will be transmitted unconditionally once, and then the code will automatically return to RANSWER (1010). The CPU can also write this code with the same effect. The remote response frame can be either a data frame or another remote request frame depending on the RTR bit value. See ”Matching Process” and ”Arbitration Process” for details.

SRR – Substitute Remote Request

SRR is a fixed recessive bit, used only in extended format. SRR must be set to one by the user for transmission (Tx buffers) and will be stored with the value received on the CAN bus for Rx receiving buffers. It can be received as either recessive or dominant. If the FlexCAN receives this bit as dominant, then it is interpreted as an arbitration loss.

- 1: Recessive value is compulsory for transmission in extended format frames.
- 0: Dominant is not a valid value for transmission in extended format frames.

IDE – ID Extended Bit

IDE identifies whether the frame format is standard or extended:

- 1: Frame format is extended.
- 0: Frame format is standard.

RTR – Remote Transmission Request

The RTR bit affects the behavior of remote frames and is part of the reception filter:

- 1: The current MB may have a remote request frame to be transmitted if MB is Tx. If the MB is Rx, then incoming remote request frames may be stored.
- 0: The current MB has a data frame to be transmitted. In an Rx MB, it may be considered in matching processes.

See “CAN message buffer code for RX buffers”) and ”CAN message buffer code for TX buffers”), and the description of the CAN_CTRL2.RRS bit for additional details.

If the FlexCAN transmits RTR as 1 (recessive) and receives it as 0 (dominant), it is interpreted as an arbitration loss. If RTR is transmitted as 0 (dominant) and received as 1 (recessive), the FlexCAN treats it as a bit error. If the value received matches the value transmitted, it is considered a successful bit transmission.

NOTE: When configuring CAN FD frames, the RTR bit must be negated.

DLC – Length of Data in Bytes

Yuntu Microelectronics

The 4-bit DLC field is the length (in bytes) of the Rx or Tx data, which is located in offset 0x8 through 0xF of the MB space(see: “message buffer structure example with 64 bytes payload”). In reception, DLC is written by the FlexCAN module, copied from the DLC (Data Length Code) field of the received frame. In transmission, DLC is written by the CPU and corresponds to the DLC field value of the frame to be transmitted. When RTR = 1, the frame to be transmitted is a remote frame and does not include the data field, regardless of the DLC field(see ”CAN data bytes validity”).

TIME STAMP – Free-Running Counter Time Stamp

The 16-bit TIME STAMP field is a copy of the Free-Running Timer or 16 bits of the 32-bit onchip timer, captured for Tx and Rx frames as described in below Table.

Table 17.76: CAN Time Stamp Operation

CAN_CTRL2			Captured time base	Capture point
TSTAMPCAP	MBTSBASE	TIMER_SRC		
00	xx	0	CAN_TIMER incremented by CAN bit clock	Second bit of identifier field
00	xx	1	CAN_TIMER incremented by tmr_tick_ext	Second bit of identifier field
xx	00	0	CAN_TIMER incremented by CAN bit clock	Second bit of identifier field
xx	00	1	CAN_TIMER incremented by tmr_tick_ext	Second bit of identifier field
01	01	x	Lower 16 bits of high-resolution on-chip timer	Seventh bit of End of Frame field for transmission; sixth bit of End of Frame field for reception
01	10	x	Upper 16 bits of high-resolution on-chip timer	Seventh bit of End of Frame field for transmission; sixth bit of End of Frame field for reception
10	01	x	Lower 16 bits of high-resolution on-chip timer	Start of Frame
10	10	x	Upper 16 bits of high-resolution on-chip timer	Start of Frame
11	01	x	Lower 16 bits of high-resolution on-chip timer	Start of Frame for Classical CAN frame format; res bit for CAN FD frame format
11	10	x	Upper 16 bits of high-resolution on-chip timer	Start of Frame for Classical CAN frame format; res bit for CAN FD frame format

PRIO – Local Priority

The 3-bit PRIO field is used only when the CAN_MCR.LPRIO_EN bit is set and only makes sense for Tx mailboxes. The PRIO field is not transmitted. It is appended to the regular ID to define the transmission priority. See “Arbitration Process”.

ID – Frame Identifier

In standard frame format, only the 11 most significant bits (28 to 18) are used for frame identification for both receive and transmit, and the 18 least significant bits are ignored. In extended frame format, all bits are used for frame identification for both receive and transmit.

Yuntu Microelectronics

DATA BYTE 0 to 63 – Data Field

Up to 64 bytes can be used for a data frame, depending on the size of payload selected for the Message Buffers.

For Rx frames, the data is stored as it is received from the CAN bus. DATA BYTE (n) is valid only if n is less than DLC as shown in below Table: “CAN data bytes validity”).

Table 17.77: CAN Data Bytes Validity

DLC	Valid Data Bytes
0	None
1	DATA BYTE 0
2	DATA BYTE 0 to 1
3	DATA BYTE 0 to 2
4	DATA BYTE 0 to 3
5	DATA BYTE 0 to 4
6	DATA BYTE 0 to 5
7	DATA BYTE 0 to 6
8	DATA BYTE 0 to 7
9	DATA BYTE 0 to 11
10	DATA BYTE 0 to 15
11	DATA BYTE 0 to 19
12	DATA BYTE 0 to 23
13	DATA BYTE 0 to 31
14	DATA BYTE 0 to 47
15	DATA BYTE 0 to 63

17.3.5 FlexCAN Memory Partition for CAN FD

When CAN FD is enabled, the FlexCAN RAM can be partitioned in blocks of 512 bytes. Each block can accommodate a number of message buffers that depends on the configuration provided by CAN_FDCTRL.MBDSRn bit fields as shown in Table: “CAN RAM partition”.

Table 17.78: CAN RAM Partition

RAM Block	Number of MBS with8 Bytes (default range)	Size Control Bit Field in CAN_FDCTRL	Number of MBs of Different Sizes, per Block
0	0 to 31	MBDSR0	MBDSR0=00, 32 MBs with 8 bytes payload MBDSR0=01, 21 MBs with 16 bytes payload MBDSR0=10, 12 MBs with 32 bytes payload MBDSR0=11, 7 MBs with 64 bytes payload
1	32 to 63	MBDSR1	MBDSR1=00, 32 MBs with 8 bytes payload MBDSR1=01, 21 MBs with 16 bytes payload MBDSR1=10, 12 MBs with 32 bytes payload MBDSR1=11, 7 MBs with 64 bytes payload

Yuntu Microelectronics

Table 17.78 continued from previous page

RAM Block	Number of MBS with8 Bytes (default range)	Size Control Bit Field in CAN_FDCTRL	Number of MBs of Different Sizes, per Block
2	64 to 95	MBDSR2	MBDSR2=00, 32 MBs with 8 bytes payload MBDSR2=01, 21 MBs with 16 bytes payload MBDSR2=10, 12 MBs with 32 bytes payload MBDSR2=11, 7 MBs with 64 bytes payload
3	96 to 127	MBDSR3	MBDSR3=00, 32 MBs with 8 bytes payload MBDSR3=01, 21 MBs with 16 bytes payload MBDSR3=10, 12 MBs with 32 bytes payload MBDSR3=11, 7 MBs with 64 bytes payload
Blocks 4-7 exist when configuration parameter ENHANCE_MB_MEM =1			
4	–	–	7 MBs with 64 bytes payload (if the total number of MBs in blocks 0-3 is 121 or less)
5	–	–	7 MBs with 64 bytes payload (if the total number of MBs in blocks 0-3 is 114 or less)
6	–	–	7 MBs with 64 bytes payload (if the total number of MBs in blocks 0-3 is 107 or less)
7	–	–	7 MBs with 64 bytes payload (if the total number of MBs in blocks 0-3 is 100 or less)

When payload sizes of 16, 32, or 64 bytes are configured in some or all RAM blocks, the total number of MBs and the respective number order may differ from the default configuration of 8 bytes. For example, “CAN RAM partition example” indicates how the message buffers will be arranged into RAM when:

- Block 0 is configured to 8 bytes payload.
- Block 1 is configured to 16 bytes payload.
- Block 2 is configured to 32 bytes payload.
- Block 3 is configured to 64 bytes payload.
- CAN_MCR.MAXMB = 95 (96 message buffers are used).
- Configuration parameter ENHANCE_MB_MEM = 1.

Table 17.79: CAN RAM Partition Example

RAM block	Payload size	Number of MBs in the RAM block	Message Buffer range
0	CAN_FDCTRL[MBDSR0]=00, 8 bytes payload	32	0 to 31
1	CAN_FDCTRL[MBDSR1]=01, 16 bytes payload	21	32 to 52
2	CAN_FDCTRL[MBDSR2]=10, 32 bytes payload	12	53 to 64
3	CAN_FDCTRL[MBDSR2]=11, 64 bytes payload	7	65 to 71
4	64 bytes payload	7	72 to 78
5	64 bytes payload	7	79 to 85
6	64 bytes payload	7	86 to 92
7	64 bytes payload	3	93 to 95

Yuntu Microelectronics

17.3.6 FlexCAN Message Buffer Memory Map

The FlexCAN memory buffers are allocated in memory according to the following tables.

Table 17.80: CAN 8-byte Message Buffers

Address Offset (hex)	MBDSR = 00 (8-Byte Payload)
0080	MB0
0090	MB1
00A0	MB2
00B0	MB3
00C0	MB4
00D0	MB5
00E0	MB6
00F0	MB7
0100	MB8
0110	MB9
0120	MB10
0130	MB11
0140	MB12
0150	MB13
0160	MB14
0170	MB15
0180	MB16
0190	MB17
01A0	MB18
01B0	MB19
01C0	MB20
01D0	MB21
01E0	MB22
01F0	MB23
0200	MB24
0210	MB25
0220	MB26
0230	MB27
0240	MB28
0250	MB29
0260	MB30
0270	MB31
0280	MB32
0290	MB33
02A0	MB34
02B0	MB35
02C0	MB36
02D0	MB37
02E0	MB38
02F0	MB39
0300	MB40

Yuntu Microelectronics

Table 17.80 continued from previous page

Address Offset (hex)	MBDSR = 00 (8-Byte Payload)
0310	MB41
0320	MB42
0330	MB43
0340	MB44
0350	MB45
0360	MB46
0370	MB47
0380	MB48
0390	MB49
03A0	MB50
03B0	MB51
03C0	MB52
03D0	MB53
03E0	MB54
03F0	MB55
0400	MB56
0410	MB57
0420	MB58
0430	MB59
0440	MB60
0450	MB61
0460	MB62
0470	MB63
0480	MB64
0490	MB65
04A0	MB66
04B0	MB67
04C0	MB68
04D0	MB69
04E0	MB70
04F0	MB71
0500	MB72
0510	MB73
0520	MB74
0530	MB75
0540	MB76
0550	MB77
0560	MB78
0570	MB79
0580	MB80
0590	MB81
05A0	MB82
05B0	MB83
05C0	MB84
05D0	MB85

Yuntu Microelectronics

Table 17.80 continued from previous page

Address Offset (hex)	MBDSR = 00 (8-Byte Payload)
05E0	MB86
05F0	MB87
0600	MB88
0610	MB89
0620	MB90
0630	MB91
0640	MB92
0650	MB93
0660	MB94
0670	MB95
0680	MB96
0690	MB97
06A0	MB98
06B0	MB99
06C0	MB100
06D0	MB101
06E0	MB102
06F0	MB103
0700	MB104
0710	MB105
0720	MB106
0730	MB107
0740	MB108
0750	MB109
0760	MB110
0770	MB111
0780	MB112
0790	MB113
07A0	MB114
07B0	MB115
07C0	MB116
07D0	MB117
07E0	MB118
07F0	MB119
0800	MB120
0810	MB121
0820	MB122
0830	MB123
0840	MB124
0850	MB125
0860	MB126
0870	MB127

Yuntu Microelectronics

Table 17.81: CAN 16-byte Message Buffers

Address Offset (hex)	MBDSR = 01 (16-Byte Payload)
0080	MB0
0098	MB1
00B0	MB2
00C8	MB3
00E0	MB4
00F8	MB5
0110	MB6
0128	MB7
0140	MB8
0158	MB9
0170	MB10
0188	MB11
01A0	MB12
01B8	MB13
01D0	MB14
01E8	MB15
0200	MB16
0218	MB17
0230	MB18
0248	MB19
0260	MB20
0280	MB21
0298	MB22
02B0	MB23
02C8	MB24
02E0	MB25
02F8	MB26
0310	MB27
0328	MB28
0340	MB29
0358	MB30
0370	MB31
0388	MB32
03A0	MB33
03B8	MB34
03D0	MB35
03E8	MB36
0400	MB37
0418	MB38
0430	MB39
0448	MB40
0460	MB41
0480	MB42
0498	MB43

Yuntu Microelectronics

Table 17.81 continued from previous page

Address Offset (hex)	MBDSR = 01 (16-Byte Payload)
04B0	MB44
04C8	MB45
04E0	MB46
04F8	MB47
0510	MB48
0528	MB49
0540	MB50
0558	MB51
0570	MB52
0588	MB53
05A0	MB54
05B8	MB55
05D0	MB56
05E8	MB57
0600	MB58
0618	MB59
0630	MB60
0648	MB61
0660	MB62
0680	MB63
0698	MB64
06B0	MB65
06C8	MB66
06E0	MB67
06F8	MB68
0710	MB69
0728	MB70
0740	MB71
0758	MB72
0770	MB73
0788	MB74
07A0	MB75
07B8	MB76
07D0	MB77
07E8	MB78
0800	MB79
0818	MB80
0830	MB81
0848	MB82
0860	MB83

Yuntu Microelectronics

Table 17.82: CAN 32-byte Message Buffers

Address Offset (hex)	MBDSR = 10 (32-Byte Payload)
0080	MB0
00A8	MB1
00D0	MB2
00F8	MB3
0120	MB4
0148	MB5
0170	MB6
0198	MB7
01C0	MB8
01E8	MB9
0210	MB10
0238	MB11
0280	MB12
02A8	MB13
02D0	MB14
02F8	MB15
0320	MB16
0348	MB17
0370	MB18
0398	MB19
03C0	MB20
03E8	MB21
0410	MB22
0438	MB23
0480	MB24
04A8	MB25
04D0	MB26
04F8	MB27
0520	MB28
0548	MB29
0570	MB30
0598	MB31
05C0	MB32
05E8	MB33
0610	MB34
0638	MB35
0680	MB36
06A8	MB37
06D0	MB38
06F8	MB39
0720	MB40
0748	MB41
0770	MB42
0798	MB43

Yuntu Microelectronics

Table 17.82 continued from previous page

Address Offset (hex)	MBDSR = 10 (32-Byte Payload)
07C0	MB44
07E8	MB45
0810	MB46
0838	MB47

Table 17.83: CAN 64-byte Message Buffers

Address Offset (hex)	MBDSR = 11 (64-Byte Payload)
0080	MB0
00C8	MB1
0110	MB2
0158	MB3
01A0	MB4
01E8	MB5
0230	MB6
0280	MB7
02C8	MB8
0310	MB9
0358	MB10
03A0	MB11
03E8	MB12
0430	MB13
0480	MB14
04C8	MB15
0510	MB16
0558	MB17
05A0	MB18
05E8	MB19
0630	MB20
0680	MB21
06C8	MB22
0710	MB23
0758	MB24
07A0	MB25
07E8	MB26
0830	MB27

Yuntu Microelectronics

17.3.7 Legacy Rx FIFO Structure

When the CAN_MCR.RFEN bit is set, the memory area from 0x80 to 0xDC (which is normally occupied by MBs 0–5) is used by the Legacy Rx FIFO engine.

The region 0x80–0x8C contains the output of the Legacy Rx FIFO, which must be read by the CPU as a message buffer. The Legacy Rx FIFO output contains the oldest message that has been received but not yet read. The region 0x90–0xDC is reserved for internal use of the Legacy Rx FIFO engine.

An additional memory area, which starts at 0xE0 and may extend up to 0x2DC (normally occupied by MBs 6–37) depending on the CAN_CTRL2.RFFN field setting, contains the ID filter table (configurable from 8 to 128 table elements), which specifies filtering criteria for accepting frames into the Legacy Rx FIFO.

Out of reset, the ID filter table flexible memory area defaults to 0xE0 and extends only to 0xFC, which corresponds to MBs 6 to 7 for RFFN = 0, for backward compatibility with previous versions of the FlexCAN.

below table shows the Legacy Rx FIFO data structure.

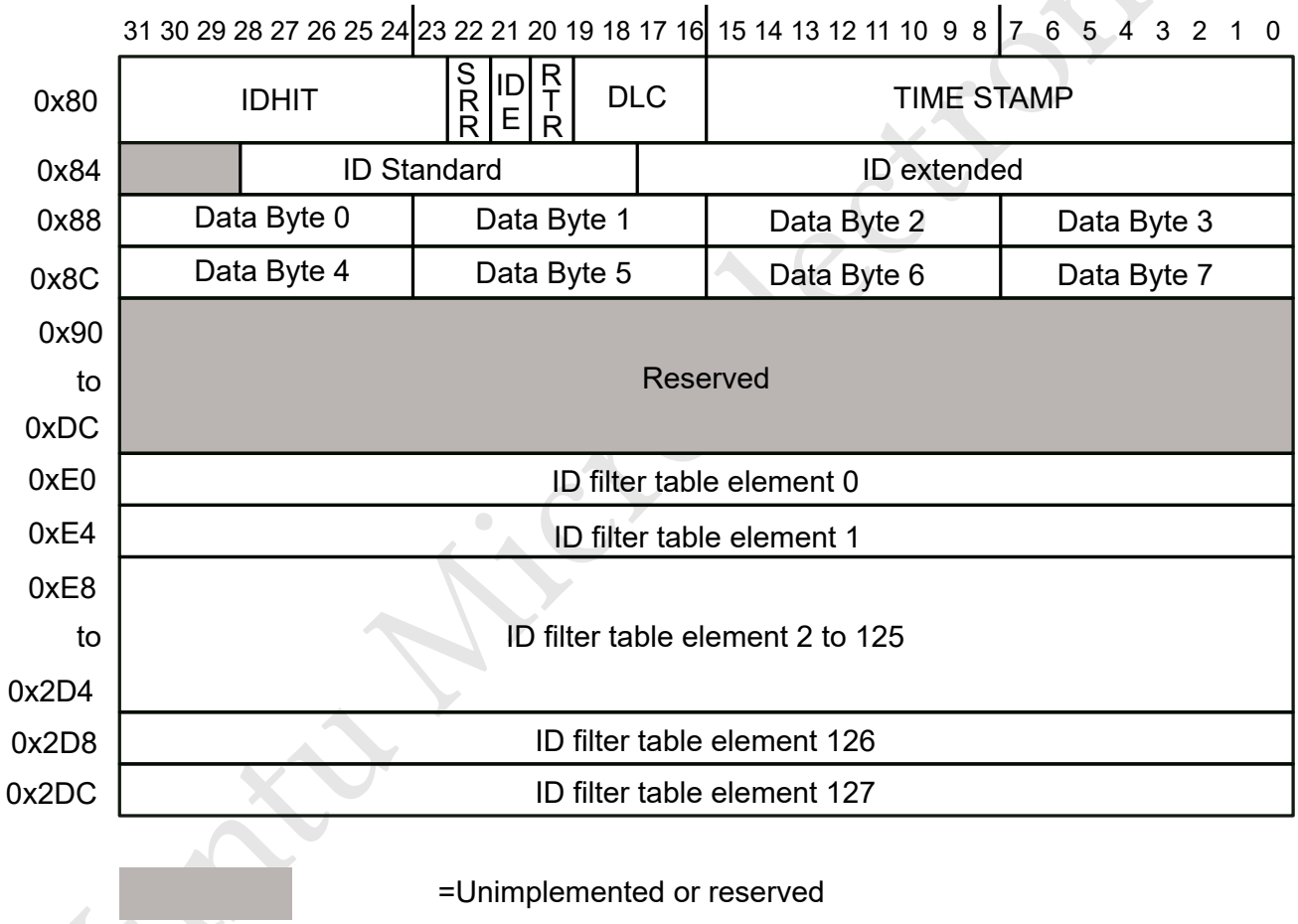


Figure 17.3: Legacy RX FIFO Structure

Each ID filter table element occupies an entire 32-bit word and can be compounded by one, two, or four Identifier Acceptance Filters (IDAF) depending on the CAN_MCR.IDAM field setting.

Below table shows the three different formats of the ID table elements. All elements of the table must have the same format, as selected by CAN_MCR.IDAM. See “Legacy Rx FIFO” for more information.

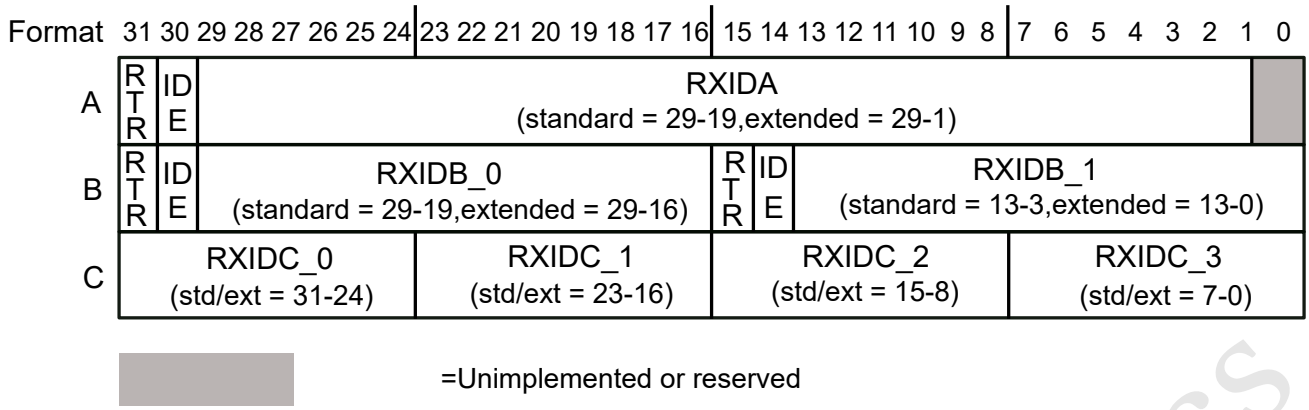


Figure 17.4: ID Filter Table Structure

RTR – Remote Frame

The RTR bit specifies if remote frames are accepted into the Legacy Rx FIFO if they match the target ID:

- 1 = Remote frames can be accepted and data frames are rejected.
- 0 = Remote frames are rejected and data frames can be accepted. IDE – Extended Frame

IDE – Extended Frame

The IDE bit specifies whether extended or standard frames are accepted into the Legacy Rx FIFO if they match the target ID:

- 1 = Extended frames can be accepted and standard frames are rejected.
- 0 = Extended frames are rejected and standard frames can be accepted.

RXIDA – Rx Frame Identifier (Format A)

Specifies an ID to be used as acceptance criteria for the Legacy Rx FIFO. In the standard frame format, only the 11 most significant bits (29 to 19) are used for frame identification. In the extended frame format, all bits are used.

RXIDB_0, RXIDB_1 – Rx Frame Identifier (Format B)

Specifies an ID to be used as acceptance criteria for the Legacy Rx FIFO. In the standard frame format, the 11 most significant bits (a full standard ID) (29 to 19 and 13 to 3) are used for frame identification. In the extended frame format, all 14 bits of the field are compared to the 14 most significant bits of the received ID.

RXIDC_0, RXIDC_1, RXIDC_2, RXIDC_3 – Rx Frame Identifier (Format C)

Specifies an ID to be used as acceptance criteria for the Legacy Rx FIFO. In both standard and extended frame formats, all 8 bits of the field are compared to the 8 most significant bits of the received ID.

IDHIT – Identifier Acceptance Filter Hit Indicator

The 9-bit IDHIT field indicates which Identifier Acceptance Filter was hit by the received message that is in the output of the Legacy Rx FIFO. See ‘‘Legacy Rx FIFO“} for more information.

Yuntu Microelectronics

17.3.8 Enhanced Rx FIFO Structure

When CAN_ERFCR.ERFEN is set, the Enhanced Rx FIFO is enabled. The region 0x2000-0x204C contains the output of the Enhanced Rx FIFO which must be read by the CPU as a message buffer. This output contains the oldest message that has been received but not yet read. "enhanced rx fifo structure" shows the Enhanced Rx FIFO data structure.

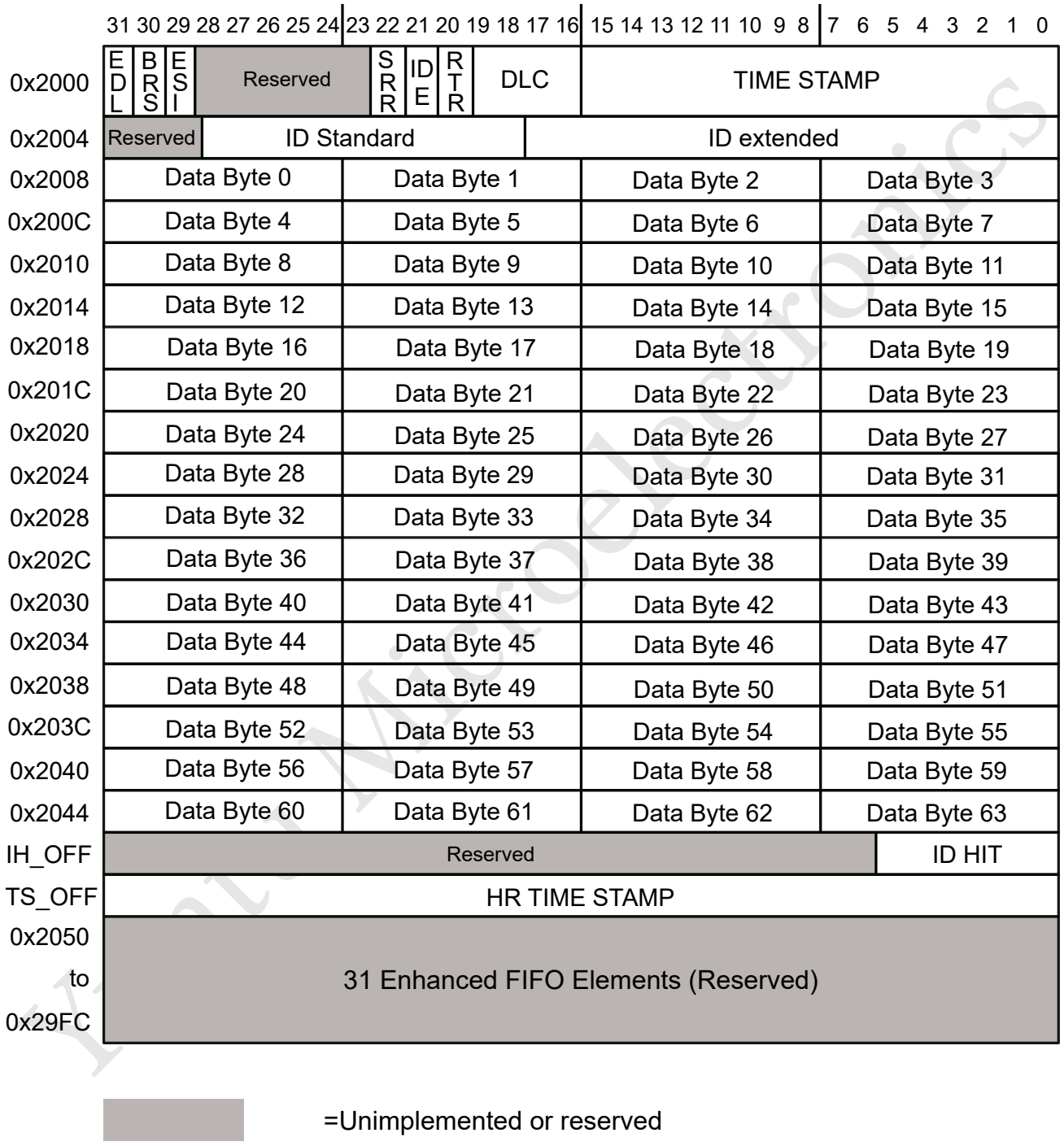


Figure 17.5: Enhanced RX FIFO Structure

NOTE: ID HIT offset and HR TIME STAMP offset are changed dynamically according to Data Length Code (DLC), as shown in below Table.

Yuntu Microelectronics

Table 17.84: CAN ID HIT and HR Time Stamp Offsets

Data Length Code (DLC)	ID HIT offset (IH_OFF)	HR TIME STAMP offset (TS_OFF)
0	0x2008	0x200C
1-4	0x200C	0x2010
5-8	0x2010	0x2014
9	0x2014	0x2018
10	0x2018	0x201C
11	0x201C	0x2020
12	0x2020	0x2024
13	0x2028	0x202C
14	0x2038	0x203C
15	0x2048	0x204C

EDL – Extended Data Length

The EDL bit distinguishes between classic CAN format and CAN FD format frames:

- 1 = CAN FD frame format.
- 0 = Classic CAN frame format.

BRS – Bit Rate Switch

The BRS bit indicates whether the bit rate is switched inside a CAN FD format frame:

- 1 = Bit rate is switched in a CAN FD frame.
- 0 = Bit rate is not switched in a CAN FD frame.

ESI – Error State Indicator

The ESI bit indicates if the transmitting node is error active or error passive:

- 1 = Transmitting node is error passive.
- 0 = Transmitting node is error active.

ESI is meaningful only if EDL = 1.

SRR – Substitute Remote Request

SRR is a fixed recessive bit, used only in extended format. Transmitting nodes always send it as recessive and receiving nodes can receive it as either recessive or dominant. If the FlexCAN receives the SRR bit as dominant, then it is interpreted as an arbitration loss.

IDE – ID Extended Bit

The IDE bit indicates whether the frame format is standard or extended:

- 1 = Frame format is extended.
- 0 = Frame format is standard.

RTR – Remote Frame

The RTR bit indicates whether the frame format is a data frame or remote request:

- 1 = Frame is remote request.
- 0 = Frame is a data frame.

Yuntu Microelectronics

DLC – Data Length Code

The 4-bit DLC field defines the number of bytes in the data field of a CAN frame. When RTR = 1, the frame is a remote request and does not include the data field, regardless of the DLC field.

TIME STAMP – 16-bit Time Stamp

The 16-bit TIME STAMP field is a copy of the Free-Running Timer, captured during the CAN frame.

ID – Frame Identifier

In the standard frame format, only the 11 most significant bits are used for frame identification. The 18 least significant bits are ignored. In the extended frame format, all 29 bits are used for frame identification.

DATA BYTE 0 to 63 – Data Field

Up to sixty four bytes can be stored in the data field.

ID HIT – Identifier Acceptance Filter Hit Indicator

The ID HIT field indicates which Enhanced Rx FIFO Filter Element (CAN_ERFFELn) was hit by the received message that is in the output of the Enhanced Rx FIFO. For each filter region, standard ID filter space and extended ID filter space, there is an independent index starting from zero. below Table shows how ID HIT is written by the FlexCAN according to each filter element.

Table 17.85: CAN ID HIT for Enhanced RX FIFO

Enhanced Rx FIFO Filter Element (CAN_ERFFEL)	ID HIT Value	Filter Element Type
CAN_ERFFEL0	0	Extended ID
CAN_ERFFEL1	1	Extended ID
...		Extended ID
CAN_ERFFEL(m-1)	m-1	Extended ID
CAN_ERFFEL(m)	0	Standard ID
CAN_ERFFEL(m+1)	1	Standard ID
...		Standard ID
CAN_ERFFEL(2n-m+1)	2x(n-m)+1	Standard ID

NOTE: m = NEXIF and n = NFE. If NEXIF = 0, then only standard ID filter elements exist. If NEXIF > NFE, then only extended ID filter elements exist.

HR TIME STAMP – High-Resolution Time Stamp

32-bit time base captured during the CAN frame. When CAN_CTRL2.TSTAMPCAP is nonzero, a 32-bit time base is captured from a dedicated on-chip timer which operates in free running mode. The 32-bit time base is captured through the timestamp_base[31:0] input.

17.4 Functional Description

The FlexCAN module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames. The mailbox system consists of a set of message buffers (MBs) that store configuration and control data, time stamp, message ID and data (see ”Message Buffer Structure”). The memory corresponding to the first 38 MBs can be configured to support a Legacy Rx FIFO reception scheme with a powerful ID filtering mechanism, capable of checking incoming frames against a table of

Yuntu Microelectronics

IDs (up to 128 extended IDs or 256 standard IDs or 512 8-bit ID slices), with individual mask register for up to 32 ID Filter Table elements.

For Classical CAN frames, simultaneous reception through the Legacy Rx FIFO and mailbox is supported. For CAN FD frames, reception is supported through mailboxes and the Enhanced Rx FIFO. For mailbox reception, a matching algorithm makes it possible to store received frames only into MBs that have the same ID programmed in the ID field. A masking scheme makes it possible to match the ID programmed on the MB with a range of IDs on received CAN frames. For transmission, an arbitration algorithm decides the prioritization of MBs to be transmitted based on the message ID (optionally augmented by 3 local priority bits) or the MB ordering.

A message buffer is said to be active at a given time if it can participate in both the matching and arbitration processes. An Rx MB with a 0b0000 code is inactive (see ”CAN message buffer code for RX buffers”). Similarly, a Tx MB with a 0b1000 or 0b1001 code is also inactive (see ”CAN message buffer code for TX buffers”).

The FlexCAN module can also receive and transmit messages in CAN FD format. The MBs are sized to adequately store the quantity of data bytes selected in the MBDSRn fields of the CAN_FDCTRL register. The quantity of FD MBs available for a given quantity of data bytes is described in ”CAN FD Control Register (CAN_FDCTRL)”. See also ”FlexCAN Memory Partition for CAN FD”.

17.4.1 Transmit Process

To transmit a CAN frame, the CPU must prepare a message buffer for transmission by executing the following procedure:

1. Check whether the respective interrupt bit is set and clear it.
2. If the MB is active (transmission pending), write the ABORT code (1001) to the CODE field of the Control and Status (C/S) word to request an abort of the transmission. Wait for the corresponding IFLAG bit to be asserted by polling the CAN_IFLAGn register or by the interrupt request if enabled by the respective IMASK bit. Then, read back the CODE field to check if the transmission was aborted or transmitted (see see ”Transmission Abort Mechanism”) and clear the corresponding interrupt flag.
3. Write the ID word, including local priority if enabled by CAN_MCR.LPRIOEN.
4. Write the payload data bytes.
5. Configure the C/S word for the MB as needed:
    - Set IDE to the ID type.
    - For a Remote Transmission Request, set RTR.
    - If CAN FD is enabled (CAN_MCR.FDEN = 1), configure the EDL and BRS bits. **NOTE:** The ESI field does not need to be written, and will automatically be transmitted dominant by error active nodes and recessive by error passive nodes. The exception is if the FlexCAN is operating as a network gateway, in which case, the CPU writes the ESI bit according to the error status of the node that sent the message. For details about the relationship between the written and transmitted values of ESI, see ”CAN written and transmitted values of ESI bit”.
    - Write the Data Length Code to the DLC field.
    - Activate the MB to transmit the CAN frame by setting the CODE field to 0xC.

NOTE: To maximize software performance, configure all the fields in the C/S word in one 32-bit write operation. If the fields are configured in separate writes, the CODE field must be the last write in the C/S word.

When the MB is activated, it participates in the arbitration process and is eventually transmitted according to its priority. When the DLC value stored in the MB selected for transmission is larger than

Yuntu Microelectronics

the respective MB payload size, the FlexCAN adds the necessary number of bytes with constant 0xCC pattern to complete the expected DLC.

At the end of the successful transmission:

- The value of the Free Running Timer is written into the TIME STAMP field.
- The CODE field in the C/S word is updated.
- CAN_CRC and CAN_FDCRC registers are updated.
- A status flag is set in CAN_IFLAGn.
- An interrupt is generated if allowed by the corresponding CAN_IFLAGn bit.

The new CODE field after transmission depends on the code that was used to activate the MB (see "CAN message buffer code for TX buffers").

When the Abort feature is enabled (CAN_MCR.AEN is asserted), after the Interrupt Flag is asserted for a mailbox configured as transmit buffer, the mailbox is blocked. Therefore, the CPU is not able to update it until the Interrupt Flag is negated by CPU. This means that the CPU must clear the corresponding CAN_IFLAGn bit before starting to prepare this MB for a new transmission or reception.

NOTE: If backwards compatibility is desired (CAN_MCR.AEN is negated), write the INACTIVE code (0b1000) to the CODE field to inactivate the MB. However, in this case, the pending frame may be transmitted without notification (see "Mailbox Inactivation").

17.4.2 Arbitration Process

The arbitration process scans the mailboxes, searching for the Tx mailbox that holds the message to be sent in the next opportunity. This mailbox is called the arbitration winner.

The scan starts from the lowest number mailbox and runs toward the higher ones.

The arbitration process is triggered in the following events:

- From the CRC field of the CAN frame. The start point depends on the CAN_CTRL2.TASD field value.
- During the Error Delimiter field of a CAN frame.
- During the Overload Delimiter field of a CAN frame.
- When the winner is inactivated and the CAN bus has still not reached the first bit of the Intermission field.
- When there is a CPU write to the C/S word of a winner MB and the CAN bus has still not reached the first bit of the Intermission field.
- When CHI is in Idle state and the CPU writes to the C/S word of any MB.
- When FlexCAN exits Bus Off state.
- Upon leaving Freeze mode or Low-Power mode.

If the arbitration process does not manage to evaluate all mailboxes before the CAN bus has reached the first bit of the Intermission field, the temporary arbitration winner is invalidated and the FlexCAN does not compete for the CAN bus in the next opportunity.

The arbitration process selects the winner among the active Tx mailboxes at the end of the scan according to the CAN_CTRL1.LBUF and CAN_MCR.LPRIOEN bit settings.

17.4.2.1 Lowest-Number Mailbox First

If CAN_CTRL1.LBUF is asserted, the first (lowest number) active Tx mailbox found is the arbitration winner. CAN_MCR.LPRIOEN bit has no effect when CAN_CTRL1.LBUF is asserted.

Yuntu Microelectronics

17.4.2.2 Highest-Priority Mailbox First

If CAN_CTRL1.LBUF is negated, then the arbitration process searches the active Tx mailbox with the highest priority, which means that this mailbox's frame would have a higher probability to win the arbitration on CAN bus when multiple external nodes compete for the bus at the same time.

The sequence of bits considered for this arbitration is called the arbitration value of the mailbox. The highest-priority Tx mailbox is the one that has the lowest arbitration value among all Tx mailboxes.

If two or more mailboxes have equivalent arbitration values, the mailbox with the lowest number is the arbitration winner.

The composition of the arbitration value depends on CAN_MCR.LPRIOEN.

17.4.2.2.1 Local Priority Disabled

If CAN_MCR.LPRIOEN is negated, Local Priority is disabled and the arbitration value is built in the exact sequence of bits as they would be transmitted in a CAN frame(see CAN composition of the arbitration value when local priority is disabled).

Table 17.86: CAN Composition of the Arbitration Value When Local Priority Is Disabled

Format	Mailbox Arbitration Value(32 bits)				
Standard(IDE=0)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bits)
Extended(IDE=1)	Extended ID[28:18] (11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

17.4.2.2.2 Local Priority Enabled

If CAN_MCR.LPRIOEN asserted, Local Priority is enabled. In this case, the mailbox PRIO field is included at the very left of the arbitration value(see CAN composition of the arbitration value when local priority is enabled).

Table 17.87: CAN Composition of the Arbitration Value When Local Priority Is Enabled

Format	Mailbox Arbitration Value(35 bits)					
Standard(IDE=0)	PRIO (3 bits)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bits)
Extended(IDE=1)	PRIO (3 bits)	Extended ID[28:18] (11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

Because the PRIO field is the most significant part of the arbitration value, mailboxes with low PRIO values have higher priority than mailboxes with high PRIO values, regardless of the rest of their arbitration values.

- **NOTE:** The PRIO field is not part of the frame on the CAN bus. Its purpose is only to affect the internal arbitration process.

Yuntu Microelectronics

17.4.2.3 Arbitration Completion

After the arbitration winner is found, its content is copied to a hidden auxiliary MB called Tx Serial Message Buffer (Tx SMB), which has the same structure as a normal MB but is not user accessible. This operation is called move-out and, after it is done, write access to the C/S word of the corresponding MB is blocked (if CAN_MCR.AEN is asserted). Write access is restored in the following events:

- After the MB is transmitted and the corresponding IFLAG bit is cleared by the CPU.
- The FlexCAN enters Freeze mode or Bus Off.
- The FlexCAN loses the bus arbitration or there is an error during the transmission.

At the first opportunity window on the CAN bus, the message on the Tx SMB is transmitted according to the CAN protocol rules.

17.4.2.4 Arbitration Start and Stop Conditions

The arbitration process can be triggered in the following situations:

- During Rx and Tx frames from CAN CRC field to end of frame. CAN_CTRL2.TASD may be changed to optimize the arbitration start point.
- During CAN Bus Off state from TX_ERR_CNT = 124 to 128. CAN_CTRL2.TASD may be changed to optimize the arbitration start point.
- During C/S write by CPU in Bus Idle. The first C/S write starts arbitration process and a second C/S write during this same arbitration restarts the process. If other C/S writes are performed, the Tx arbitration process is pending. If there is no arbitration winner after the arbitration process has finished, the TX arbitration machine begins a new arbitration process. If there is a pending arbitration and Bus Idle state starts, an arbitration process is triggered. In this case, the first and second C/S write in Bus Idle do not restart the arbitration process. It is possible that there is not enough time to finish arbitration in Wait For Bus Idle state and the next state is Idle. In this case, the scan is not interrupted, and it is completed during Bus Idle state. During this arbitration, a C/S write does not cause arbitration restart.
- Arbitration winner deactivation during a valid arbitration window.
- Upon exiting Freeze mode (first bit of the Wait For Bus Idle state). If there is a resynchronization during Wait For Bus Idle, the arbitration process is restarted.

Arbitration process stops in the following situations:

- All mailboxes were scanned.
- A Tx active mailbox is found when the lowest buffer feature enabled.
- Arbitration winner inactivation or abort during any arbitration process.
- There was not enough time to finish the Tx arbitration process (for example, when a deactivation was performed near the end of frame). In this case, the arbitration process is pending.
- Error or Overload flag in the bus.
- Low-Power or Freeze mode request in Idle state.

Arbitration is considered pending as described below:

- It was not possible to finish the arbitration process in time.
- C/S write during arbitration if write is performed in a MB whose number is lower than the Tx arbitration pointer.
- Any C/S write if there is no Tx arbitration process in progress.
- Rx Match has just updated an Rx code to Tx code
- Entering Bus Off state.

Yuntu Microelectronics

C/S write during arbitration has the following effect:

- If the C/S write is performed in the arbitration winner, a new process is restarted immediately.
- If the C/S write is performed in an MB whose number is higher than the Tx arbitration pointer, the ongoing arbitration process will scan this MB as normal.

17.4.3 Receive Process

To be able to receive CAN frames into a mailbox, the CPU must prepare the mailbox for reception by executing the following steps:

1. If the mailbox is active (either Tx or Rx), inactivate the mailbox (see ”Mailbox Inactivation”), preferably with a safe inactivation (see ”Transmission Abort Mechanism”).
2. Write the ID word.
3. Write the EMPTY code (0b0100) to the CODE field of the C/S word to activate the mailbox. No setup is required for the EDL, BRS, and ESI bits; they are overwritten by the respective bit fields in the received message.

After the MB is activated, it will be able to receive frames that match the programmed filter.

At the end of a successful reception, the mailbox is updated by the move-in process (see ”Move-in”) as follows:

1. The received Data field (8 bytes at most for Classical CAN message format and up to 64 bytes for CAN FD message format) is stored.
2. The received ID field is stored.
3. The value of the Free Running Timer at the time of the second bit of frame’s Identifier field is written into the mailbox’s TIME STAMP field.
4. The received SRR, IDE, RTR, EDL, BRS, ESI, and DLC fields are stored.
5. The CODE field in the C/S word is updated(see ”CAN message buffer code for RX buffers”) and ”CAN message buffer code for TX buffers”)).
6. A status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit.

The recommended procedure for the CPU to service (read) the frame received in a mailbox is:

1. Read the C/S word of that mailbox.
2. Check if the BUSY bit is deasserted, indicating that the mailbox is locked. Repeat step 1) while it is asserted. See ”Mailbox Lock Mechanism”.
3. Read the contents of the mailbox. After the mailbox is locked, its contents won’t be modified by FlexCAN Move-in processes. See ”Move-in”.
4. Acknowledge the proper flag at the CAN_IFLAG registers.
5. Read the Free Running Timer to unlock the mailbox.

The CPU should poll for frame reception by checking the status flag bit for the specific mailbox in the respective CAN_IFLAGn register and not by polling the CODE field of that mailbox. Polling the CODE field does not work because, once a frame is received and the CPU services the mailbox (by reading the C/S word followed by unlocking the mailbox), the CODE field does not return to EMPTY. It remains FULL, as expained in Table: ”CAN message buffer code for RX buffers”). If the CPU tries to work around this behavior by writing to the C/S word to force an EMPTY code after reading the mailbox without a prior safe inactivation, a newly received frame matching the filter of that mailbox may be lost.

CAUTION: In summary, never do polling by reading directly the C/S word of the mailboxes. Instead, read the CAN_IFLAGn registers.

Yuntu Microelectronics

The received frame's Identifier field is always stored in the matching mailbox. Therefore, the contents of the ID field in a mailbox may change if the match was due to masking.

When CAN_MCR.SRXDIS is asserted, the FlexCAN does not store frames transmitted by itself in any MB, even if it contains a matching Rx mailbox, and no interrupt flag or interrupt signal is generated. Otherwise, when CAN_MCR.SRXDIS is deasserted, the FlexCAN can receive frames transmitted by itself if there is a matching Rx mailbox.

To be able to receive CAN frames through the Legacy Rx FIFO, the CPU must enable and configure the Legacy Rx FIFO during Freeze mode (see "Legacy Rx FIFO"). Upon receiving the Frames Available in Rx FIFO interrupt (see the description of the BUF5I bit in "CAN IFLAG1 Register Interrupt Flags 1 Register (CAN_IFLAG1)"), the CPU should service the received frame using the following procedure:

1. Read the C/S word (optional: needed only if a mask was used for the IDE and RTR bits).
2. Read the ID field (optional: needed only if a mask was used).
3. Read the Data field.
4. Read the CAN_RXFIR register (optional).
5. Clear the Frames Available in Legacy Rx FIFO interrupt by writing 1 to CAN_IFLAG1.BUF5I (mandatory: releases the MB and allows the CPU to read the next Legacy Rx FIFO entry).

When CAN_MCR.DMA is asserted, upon receiving a frame in the Legacy Rx FIFO, CAN_IFLAG1.BUF5I generates a DMA request and does not generate a CPU interrupt (see "Rx FIFO under DMA operation"). The CAN_IMASK1 bits in the Legacy Rx FIFO region are not used.

The DMA controller must service the received frame using the following procedure:

1. Read the C/S word (read 0x80 address, optional).
2. Read the ID field (read 0x84 address, optional).
3. Read all Data Bytes (start read at 0x88 address, optional).
4. Read the last Data Bytes (read 0x8C address is mandatory).

17.4.4 Matching Process

The matching process scans the MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus. If the Legacy Rx FIFO is enabled, the priority of scanning can be selected between mailboxes and FIFO filters. The matching starts from the lowest number MB toward the higher ones. If no match is found within the first structure then the other is scanned subsequently. In the event that the FIFO is full, the matching algorithm always looks for a matching MB outside the FIFO region.

For details about the Legacy Rx FIFO, see "Legacy Rx FIFO".

For details about the Enhanced Rx FIFO, see "Enhanced Rx FIFO".

As the frame is being received, it is stored in a hidden auxiliary MB called Rx Serial Message Buffer (Rx SMB).

The matching process start point depends on the following conditions:

- If the received frame is a remote frame, the start point is the CRC field of the frame.
- If the received frame is a data frame with DLC field equal to zero, the start point is the CRC field of the frame.
- If the received frame is a data frame with DLC field different than zero, the start point is the DATA field of the frame.

If a matching ID is found in the FIFO table or in one of the mailboxes, the contents of the Rx SMB are transferred to the FIFO or to the matched mailbox by the move-in process. If any CAN protocol error is

Yuntu Microelectronics

detected, then no match results are transferred to the FIFO or to the matched mailbox at the end of reception.

The matching process scans all matching elements of both Rx FIFO (if enabled) and the active Rx mailboxes (CODE is EMPTY, FULL, OVERRUN, or RANSWER) in search of a successful comparison with the matching elements of the Rx SMB that is receiving the frame on the CAN bus. The Rx SMB has the same structure as a mailbox. The reception structures (Rx FIFO or mailboxes) associated with the matching elements that had a successful comparison are the matched structures. The matching winner is selected at the end of the scan among those matched structures and depends on conditions described below.

Table 17.88: CAN Matching Architecture

Structure	SMB.RTR	CTRL2.RRS	CTRL2.EACEN	MB.IDE	MB.RTR	MB.ID[1]	MB.CODE
Mailbox	0	–	0	cmp[2]	nocmp[3]	cmp_msk[4]	EMPTY or FULL or OVERRUN
Mailbox	0	–	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	0	–	cmp	no_cmp	cmp	RANSWER
Mailbox	1	1	0	cmp	no_cmp	cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	1	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
FIFO[5]	–	–	–	cmp_msk	cmp_msk	cmp_msk	–

1. For mailbox structure, if SMB[IDE] is asserted, the ID is 29 bits(ID Standard ID Extended). If SMB[IDE] is negated, the ID is only 11 bits(ID Standard). For Rx FIFO structure, the ID depends on CAN_MCR[IDA].
2. cmp: Compares the Rx SMB contents with the MB contents regardless the masks.
3. no_cmp: The Rx SMB contents are not compared with the MB contents.
4. cmp_msk: Compares the Rx SMB contents with MB contents taking into account the masks.
5. SMB[IDE] and SMB[RTR] are not taken into account when CAN_MCR[IDAM] is format C.

NOTE: For details about the matching process for the Enhanced Rx FIFO, see ”Enhanced Rx FIFO Matching Process”.

A reception structure is free-to-receive when any of the following conditions is satisfied:

- The CODE field of the mailbox is EMPTY.
- The CODE field of the mailbox is either FULL or OVERRUN and it has already been serviced (the C/S word was read by the CPU and unlocked as described in ”Mailbox Lock Mechanism”).
- The CODE field of the mailbox is either FULL or OVERRUN and an inactivation is performed. See

Yuntu Microelectronics

"Mailbox Inactivation".

- The Legacy Rx FIFO or Enhanced Rx FIFO is not full.

The scan order for mailboxes and the legacy Rx FIFO is from the matching element with lowest number to the higher ones.

The matching winner search for mailboxes is affected by the CAN_MCR.IRMQ bit. If IRMQ is negated, the matching winner is the first matched mailbox regardless if it is free-toreceive or not. If IRMQ is asserted, the atching winner is selected according to the priority below:

1. The first free-to-receive matched mailbox.
2. The last non free-to-receive matched mailbox.

It is possible to select the priority of scan between mailboxes and the legacy Rx FIFO or Enhanced Rx FIFO by using the CAN_CTRL2.MRP bit.

If the selected priority is Rx FIFO first:

- If the Rx FIFO is a matched structure and is free-to-receive, then the Rx FIFO is the matching winner regardless of the scan for mailboxes.
- Otherwise (the Rx FIFO is not a matched structure or is not free-to-receive), then the matching winner is searched for among mailboxes as described above.

If the selected priority is mailboxes first:

- If a free-to-receive matched mailbox is found, it is the matching winner regardless of the scan for the Rx FIFO.
- If no matched mailbox is found, then the matching winner is searched for in the scan for the Rx FIFO.

If both conditions above are not satisfied and a non-free-to-receive matched mailbox is found, then the matching winner determination is conditioned by the CAN_MCR.IRMQ bit:

- If CAN_MCR.IRMQ is negated, the matching winner is the first matched mailbox.
- If CAN_MCR.IRMQ is asserted, the matching winner is the Rx FIFO if it is a free-toreceive matched structure; otherwise, the matching winner is the last non-free-toreceive matched mailbox.

See below Table for a summary of matching possibilities.

Table 17.89: CAN Matching Possibilities and Resulting Reception Structures

RFEN	IRMQ	MRP	Matched in MB	Matched in FIFO	Reception structure	Description
No FIFO, only MB, match is always MB first:						
0	0	X[1]	None[2]	–[3]	None	Frame lost by no match
0	0	X	Free[4]	–	First MB	
0	1	X	None	–	None	Frame lost by no match
0	1	X	Free	–	First MB	
0	1	X	Not Free	–	Last MB	Overrun
FIFO enabled, no match in FIFO is as if FIFO does not exist:						
1	0	X	None	None[5]	None	Frame lost by no match
1	0	X	Free	None	First MB	

Yuntu Microelectronics

Table 17.89 continued from previous page

RFEN	IRMQ	MRP	Matched in MB	Matched in FIFO	Reception structure	Description
1	1	X	None	None	None	Frame lost by no match
1	1	X	Free	None	First MB	
1	1	X	Not Free	None	Last MB	Overrun
FIFO enabled, Queue disabled:						
1	0	0	X	Not Full[6]	FIFO	
1	0	0	None	Full[7]	None	Frame lost by FIFO full (FIFO Overflow)
1	0	0	Free	Full	First MB	
1	0	0	Not Free	Full	First MB	
1	0	1	None	Not Full	FIFO	
1	0	1	None	Full	None	Frame lost by FIFO full (FIFO Overflow)
1	0	1	Free	X	First MB	
1	0	1	Not Free	X	First MB	Overrun
FIFO enabled, Queue enabled:						
1	1	0	X	Not Full	FIFO	
1	1	0	None	Full	None	Frame lost by FIFO full (FIFO Overflow)
1	1	0	Free	Full	First MB	
1	1	0	Not Free	Full	Last MB	Overrun
1	1	1	None	Not Full	FIFO	
1	1	1	Free	X	First MB	
1	1	1	Not Free	Not Full	FIFO	
1	1	1	Not Free	Full	Last MB	Overrun

1. This is a ”don’t care” condition.
2. Matched in MB ”None” means that the frame has not matched any MB (free-to-receive or non-freetoreceive).
3. This is a forbidden condition.
4. Matched in MB ”Free” means that the frame matched at least one MB free-to-receive regardless of whether it has matched MBs non-free-to-receive.
5. Matched in FIFO ”None” means that the frame has not matched any filter for the Rx FIFO. It is as if the Rx FIFO does not exist (CAN_CTRL2.RFEN = 0).
6. Matched in FIFO ”Not Full” means that the frame has matched a FIFO filter and has empty slots to receive it.
7. Matched in FIFO ”Full” means that the frame has matched a FIFO filter but the FIFO couldn’t store it because the FIFO has no empty slots to receive it.

If a non-safe mailbox inactivation (see ”Mailbox Inactivation”) occurs during the matching process and the mailbox inactivated is the temporary matching winner, then the temporary matching winner is invalidated. The matching elements scan is not stopped nor restarted; it continues normally. The consequence is that the current matching process works as if the matching elements compared before the inactivation did not exist, therefore a message may be lost.

Suppose, for example, that the Rx FIFO is disabled, IRMQ is asserted, there are two MBs with the same ID, and the FlexCAN starts receiving messages with that ID. If the two MBs are the second and the fifth in the array, then when the first message arrives, the matching algorithm finds the first match in MB

Yuntu Microelectronics

number 2. The code of this MB is EMPTY, so the message is stored there. When the second message arrives, the matching algorithm finds MB number 2 again, but it is not “free-to-receive”, so it keeps looking, finds MB number 5 and stores the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching MBs that are “free-to-receive”, so it decides to overwrite the last matched MB, which is number 5. In doing so, it sets the CODE field of the MB to indicate OVERRUN.

The ability to match the same ID in more than one MB can be exploited to implement a reception queue (in addition to the full-featured FIFO) to allow more time for the CPU to service the MBs. By programming more than one MB with the same ID, received messages are queued into the MBs. The CPU can examine the TIME STAMP field of the MBs to determine the order in which the messages arrived.

Matching to a range of IDs is possible by using ID Acceptance Masks. The FlexCAN supports individual masking per MB (see ”Rx Individual Mask Registers (CAN_RXIMRn)”). During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is a “don’t care”. The Individual Mask Registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed when the FlexCAN is in Freeze mode; otherwise, they are blocked by hardware.

The FlexCAN also supports an alternate masking scheme with only four mask registers (CAN_RXFGMASK, CAN_RXMGMASK, CAN_RX14MASK, and CAN_RX15MASK) for backward compatibility with legacy applications. This alternate masking scheme is enabled when CAN_MCR.IRMQ is negated.

17.4.5 Receive Process under Pretended Networking Mode

Pretended Networking mode adds specific Wake-Up functionality in Low-Power modes (Doze and Deepsleep mode). When Pretended Network mode (PN) is enabled by asserting CAN_MCR.PNET_EN, the FlexCAN continues processing Rx CAN messages under Low-Power mode, and is able to detect specific Wake-Up messages by filtering them against ID and payload target values using a pre-selected matching criteria. Wake-Up functionality is not available for messages in CAN FD format. When the FlexCAN is in Pretended Networking mode, CAN FD format messages are ignored.

PN registers are located in the 0x0B00–0x0B7C address range and can be written only in Freeze mode. These registers are used for writing PN configuration (both control and target values) prior to entering Pretended Networking mode, and for reading Wake-Up flags and the received message ID and data when returning back to normal mode after Wake-Up. The CPU must wait for CAN_MCR.LPMACK to be negated before performing any access to FlexCAN PN registers.

PN control registers are described in ”CAN CTRL1 PN Register (CAN_CTRL1_PN)” and ”CAN CTRL2 PN Register (CAN_CTRL2_PN)”. The control bit fields that configure the filtering criteria are:

- PLFS: Payload filtering selection.
- IDFS: ID filtering selection.
- FCS: Filtering combination selection.

PN target values are:

- FLT_IDE: IDE target value used to filter the incoming message by its format (standard or extended).
- FLT_RTR: RTR target value used to filter the incoming message by its type (data or remote frame).
- FLT_DLC_HI and FLT_DLC_LO: Target DLC range used to filter the size of payload part of an incoming message.

Yuntu Microelectronics

- FLT_ID1: ID target value used to filter the incoming message ID (equal to, smaller than or equal, greater than or equal, or the lower limit value in an ID range).
- FLT_ID2: ID target value used as the upper limit in an ID range.
- PL1: Payload target value used to filter the incoming message payload (equal to, smaller than or equal, greater than or equal, or the lower limit value in a payload range).
- PL2: Payload target value used as the upper limit in a payload range.

The IDE, RTR, ID, and payload filters have their respective masks. These masks determine which bits are taken into account in equality comparisons (1 in certain mask positions) and which ones are don't care (0 in other mask positions). ID and payload masks are used only for exact ID and/or exact payload comparisons.

The ID of Rx incoming messages can be filtered based on the following criteria:

- A match with the exact ID value by detecting the equality between the ID field of the incoming message and the content of target CAN_FLT_ID1 register. The ID mask is used.
- A match with the maximum range of ID; in other words, any message with ID value smaller than or equal to the content of target CAN_FLT_ID1 register is accepted. The ID mask is not used.
- A match with the minimum range of ID; in other words, any message with ID value greater than or equal to the content of target CAN_FLT_ID1 register is accepted. The ID mask is not used.
- A match inside a range of IDs; in other words, any message with an ID value that is greater than or equal to the content of target CAN_FLT_ID1 register and smaller than or equal to the content of target CAN_FLT_ID2_IDMASK register is accepted. The ID mask is not used.

The above criteria for ID filtering must be coherent with FLT_IDE and FLT_RTR target values in CAN_FLT_ID1 register. Only Rx frames that match the respective IDE and RTR bits to the contents of FLT_IDE and FLT_RTR bit fields are compared. When range of IDs is selected (CAN_CTRL1_PN.IDFS = 11), both FLT_ID1 and FLT_ID2 are referred to the same FLT_IDE and FLT_RTR bits in the CAN_FLT_ID1 register.

The ID mask is applied only to the exact ID comparison filtering option (CAN_CTRL1_PN.IDFS = 00) to determine which bits are taken into account in the comparison. For the exact match option, the mask can select any bit within the ID field. For maximum range, minimum range, and inside range comparisons, the ID mask is not considered.

The IDE and RTR masks are applied in both exact and range ID comparison filtering options to determine which bits are taken into account in comparison.

Similarly to the ID criteria, 64-bit data or payloads (PL) of Rx incoming messages can be filtered based on the following criteria:

- A match with the exact payload value by detecting the equality between the payload field of the incoming message and the content of PL1 register. The payload mask is used.
- A match with the maximum range of payload; in other words, any message with payload value smaller than or equal to the content of PL1 register is accepted. The payload mask is not used.
- A match with the minimum range of payload; in other words,. any message with payload value greater than or equal to the content of PL1 register is accepted. The payload mask is not used.
- A match inside a range of payloads; in other words, any message with a payload value that is greater than or equal to the content of PL1 register and smaller than or equal to the content of PL2 register is accepted. The payload mask is not used.

The above criteria for payload filtering must be coherent with FLT_DLC upper and lower limit values in CAN_FLT_DLC register. The payload of a Rx incoming message is filtered in accordance to the selected criteria only if the DLC value of the Rx incoming message is inside a DLC range:

Yuntu Microelectronics

- Greater than or equal to the FLT_DLC_LO (lower limit) and
- Lower than or equal to the FLT_DLC_HI (upper limit)

Conversely, a DLC value out of the specified range results in mismatch. By making FLT_DLC_LO = FLT_DLC_HI, only payloads of the specified quantity of bytes will be filtered. DLC is not maskable.

When the inside range of payloads option is selected (CAN_CTRL1_PN.PLFS = 11), both PL1 and PL2 are considered with the 8-byte data length. All the data bytes excluded by the DLC of the received message are considered with value zero.

Payload mask is only used in the exact match option (CAN_CTRL1_PN.PLFS = 00) to select which bits or bytes in the 8-byte data field of both Rx incoming message and the contents of PL1 register are selected for matching. Mask length must be in accordance with the expected range of DLC values. For maximum range, minimum range, and inside range comparisons, the payload mask is not considered.

When a remote frame is received by the FlexCAN and CAN_CTRL1_PN.FCS is configured to select the payload comparison, the payload filtering is not considered and the comparison results in a mismatch.

Rx incoming messages can also be filtered based upon the quantity and rate of message reception, specifically:

- Several messages that match the filtering criteria for ID or payload for a predefined number of times. This number can be configured in the 1 to 255 range. See ”CAN CTRL1 PN Register (CAN_CTRL1_PN)”.
- No message matching the filtering criteria for ID or payload up to a timeout trigger. That is, non-reception of a matching message for a defined quantity of time. ”CAN CTRL2 PN Register (CAN_CTRL2_PN)”.

The FlexCAN can generate a Wake-Up timeout event from an internal timer with associated comparator circuitry capable of generating a timeout flag when the counting reaches the pre-defined timeout value, as specified in CAN_CTRL2_PN.MATCHTO.

The above filtering criteria can be used together as follows:

- Message ID filtering only.
- Message ID filtering and Payload filtering.
- Message ID filtering only occurring N times.
- Message ID filtering and Payload filtering occurring N times.

The timeout counter runs concurrently with the reception filtering process. Both engines, timeout counter and message filtering, are independent. If an incoming message matches the selected filter criteria, the timeout counter keeps counting until the CPU wakes up.

Conversely, if the timeout counter reaches the target value, then the message filtering process continue to filter incoming messages until the CPU wakes up. The CAN_WU_MTC.MCOUNTER field reports the number of matched messages that occurred under Pretended Networking mode up to the moment the CPU wakes up.

Under Pretended Networking mode, a Wake-Up event that occurs sets the respective Wake-Up flag (see ”Pretended Networking Wake Up Match Register (CAN_WU_MTC)” :

- In case of a successful match in accordance with the selected filtering criteria, CAN_WU_MTC.WUMF.
- In case of a timeout trigger, CAN_WU_MTC.WTOF.

Either of these flags generates an interrupt to the CPU if the respective mask bit is asserted (WUMF_MSK or WTOF_MSK in the CAN_CTRL1_PN register).

Yuntu Microelectronics

There are four Wake-Up Message Buffers (WMBs) used to store incoming messages in Pretended Networking mode. Up to four messages can be stored (see ”Wake Up Message Buffer Register for ID (CAN_WMBn_ID)”).

When the CAN_CTRL1_PN.NMATCH value is 1, only one message is received if the filtering criteria are matched, and this message is stored in CAN_WMB0.

If the NMATCH value is between 2 and 4, CAN_WMB1, CAN_WMB2, and CAN_WMB3 are used to store the second, third, and fourth matching messages, respectively.

If NMATCH is greater than 4, the last four matching messages are stored in the WMBs. The WMB index indicates the arrival order. The last message is stored in CAN_WMB3.

Only the valid data bytes of the incoming match message is stored in the data field of WMBs. The non-valid data bytes are read as zero. In case the of DLC = 0 and RTR = 1, the data field is filled with zeros. In any of the above cases, the Wake-Up interrupt is generated

when the filtering criteria is completed and CAN_CTRL1_PN.WUMF_MSK is asserted. When a non-match Wake-Up event occurs (timeout or external) and CAN_WU_MTC.MCOUNTER is equal to or greater than 4, the message stored in WMB0 does not have a valid content. The CAN_WMB0 is used as the buffer for the current message on the CAN bus. Messages received during Pretended Networking mode do not have timestamps and respective field in the WMB structure must be ignored.

Under Low-Power mode (Doze or Deepsleep), all processes are shut down except for the PN functionality in the PE submodule, which is kept clocked by the Oscillator clock (see ”Clock Domains and Restrictions”). The FlexCAN continues to receive Rx incoming messages, but only compares them to the predefined target values and according to the selected filtering criteria. The matching, arbitration, move-in, and move-out processes, normally available in Normal mode, are not performed under Pretended Networking mode.

Under Pretended Networking, the FlexCAN reacts to messages on the CAN bus in the same manner as in Normal mode (generates acknowledge bits, detects and counts errors, and so on).

17.4.6 Move Process

There are two types of move process: move-in and move-out.

17.4.6.1 Move-in

The move-in process is the copy of a message received by an Rx SMB to an Rx mailbox or FIFO that has matched it. If the move destination is the Legacy Rx FIFO, attributes of the message are also copied to the CAN_RXFIR FIFO. Each Rx SMB has its own move-in process, but only one is performed at a given time as described below. The move-in starts only when the message held by the Rx SMB has a corresponding matching winner (see ”Matching Process”) and all of the following conditions are true:

- The CAN bus has reached or let past either:
  - The second bit of the Intermission field next to the frame that carried the message that is in the Rx SMB, or
  - The first bit of an overload frame next to the frame that carried the message that is in the Rx SMB
- There is no ongoing matching process.
- The destination mailbox is not locked by the CPU.

Yuntu Microelectronics

- There is no ongoing move-in process from another Rx SMB. If more than one move-in processes are to be started at the same time, both are performed and the newest substitutes for the oldest.

The term pending move-in is used throughout the documentation and stands for a move-tobe that still does not satisfy all of the aforementioned conditions.

The move-in is canceled and the Rx SMB is able to receive another message if any of the following conditions is satisfied:

- The destination mailbox is inactivated after the CAN bus has reached the first bit of Intermission field next to the frame that carried the message and its matching process has finished.
- There is a previous pending move-in to the same destination mailbox.
- The Rx SMB is receiving a frame transmitted by the FlexCAN itself and self-reception is disabled (CAN_MCR.SRXDIS bit is asserted)
- Any CAN protocol error is detected.

Note that the pending move-in is not canceled if the FlexCAN enters Freeze or Low-Power mode. The pending move-in stays on hold, waiting for exit from Freeze or Low-Power mode and to be unlocked. If an MB is unlocked during Freeze mode, the move-in happens immediately.

The move-in process is the execution by the FlexCAN of the following steps:

1. Push IDHIT into the RXFIR FIFO if the message is destined to the Legacy Rx FIFO.
2. Read all data words from the Rx SMB according to the selected payload size for the Rx storage element.
3. Write all data words to the Rx mailbox according to the selected payload size for the Rx storage element. If the data size of the storage element is smaller than the original payload size described in the message's DLC field, the payload is truncated and the high order bytes that do not fit the destination size are lost.
4. Read the C/S and ID words from the Rx SMB.
5. Write C/S and ID words to the Rx mailbox, and update the CODE field.

The move-in process is not atomic, in such a way that it is immediately canceled by the inactivation of the destination mailbox (see "Mailbox Inactivation"). In this case, the mailbox may be left partially updated, thus incoherent. The exception is if the move-in destination is a Legacy or Enhanced Rx FIFO message buffer, in which case the process cannot be canceled.

The BUSY bit (least significant bit of the CODE field) of the destination message buffer is asserted while the move-in is being performed to alert the CPU that the message buffer content is temporarily incoherent.

17.4.6.2 Move-out

The move-out process is the copy of the content from a Tx mailbox to the Tx SMB when a message for transmission is available (see "Arbitration Process"). The moveout occurs in the following conditions:

- The first bit of Intermission field.
- During Bus Off state when TX Error Counter is in the 124 to 128 range.
- During Bus Idle state.
- During Wait For Bus Idle state.

The move-out process is not atomic. Only the CPU has priority to access the memory concurrently out of Bus Idle state. In Bus Idle, the move-out has the lowest priority to the concurrent memory accesses.

Yuntu Microelectronics

17.4.7 Data Coherence

In order to maintain data coherency and FlexCAN proper operation, the CPU must obey the rules described in ”Transmit Process” and ”Receive Process”.

17.4.7.1 Transmission Abort Mechanism

The abort mechanism provides a safe way to request the abortion of a pending transmission. A feedback mechanism is provided to inform the CPU if the transmission was aborted or if the frame could not be aborted and was transmitted instead.

Two primary conditions must be fulfilled in order to abort a transmission:

- CAN_MCR.AEN must be asserted.
- The first CPU action must be the writing of abort code (1001) into the CODE field of the C/S word.

Active MBs configured for transmission must be aborted first before they can be updated. If the abort code is written to a mailbox that is currently being transmitted or to a mailbox that was already loaded into the Tx SMB for transmission, the write operation is blocked and the transmission is not disturbed. However, the abort request is captured and kept pending until one of the following conditions is satisfied:

- The FlexCAN loses the bus arbitration.
- There is an error during the transmission.
- The FlexCAN is put into Freeze mode.
- The FlexCAN enters the Bus Off state.
- There is an overload frame.

If none of the conditions above are reached, the MB is transmitted correctly, the interrupt flag is set in the respective CAN_IFLAGn register, and an interrupt to the CPU is generated (if enabled). The abort request is automatically cleared when the interrupt flag is set. On the other hand, if one of the above conditions is reached, the frame is not transmitted; therefore, the abort code is written into the CODE field, the interrupt flag is set in the respective CAN_IFLAGn register, and an interrupt is (optionally) generated to the CPU.

If the CPU writes the abort code before the transmission begins internally, then the write operation is not blocked; therefore, the MB is updated and the interrupt flag is set. In this way, the CPU only needs to read the abort code to make sure the active MB was safely inactivated. Although the AEN bit is asserted and the CPU wrote the abort code, in this case the MB is inactivated and not aborted, because the transmission did not start yet. One mailbox is only aborted when the abort request is captured and kept pending until one of the previous conditions is satisfied.

The abort procedure can be summarized as follows:

- CPU checks the corresponding IFLAG and clears it, if asserted.
- CPU writes 1001 into the CODE field of the C/S word.
- CPU waits for the corresponding IFLAG indicating that the frame was either transmitted or aborted.
- CPU reads the CODE field to check if the frame was either transmitted (CODE = 1000) or aborted (CODE = 1001).
- It is necessary to clear the corresponding IFLAG in order to allow the MB to be reconfigured.
- It is necessary to reconfigure the EDL, BRS, and ESI fields of the aborted MB before transmitting it again.

Yuntu Microelectronics

17.4.7.2 Mailbox Inactivation

Inactivation is a mechanism provided to protect the mailbox against updates by the FlexCAN internal processes, thus allowing the CPU to rely on mailbox data coherence after having updated it, even in Normal mode.

Inactivation of transmission mailboxes must be performed only when MCR.AEN bit is deasserted.

If a mailbox is inactivated, it participates in neither the arbitration process nor the matching process until it is reactivated. See ”Transmit Process” and ”Receive Process” for more detailed instructions on how to inactivate and reactivate a mailbox.

To inactivate a mailbox, the CPU must update its CODE field to INACTIVE (either 0000 or 1000).

Because the CPU is not able to synchronize the CODE field update with the FlexCAN internal processes, an inactivation can have the following consequences:

- A frame on the bus that matches the filtering of the inactivated Rx mailbox may be lost without notice, even if there are other mailboxes with the same filter.
- A frame containing the message within the inactivated Tx mailbox may be transmitted without setting the respective IFLAG.

To perform a safe inactivation and avoid the above consequences for Tx mailboxes, the CPU must use the Transmission Abort mechanism (see ”Transmission Abort Mechanism”).

The inactivation automatically unlocks the mailbox (see ”Mailbox Lock Mechanism”).

NOTE: Message buffers that are part of the Legacy Rx FIFO or Enhanced Rx FIFO cannot be inactivated. There is no write protection on the Legacy Rx FIFO region by FlexCAN. The CPU must maintain data coherency in the Legacy Rx FIFO region when RFEN is asserted.

17.4.7.3 Mailbox Lock Mechanism

Other than mailbox inactivation, the FlexCAN has another data coherence mechanism for the receive process. When the CPU reads the C/S word of an Rx MB with codes FULL or OVERRUN, the FlexCAN assumes that the CPU wants to read the whole MB in an atomic operation and therefore it sets an internal lock flag for that MB. The lock is released when the CPU reads the Free Running Timer (global unlock operation), or when it reads the C/S word of another MB regardless of its code. A CPU write into the C/S word also unlocks the MB, but this procedure is not recommended for normal unlock use because it cancels a pending move and potentially may lose a received message. The MB locking prevents a new frame from being written into the MB while the CPU is reading it.

NOTE: The locking mechanism applies only to Rx MBs that are not part of the Legacy Rx FIFO and have a code other than INACTIVE (0000) or EMPTY (0100). Tx MBs cannot be locked.

NOTE: In previous FlexCAN versions, reading the C/S word locked the MB even if it was EMPTY. This behavior is maintained when the IRMQ bit is negated.

Suppose, for example, that the Legacy Rx FIFO is disabled and the second and the fifth MBs of the array are programmed with the same ID, and the FlexCAN has already received and stored messages into these two MBs. Suppose now that the CPU decides to read MB number 5 and at the same time another message with the same ID is arriving. When the CPU reads the C/S word of MB number 5, this MB is locked. The new message arrives and the matching algorithm finds out that there are no free-to-receive MBs, so it decides to override MB number 5. However, this MB is locked, so the new message cannot be written there. It will remain in the Rx SMB waiting for the MB to be unlocked, and only then will be written to the MB.

Yuntu Microelectronics

If the MB is not unlocked in time and yet another new message with the same ID arrives, then the new message overwrites the message in the Rx SMB and there is no indication of lost messages either in the CODE field of the MB or in the Error and Status Register.

While the message is being moved in from the Rx SMB to the MB, the BUSY bit on the CODE field is asserted. If the CPU reads the C/S word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is negated.

NOTE: If the BUSY bit is asserted or if the MB is empty, then reading the C/S word does not lock the MB.

Inactivation takes precedence over locking. If the CPU inactivates a locked Rx MB, then its lock status is negated and the MB is marked as invalid for the current matching round. Any pending message on the Rx SMB will not be transferred to the MB. An MB is unlocked when the CPU reads the Free Running Timer Register, or the C/S word of another MB.

Lock and unlock mechanisms have the same functionality in both Normal and Freeze modes.

An unlock during Normal or Freeze mode results in the move-in of the pending message. However, the move-in is postponed if an unlock occurs during a Low-Power mode (see ”Modes of operation”), and it takes place only when the module returns to Normal or Freeze mode.

17.4.8 Enhanced Rx FIFO

As an alternative to the 6-message Legacy Rx FIFO, the FlexCAN supports an Enhanced Rx FIFO, which can store up to 32 CAN FD messages. The region 0x2000–0x204F contains the output of the Enhanced Rx FIFO, which should be read by the CPU. The Enhanced Rx FIFO is enabled by setting CAN_EFRCR.ERFEN.

The Legacy Rx FIFO and Enhanced Rx FIFO cannot be enabled at the same time.

The Enhanced Rx FIFO has a watermark, which is configured by setting CAN_EFRCR.ERFWM. If CAN_EFRCR.ERFWM= 1, the CPU is notified only if a minimum number of messages is stored in the FIFO. When the number of stored messages is greater than the value in CAN_EFRCR.ERFWM, CAN_ERFSR.ERFWMI is set by the hardware. An interrupt can be triggered by setting CAN_ERFIER.ERFWMIIE. A DMA transfer can be triggered by setting CAN_MCR.DMA.

For the Enhanced Rx FIFO to receive, the CPU must execute the following configuration procedure:

1. Enter Freeze mode.
2. Enable the Enhanced Rx FIFO (if it is not already enabled) by setting ERFCR.ERFEN.

   NOTE: CAN_MCR.RFEN must be 0 when enabling the Enhanced Rx FIFO.
3. Write 1 to CAN_ERFSR.ERFCLR to reset Enhanced Rx FIFO engine.
4. Clear CAN_EFRSR.ERFUFW, CAN_EFRSR.ERFOVF, CAN_EFRSR.ERFWMI, and CAN_EFRSR.ERFDA, if they are set.
5. Write to CAN_EFRCR.NFE to configure the total number of Enhanced Rx FIFO filter elements to be used in Enhanced Rx FIFO reception.
6. Write to CAN_ERFCR.NEXIF to configure the number of extended ID and standard ID filter elements to be used in Enhanced Rx FIFO reception. CAN_ERFCR.NEXIF must be less than or equal to EFRCR.NFE + 1.
7. Configure the Enhanced Rx FIFO watermark by writing CAN_ERFCR.ERFWM.
8. If interrupts are to be used, set the interrupt enables in the ERIER register.

Yuntu Microelectronics

9. If DMA is to be used, set CAN_MCR.DMA to enable DMA operation and write CAN_ERFCR.DMALW to configure the number of words to transfer for each Enhanced Rx FIFO data element.

10. Configure the filter elements by writing to the CAN_ERFFELn registers.

    NOTE: The CAN_ERFFELn registers are in RAM and need to be explicitly initialized before any reception.

11. Exit Freeze mode.

The above procedure must be done if the CPU needs to change any of the configuration of the Enhanced Rx FIFO.

17.4.8.1 Enhanced Rx FIFO Filter Elements

There are two types of Enhanced Rx FIFO filter elements that can be stored in the CAN_ERFFELn registers: extended ID filter element and standard ID filter element. Each extended ID filter element is stored in two CAN_ERFFELn registers; each standard-ID filter element is stored in one CAN_ERFFELn register. The total number of Enhanced RX FIFO filter elements is defined by CAN_ERFCR.NFE.

The filter memory space can be split into two regions: one for extended ID filter elements and one for standard ID filter elements, as controlled by CAN_ERFCR.NEXIF. Figure "Number of enhanced RX FIFO filter elements" shows how the Enhanced Rx FIFO filter elements are defined for the following example configuration:

- Configuration parameter ERX_FIFO_FLT_ELEM = 128
- CAN_ERFCR.NEXIF = 3 (There are 3 extended ID filter elements, each occupying two CAN_ERFFELn registers.)
- CAN_ERFCR.NFE = 63 (There are 2 x (63 - 3 + 1) = 122 standard ID filter elements, each occupying one CAN_ERFFELn register.)

Yuntu Microelectronics

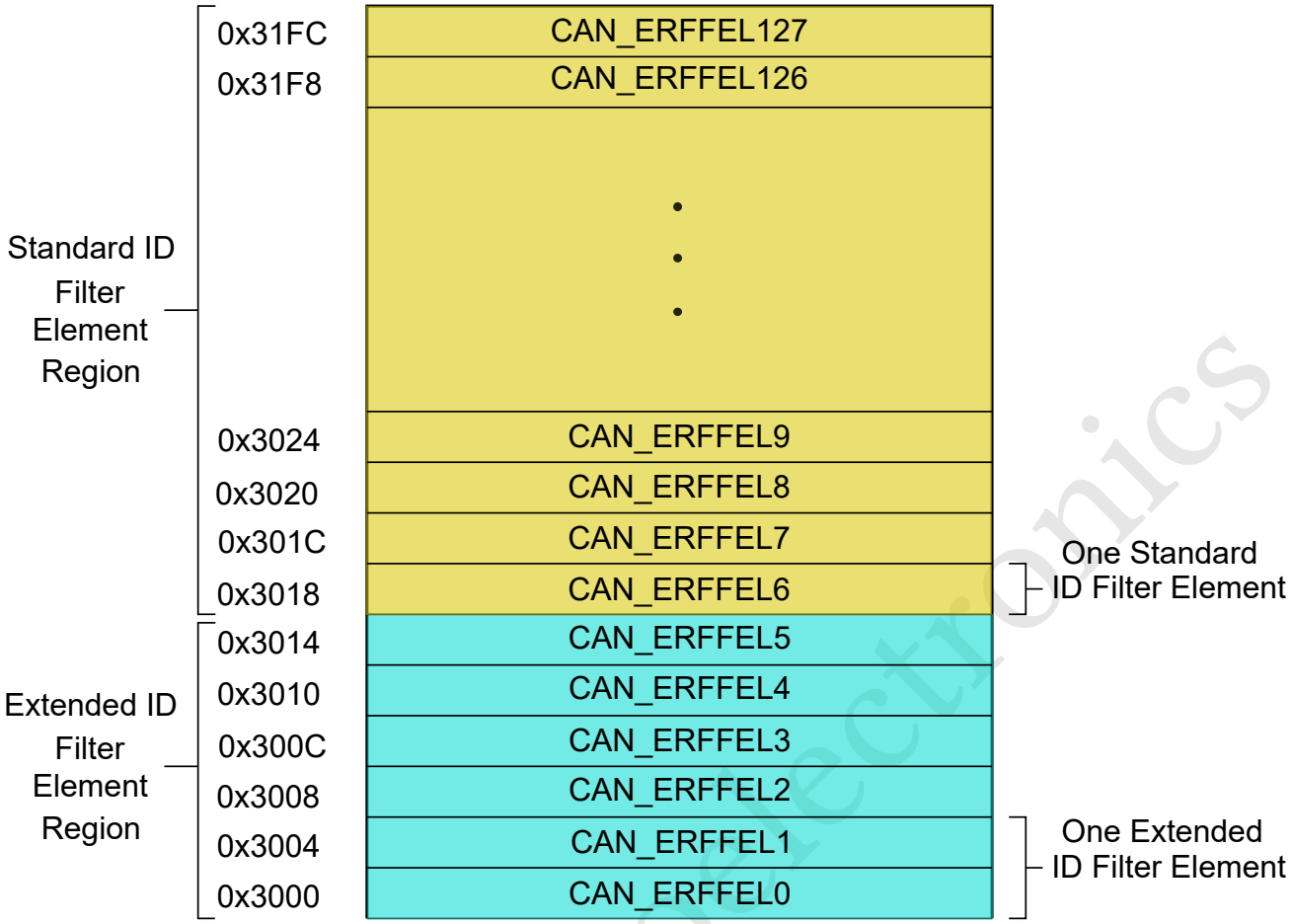


Figure 17.6: Number of Enhanced RX FIFO Filter Elements

17.4.8.2 Enhanced Rx FIFO Matching Process

When CAN_ERFCR.ERFEN = 1, the FlexCAN scans the CAN_ERFFELn memory region. If at least one filter element satisfies the matching criteria, the CAN message content is transferred to the Enhanced RX FIFO memory.

NOTE: If multiple filters match the incoming message ID, then the first matching filter found by the matching process is indicated in ID HIT.

Each CAN_ERFFELn register can store one standard filter element. The matching criteria are defined by CAN_ERFFELn.FSCH as follows:

- If FSCH = 00, the filter scheme is based on mask and filter. A CAN message matches a standard ID filter element only if the following criteria are met:
  - CAN message is base frame format (IDE = 0)
  - (ID[n] = STD ID Filter [n]) or (STD ID Mask [n] =0) for each bit n from 0 to 10
  - (RTR = RTR Filter) or (RTR Mask = 0)

  RTR and ID are the Remote Transmit Request bit and the ID from a CAN message. If FSCH = 00, the filters and masks are defined as shown in below Figure.

Yuntu Microelectronics

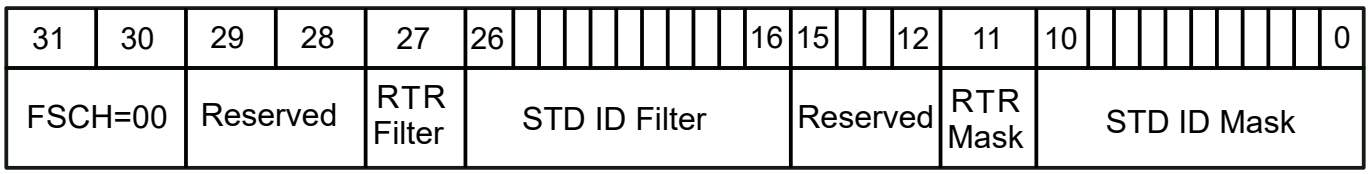


Figure 17.7: Standard ID Filter Element with Filter Mask Scheme FSCH 00

- If FSCH = 01, the filter scheme is based on range. A CAN message matches a standard ID filter element only if the following criteria are met:
  - CAN message is base frame format (IDE = 0)
  - ID ≥ STD ID Filter1
  - ID ≤ STD ID Filter2
  - (RTR = RTR Filter) or (RTR Mask = 0)

  RTR and ID are the Remote Transmit Request bit and ID from a CAN message. If FSCH = 01, the filters and masks are defined as shown in below Figure.

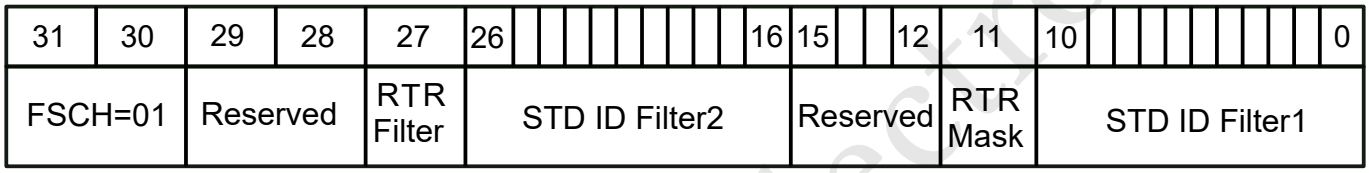


Figure 17.8: Standard ID Filter Element with Range Scheme FSCH 01

- If FSCH = 10, the filter scheme is based on two filters without masks. A CAN message matches a standard ID filter element only if the following criteria are met:
  - CAN message is base frame format (IDE = 0)
  - (ID[n] = STD ID Filter1[n]) or (ID[n] = STD ID Filter2[n]) for each bit n from 0 to 10
  - (RTR = RTR Filter1) or (RTR = RTR Filter2)

  RTR and ID are the Remote Transmit Request bit and ID from a CAN message. If FSCH = 10, the filters and masks are defined as shown in below Figure.

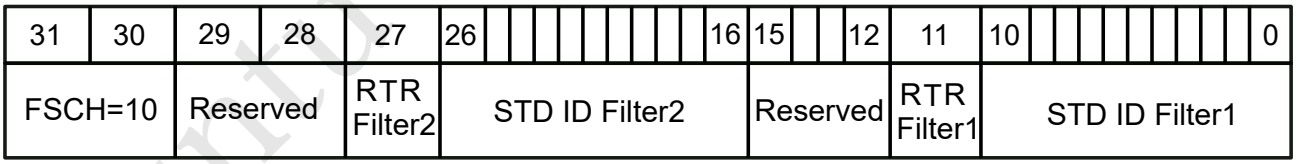


Figure 17.9: Standard ID Filter Element with Two Filter Scheme FSCH 10

Each pair of CAN_ERFFELn registers can store one extended filter element. The matching criteria are defined by CAN_ERFFELn.FSCH as follows:

- If FSCH = 00, the filter scheme is based on mask and filter. A CAN message matches an extended ID filter element only if the following criteria are met:
  - CAN message is extended frame format (IDE = 1)
  - (ID[n] = EXT ID Filter [n]) or (EXT ID Mask [n] =0) for each bit n from 0 to 28
  - (RTR = RTR Filter) or (RTR Mask = 0)

  If FSCH = 00, the filters and masks are defined as shown in below Figure.

Yuntu Microelectronics

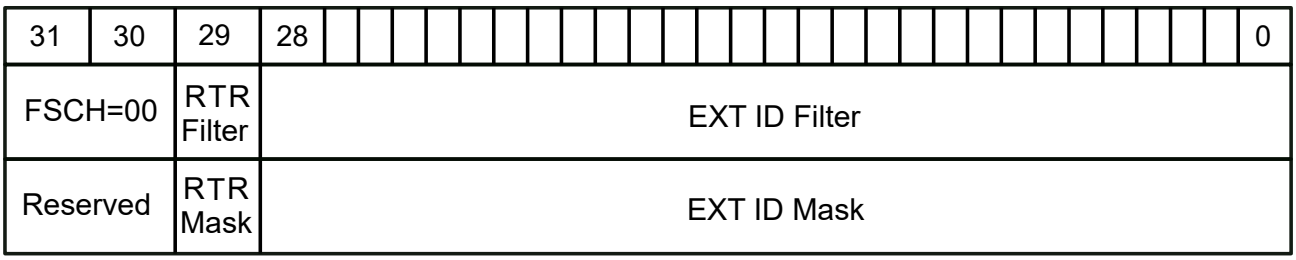


Figure 17.10: Extended ID Filter Element with Filter Mask Scheme FSCH 00

- If FSCH = 01, the filter scheme is based on range. A CAN message matches an extended ID filter element only if the following criteria are met:
  - CAN message is extended frame format (IDE = 1)
  - ID ≥ STD ID Filter1
  - ID ≤ STD ID Filter2
  - (RTR = RTR Filter) or (RTR Mask = 0)

  If FSCH = 01, the filters and masks are defined as shown in below Figure.

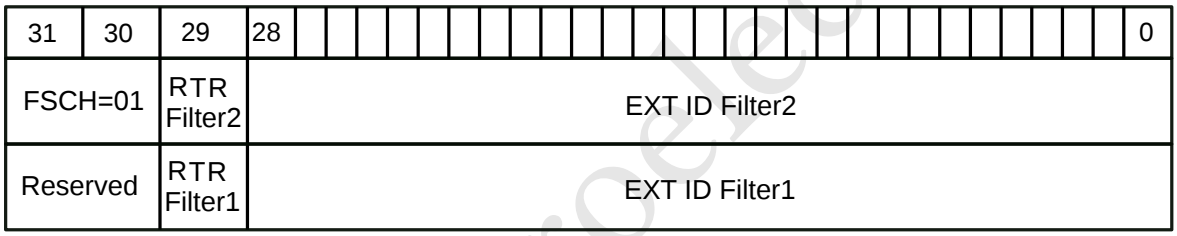


Figure 17.11: Extended ID Filter Element with Filter Mask Scheme FSCH 01

- If FSCH = 10, the filter scheme is based on two filters without masks. A CAN message matches an extended ID filter element only if the following criteria are met:
  - CAN message is extended frame format (IDE = 1)
  - (ID[n] = EXT ID Filter1[n]) or (ID[n] = EXT ID Filter2[n]) for each bit n from 0 to 28
  - (RTR = RTR Filter1) or (RTR = RTR Filter2)

  If FSCH = 10, the filters and masks are defined as shown in below Figure.

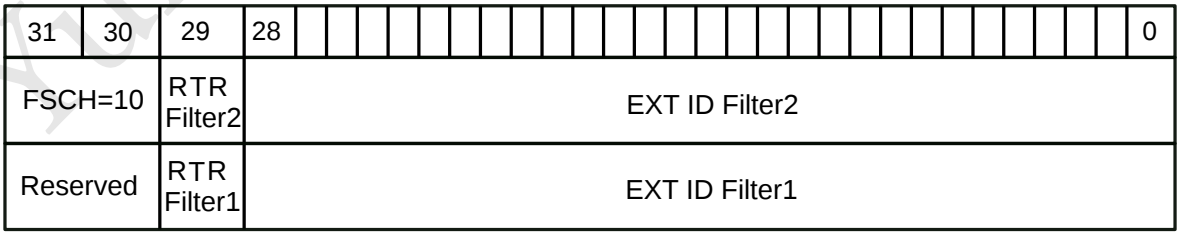


Figure 17.12: Extended ID Filter Element with Two Filter Scheme FSCH 10

17.4.8.3 Enhanced Rx FIFO under DMA Operation

The DMA feature is enabled by asserting both CAN_ERFCR.ERFEN and CAN_MCR.DMA.

Yuntu Microelectronics

The DMA controller can read the received message by reading a message buffer structure at the Enhanced Rx FIFO output port at the address range defined in Enhanced Rx FIFO structure.

The CPU should not access the Enhanced Rx FIFO output port address range during DMA operation, so that the FIFO engine operates properly. Before setting CAN_MCR.DMA, the CPU must service Enhanced Rx FIFO status bits. Otherwise, these bits may show that the FIFO has data to be serviced, and mistakenly generate a DMA request.

Before clearing CAN_MCR.DMA, the CPU must first clear the ERFUFW, ERFOVF, ERMWMI, and ERFDA fields in the CAN_ERFSR register, then clear the Enhanced Rx FIFO engine by writing 1 to CAN_ERFSR.ERFCLR.

CAN_ERFSR.ERFDA is set by the hardware when there is at least one frame available to be read from the Enhanced Rx FIFO, and CAN_ERFSR.ERFWMI is set by the hardware when the number of frames stored in the FIFO is greater than the watermark defined in CAN_ERFCR.ERFWM. A DMA request is generated when CAN_ERFSR.ERFWMI is set.

Upon receiving the request, the DMA controller.can read the message in the Enhanced Rx FIFO output. The DMA controller must perform the number of read loops in the FIFO output defined by CAN_ERFCR.ERFWM. Also, each message reading process must end at the address defined in CAN_ERFCR.DMALW.

The following rules must be followed for Enhanced Rx FIFO DMA operation:

- Because a DMA transfer cannot be dynamically changed, CAN_ERFCR.DMALW should be programmed so that the Enhanced Rx FIFO element can store the largest CAN message present on the CAN bus.
- Data bytes are valid according to the DLC field. See ”CAN data bytes validity”.

Each time one message is read from Enhanced Rx FIFO by the DMA controller, CAN_ERFSR.ERFDA is cleared by the FlexCAN and set again if there is at least one message stored in the FIFO. CAN_ERFSR.ERFWMI is cleared by the hardware when the messages are read in the FIFO output, and set again if the number of messages in the FIFO is greater than the value defined by CAN_ERFCR.ERFWM.

17.4.8.4 Enhanced Rx FIFO Clear Operation

When CAN_ERFCR.ERFEN is set, the CPU can clear the Enhanced Rx FIFO by writing 1 to CAN_ERFSR.ERFCLR. The clear operation resets the internal FIFO pointers, although the FIFO content stored in RAM is not changed. This operation can only be performed in Freeze mode and is blocked by hardware in other modes. The clear operation does not clear the ERFUFW, ERFOVF, ERFDA, and ERFWMI fields in CAN_EFRSR. Consequently, the CPU must service all these fields before executing the clear FIFO task.

17.4.9 Legacy Rx FIFO

The Legacy Rx FIFO is receive-only and is enabled by asserting the CAN_MCR.RFEN bit.

The reset value of this bit is zero to maintain software backward compatibility with previous versions of the FlexCAN module that did not have the Legacy Rx FIFO feature.

CAUTION: The Legacy Rx FIFO must not be enabled when the CAN FD feature is enabled.

The Legacy Rx FIFO is 6 messages deep. The memory region occupied by the Legacy Rx FIFO structure (both message buffers and Legacy Rx FIFO engine) is described in ”Rx FIFO Structure”. The CPU can

Yuntu Microelectronics

read the received messages sequentially, in the order they were received, by repeatedly reading a message buffer structure at the output of the Legacy Rx FIFO.

CAN_IFLAG1.BUF5I (Frames available in Legacy Rx FIFO) is asserted when there is at least one frame available to be read from the Legacy Rx FIFO. An interrupt is generated if it is enabled by the corresponding mask bit. Upon receiving the interrupt, the CPU can read the message (accessing the output of the Legacy Rx FIFO as a message buffer) and the CAN_RXFIR register, then clear the interrupt. If there are more messages in the Legacy Rx FIFO, the act of clearing the interrupt updates the output of the FIFO with the next message and updates CAN_RXFIR with the attributes of that message, reissuing the interrupt to the CPU. Otherwise, the flag remains negated. The output of the Legacy Rx FIFO is only valid while CAN_IFLAG1.BUF5I is asserted.

CAN_IFLAG1.BUF6I (Legacy Rx FIFO Warning) is asserted when the number of unread messages in the Legacy Rx FIFO is increased to 5 from 4 due to the reception of a new message, meaning that the Legacy Rx FIFO is almost full. The flag remains asserted until the CPU clears it.

CAN_IFLAG1.BUF7I (Legacy Rx FIFO Overflow) is asserted when an incoming message was lost because the Legacy Rx FIFO is full. CAN_IFLAG1.BUF7I is not asserted when the Legacy Rx FIFO is full and the message was captured by a mailbox. BUF7I remains asserted until the CPU clears it.

Clearing one of those three flags does not affect the state of the other two.

An interrupt is generated if an IFLAG bit is asserted and the corresponding mask bit is asserted.

A powerful filtering scheme is provided to accept only frames intended for the target application, reducing the interrupt servicing workload. The filtering criteria is specified by programming a table of up to 128 32-bit registers, according to the CAN_CTRL2.RFFN setting, that can be configured to one of the following formats:

- Format A: 128 IDAFs (extended or standard IDs including IDE and RTR)
- Format B: 256 IDAFs (standard IDs or extended 14-bit ID slices including IDE and RTR)
- Format C: 512 IDAFs (standard or extended 8-bit ID slices)

NOTE: A chosen format is applied to all entries of the filter table. It is not possible to mix formats within the table.

See also "Legacy Rx FIFO Structure".

Every frame available in the Legacy Rx FIFO has a corresponding IDHIT (Identifier Acceptance Filter Hit Indicator) that can read in the IDHIT field from C/S word, as shown in the Legacy Rx FIFO structure description. Another way the CPU can obtain this information is by accessing the CAN_RXFIR register. The CAN_RXFIR.IDHIT field refers to the message at the output of the Legacy Rx FIFO and is valid while the CAN_IFLAG1.BUF5I flag is asserted. The CAN_RXFIR register must be read only before clearing the flag, which guarantees that the information refers to the correct frame within the Legacy Rx FIFO.

Up to 32 elements of the filter table are individually affected by the individual Mask Registers (CAN_RXIMRn), according to the setting of CAN_CTRL2.RFFN, allowing very powerful filtering criteria to be defined. If CAN_MCR.IRMQ is negated, the Legacy Rx FIFO filter table is affected by CAN_RXFGMASK.

17.4.9.1 Rx FIFO under DMA Operation

The Legacy Rx FIFO can support DMA. The Legacy Rx FIFO with DMA feature is enabled by asserting both CAN_MCR.RFEN and CAN_MCR.DMA. The reset value of CAN_MCR.DMA is zero to maintain backward compatibility with previous versions of the FlexCAN that did not have the DMA feature.

Yuntu Microelectronics

The DMA controller can read the received message by reading a message buffer structure at the Legacy Rx FIFO output port at the 0x80–0x8C address range.

When CAN_MCR.DMA is asserted, the CPU must not access the Legacy Rx FIFO output port address range. Before asserting CAN_MCR.DMA, the CPU must service the IFLAGs asserted in the Legacy Rx FIFO region. Otherwise, these IFLAGs may show that the Legacy Rx FIFO has data to be serviced, and mistakenly generate a DMA request. Before negating CAN_MCR.DMA, the CPU must perform a clear Legacy Rx FIFO operation.

CAN_IFLAG1.BUF5I (Frames available in Legacy Rx FIFO) is asserted when there is at least one frame available to be read from the Legacy Rx FIFO; consequently, a DMA request is generated simultaneously. Upon receiving the request, the DMA controller can read the message (accessing the output of the Legacy Rx FIFO as a message buffer). The DMA reading process must end by reading address 0x8C, which clears CAN_IFLAG1.BUF5I and updates both the Legacy Rx FIFO output with the next message (if the Legacy Rx FIFO is not empty) and the CAN_RXFIR register with the attributes of the new message. If there are more messages stored in the Legacy Rx FIFO, CAN_IFLAG1.BUF5I is re-asserted and another DMA request is issued. Otherwise, CAN_IFLAG1.BUF5I remains negated.

NOTE: CAN_RXFIR register contents cannot be read after DMA completes the Legacy Rx FIFO read. The IDHIT information is also available in the C/S word at address 0x080 (see ”Legacy Rx FIFO Structure”).

The CAN_IFLAG1.BUF6I and CAN_IFLAG1.BUF7I flags are not used when the DMA feature is enabled.

When the FlexCAN is working with DMA, the CPU does not receive any Legacy Rx FIFO interrupt and must not clear the related IFLAGs. In addition, the related IMASKs are not used to mask the generation of DMA requests.

17.4.9.2 Clear Legacy Rx FIFO Operation

When CAN_MCR.RFEN is asserted, the clear Legacy Rx FIFO operation is a feature used to empty Legacy Rx FIFO contents. With CAN_MCR.RFEN asserted, the clear Legacy Rx FIFO occurs when the CPU writes 1 to CAN_IFLAG1.BUF0I. This operation can only be performed in Freeze mode and is blocked by hardware in other modes. This operation does not clear the Legacy Rx FIFO IFLAGs; consequently, the CPU must service all Legacy Rx FIFO IFLAGs before executing the clear Legacy Rx FIFO task.

When the Legacy Rx FIFO is working with DMA, the clear Legacy Rx FIFO operation clears the CAN_IFLAG1.BUF5I flag and the DMA request is canceled.

CAUTION: The clear Legacy Rx FIFO operation does not clear IFLAGs, except when CAN_MCR.DMA is asserted. When CAN_MCR.DMA is asserted, only the CAN_IFLAG1.BUF5I flag is cleared.

17.4.10 CAN Protocol Related Features

This section describes the CAN protocol related features.

17.4.10.1 CAN FD Frames

The ISO 11898-1 standard specifies the Classical Frame format compliant to ISO 11898-1 (2003) and introduces the CAN Flexible Data Rate Frame format. The Classical Frame format allows bit rates up to 1 Mbit/s and payloads up to 8 bytes per frame. The Flexible Data Rate Frame format allows bit rates higher than 1 Mbit/s and payloads longer than 8 bytes per frame. The FlexCAN can receive and transmit CAN FD messages interleaved with Classical CAN messages.

There are three additional control bits in the CAN FD frame:

Yuntu Microelectronics

- The Extended Data Length (EDL) bit enables a longer data payload with different data length coding.
- The Bit Rate Switch (BRS) bit decides whether the bit rate is switched inside a CAN FD format frame.
- The Error State Indicator (ESI) flag is transmitted dominant by error active nodes, and recessive by error passive nodes.

There are no remote frames in the CAN FD format. A message configured to transmit a remote frame is always sent out in the Classical CAN format. When a FD frame is received and matches a mailbox, the RTR bit in the receiving message buffer is negated. The RTR bit must be considered in Classical CAN frames only.

CAN FD messages may be formatted as long frames, in which the data field exceeds 8 bytes and may range from 12 up to 64 bytes. CAN FD messages can also be configured to support bit rate switching, where the control field, the data field, and the CRC field of a CAN frame are transmitted with a higher bit rate than the beginning and the end of the frame. Messages in Classical CAN format are limited to transport a maximum payload of 8 bytes at nominal rate. Figure: ”CAN message formats” illustrates the message formats for Classical and FD frames with either standard or extended ID.

The ability to receive and transmit CAN FD messages is enabled by the CAN_MCR.FDEN bit. Either a recessive R0 bit in CAN frames with 11-bit identifiers or a recessive R1 bit in CAN frames with 29-bit identifiers are decoded as an EDL bit (not a reserved one). A CAN FD frame is recognized by a recessive EDL bit, while a Classical CAN frame is recognized by a dominant EDL bit. The BRS bit specifies whether this frame switches the bit rate in its data phase. A long frame is decoded according to the DLC field value.

CAN FD messages can be transmitted with two different bit rates. The first part of a CAN FD frame, from the Start Of Frame (SOF) bit until the Bit Rate Switch (BRS) bit, also called the arbitration phase, is transmitted with the nominal bit rate based on a set of nominal CAN bit timing configuration values. The second part, from the BRS bit until the CRC Delimiter bit, also named the data phase, is transmitted with the data bit rate defined by a second set of CAN data bit timing configuration values. Finally, from the CRC Delimiter until the Intermission bits, the transmission resumes to nominal bit rate. In CAN FD frames with bit rate switching, the bit timing is changed inside the frame at the sample point of the BRS bit if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the nominal CAN bit timing is used as defined by the CAN_CBT register (also by CAN_CTRL1 register for backward compatibility). Upon detecting a recessive BRS bit, the CAN data bit timing is used as defined by the CAN_FDCBT register.

NOTE: If the length of the time quantum in the nominal bit timing and the length of the time quantum in the data bit timing are not identical, a quantization error of up to one time quantum of the arbitration phase may be present as a phase error. This situation can occur after the switch from arbitration to data phase and lasts until the next synchronization event. Thus, the length of the time quantum should be the same in nominal and data bit timing in order to minimize the chance of error frames on the CAN bus, and to optimize the clock tolerance in networks that use CAN FD frames.

Yuntu Microelectronics

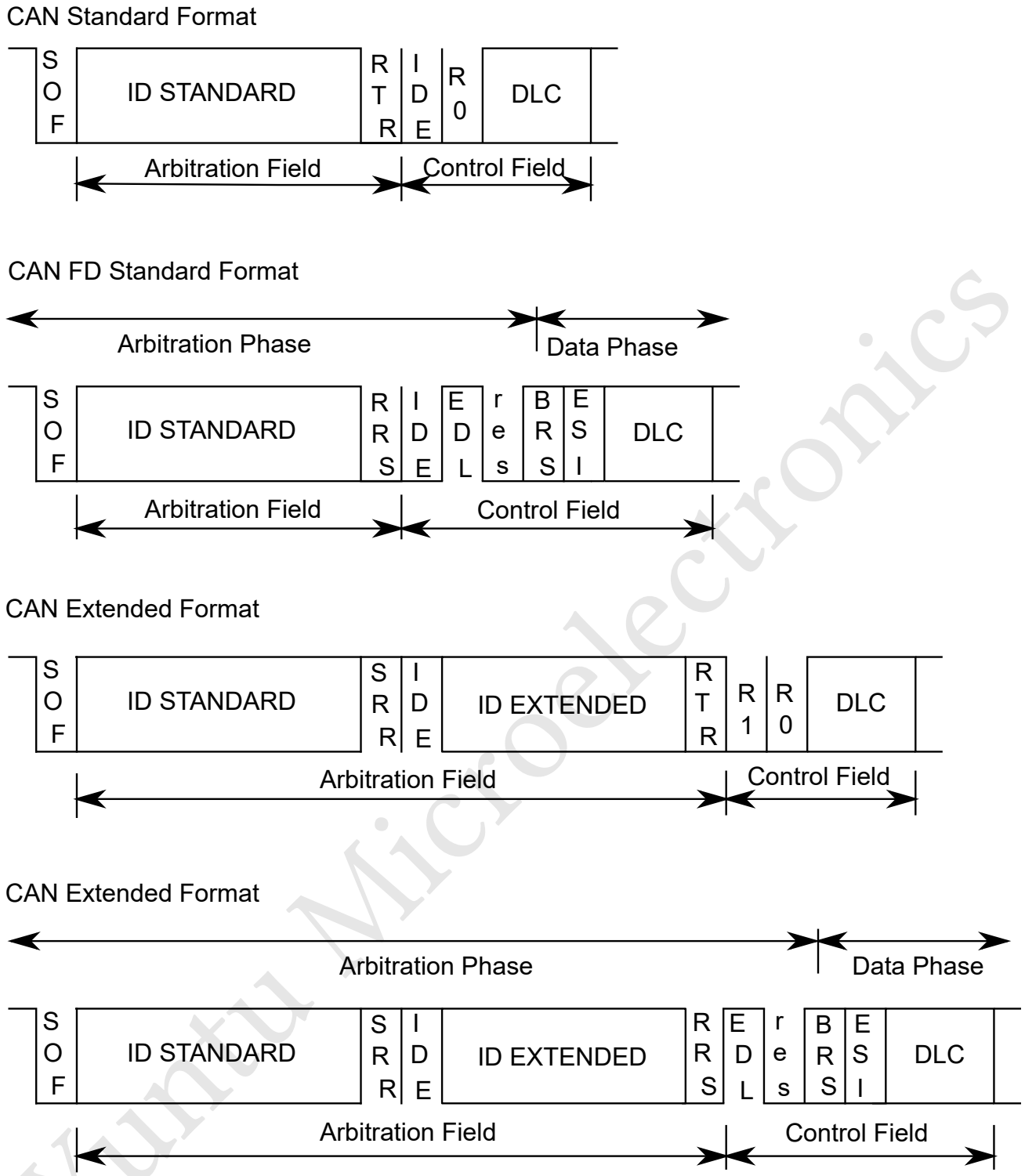


Figure 17.13: CAN Message Formats

CAN_FDCTRL.FDRATE enables the transmission of all frames with bit rate switching if the BRS bit in the selected Tx MB is set. If FDRATE is negated, the transmission is performed at nominal rate regardless of the BRS bit value. The CAN_FDCTRL.FDRATE bit can be written any time but takes effect only for the next message transmitted or received.

The nominal bit timing is resumed at either the sample point of the CRC Delimiter bit or when an error is detected, whichever occurs first. Figure: ”Bit rate switching mechanism for can fd messages” describes

Yuntu Microelectronics

the mechanism for entering and leaving the data phase when BRS bit is recessive.

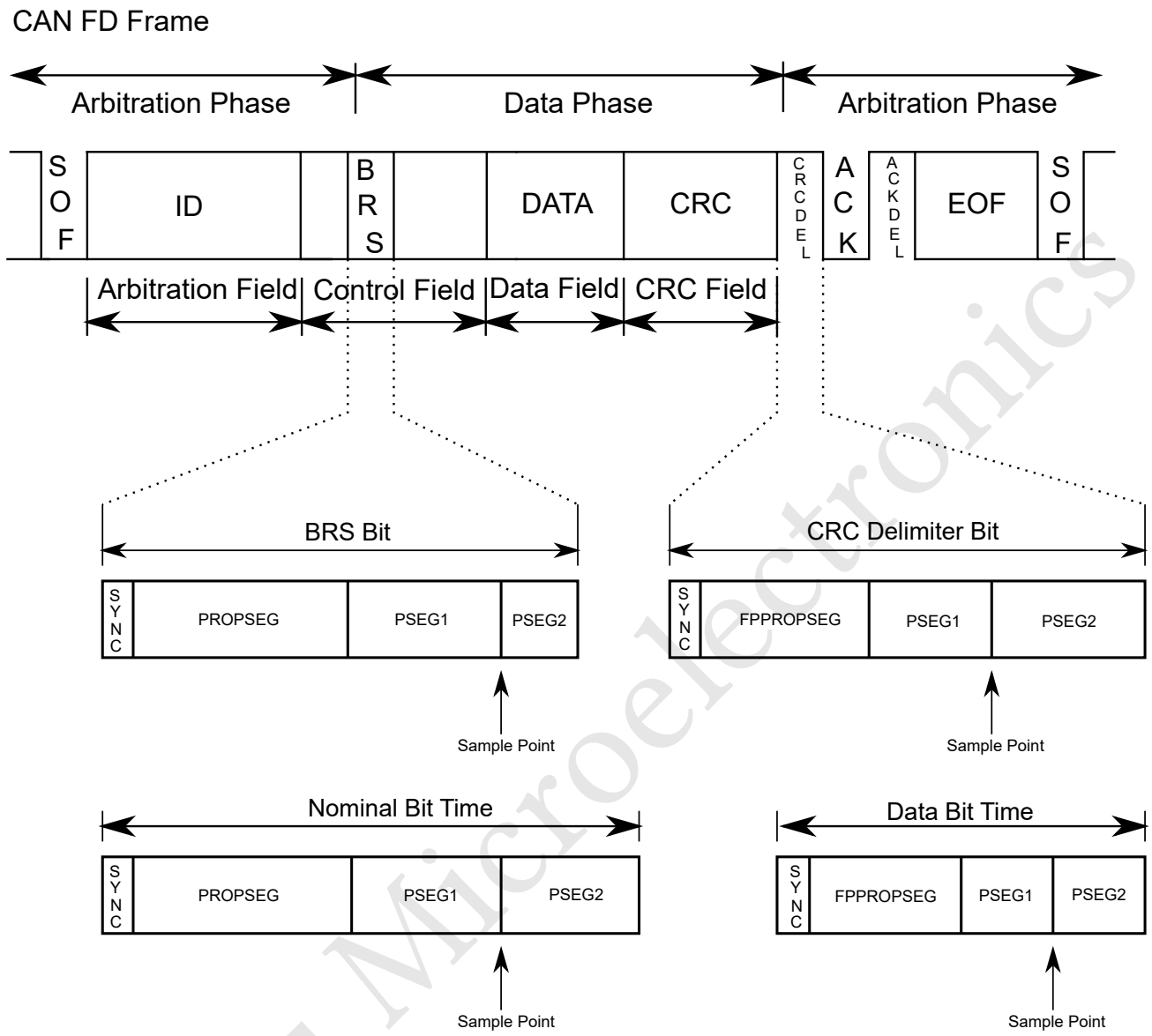


Figure 17.14: Bit Rate Switching Mechanism for CAN FD Messages

NOTE: In Classical CAN frames, the CRC delimiter is one single recessive bit. In CAN FD frames, the CRC delimiter may consist of one or two recessive bits. The FlexCAN sends only one recessive bit as the CRC delimiter, but it accepts two recessive bits before the edge from recessive to dominant that starts the acknowledge slot. As a receiver, the FlexCAN sends its acknowledge bit after the first CRC delimiter bit. In CAN FD frames, the FlexCAN accepts a two-bit dominant ACK slot as a valid ACK to compensate for phase shifts between the receivers.

The maximum configurable bit rate in the CAN FD data phase depends on the clock frequency of the PE submodule. For example, with a PE clock frequency of 40 MHz and the shortest configurable bit time of 5 time quanta, the bit rate in the data phase is 8 Mbit/s.

The value of the ESI bit is determined either by the transmitter's error state at the start of the transmission, if the frame is originated in the FlexCAN node, or by the original transmitting node if the FlexCAN is acting as a gateway for the message. If the transmitter is error passive, ESI is transmitted

Yuntu Microelectronics

recessive; otherwise, it is transmitted dominant(see ”CAN written and transmitted values of ESI bit”).

Table 17.90: CAN Written and Transmitted Values of ESI Bit

FlexCAN Fault Confinement Status at Start of Frame	ESI Bit of Tx MB	Transmitted ESI
Error active	0	0 (Error active)
Error passive	0	1 (Error passive)
Error active	1	1 (Error passive)
Error passive	1	1 (Error passive)

There are different CRC polynomials for different CAN frame formats:

- The first polynomial, CRC_15, is used for all frames in Classical CAN format.
- The second, CRC_17, is used for frames in CAN FD format with a data field up to sixteen bytes long.
- The third, CRC_21, is used for frames in CAN FD format with a data field longer than sixteen bytes.

Each polynomial results in a Hamming distance of 6. At the start of the frame, all three CRC polynomials are calculated concurrently. The CRC sequence to be transmitted is selected by the values of the EDL bit and the DLC bit field. When receiving a message, the FlexCAN decodes EDL and DLC to select the proper CRC polynomial to check for a CRC error.

In CAN FD format frames, stuff bits are included in the bit stream for CRC calculation. In Classical CAN format frames, stuff bits are not included. After the transmission of the last bit relevant to the CRC calculation, the CAN_FDCRC register stores the calculated CRC for the transmitted message, with the adequate length for the type of message, for both CAN FD and non-FD messages. The CAN_CRCR register reports a valid CRC for Classical CAN messages only.

In CAN FD format frames, the CAN bit stuffing method is changed for the CRC sequence so that the stuff bits are inserted at fixed positions. When the FlexCAN is transmitting a CAN FD frame, a fixed stuff bit is inserted just before the first bit of the CRC sequence, even if the last bits of the preceding field do not fulfill the CAN stuff condition. Additional stuff bits are inserted after each fourth bit of the CRC sequence. The value of any fixed stuff bit is the inverse value of its preceding bit. When the FlexCAN is receiving a CAN FD frame, it discards the fixed stuff bits from the bit stream for the CRC check. A Stuff Error is detected if the fixed stuff bit has the same value as its preceding bit.

The FlexCAN detects errors in CAN FD frames the same way as in Classical CAN frames. The error counters RXERRCNT and TXERRCNT in the CAN_ECR register accumulate the counts of Rx and Tx errors, respectively, for both FD and non-FD frames indiscriminately. There are two extra error counters (RXERRCNT_FAST and TXERRCNT_FAST) that accumulate Rx and Tx errors occurring in the data phase of CAN FD frames with the BRS bit set only. The rules for updating the error counters are the same for both CAN FD and non-FD frames (see ”Error Counter (CAN_ECR)”).

Error flags BITERR1, BITERR0, ACKERR, CRCERR, FRMERR, and STFERR in the CAN_ESR1 register report errors in both CAN FD and non-FD frames. They also generate the ERRINT interrupt if CAN_CTRL1.ERRMSK is asserted. The CAN_ESR1 register has additional error flags (BITERR1_FAST, BITERR0_FAST, CRCERR_FAST, FRMERR_FAST, and STFERR_FAST) to individually indicate the occurrence of errors in the data phase of CAN FD frames with the BRS bit set. There is no ACKERR detected in the data phase of a CAN FD frame. Fault confinement status reported in CAN_ESR1.FLTCONF is the same for both CAN FD and Classical CAN

Yuntu Microelectronics

frames, and is based on RXERRCNT and TXERRCNT error counters only. Information contained in RXERRCNT_FAST and TXERRCNT_FAST counters may be considered as status to help detect the error nature related to the bit rate value.

When the FlexCAN is in the data phase, either transmitting or receiving a CAN FD message, and detects an error, it immediately switches back to the arbitration phase and to the nominal rate to start an Error Flag.

Resynchronization and Hard Synchronization occur in CAN FD frames in the same way as in Classical CAN frames. A Hard Synchronization is also performed at the recessive-todominant edge from EDL to R0 in CAN FD format frames. The FlexCAN does not resynchronize while transmitting in the CAN FD data phase.

17.4.10.2 Transceiver Delay Compensation

The CAN FD protocol allows the transmission and reception of data at a higher bit rate than the nominal rate used in the arbitration phase when the message's BRS bit is set. This feature enables the use of rates up to 8 Mbps.

During the data phase of a CAN FD frame, the transmitter detects a bit error if it cannot receive its own latest transmitted bit at the sample point of that bit. When bit rate switching is enabled (BRS bit is asserted), the length of the CAN bit time in the data phase can become shorter than the transceiver's loop delay, thus impeding the correct comparison between the transmitted bit and the received bit within the current CAN bit time interval.

The FlexCAN supports an optional Transceiver Delay Compensation (TDC) mechanism that defines a secondary sample point where the transmitted bit is correctly compared with the received bit in order to check for bit errors.

The TDC mechanism is enabled by the CAN_FDCTRL.TDCEN bit and is effective only during the data phase of CAN FD frames having the BRS bit set. It has no effect either on non-FD frames, or on FD frames transmitted at normal bit rate. The TDC is active from the sample point of the BRS bit until the sample point of the CRC Delimiter bit, provided the respective message under transmission has the BRS bit set. When it is active, a comparison is done between the real received bit and the delayed transmitted bit, where the delay is calculated based on the measured transceiver loop delay.

NOTE: The actual value of the CRC Delimiter bit is disregarded by transmitters using the Transceiver Delay Compensation mechanism. A global error at the end of the CRC Field will cause the receivers to send error frames that the transmitter will detect during Acknowledge or End of Frame.

For every transmitted FD frame having the BRS bit set, the delay measurement is triggered by the transition from the recessive EDL bit to the dominant R0 bit (as shown in Figure: "Transceiver loop delay measurement".

The loop delay is measured in PE clock periods (CANCLK, see "Protocol Timing"), from the transmitted EDL-R0 edge to the received EDL-R0 edge. The position of the secondary sample point is defined by the measured loop delay time added to an offset value specified in CAN_FDCTRL.TDCOFF. The CAN_FDCTRL.TDCVAL bit field stores the result of this calculation. The TDCVAL value saturates at its maximum value of 15 CANCLK when the delay measurement is too long.

Yuntu Microelectronics

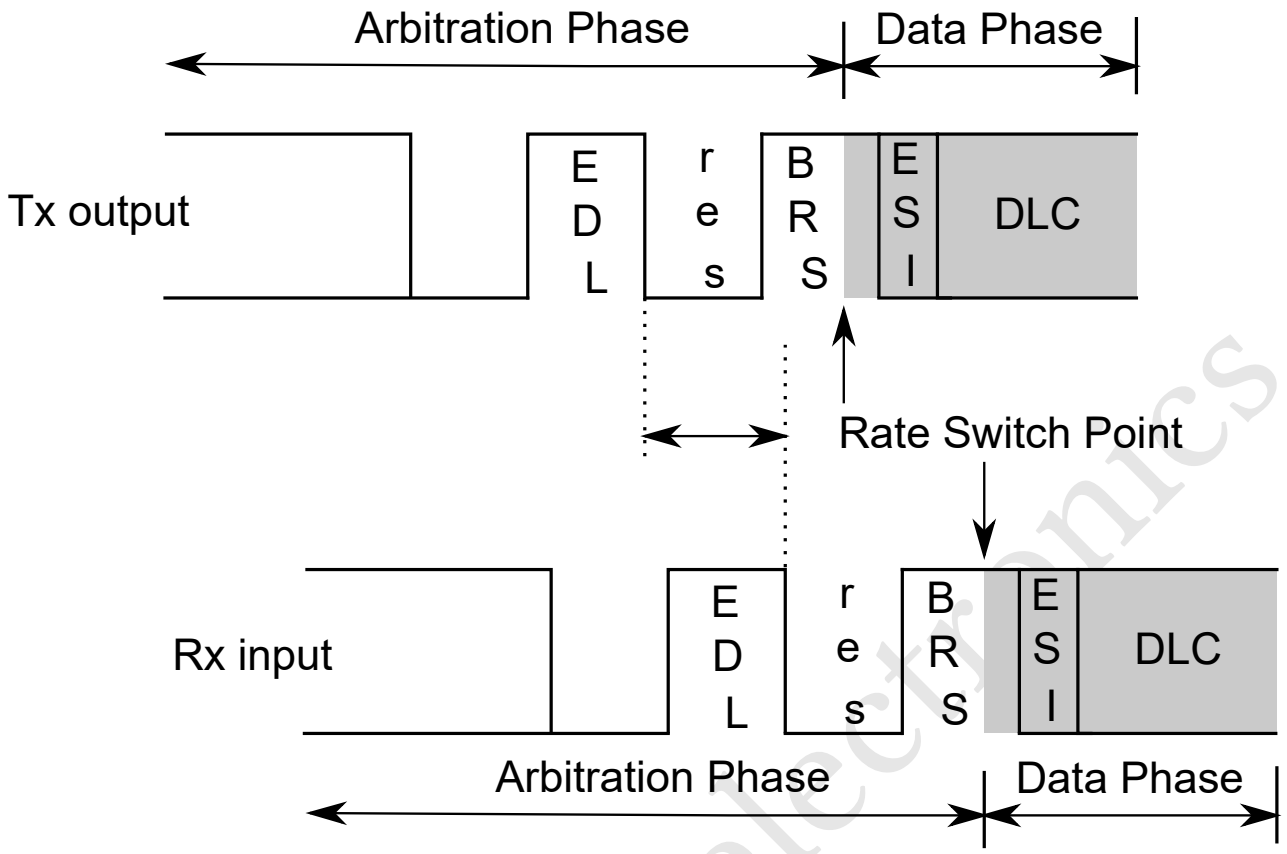


Figure 17.15: Transceiver Loop Delay Measurement

The measured loop delay is not enough to be used to define the secondary sample point because it relates to the CAN bit edges. The transceiver delay compensation offset TDCOFF or ETDCOFF is used to shift the secondary sample point from the edge to an intermediate point inside the bit time (for example, half of the bit time in the data phase), far away from its edges. Therefore, the TDCOFF or ETDCOFF value cannot be larger than the CAN bit duration in the data phase.

If the secondary sample point is set very near the CAN bit edge (SYNC field), then problems may occur during the bit sampling in the data phase. For the TDC to work reliably, the offset has to use optimal settings. To be sure the bit sampling is performed in the best region, the TDC offset should be configured as shown in the following equation:

Offset = (FPSEG1 + FPROPSEG + 2) x (FPRESDIV + 1)

Offset = (DTSEG1 + 2) x (EDPRESDIV + 1), if ETDCEN = 1

Below Figure shows the secondary sample point (SSP) position when these settings are used.

Yuntu Microelectronics

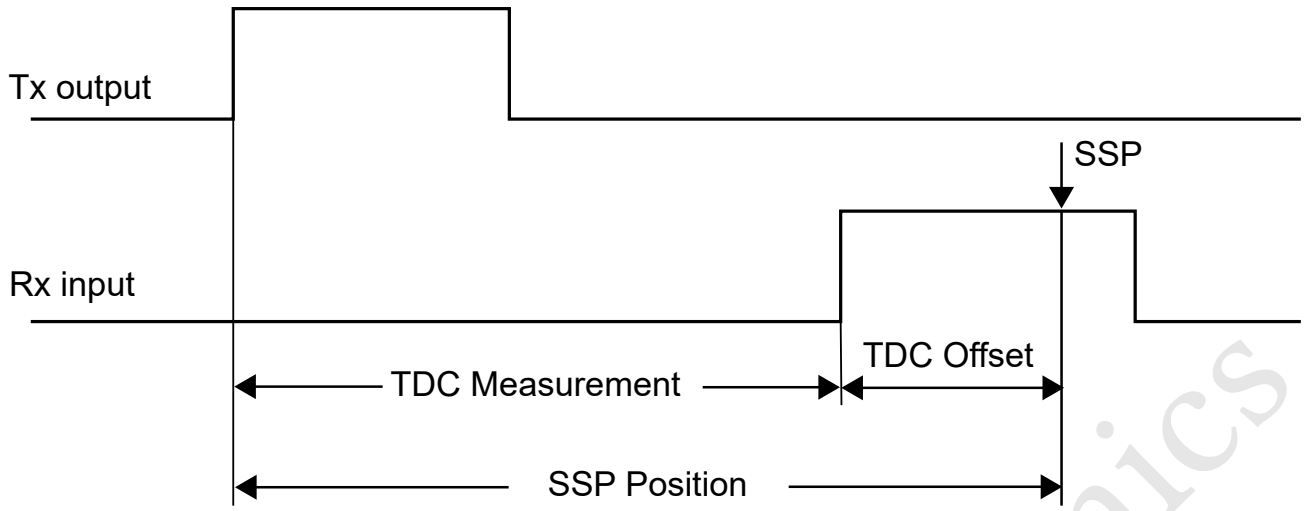


Figure 17.16: SSP Position with Optional Values

Alternatively, if CAN_CTRL2.BTE and CAN_ETDC.ETDCEN are set, the CAN_ETDC.TDMDIS bit can be set to disable the transceiver delay measurement. In this case, the SSP position is defined only by the CAN_ETDC.ETDCOFF field. Figure 64 shows the SSP position when the transceiver delay measurement is disabled.

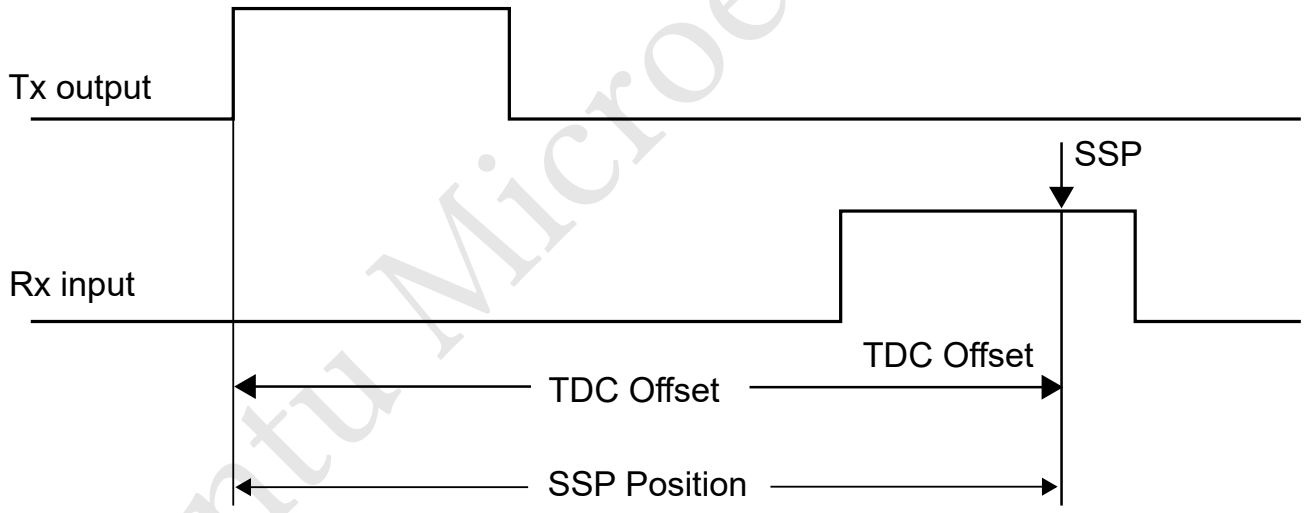


Figure 17.17: SSP Position with TDC Measurement Disabled by ETDCEN and TDMDIS

During the data phase of CAN FD frames with bit rate switching enabled, at the onset of every Tx CAN bit, the transmitted Tx bit value is temporarily stored in a buffer and a time countdown based on TDCVAL is started which ends with the comparison of the received Rx bit (delayed by the external loop delay plus the specified offset) with the stored Tx bit. If a bit error is detected at the secondary sample point, the FlexCAN issues an error flag to the CAN bus at the next sample point.

During the arbitration phase, the delay compensation is always disabled. The maximum delay that can be compensated by the FlexCAN's transceiver delay compensation during the data phase is 3 CAN bit times - 2 Tq. Beyond this limit, the CAN_FDCTRL.TDCFAIL flag is set to indicate when the Transceiver Delay Compensation mechanism is out of range and unable to compensate the transceiver loop delay.

Yuntu Microelectronics

17.4.10.3 Remote Frame Transmission

The CPU can program a mailbox to be a remote request frame by configuring the mailbox as transmit with the RTR bit set to '1'. After the remote request frame is transmitted successfully, the mailbox becomes a receive message buffer, with the same ID as before.

The FlexCAN receives the resulting answer frame into the same message buffer that transmitted the remote frame. Filtering for that message buffer must not consider the RTR bit when comparing the incoming answer frame. The answer frame will have RTR = 0, whereas the remote frame is configured with RTR = 1. To exclude the RTR bit when filtering the incoming answer frame, make sure that either or both of the following bits are 0:

- The RTR bit of the respective mask register, either individual mask register RXIMRn[31] or global mask register RXMGMASK[31], depending on the value of CAN_MCR[IRMQ]. RXIMRn[31] disables RTR comparison for message buffer n (effective when CAN_MCR[IRMQ] = 1). RXMGMASK[31] disables RTR comparison for all message buffers (effective when CAN_MCR[IRMQ] = 0).
- CAN_CTRL2.EACEN = 0 globally disables comparing the RTR bit for incoming frames, regardless of a message buffer's RTR mask bit.

17.4.10.4 Remote Frame Reception

When a remote request frame is received by the FlexCAN, it can be treated in one of four ways, depending on the Remote Request Storing (CAN_ERFCR.RRS]), Legacy Rx FIFO Enable (CAN_MCR.RFEN), and Enhanced Rx FIFO Enable (CAN_MCR.ERFEN) bits:

- If RRS is negated, the frame's ID is compared to the IDs of the transmit message buffers with the CODE field 1010. If there is a matching ID, then this mailbox frame is transmitted. Note that if the matching mailbox has the RTR bit set, then the FlexCAN transmits a remote frame as a response. The received remote request frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame should match. In the case that a remote request frame is received and matches a mailbox, this message buffer immediately enters the internal arbitration process, but is considered as a normal Tx mailbox, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.
- If RRS is asserted, the frame's ID is compared to the IDs of the receive mailboxes with the CODE field 0100, 0010, or 0110. If there is a matching ID, then this mailbox stores the remote frame in the same fashion of a data frame. No automatic remote response frame is generated. The mask registers are used in the matching process.
- If RFEN is asserted, the FlexCAN does not generate an automatic response for remote request frames that match the Legacy Rx FIFO filtering criteria. If the remote frame matches one of the target IDs, it is stored in the Legacy Rx FIFO and presented to the CPU. Note that for filtering formats A and B, it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted (if they match the ID). Remote request frames are considered as normal frames, and generate a Legacy Rx FIFO overflow when a successful reception occurs and the Legacy Rx FIFO is already full.
- If ERFEN is set, the FlexCAN does not generate an automatic response for remote request frames that match the Enhanced Rx FIFO filtering criteria. Remote Request Frames are considered normal frames and generate an Enhanced Rx FIFO overflow when a successful reception occurs and the Enhanced Rx FIFO is already full.

NOTE: There is no remote frame in the CAN FD format. The RTR bit is replaced by a fixed dominant

Yuntu Microelectronics

RRS bit. The FlexCAN receives and transmits remote frames in the Classical CAN format.

17.4.10.5 Overload Frames

The FlexCAN does transmit overload frames when the following conditions are detected on the CAN bus:

- Detection of a dominant bit in the first/second bit of Intermission.
- Detection of a dominant bit at the 7th bit (last) of End of Frame field (Rx frames).
- Detection of a dominant bit at the 8th bit (last) of Error Frame Delimiter or Overload Frame Delimiter.

17.4.10.6 Message Buffer Time Stamp

The value of the Free Running Timer is sampled at the beginning of the Identifier field on the CAN bus, and is stored at the end of move-in in the TIME STAMP field of a message buffer, providing network behavior with respect to time.

When CAN_CTRL2.TIMER_SRC is asserted, the Free Running Timer is continuously clocked by an external time tick.

When CAN_CTRL2.TIMER_SRC is negated, the Free Running Timer is clocked by the FlexCAN bit-clock, which defines the baud rate on the CAN bus. During a message transmission/reception, it increments by one for each bit that is received or transmitted.

When there is no message on the bus, it counts using the previously programmed baud rate.

The Free Running Timer is not incremented during Module Disable, Doze, Deepsleep, and Freeze modes. It can be reset upon a specific frame reception, enabling network time synchronization. See the TSYN description in ”Control 1 Register (CAN_CTRL1)”.

Alternatively, if the high-resolution time stamp feature is implemented (TIME_STAMP_EN = 1), the time stamp of the message buffer can capture either the upper or lower 16 bits of the 32-bit time base (timestamp_base[31:0]), as controlled by CAN_CTRL2.MTSBASE.

17.4.10.7 High-Resolution Time Stamp

The high-resolution time stamp feature, implemented when configuration parameter TIME_STAMP_EN = 1, uses a dedicated timer with a 32-bit counter operating in freerunning mode.

The high-resolution time stamp is enabled by setting CAN_CTRL2.TSTAMPCAP to a nonzero value. When enabled, the dedicated 32-bit counter value at the timestamp_base[31:0] input is captured during a valid CAN frame and stored in the CAN_HR_TIME_STAMPn register.

Each CAN_HR_TIME_STAMPn register corresponds to a specific message buffer. For example, CAN_HR_TIME_STAMP0 stores the 32-bit time stamp associated with message buffer 0; HR_TIME_STAMP1 stores the 32-bit time stamp associated with message buffer 1; and so on.

The counter value is captured according to the CAN_CTRL2.TSTAMPCAP value. For classical CAN frames, the capture point can be the Start of Frame bit or the point in time at which a CAN frame is considered valid, which is the seventh bit of End of Frame for transmission and the sixth bit of End of Frame for reception. For CAN FD frames, the capture points can be the Start of Frame, the point in time at which a CAN FD frame is considered valid, or the res bit of a CAN FD frame.

The 16-bit TIME STAMP field of the message buffer can be configured to capture the lower or higher 16 bits of the high-resolution timer, as controlled by CAN_CTRL2.MBTSBASE.

Yuntu Microelectronics

17.4.10.8 Protocol Timing

Figure: ”CAN engine clocking scheme” shows the structure of the clock generation circuitry that feeds the PE submodule. The clock source bit (CAN_CTRL1.CLKSRC) defines whether the internal clock is connected to the output of a crystal oscillator (Oscillator Clock) or to the Peripheral Clock.

To ensure reliable operation, the clock source should be selected while the FlexCAN is in Module Disable mode (CAN_MCR.MDIS bit is asserted).

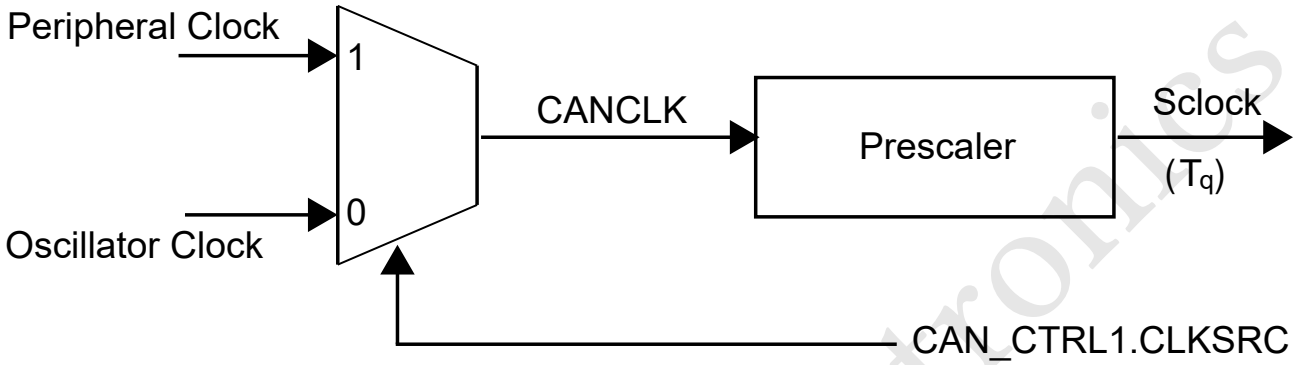


Figure 17.18: CAN Engine Clocking Scheme

The oscillator clock should be selected whenever a tight tolerance (up to 0.1%) is required in the CAN bus timing. The crystal oscillator clock has better jitter performance than the peripheral clock.

The FlexCAN module supports a variety of means to set up bit timing parameters that are required by the CAN protocol. The Control 1 Register (CAN_CTRL1) has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2, and RJW.

The CAN Bit Timing register (CAN_CBT) extends the range of the CAN bit timing variables in CAN_CTRL1. The CAN FD Bit Timing register (CAN_FDCBT) provides a second set of CAN bit timing variables to be applied at the data phase of CAN FD frames with the Bit Rate Switch (BRS) set.

The Enhanced Nominal CAN Bit Timing register (CAN_ENCBT) extends the range of CAN bit timing variables in CAN_CBT. The Enhanced Data Phase CAN Bit Timing register (CAN_EDCBT) extends the range of CAN bit timing variables in CAN_FDCBT. When using CAN_ENCBT and CAN_EDCBT, the bit timing nominal and data phase serial clock (Sclock) dividers must be programed in the Enhanced CAN Bit Timing Prescalers register (CAN_EPRS).

NOTE: When the CAN FD feature is enabled, always set CAN_CBT.BTF and configure the CAN bit timing variables in CAN_CBT. See or. When the CAN FD feature is enabled, always set CAN_CBT.BTF or CAN_CTRL2.BTE and configure the CAN bit timing variables in CAN_CBT CAN_ENCBT. See ”CAN FD Bit Timing Register (CAN_FDCBT)” or ”Enhanced Nominal CAN Bit Timing Register (CAN_ENCBT)”.

The PRESDIV field (as well as its extended range EPRESDIV (or ENPRESDIV) and FDPRESDIV (or EDPRESDIV) for the data phase bits of CAN FD messages) defines the Prescaler Value (see the equation below) that generates the Serial Clock (Sclock), whose period defines the time quantum used to compose the CAN waveform. A time quantum (Tq) is the atomic unit of time handled by the CAN engine.

$$T_q = \frac{(PRESDIV + 1)}{f_{CANCLK}}$$

Yuntu Microelectronics

The bit rate, which defines the rate at which the CAN message is either received or transmitted, is given by the formula:

CAN Bit Time = (Number of Time Quanta in 1 bit time) * Tq

$$BitRate = \frac{1}{CANBitTime}$$

A bit time is subdivided into three segments:

- SYNC_SEG – Has a fixed length of one time quantum. Signal edges are expected to occur within this segment.
- Time Segment 1 – Includes the Propagation Segment and the Phase Segment 1 of the CAN standard. Time Segment 1 can be programmed by setting the PROPSEG and the PSEG1 fields of the CAN_CTRL1 register so that their sum (plus 2) is in the range of 2 to 16 time quanta. When CAN_CBT.BTF is asserted, the FlexCAN uses the EPROPSEG and EPSEG1 fields from the CAN_CBT register so that their sum (plus 2) is in the range of 2 to 96 time quanta. For messages in CAN FD format with the BRS bit set, the FlexCAN uses FDPROPSEG and FDPSEG1 from CAN_FDCBT instead, so that their sum (plus 1) is in the range of 2 to 39 time quanta.

If CAN_CTRL2.BTE is set, the FlexCAN uses CAN_ENCBT.NTSEG1 to configure time segment 1 in the range of 2 to 256 time quanta. For the data phase in CAN FD messages with BRS set, CAN_EDCBT.DTSEG1 must be used for configuring time segment 1 in the range of 2 to 32 time quanta.

- Time Segment 2 – Represents the Phase Segment 2 of the CAN standard. It can be programmed by setting the PSEG2 field of the CAN_CTRL1 register (plus 1) to be 2 to 8 time quanta long. When CAN_CBT.BTF is asserted, the FlexCAN uses the EPSEG2 field of CAN_CBT register so that its value (plus 1) is in the range of 2 to 32 time quanta. For messages in CAN FD format with the BRS bit set, the FlexCAN uses FDPSEG2 from CAN_FDCBT instead, so that its value (plus 1) is in the range of 2 to 8 time quanta. The Time Segment 2 cannot be smaller than the Information Processing Time (IPT), which is 2 time quanta in the FlexCAN.

If CAN_CTRL2.BTE is set, the FlexCAN uses CAN_ENCBT.NTSEG2 to configure time segment 2 in the range of 2 to 128 time quanta. For the data phase in CAN FD messages with BRS set, CAN_EDCBT.DTSEG2 must be used for configuring time segment 2 in the range of 2 to 16 time quanta.

see also ”Segments within the bit time (example using CAN_ CBT and CAN_FDCBT bit timing variables for CAN FD format)”, ”Segments within the bit time (example using CAN_CTRL1 bit timing variables for classical CAN format)” and ”CAN time segment syntax”.

NOTE: For further explanation of the underlying concepts, see ISO 11898-1. See also the CAN 2.0A/B protocol specification for bit timing.

NOTE: The bit time defined by the above time segments must not be smaller than 5 time quanta. For bit time calculations, use an Information Processing Time (IPT) of 2, which is the value implemented in the FlexCAN module.

Yuntu Microelectronics

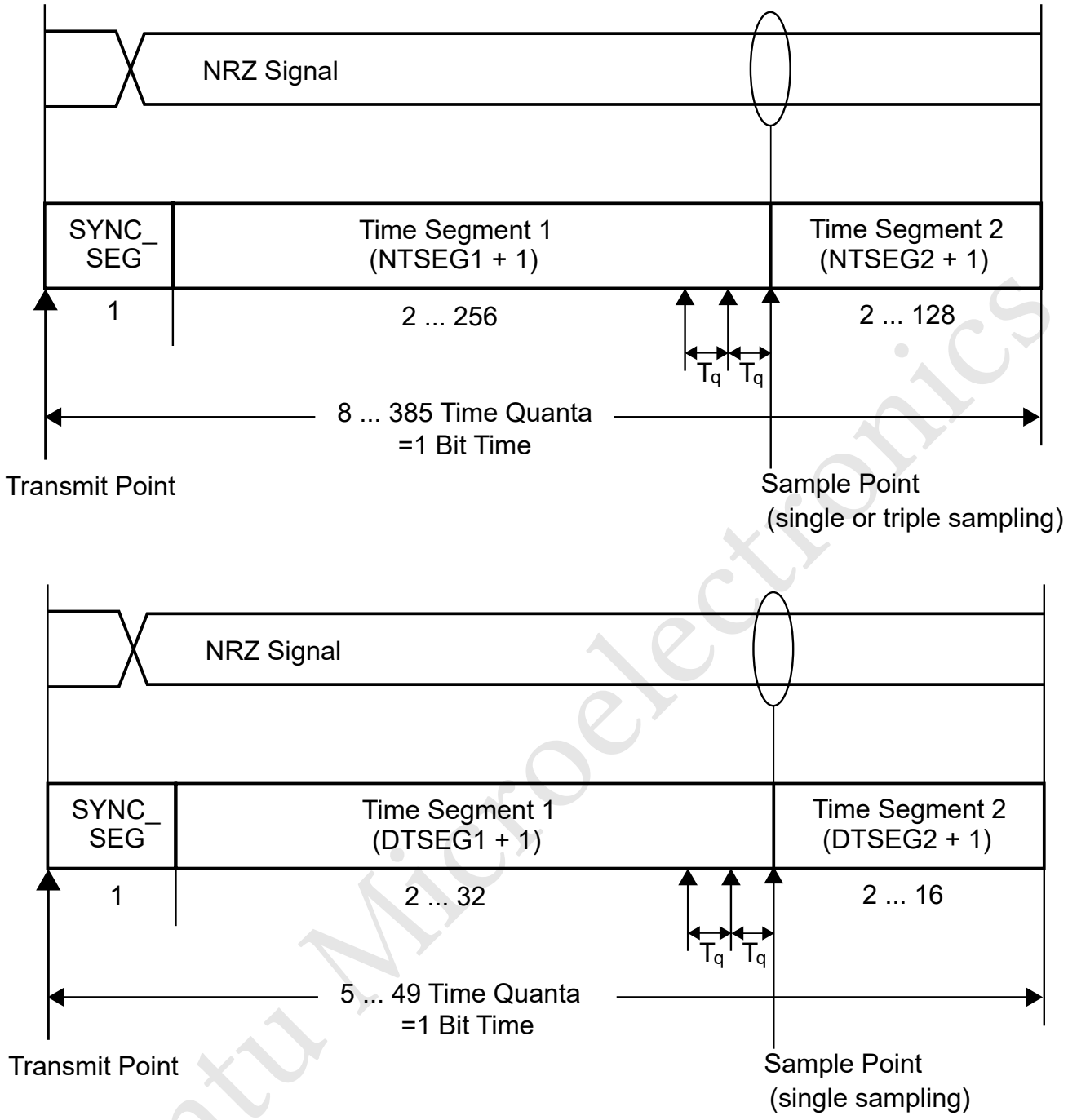


Figure 17.19: Segments within the Bit Time Example Using CAN ENCBT and CAN EDCBT Bit Timing Variables

Yuntu Microelectronics

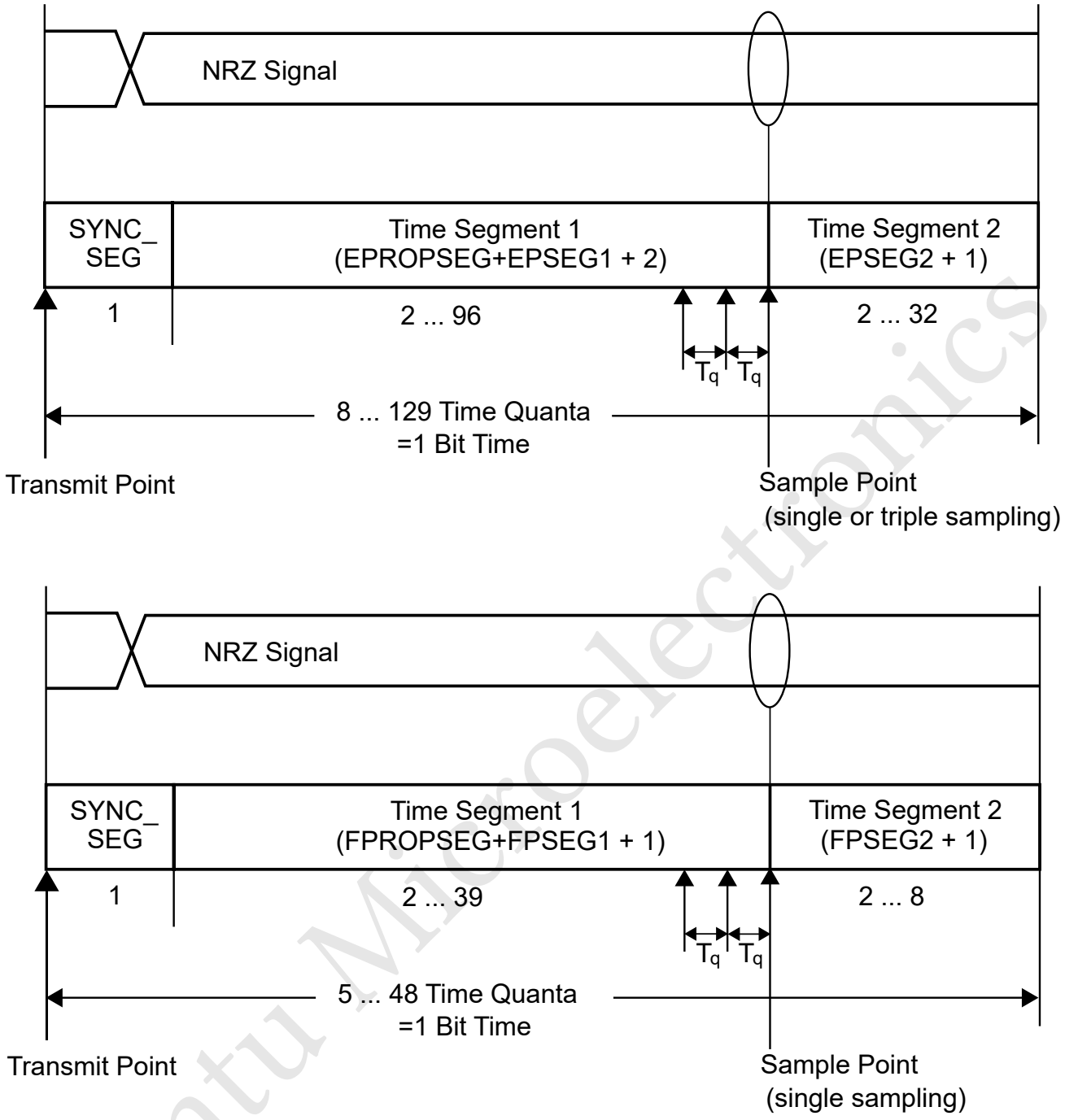


Figure 17.20: Segments within the Bit Time Example Using CAN CBT and CAN FDCBT Bit Timing Variables for CAN FD Format

Yuntu Microelectronics

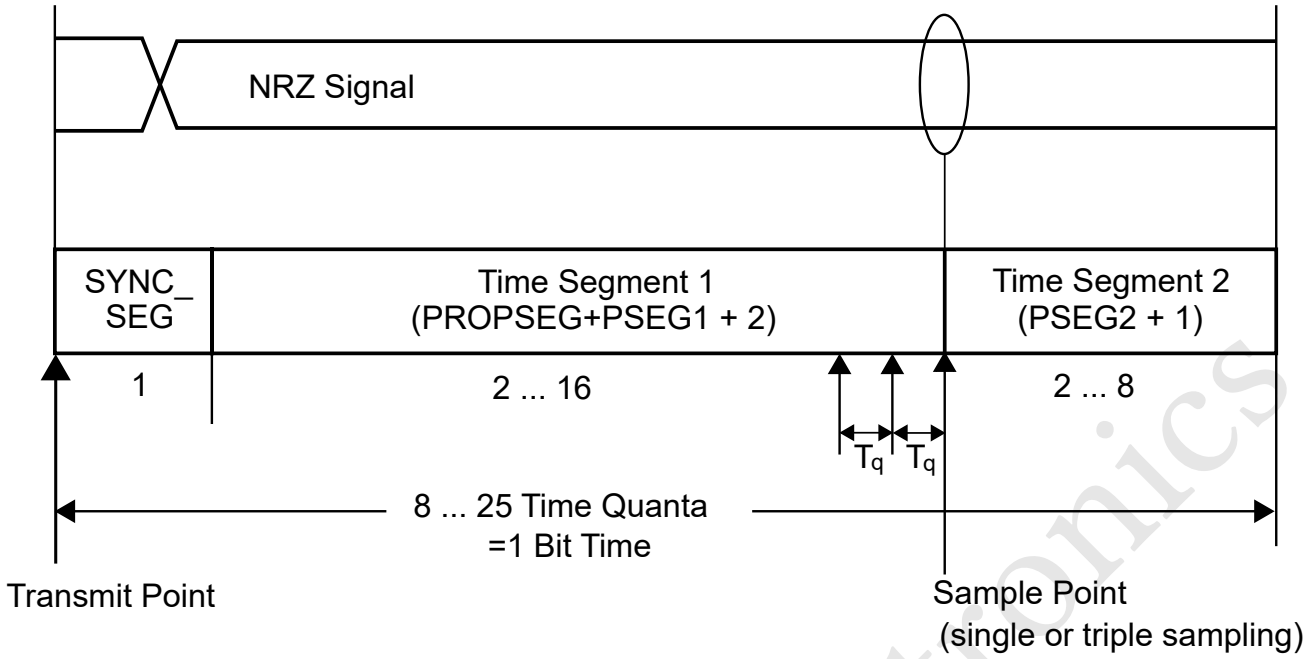


Figure 17.21: Segments within the Bit Time Example Using CAN CTRL1 Bit Timing Variables for Classical CAN Format

Table 17.91: CAN Time Segment Syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period
TSEG1	Corresponds to the sum of PROPSEG and PSEG1
TSEG2	Corresponds to the PSEG2 value
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

Table: ”CAN BOSCH CAN 2P0B standard complicant bit segment settings” gives some examples of the CAN compliant segment settings for Classical CAN format (Bosch CAN 2.0B) (non-FD) messages.

Table 17.92: CAN BOSCH CAN 2.0B Standard Complicant Bit Segment Settings

Time Segment 1	Time Segment 2	Re-Synchronization Jump Width
5 ... 10	2	1 ... 2
4 ... 11	3	1 ... 3
5 ... 12	4	1 ... 4
6 ... 13	5	1 ... 4
7 ... 14	6	1 ... 4
8 ... 15	7	1 ... 4
9 ... 16	8	1 ... 4

Yuntu Microelectronics

NOTE: The user must ensure the bit time settings are in compliance with the CAN Protocol standard (ISO 11898-1).

Whenever CAN bit is used as a measure of time duration (for example, estimating the occurrence of a CAN bit event in a message), the number of peripheral clocks in one CAN bit (NumClkBit) can be calculated as:

$$NumClkBit = (\frac{f_{SYS}}{f_{CANCLK}}) \times (PRESDIV + 1) \times (PROPSEG + PSEG1 + PSEG2 + 4)$$

or, if CAN_CTRL2.BTE = 1:

$$NumClkBit = (\frac{f_{SYS}}{f_{CANCLK}}) \times (ENPRESDIV + 1) \times (NTSEG1 + NTSEG2 + 3)$$

where:

- NumClkBit is the number of peripheral clocks in one CAN bit.
- fCANCLK is the PE clock in Hz (see Figure: ”CAN engine clocking scheme”).
- fSYS is the frequency of operation of the system (CHI) clock in Hz.
- PSEG1 is the CAN_CTRL1.PSEG1 value.
- PSEG2 is the CAN_CTRL1.PSEG2 value.
- PROPSEG is the CAN_CTRL1.PROPSEG value.
- PRESDIV is the CAN_CTRL1.PRESDIV value.
- ENPRESDIV is the CAN_ENPRS.ENPRESDIV value.
- NTSEG1 is the CAN_ENCBT.NTSEG1 value.
- NTSEG2 is the CAN_ENCBT.NTSEG2 value.

The formula above is also applicable to the alternative CAN bit timing variables described in the CAN Bit Timing Register (CAN_CBT) and the CAN FD Bit Timing Register (CAN_FDCBT).

For example, 180 CAN bits = (180 x NumClkBit) peripheral clock periods.

17.4.10.9 Arbitration and Matching Timing

During normal reception and transmission, the matching, arbitration, move-in, and move-out processes are executed during certain time windows inside the CAN frame, as shown in the following figures.

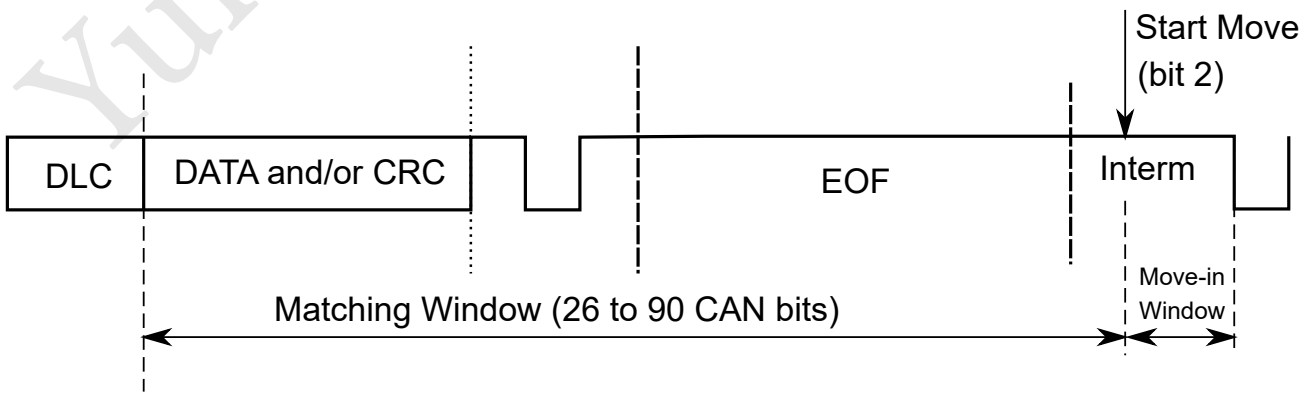


Figure 17.22: Matching and Move-in Time Windows

Yuntu Microelectronics

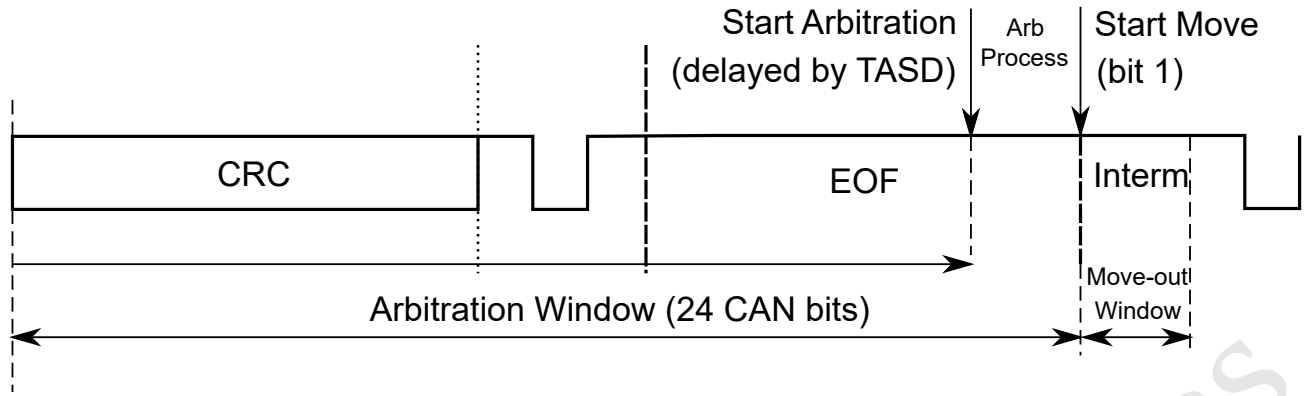


Figure 17.23: Arbitration and Move-out Time Windows

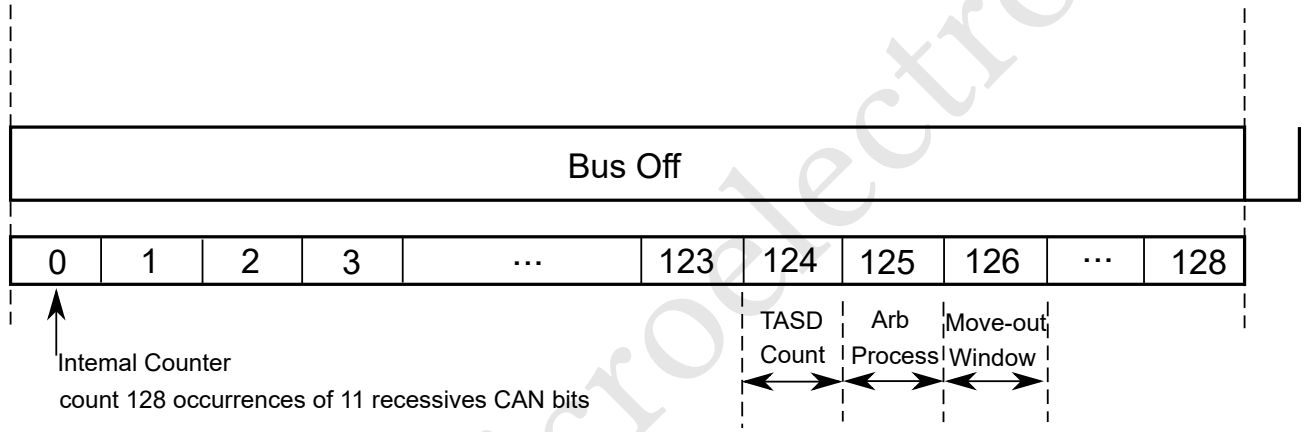


Figure 17.24: Arbitration at the End of Bus Off and Move-out Time Window

NOTE: In the preceding figures, the matching and arbitration timing does not take into account the delay caused by the concurrent memory access due to the CPU or other internal FlexCAN submodules.

17.4.10.10 Tx Arbitration Start Delay

The Tx Arbitration Start Delay (TASD) bit field in Control 2 register (CAN_CTRL2.TASD) is a variable that indicates the number of CAN bits used by the FlexCAN to delay the Tx arbitration process start point from the first bit of the CRC field of the current frame.

CAN_CTRL2.TASD can be written only in Freeze mode because it is blocked by hardware in other modes.

The transmission performance is impacted by the ability of the CPU to reconfigure MBs for transmission after the end of the internal arbitration process, in which the FlexCAN finds the winner MB for transmission (see ”Arbitration Process”). If the arbitration ends too early before the first bit of the Intermission field, then there is a chance that the CPU reconfigures some Tx MBs and the winner MB is no longer the best candidate to be transmitted.

TASD is useful to optimize the transmission performance by defining the arbitration start point, as shown in the next figure, based on factors such as:

Yuntu Microelectronics

- The peripheral-to-oscillator clock ratio
- CAN bit timing variables that determine the CAN bit rate
- The number of MBs in use by the matching and arbitration processes

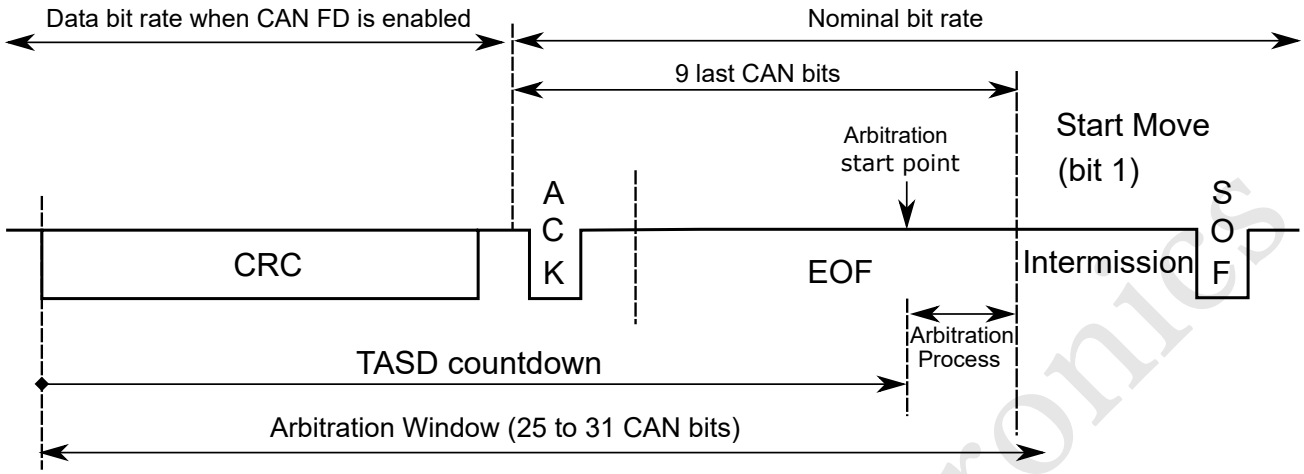


Figure 17.25: Optional TX Arbitration Start Point

The duration of an arbitration process, in terms of CAN bits, is directly proportional to the number of available MBs and to the CAN bit rate, and inversely proportional to the peripheral clock frequency.

The optimal arbitration timing is that in which the last MB is scanned right before the first bit of the Intermission field of a CAN frame. For instance, if there are few MBs and the peripheral/oscillator clock ratio is high and the CAN baud rate is low, then the arbitration can be placed closer to the frame's end, adding more delay to its start point, and vice-versa.

If TASD is set to 0, the arbitration start is not delayed and more time is reserved for arbitration. On the other hand, if TASD is close to 24, then the CPU can configure a Tx MB later and less time is reserved for arbitration. If too little time is reserved for arbitration, the FlexCAN may be not be able to find a winner MB in time to be transmitted with the best chance to win the bus arbitration against external nodes on the CAN bus.

The optimal TASD value can be calculated as follows:

For CAN FD frames and $(MAXMB + 1) \leqslant NMB_{END}$

$$TASD = 31 - \frac{2 * (MAXMB + 1) + 4}{CPCN_N}$$

For CAN FD frames and $(MAXMB + 1) > NMB_{END}$

$$TASD = 22 - \frac{2 * (MAXMB + 1) - NMB_{END}}{CPCB_F}$$

For non-FD frames

$$TASD = 25 - \frac{2 * (MAXMB + 1) + 4}{CPCB}$$

Yuntu Microelectronics

where:

$$NMB_{END} = \frac{9 * CPCN_N - 4}{2}$$

$$BITRATE_F = \frac{f_{CANCLK}}{[1 + (FPSEG1 + 1) + (FPSEG2 + 1) + FPROPSEG] * (FPRESDIV + 1)}$$

$$BITRATE_N = \frac{f_{CANCLK}}{[1 + (EPSEG1 + 1) + (EPSEG2 + 1) + (EPROPSEG + 1)] * (EPRESDIV + 1)}$$

$$CPCB_N = \frac{f_{SYS}}{BITRATE_N}$$

$$CPCB_F = \frac{f_{SYS}}{BITRATE_F}$$

$$CPCB = CPCB_N$$

- MAXMB is the CAN_MCR.MAXMB value.
- NMBEND is the number of MBs that can be scanned by the arbitration process during the 9 last CAN bits at the end of a frame, see the figure above.
- BITRATEN is the CAN bit rate in bits per second calculated by the nominal CAN bit time variables.
- BITRATEF is the CAN bit rate in bits per second calculated by the data CAN bit time variables.
- CPCBN is the number of peripheral clocks per CAN bit in nominal bit rate for CAN FD frames.
- CPCBF is the number of peripheral clocks per CAN bit in data bit rate for CAN FD frames.
- CPCB is the number of peripheral clocks per CAN bit for non-FD frames.
- fCANCLK is the oscillator clock, in Hz.
- fSYS is the peripheral clock, in Hz.
- EPSEG1 is the CAN_CBT.EPSEG1 value. (CAN_CTRL1.PSEG1 can also be used.)
- EPSEG2 is the CAN_CBT.EPSEG2 value. (CAN_CTRL1.PSEG2 can also be used.)
- EPROPSEG is the CAN_CBT.EPROPSEG value. (CAN_CTRL1.PROPSEG can also be used.)
- EPRESDIV is the CAN_CBT.EPRESDIV value. (CAN_CTRL1.PRESDIV can also be used.)
- FPSEG1 is the CAN_FDCBT.FPSEG1 value.
- FPSEG2 is the CAN_FDCBT.FPSEG2 value.
- FPROPSEG is the CAN_FDCBT.FPROPSEG value.
- FPRESDIV is the CAN_FDCBT.FPRESDIV value.

If CAN_CTRL2.BTE = 1:

$$BITRATE_N = \frac{f_{CANCLK}}{[1 + (NTSEG1 + 1) + (NTSEG2 + 1)] * (ENPRESDIV + 1)}$$

See "Protocol Timing" for additional details.

The following tables give the TASD values calculated for some configuration cases.

Case 1:

Yuntu Microelectronics

- Clock ratio = 2:1 (example: peripheral clock 80 MHz and oscillator clock 40 MHz)
- Bit rate in arbitration phase = 1 Mbaud

Table 17.93: CAN TASD Values for Case1 Example

Number of MBs	TASD Value	Maximum Bit Rate in Data Phase (Mbaud)
16	24	Invalid
32	24	8.0
64	23	8.0
96	22	8.0
100	22	8.0
128	21	3.6

Case 2:

- Clock ratio = 1:1 (example: peripheral clock 40 MHz and oscillator clock 40 MHz)
- Bit rate in arbitration phase = 1 Mbaud

Table 17.94: CAN TASD Values for Case2 Example

Number of MBs	TASD Value	Maximum Bit Rate in Data Phase (Mbaud)
16	24	Invalid
32	23	6.67
54	22	5.0
64	21	3.33
96	20	1.6
128	18	1.05

Case 3:

- Clock ratio = 2:1 (example: peripheral clock 40 MHz and oscillator clock 20 MHz)
- Bit rate in arbitration phase = 1 Mbaud

Table 17.95: CAN TASD Values for Case3 Example

Number of MBs	TASD Value	Maximum Bit Rate in Data Phase (Mbaud)
16	24	Invalid
32	23	4.0
54	22	4.0
64	21	3.33
96	20	1.54
128	18	1.05

17.4.11 Clock Domains and Restrictions

The FlexCAN module has two clock domains asynchronous to each other:

Yuntu Microelectronics

- The Bus Domain feeds the CHI submodule and is derived from the peripheral clock.
- The Oscillator Domain feeds the PE submodule and is derived directly from a crystal oscillator clock, so that very low jitter performance can be achieved on the CAN bus.

When CAN_CTRL1.CLKSRC is set, synchronous operation occurs because both domains are connected to the peripheral clock (creating a 1:1 ratio between the peripheral and oscillator clocks).

When the two domains are connected to clocks with different frequencies and/or phases, there are restrictions on the frequency relationship between the two clock domains. In the case of asynchronous operation, the Bus Domain clock frequency must always be greater than the Oscillator Domain clock frequency.

NOTE: Asynchronous operation with a 1:1 ratio between peripheral and oscillator clocks is not allowed.

When doing matching and arbitration, the FlexCAN needs to scan the entire MB memory during the time slot of one CAN frame, comprising a number of CAN bits. In order to have sufficient time to do that, the following requirements must be observed:

- The peripheral clock frequency can not be lower than the oscillator clock frequency.
- There must be a minimum number of peripheral clocks per CAN bit, see below table:

Table 17.96: CAN Minimum Number of Peripheral Clocks per CAN Bit for Classical CAN Format

Number of Mailboxes	Value of CAN_MCR.RFEN bit	Minimum Number of Peripheral Clocks per CAN Bit
16	0	16
32	0	16
16	1	16
32	1	17

For Classical CAN frame format, the minimum number of peripheral clocks per CAN bit specified in ”CAN minimum number of peripheral clocks per CAN bit for classical CAN format” determines the minimum peripheral clock frequency for a given number of mailboxes and for an expected CAN bit rate. The CAN bit rate depends on the number of time quanta in a CAN bit, which can be defined by adjusting one or more of the bit timing values contained in either the CAN_CTRL1 register or the CAN_CBT register or the CAN_ENCBT register. The time quantum (Tq) is defined in ”Protocol Timing”. The minimum number of time quanta per CAN bit must be 8; therefore, the oscillator clock frequency should be at least 8 times the CAN bit rate.

For CAN FD frame format, there are some constraints that need to be satisfied. The number of peripheral clocks per CAN bit in nominal bit rate (NumClkNomBit) can be calculated by the following equation:

$$NumClkNomBit = (\frac{f_{SYS}}{f_{CANCLK}}) \times (PRESDIV + 1) \times (PROPSEG + PSEG1 + PSEG2 + 4) = \frac{f_{SYS}}{NomBitRate}$$

where PRESDIV, PSEG1 and PSEG2 are CAN bit time values in the CAN_CTRL1 register. Alternatively, the EPRESDIV, EPSEG1 and EPSEG2 values in the CAN_CBT register or the CAN_EPRS.ENPRESDIV, CAN_ENCBT.NTSEG1, and CAN_ENCBT.NTSEG2 values can be used instead. NumClkNomBit can also be calculated as a function of the expected nominal bit rate used in the Arbitration Phase (NomBitRate) as shown in the equation above.

Yuntu Microelectronics

The number of CAN bits in the Data Phase of a FD Frames with the BRS bit set (fast CAN bits) depends on the number of data bytes in the payload. The number of fast CAN bits (NumOfFastBits) can be determined as shown in Table: "CAN number of fast CAN bits in a CAN FD frame". The fewer the number of data bytes, the fewer the number of fast CAN bits, and less time is available for the FlexCAN to scan the whole MB memory during the internal matching and arbitration processes.

Table 17.97: CAN Number of Fast CAN Bits in a CAN FD Frame

Minimum Number of Data Bytes	DLC Field	NumOfFastBits
0	0x0	21
1	0x1	29
2	0x2	37
3	0x3	45
4	0x4	53
5	0x5	61
6	0x6	69
7	0x7	77
8	0x8	85
12	0x9	117
16	0xA	149
20	0xB	186
24	0xC	218
32	0xD	282
48	0xE	410
64	0xF	538

The critical part of a CAN FD frame is during the Data Phase, where the CAN bit rate is faster than in the Arbitration Phase. The minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) can be calculated to guarantee that enough time is available for the FlexCAN to scan the MB memory during reception and transmission. The following equation below calculates this constraint:

$$MinNumClkFastBitA = \frac{(8.5 * MaxNumOfMb) + (ERFEN * (2 * NFE + 4)) + 64 - (9 * NumClkNomBit)}{NumOfFastBits}$$

where MaxNumOfMb is the maximum number of available mailboxes defined in CAN_MCR.MAXMB. ERFEN and NFE are fields in CAN_ERFCR.

The clock domain crossing circuit between the CHI and PE submodules also imposes a minimum number of peripheral clocks per fast CAN bit for the handshake mechanism to work properly without losing status information through the interface, as shown in the following equation:

$$MinNumClkFastBit_B = 3 * (1 + \frac{f_{SYS}}{f_{CANCLK}})$$

Therefore, the minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) is determined by the larger of the two values calculated above:

MinNumClkFastBit = Maximum (MinNumClkFastBitA, MinNumClkFastBitB)

Yuntu Microelectronics

Then, the maximum CAN bit rate in the Data Phase of CAN FD frames (DataBitRateMAX) can be calculated as follows:

$$DataBitRate_{MAX} = \frac{f_{CANCLK}}{ROUNDUP(\frac{MinNumClkFastBit * f_{CANCLK}}{f_{SYS}})}$$

The peripheral and oscillator clock frequencies, the maximum number of mailboxes, and the expected nominal bit rate affect the maximum data bit rate attainable by the FlexCAN in CAN FD mode. In addition, the data bit rate depends on the minimum payload size of FD frames used in a given application.

To illustrate how the CAN FD bit rate is affected by the configuration of FlexCAN variables, an application example with the peripheral and oscillator clock frequencies set to 50 MHz and 40 MHz, respectively, is considered.

Step 1 - Considering the nominal bit rate as 1 Mbps, the number of peripheral clocks per CAN bit in nominal bit rate is calculated as follows:

$$NumClkNomBit = \frac{50 * 10^6}{1 * 10^6} = 50$$

Step 2 - The number of fast CAN bits (NumOfFastBits) is determined as shown in table ”CAN number of fast CAN bits in a CAN FD frame”. For example, if the minimum payload in FD frames is 8 bytes, then there are 85 CAN bits in the Data Phase.

Step 3 - Assuming the maximum number of mailboxes is 96 and the Enhanced Rx FIFO feature is disabled, the minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) can be calculated as follows:

$$MinNumClkFastBit_A = \frac{(8.5 * 96) + 64 - (9 * 50)}{85} = 5.06$$

$$MinNumClkFastBit_B = 3 * (1 + \frac{50}{40}) = 6.75$$

$$MinNumClkFastBit = Maximum(5.06, 6.75) = 6.75$$

Step 4 - The maximum CAN bit rate in the Data Phase can then be calculated as follows:

$$DataBitRate_{MAX} = \frac{40 * 10^6}{ROUNDUP(\frac{6.75 * 40 * 10^6}{50 * 10^6})} = 6.667 Mbps$$

As demonstrated in this example, even though the oscillator clock frequency (40 MHz) is adequate to generate a data rate of 8 Mbps in CAN FD mode, the specific FlexCAN configuration limits this rate to 6.667 Mbps. This limitation is mainly due to the low peripheral clock frequency that imposes the MinNumClkFastBitB limit.

Below table shows the maximum data rate for CAN FD according to clock frequencies, payload size, and number of available mailboxes. As shown in below table, for some cases, if the number of available mailboxes is reduced, the FlexCAN can then achieve a data rate up to 8 Mbps.

Yuntu Microelectronics

Table 17.98: CAN Maximum CAN Bit Rate in Data Phase of CAN FD Frames

Peripheral Clock Frequency (MHz)	Payload Size	Number of Available Mailboxes	Maximum Data Rate (Mbps)
40	8	94	6.667
40	8	114	5.0
40	12	117	6.667
40	12	128	5.714
50	12 to 64	128	6.667
60	8	126	8.0
60	12	128	8.0
67	6	128	8.0
80	3	128	8.0
100	0	128	8.0

17.4.12 Modes of Operation Details

The FlexCAN module has functional modes and Low-Power modes. See "Modes of Operation" for an introductory description of all the modes of operation. The following subsections contain functional details on Freeze mode and the Low-Power modes.

17.4.12.1 Freeze Mode

Freeze mode is requested either by the CPU through the assertion of the CAN_MCR.HALT bit or when the chip is put into Debug mode. In both cases, it is also necessary that the CAN_MCR.FRZ bit is asserted and the FlexCAN is not in a Low-Power mode. Freeze mode is also requested by FlexCAN through the automatic assertion of both HALT and FRZ bits when the CAN_MECR.NCEFAFRZ bit is set and a non-correctable error is detected in a memory read access performed by FlexCAN internal processes (see "response to Errors").

The acknowledgement is obtained through the assertion by the FlexCAN of CAN_MCR.FRZACK bit. The CPU must only consider the FlexCAN to be in Freeze mode when both the request and acknowledge conditions are satisfied.

When Freeze mode is requested, the FlexCAN performs the following operations:

- Waits to be in either the Intermission, Passive Error, Bus Off, or Idle state.
- Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. A pending move-in does not prevent going to Freeze mode.
- Ignores the Rx input pin and drives the Tx pin as recessive.
- Stops the prescaler, thus halting all CAN protocol activities.
- Grants write access to the Error Counters Register, which is read-only in other modes.
- Sets the NOTRDY and FRZACK bits in CAN_MCR.

After requesting Freeze mode, the user must wait for the CAN_MCR.FRZACK bit to be asserted before executing any other action. Otherwise, the FlexCAN may operate in an unpredictable way. In Freeze mode, all memory mapped registers are accessible, except for the CAN_CTRL1.CLKSRC bit, which can be read but cannot be written.

Exiting Freeze mode is done in one of the following ways:

- CPU negates the CAN_MCR.FRZ bit.

Yuntu Microelectronics

- The chip is removed from Debug Mode and/or the HALT bit is negated.

The CAN_MCR.FRZACK bit is negated after the PE recognizes the negation of the freeze request. When out of Freeze mode, the FlexCAN tries to re-synchronize to the CAN bus by waiting for 11 consecutive recessive bits.

17.4.12.2 Module Disable Mode

Module Disable is a Low-Power mode normally used to temporarily disable a complete FlexCAN block, with no power consumption. Module Disable is requested by the CPU through the assertion of the CAN_MCR.MDIS bit, and the acknowledged through the assertion by the FlexCAN of the CAN_MCR.LPMACK bit. The CPU must only consider the FlexCAN to be in Disable mode when both request and acknowledge conditions are satisfied.

If the FlexCAN is disabled during Freeze mode, it requests to disable the clocks to the PE and CHI submodules, sets the LPMACK bit, and negates the FRZACK bit.

It is not recommended to use Module Disable mode under Pretended Networking mode. Negate the MDIS bit and wait for LPMACK to negate before setting CAN_MCR.PNET_EN.

If disabled during transmission or reception, the FlexCAN module performs the following operations:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and then checks it to be recessive.
- Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive.
- Shuts down the clocks to the PE and CHI submodules.
- Sets the NOTRDY and LPMACK bits in CAN_MCR.

The Bus Interface Unit continues to operate, enabling the CPU to access memory mapped registers, except the Rx Mailboxes Global Mask Registers, the Rx Buffer 14 Mask Register, the Rx Buffer 15 Mask Register, and the Legacy Rx FIFO Global Mask Register. The Legacy Rx FIFO Information Register, the Message Buffers, the Rx Individual Mask Registers, and the reserved words within RAM may not be accessed when the FlexCAN is in Module Disable mode. Exiting from Module Disable mode is done by negating the MDIS bit by the CPU, which causes the FlexCAN to request to resume the clocks and to negate the LPMACK bit after the CAN PE recognizes the negation of the Module Disable mode request by the CPU.

17.4.12.3 Doze Mode

Doze mode is a system Low-Power mode in which the CPU bus is kept alive and a global Doze mode request is sent to all peripherals asking them to enter Low-Power mode. When Doze mode is globally requested, the CAN_MCR.DOZE bit needs to have been asserted previously for Doze mode to be triggered. The FlexCAN acknowledges the request by asserting the CAN_MCR.LPMACK bit. The CPU must only consider the FlexCAN to be in Doze mode when both request and acknowledge conditions are satisfied.

If Doze mode is triggered during Freeze mode, the FlexCAN requests to shut down the clocks to the PE and CHI submodules, sets the LPMACK bit and, negates the FRZACK bit.

If Doze mode is triggered during transmission or reception, the FlexCAN performs the following operations:

Yuntu Microelectronics

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and checks it to be recessive.
- Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive.
- Shuts down the clocks to the PE and CHI submodules.
- Sets the NOTRDY and LPMACK bits in CAN_MCR.

The Bus Interface Unit continues to operate, enabling the CPU to access memory mapped registers, except the Rx Mailboxes Global Mask Registers, the Rx Buffer 14 Mask Register, the Rx Buffer 15 Mask Register, and the Legacy Rx FIFO Global Mask Register. The Legacy Rx FIFO Information Register, the Message Buffers, the Rx Individual Mask Registers, and the reserved words within RAM may not be accessed when the FlexCAN is in Doze mode.

Exiting Doze mode is done in one of the following ways:

- CPU removing the Doze mode request.
- CPU negating the CAN_MCR.DOZE bit.
- Self Wake Up mechanism.

In the Self Wake Up mechanism, if the CAN_MCR.SLFWAK bit was set at the time that the FlexCAN entered Doze mode, then upon detection of a recessive-to-dominant transition on the CAN bus, the FlexCAN negates the DOZE bit, requests to resume its clocks, and negates LPMACK after the CAN PE recognizes the negation of the Doze mode request.

The FlexCAN also sets the CAN_ESR.WAKINT bit and, if enabled by the CAN_MCR.WAKMSK bit, generates a Wake Up interrupt to the CPU. The FlexCAN then waits for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, the FlexCAN does not receive the frame that woke it up. Table: ”CAN wake-up from DOZE mode” details the effect of SLFWAK and WAKMSK upon wake-up from Doze mode.

Table 17.99: CAN Wake-up from DOZE Mode

SLFWAK	WAKINT	WAKMSK	FlexCAN Clocks Enabled	Wake-Up Interrupt Generated
0	–	–	No	No
0	–	–	No	No
1	0	0	No	No
1	0	1	No	No
1	1	0	Yes	No
1	1	1	Yes	Yes

The sensitivity to CAN bus activity can be modified by applying a low-pass filter function to the Rx CAN input line while in Doze mode. See the CAN_MCR.WAKSRC bit description.

This feature can be used to protect FlexCAN from waking up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic interference in noisy environments.

17.4.12.4 Deepsleep Mode

Deepsleep mode is a system Low-Power mode in which all chip clocks can be stopped for maximum power savings. Deepsleep mode is globally requested by the CPU and the acknowledged by the FlexCAN through the assertion of a Deepsleep Acknowledgement signal (ipg_deepsleep_ack). The CPU must only

consider the FlexCAN to be in Deepsleep mode when both request and acknowledge conditions are satisfied.

If the FlexCAN receives the global Deepsleep mode request during Freeze mode, it sets the LPMACK bit, negates the FRZACK bit, and then sends the Deepsleep Acknowledge signal to the CPU, in order to shut down the clocks globally.

If Deepsleep mode is requested during transmission or reception, the FlexCAN performs the following operations:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and checks it to be recessive.
- Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive.
- Sets the NOTRDY and LPMACK bits in CAN_MCR.
- Sends a Deepsleep Acknowledge signal to the CPU, so that it can shut down the clocks globally.

Deepsleep mode is exited when the CPU resumes the clocks and removes the Deepsleep mode request. This can be as a result of the Self Wake Up mechanism.

In the Self Wake Up mechanism, if the CAN_MCR.SLFWAK bit was set at the time that the FlexCAN entered Deepsleep mode, then upon detection of a recessive-to-dominant transition on the CAN bus, the FlexCAN sets the CAN_ESR1.WAKINT bit and, if enabled by the CAN_MCR.WAKMSK bit, generates a Wake Up interrupt to the CPU. Upon receiving the interrupt, the CPU should resume the clocks and remove the Deepsleep mode request. The FlexCAN then waits for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, the FlexCAN does not receive the frame that woke it up. Table: ”Wake-up From Deepsleep Mode” details the effect of SLFWAK and WAKMSK upon wake-up from Deepsleep mode. Note that Wake-Up from Deepsleep mode only works when both bits are asserted.

After the CAN PE recognizes the negation of the Deepsleep mode request, the FlexCAN negates the LPMACK bit. The FlexCAN then waits for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, the FlexCAN does not receive the frame that woke it up.

Table 17.100: CAN Wake-up from Deepsleep Mode

SLFWAK	WAKINT	WAKMSK	Chip Clocks Enabled	Wake-Up Interrupt Generated
0	–	–	No	No
0	–	–	No	No
1	0	0	No	No
1	0	1	No	No
1	1	0	No	No
1	1	1	Yes	Yes

The sensitivity to CAN bus activity can be modified by applying a low-pass filter function to the Rx CAN input line while in Deepsleep mode. See the CAN_MCR.WAKSRC bit description.

This feature can be used to protect FlexCAN from waking up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic interference in noisy environments.

Yuntu Microelectronics

17.4.12.5 Pretended Networking Mode

Pretended Networking is a special Low-Power mode used to receive wake up messages with Low-Power consumption. Pretended Networking mode can be selected to operate together with Doze mode or Deepsleep mode. Before entering in one of these Low-Power modes, the CAN_MCR.PNET_EN bit must be asserted. Once in Low-Power mode, CHI submodule is shut down and CAN PE submodule is kept active so that the Rx process is still active to filter incoming messages (see ”Receive Process under Pretended Networking Mode”) as defined by the configuration registers (see ”CAN CTRL1 PN Register (CAN_CTRL1_PN)”). Upon detecting a Wake-Up event, a Wake Up interrupt is issued to the system.

To enter Pretended Networking mode, the FlexCAN must be in Normal mode (not in Freeze or Module Disable mode). Under Pretended Networking mode, the FlexCAN keeps itself synchronized with the CAN bus in Doze or Deepsleep mode. Then, when either Doze or Deepsleep mode is requested, FlexCAN performs the following steps:

- Waits to be in Idle state, or else waits for the third bit of Intermission, and then checks it to be recessive.
- Sets the LPMACK bit in CAN_MCR.
- Requests the shutdown of the CHI submodule clock, while keeping the PE submodule clock active.

The FlexCAN can exit Pretended Networking mode in the following ways:

- The CPU removing the Doze mode request.
- The CPU negating the CAN_MCR.DOZE bit.
- The CPU removing the Deepsleep mode request.

The FlexCAN will wait until Bus Idle or third bit of the Intermission state to negate the CAN_MCR.LPMACK bit.

The above exit methods can be triggered either by the FlexCAN action upon detecting a wake up event and issuing the respective interrupt, or by the CPU itself upon being waken up by another method. As a consequence, the FlexCAN will wait until the Bus Idle state or until the third bit of the Intermission state to negate the CAN_MCR.LPMACK bit and resume Normal mode. This procedure ensures that the FlexCAN will be synchronized to the CAN bus after exiting Pretended Networking mode. The CPU must wait for the CAN_MCR.LPMACK bit to be negated before performing any access to the FlexCAN.

When the CAN_MCR.PNET_EN bit is asserted, the CPU must disable the Self Wake Up feature by negating the CAN_MCR.SLFWAK bit.

17.4.13 Interrupts

The FlexCAN module has many interrupt sources: interrupts due to MBs and interrupts due to the OR’d interrupts from MBs, Bus Off, Bus Off Done, Error, Error Fast (errors detected in the data phase of CAN FD format messages with the BRS bit set), Wake Up, Wake Up Match, Wake Up Timeout, Tx Warning, and Rx Warning.

Each one of the MBs can be an interrupt source if its corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each of the buffers has an assigned flag bit in the CAN_IFLAG registers. The bit is set when the corresponding buffer completes a successful transfer and is cleared when the CPU writes ‘1’ to it (unless another interrupt is generated at the same time).

NOTE: It must be guaranteed that the CPU clears only the bit causing the current interrupt. For this reason, bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions

Yuntu Microelectronics

may cause accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

If the Legacy Rx FIFO is enabled (CAN_MCR.RFEN = 1) and DMA is disabled (CAN_MCR.DMA = 0), the interrupts corresponding to MBs 0 to 7 have different meanings:

- Bit 7 of the CAN_IFLAG1 register becomes the “Legacy Rx FIFO Overflow” flag.
- Bit 6 becomes the “Legacy Rx FIFO Warning” flag.
- Bit 5 becomes the “Frames Available in Legacy Rx FIFO” flag.
- Bit 4-0 are unused

See ”Interrupt Flags 1 Register (CAN_IFLAG1)” for more information.

If both Legacy Rx FIFO and DMA are enabled (CAN_MCR.RFEN =1 and CAN_MCR.DMA = 1), the FlexCAN does not generate any Legacy Rx FIFO interrupt. Bit 5 of the CAN_IFLAG1 register still indicates “Frames Available in Legacy Rx FIFO” and generates a DMA request. Bits 7, 6, and 4–0 are unused.

NOTE: The Legacy Rx Rx FIFO cannot be enabled when CAN FD feature is enabled.

For a combined interrupt where multiple MB interrupt sources are OR’d together, the interrupt is generated when any of the associated MBs (or FIFO, if applicable) generates an interrupt. In this case, the CPU must read the CAN_IFLAG registers to determine which MB or FIFO source caused the interrupt.

The interrupt sources for Bus Off, Bus Off Done, Error, Error Fast, Wake Up, Tx Warning and Rx Warning generate interrupts like the MB interrupt sources, and can be read from CAN_ESR1 register. The Bus Off, Error, Tx Warning, and Rx Warning interrupt mask bits are located in the CAN_CTRL1 register; the Wake Up interrupt mask bit is located in CAN_MCR.

The interrupt sources for Pretended Networking (Wake up by Match Flag and Wake Up by Timeout Flag) can be read in the CAN_WU_MTC register and the respective interrupts masks bits are located in CAN_CTRL1_PN register.

17.4.14 Bus Interface

CPU accesses to FlexCAN registers are subject to the following rules:

- Unrestricted read and write access to supervisor registers (registers identified with S/U in table: ”CAN register access and reset information” in Supervisor Mode or with S only) results in an access error.
- Read and write access to implemented reserved address space results in an access error.
- Write access to positions whose bits are all currently read-only results in an access error. If at least one of the bits is not read-only then no access error is issued. Write permission to positions or some of their bits can change depending on the mode of operation or transitory state. Refer to the register and bit descriptions for details.
- Read and write access to unimplemented address space results in an access error.
- Read and write access to RAM located positions during Low-Power modes results in an access error.
- If CAN_MCR.MAXMB is programmed with a value smaller than the available number of MBs, then the unused memory space can be used as general purpose RAM space. Reserved words within RAM cannot be used. As an example, suppose FlexCAN’s RAM can support up to 16 MBs, CAN_CTRL2.RFFN is 0x0, and CAN_MCR.MAXMB is programmed with zero. The maximum number of MBs in this case becomes one. The RAM starts at 0x0080, and the space from 0x0080 to 0x008F is used by the one MB. The memory space from 0x0090 to 0x017F is available. The space

Yuntu Microelectronics

between 0x0180 and 0x087F is reserved. The space from 0x0880 to 0x0883 is used by the one Individual Mask and the available memory in the Mask Registers space would be from 0x0884 to 0x08BF. From 0x08C0 through 0x09DF, there are reserved words for internal use which cannot be used as general purpose RAM. As a general rule, free memory space for general purpose depends only on MAXMB.

17.4.15 Detection and Correction of Memory Errors

The FlexCAN supports detection and correction of errors in memory read accesses. Each byte of FlexCAN memory is associated with 5 parity bits, which ensures a Hamming distance of 4. The error correction mechanism ensures that in this 13-bit word, errors in one bit can be corrected (correctable errors) and errors in 2 bits can be detected but not corrected (non-correctable errors). Errors in more than 2 bits may not be detected. In case of non-correctable errors, the corrupted data is not changed by the error correction logic.

When a read access is performed, the parity bits are used to calculate a syndrome, which indicates the error in each byte.

The FlexCAN detects a non-correctable error in the event either an all-zeros or an all-ones read occurs. See ”Error Report Syndrome Register (CAN_RERRSYNR)”.

Memory errors are indicated to the host through CAN_ERRSR and bus transfer errors, and reported through CAN_RERRAR, CAN_RERRDR, and CAN_RERRSYNR.

The error detection and correction mechanism is activated by the ECCDIS bit in the Memory Error Control Register (CAN_MECR). When error detection and correction is disabled, updates on indications and reporting registers are stopped, but the parity bits are still calculated and written along with data in memory write operations to ensure that memory has consistent parity bits associated with the data.

NOTE: All FlexCAN memory must be initialized before starting its operation in order to have the parity bits in memory properly updated. Refer to ”RAM Initialization Addresses” for the peripheral bus addresses that must be written to initialize the FlexCAN memory for your FlexCAN hardware configuration. The WRMFRZ bit in the Control 2 Register (CAN_CTRL2) grants write access to all memory positions that require initialization. The CAN_MCR.RFEN bit must not be set during memory initialization.

To avoid accidentally changing the critical error correction configuration, the following protocol must be used to enable the update of the Memory Error Control Register (CAN_MECR):

- By default, the ECRWRE bit in Control 2 Register (CAN_CTRL2) is 0 and the ECRWRDIS bit in Memory Error Control Register (CAN_MECR) is 1.
- Set CAN_CTRL2.ECRWRE.
- Clear CAN_MECR.ECRWRDIS.
- All writes to CAN_MECR must keep ECRWRDIS cleared.
- After configuration is done, lock CAN_MECR by either setting ECRWRDIS or clearing ECRWRE.

17.4.15.1 Sources of Memory Access

The FlexCAN memory can be accessed by two major sources (or requestors):

- Host (CPU): The largest word accessed is 32 bits wide.
- FlexCAN internals processes (Rx Matching, Tx Arbitration, Move-in on reception, Move-out on transmission): The largest word accessed is 64 bits wide.

The way that non-correctable errors are indicated and reported depends on the source of access.

Yuntu Microelectronics

17.4.15.2 Error Indication

Memory errors are indicated by flags HANCEIF, FANCEIF, and CEIF in CAN_ERRSR. Noncorrectable errors detected in memory reads requested by the host are indicated separately from the errors detected in requests by FlexCAN internal processes. The FlexCAN makes no distinction of the source of the access when correctable errors are detected. There are 3 independent flags for these 3 cases, and each flag will raise an interrupt unless it is masked by mask bits in CAN_MECR. If both non-correctable and correctable errors are found in different bytes in the same read operation, both flags are set.

A non-correctable error detected in host access is also indicated as a bus transfer error. A bus wait request may be asserted to extend the memory transaction to the moment the report registers are updated. This indication cannot be masked. If the flag bit CAN_ERRSR.HANCEIF is not masked, the same non-correctable error will raise a bus transfer error and an interrupt request.

Each indication flag has one Overrun flag in CAN_ERRSR. The Overrun flags do not request interrupts. Overrun flags for non-correctable errors indicate that other errors of the same nature were detected after current error being treated, while overrun flags for correctable errors indicate that other errors of the same nature were detected before the current error being treated. The recommended handling sequence for error indication is:

- Get error report information from report registers.
- Use this information to take proper measures in the application.
- Clear the HANCEIF, FANCEIF, CEIF flags.
- If the Overrun flag is active:
  - Alert application that at least one error could not be managed.
  - Clear the Overrun flag.

The FlexCAN internal processes can access memory in transactions larger than 32 bits. For the indication, this kind of access is considered a consecutive sequence of 32-bit accesses.

If errors are found in 2 or more 32-bit words, the Interrupt and Overrun flags are set simultaneously.

17.4.15.3 Error Reporting

The report registers CAN_RERRAR, CAN_RERRDR, and CAN_RERRSYNR provide detailed information about the address read, raw data, and syndrome read with error and indicated by the flags described in ”Error Indication”. The address, data, and syndrome registers are updated simultaneously along with the error flags, according to these rules:

- If either of the 2 non-correctable error flags is currently set, the report registers are not updated (the previous non-correctable error reporting is preserved).
- Otherwise (either no error flag is currently set or only the correctable error flag is currently set), the report registers are updated according to the new error, or according to the most severe of new errors if non-correctable and correctable errors are simultaneously detected.

Reporting of errors detected in accesses larger than 32 bits follows the rules described in Error Indication and in ”Error Report Address Register (CAN_RERRAR)”.

The address reported in CAN_RERRAR and defined in CAN_ERRIAR are not the same as listed in the module memory map. The relation between the reported addresses and the respective ones in the module memory map is shown in ”Error Injection Address Register (CAN_ERRIAR)”.

Addresses reported when reading memory portions organized as FIFOs, such as the Legacy Rx FIFO Structure, Enhanced Rx FIFO Structure, and the Legacy Rx FIFO Information Register (CAN_RXFIR), refer to the address of the specific entry accessed in the FIFO, not to the FIFO base address.

Yuntu Microelectronics

To ensure coherence of the error report registers, it is necessary to turn off the report update by setting the CAN_MECR.RERRDIS bit before reading the report registers.

17.4.15.4 Response to Errors

Correctable errors have no consequence on FlexCAN operation because affected data is corrected before its use by the host or FlexCAN internal processes.

For host-initiated reads, a non-correctable error may affect the host, but does not affect FlexCAN operation.

Non-correctable errors detected on memory reads requested by the FlexCAN internal processes may result in incorrect operation depending on the state of the CAN_MECR.NCEFAFRZ bit, as follows:

- During reception (either matching or move-in processes), when a non-correctable error occurs, an incorrect destination may be selected to store the incoming frame, a corrupted frame may be stored in the correct destination, or both. If NCEFAFRZ is set, the FlexCAN stops operation automatically and enters Freeze mode to prevent corrupted data from being treated as valid by FlexCAN internal processes. If NCEFAFRZ is negated, the FlexCAN continues working and a corrupted frame is received.
- During the arbitration process, when a non-correctable error occurs, either a nonhighest priority Tx message buffer may be mistakenly selected for transmission or its data may be corrupted. If NCEFAFRZ is set, the FlexCAN stops operation automatically and enters Freeze mode before starting the move-out. If NCEFAFRZ is negated, the FlexCAN proceeds to move-out with a corrupted frame that will be transmitted on the CAN bus.
- During the move-out process, when a non-correctable error occurs, a corrupted frame is copied from the selected Tx MB that won the arbitration to the Tx SMB for transmission. If NCEFAFRZ is set, the FlexCAN stops operation automatically and enters Freeze mode before starting the transmission. If NCEFAFRZ is negated, the corrupted frame is transferred from the Tx SMB to the PE submodule and is transmitted on the CAN bus.
- A non-correctable error can also be detected beyond the move-out process, when Tx data is read from Tx SMB (buffer located in RAM) to be transferred to the PE submodule for transmission. In this case, a frame with corrupted ID and/or data is transmitted on the CAN bus. To prevent the frame from being successfully received by the external nodes, the FlexCAN inverts all bits in the CRC field (CRC sequence plus CRC Delimiter), and transmits an Error Flag just after CRC Delimiter as a result of self-detecting a Bit1 Error and a Form Error due to the CRC field inversion. If NCEFAFRZ is set, the FlexCAN stops operation automatically and enters Freeze mode just after the Error Frame. If NCEFAFRZ is negated, the FlexCAN may attempt to re-transmit the same frame, as long as no other higher priority Tx MB is subsequently configured for transmission. In the event the non-correctable error persists, the FlexCAN eventually reaches the Bus Off state because of consecutive error detections. The CAN_ECR.TXERRCNT field is updated every time the FlexCAN inverts the CRC field causing errors as described above.

When NCEFAFRZ is set and the FlexCAN enters Freeze mode, only the CPU can cause then FlexCAN to exit Freeze mode and resume Normal mode. Assertion of CAN_MECR.NCEFAFRZ is the only way to prevent corrupted frames from being transmitted on the CAN bus up to the move-out internal process.

The error report registers can provide information to the application for customized management of these situations.

Yuntu Microelectronics

17.4.15.5 Error Injection

The error injection registers CAN_ERRIAR, CAN_ERRIDPR, and CAN_ERRIPPR are used to inject errors in memory reads in order to force errors and consequently update the indication and reporting registers. The relation between the error injection addresses and the respective ones in the module memory map is shown in ”Error Injection Address Register (CAN_ERRIAR)”.

The injection is done by flipping the data and parity bits corresponding to the bits set to 1 in CAN_ERRIDPR and CAN_ERRIPPR. Injection can be selected specifically for memory accesses requested by host or by FlexCAN internal processes.

In case of accesses larger than 32 bits, the CAN_MECR.EXTERRIE bit extends the injection pattern, replicating it in 32-bit words to fill the width of the access.

NOTE: It is very unlikely, but error injection may correct a bit with error. This will not raise the error flags and reports as expected.

To ensure coherence among error injection registers and avoid spurious error injections, the HAERRIE and FAERRIE bits in CAN_MECR must be cleared when configuring the memory injection registers.

17.5 Initialization / Application Information

The FlexCAN module may be reset in three ways:

- Chip level hard reset, which resets all memory mapped registers asynchronously.
- SOFTRST bit in CAN_MCR, which resets some of the memory mapped registers synchronously. See ”CAN register access and reset information” to see what registers are affected by soft reset.
- Chip level soft reset, which has the same effect as the SOFTRST bit in CAN_MCR, see chapter IPC.

Soft reset is synchronous and has to follow an internal request/acknowledge procedure across clock domains. Therefore, it may take some time to fully propagate its effects. The CAN_MCR.SOFTRST bit remains asserted while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft reset cannot be applied while clocks are shut down in a Low-Power mode. The Low-Power mode should be exited and the clocks resumed before applying soft reset.

The clock source should be selected when the FlexCAN is in Module Disable mode (see CAN_CTRL1.CLKSRC bit). After the clock source is selected and the module is enabled (CAN_MCR.MDIS bit negated), the FlexCAN automatically enters Freeze mode. In Freeze mode, the FlexCAN is un-synchronized to the CAN bus, the HALT and FRZ bits in CAN_MCR are set, the internal state machines are disabled, and the FRZACK and NOTRDY bits in CAN_MCR are set. The Tx pin is in recessive state and the FlexCAN does not initiate any transmission or reception of CAN frames. Note that the MBs and the Rx Individual Mask Registers are not affected by reset, so they are not automatically initialized.

17.5.1 RAM Initialization

The FlexCAN RAM should be initialized prior to use to avoid ‘X’ propagation in simulation.

When ECC is enabled for the FlexCAN RAM (configuration parameter ECC_EN = 1), all FlexCAN memory must be initialized before starting its operation so that the parity bits in memory are properly updated.

NOTE: The CAN_MCR.RFEN bit must not be set during memory initialization.

To initialize the FlexCAN RAM:

Yuntu Microelectronics

1. Set CAN_CTRL2.WRMFRZ to enable write access to all memory positions that require initialization.
2. Write the addresses needed to initialize the complete FlexCAN RAM according to your selected FlexCAN hardware configuration, as described in "RAM Initialization Addresses".

17.5.2 FlexCAN Initialization Sequence

For any configuration change/initialization, it is required that the FlexCAN is put into Freeze mode (see "Freeze Mode"). The following steps are a generic initialization sequence applicable to the FlexCAN module:

1. Initialize the FlexCAN RAM as described in "RAM Initialization"RAM Initialization.
2. Initialize the Module Configuration Register (CAN_MCR):
   - Enable the individual filtering per MB and reception queue features by setting IRMQ.
   - Enable the warning interrupts by setting WRNEN.
   - If required, disable frame self reception by setting SRXDIS.
   - Enable the Legacy Rx FIFO by setting RFEN or the Enhanced Rx FIFO by setting ERFEN.
   - If the Legacy Rx FIFO or Enhanced Rx FIFO is enabled and DMA is required, set the DMA bit.
   - If Pretended Networking mode is required, set PNET_EN.
   - Enable the abort mechanism by setting AEN.
   - Enable the local priority feature by setting LPRIOEN.
3. Initialize the Control 1 Register (CAN_CTRL1) and optionally the CAN Bit Timing Register (CAN_CBT). Initialize also the CAN FD CAN Bit Timing Register (CAN_FDCBT):
   - Determine the bit timing parameters: PROPSEG, PSEG1, PSEG2, and RJW.
   - Optionally determine the extended bit timing parameters: EPROPSEG, EPSEG1, EPSEG2, and ERJW.
   - Determine the CAN FD bit timing parameters: FPROPSEG, FPSEG1, FPSEG2, and FRJW.
   - Determine the bit rate by programming the PRESDIV field and optionally the EPRESDIV field.
   - Determine the CAN FD bit rate by programming the FPRESDIV field.
   - Determine the internal arbitration mode (LBUF bit).
4. Initialize the Message Buffers:
   - The C/S word of all MBs must be initialized.
   - If either the Legacy Rx FIFO or Enhanced Rx FIFO was enabled, the corresponding ID filter table must be initialized.
   - Other entries in each MB should be initialized as required.
5. Initialize the Rx Individual Mask Registers (CAN_RXIMRn).
6. Set required interrupt mask bits in the CAN_IMASKn registers (for all MB interrupts), in CAN_MCR for Wake Up interrupt, and in CAN_CTRL1/CAN_CTRL2 for Bus Off and Error interrupts.
7. Negate the HALT bit in CAN_MCR.

After the last step listed above, the FlexCAN attempts to synchronize to the CAN bus.

17.6 FlexCAN Ram Usage And Initialization

The FlexCAN Controller requires an externally connected RAM for storage of message buffers and register data for memory-mapped register that are implemented in external RAM. As shown in below Figure, the RAM is accessed:

Yuntu Microelectronics

- By the host CPU (through the IPS bus) when writing/reading message buffers and registers that are implemented in RAM.
- By FlexCAN internal processes when accessing RAM data that is shared with the CPU and for storage of data such as SMBs, which are used only by FlexCAN internal processes and not directly accessible by the CPU when the FlexCAN is operating in Normal mode.

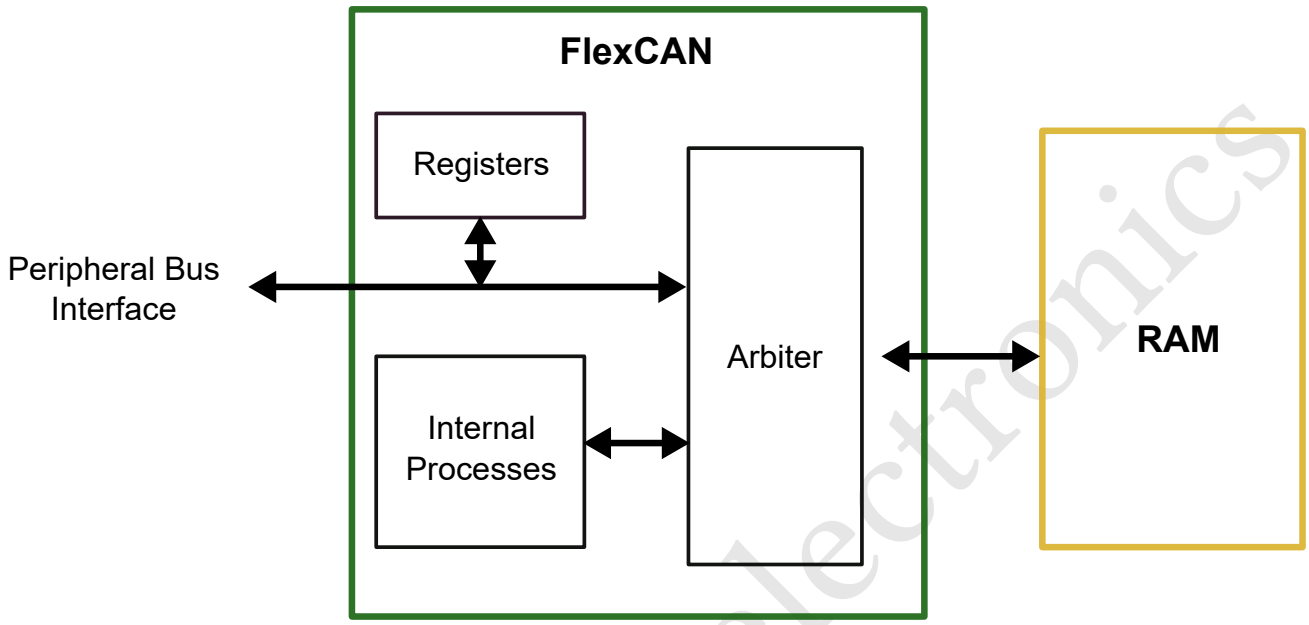


Figure 17.26: Sharing of RAM between CPU and CAN Internal Processes

17.6.1 RAM Size

The size required for the RAM depends on FlexCAN configuration. The data bus width is:

- 64 bits when ECC_EN = 0 (ECC not supported)
- 104 bits when ECC_EN = 1 (ECC is supported)

The number of addresses required and, therefore, the required RAM address bus width depends on the following configuration parameters, as detailed in the FlexCAN Controller Integration Guide:

- NUMBER_OF_MB – Number of message buffers supported
- IRMQ_EN – Support for Rx Individual Mask Registers
- FD_EN – Support for CAN FD
- ENHANCE_MB_MEM – Support for additional message buffer memory space
- TIME_STAMP_EN – Support for high-resolution time stamp
- ERX_FIFO_EN – Support for Enhanced Rx FIFO
- ERX_FIFO_SIZE – Enhanced Rx FIFO size (number of messages that can be stored in the Enhanced Rx FIFO)
- ERX_FIFO_FLT_ELEM – Number of Enhanced Rx FIFO filter elements

17.6.2 RAM Usage

As shown below 5 Figures, the RAM stores the following groups/regions of data:

- Message buffers – 16 bytes for each configured message buffer.

Yuntu Microelectronics

- Rx Individual Mask Registers (CAN_RXIMRn) – 4 bytes per message buffer when configuration parameter IRMQ_EN = 1.
- Rx Mask Registers CAN_RXMGMASK, CAN_RX14MASK, CAN_RX15MASK, CAN_RXFGMASK – 4 bytes per register.
- Rx FIFO Information Register – 24 bytes accessed as FIFO through CAN_RXFIR during normal operation.
- Classic (non-CAN-FD) mode SMBs – One Tx SMB and two Rx SMBs used when CAN_MCR.FDEN = 0. Each SMB is 16 bytes.
- CAN-FD mode SMBs – One Tx SMB and two Rx SMBs used when CAN_MCR.FDEN = 1. Each SMB is 72 bytes.
- Enhanced Rx FIFO – When configuration parameter ERX_FIFO_EN = 1, there is an 80-byte (FD_EN = 1) or 24-byte (FD_EN = 0) region for each message that can be stored in the Enhanced Rx FIFO, as determined by configuration parameter ERX_FIFO_SIZE.
- Enhanced Rx FIFO filter elements – When configuration parameter ERX_FIFO_EN = 1, there is a 4-byte region for each Enhanced Rx FIFO filter element, as determined by configuration parameter ERX_FIFO_FLT_ELEM.

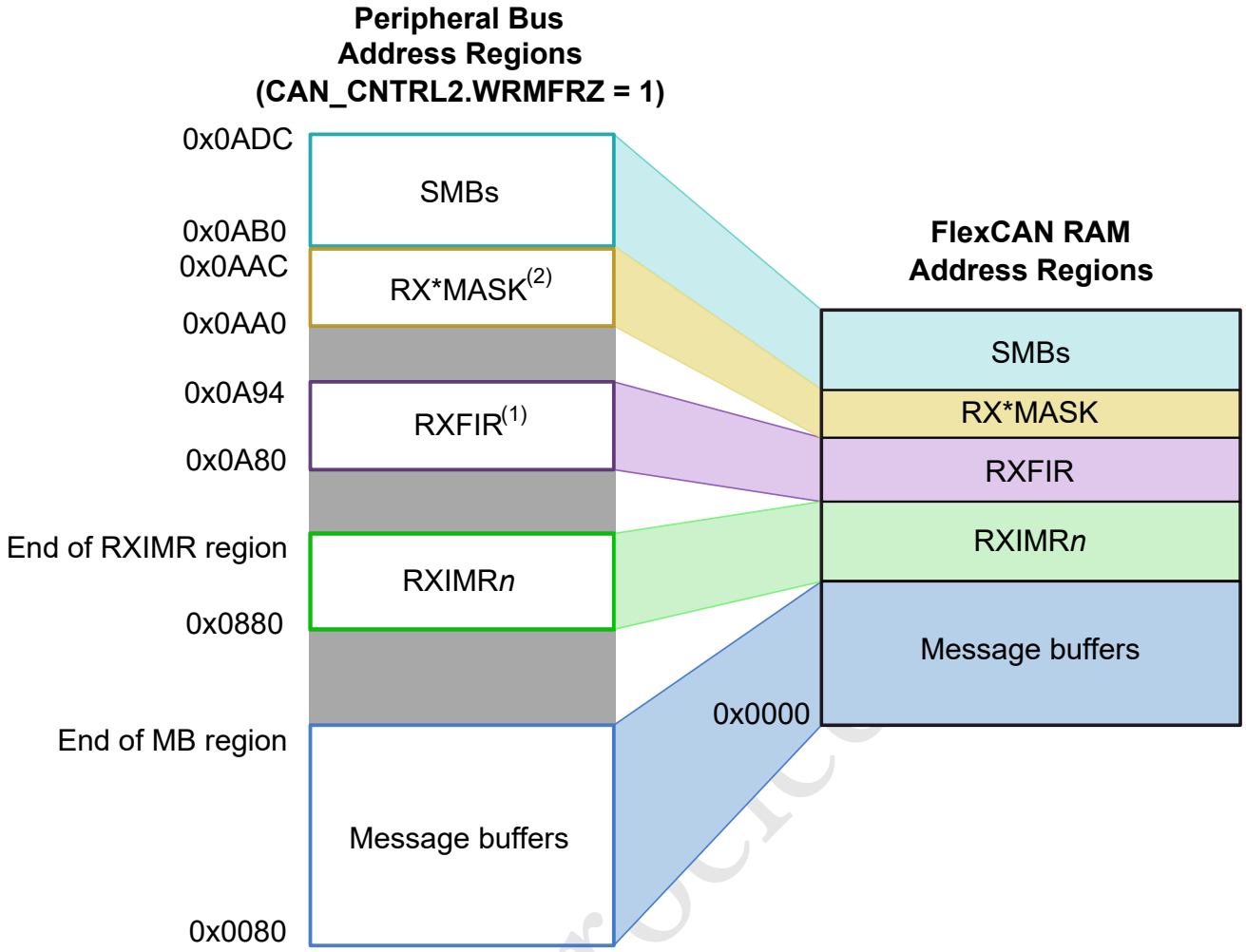
From the peripheral (IPS) bus, the base address of each region is the same regardless of the selected FlexCAN configuration, although the sizes of the message buffer and CAN_RXIMRn regions depend on configuration parameter NUMBER_OF_MB.

On the RAM interface, the message buffer region begins at address 0x0000. The addresses of regions above the message buffer region scale according to NUMBER_OF_MB and IRMQ_EN, in order to fully utilize the smallest possible RAM required for the selected configuration.

Not all RAM locations are accessible from the peripheral bus during normal operation. Therefore, when ECC_EN = 1 (ECC is enabled), the CAN_CNTRL2.WRMFRZ bit can be set to enable write access to all RAM locations so that the ECC bits can be initialized.

Below figure shows the peripheral bus to RAM address mapping when CAN-FD is not supported (configuration parameter FD_EN = 0).

Yuntu Microelectronics



1.Accessed as FIFO at 0x004C in Normal mode.

2.Accessed at addresses 0x0010,0x0014,0x0018,0x0048 in Normal mode.

Figure 17.27: Mapping of Peripheral Bus Address to CAN RAM When CAN FD Is Not Supported $(FD_EN = 0)$

Below figure shows the peripheral bus to RAM address mapping when CAN-FD is supported (configuration parameter FD_EN = 1) and enabled (CAN_MCR.FDEN = 1), and ENHANCE_MB_MEM = 0.

Yuntu Microelectronics

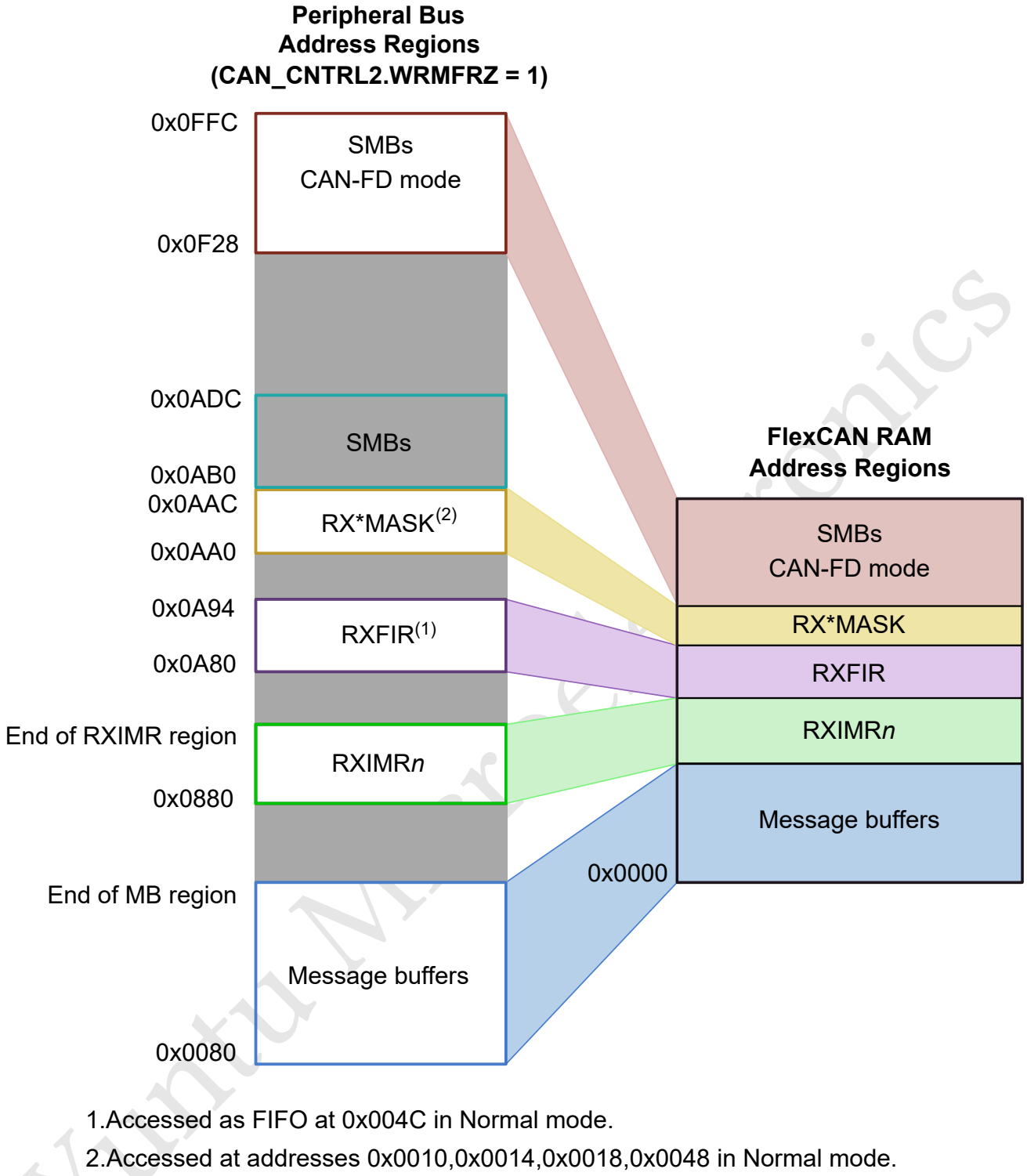


Figure 17.28: Mapping of Peripheral Bus Address to CAN RAM When CAN FD Is Supported and Enabled $(FD_EN = 1, CAN_MCR.FDEN = 1)$

Below figure shows the peripheral bus to RAM address mapping when CAN-FD is supported (configuration parameter FD_EN = 1) and disabled (CAN_MCR.FDEN = 0). The implemented RAM size supports the larger SMBs used in CAN-FD mode. However, since CAN-FD mode is not enabled and only the smaller non-FD mode SMBs are used, there is unused RAM space.

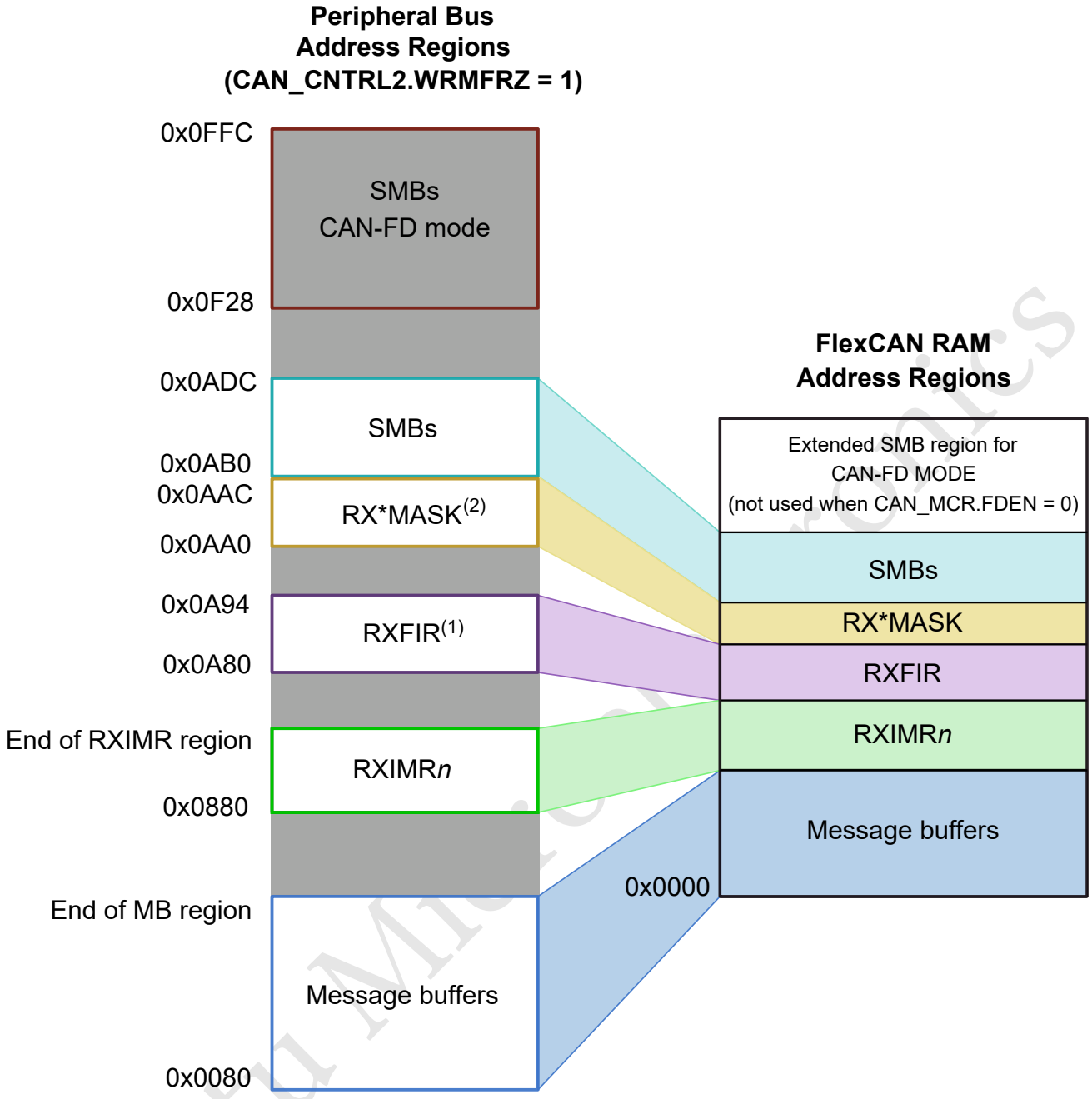


Figure 17.29: Mapping of Peripheral Bus Address to CAN RAM When CAN FD Is Supported But Not Enabled $(FD_EN = 1, CAN_MCR.FDEN = 0)$

Below figure shows the peripheral bus to RAM address mapping when the enhanced MB memory feature is implemented (ENHANCE_MB_MEM = 1). Enhanced MB memory can only be implemented when CAN-FD is supported (FD_EN = 1) and NUMBER_OF_MB = 128. Therefore, the RAM address region for enhanced MB memory is always at RAM addresses 0x0800–0x0FFC and maps to peripheral bus addresses 0x1000–0x17FC.

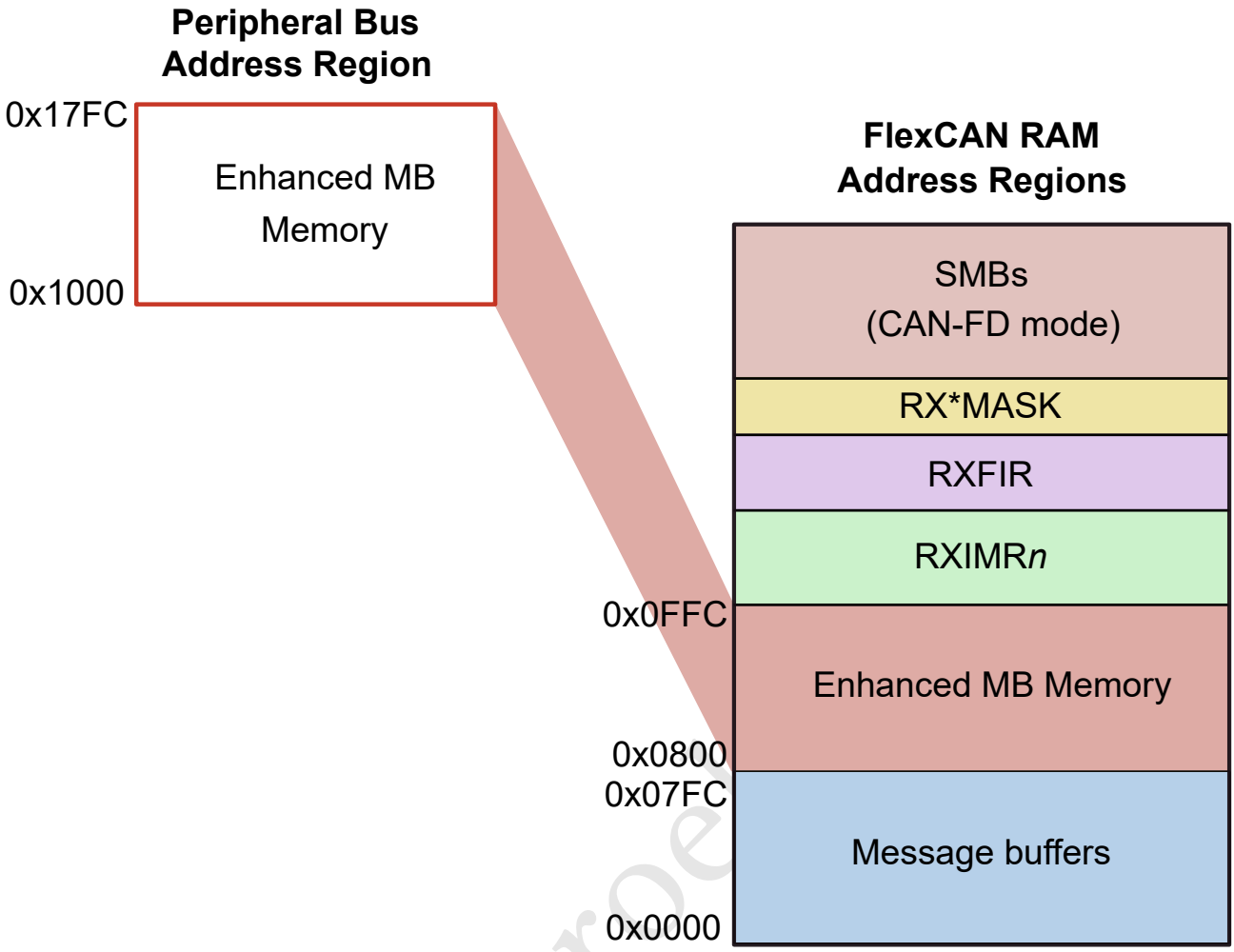


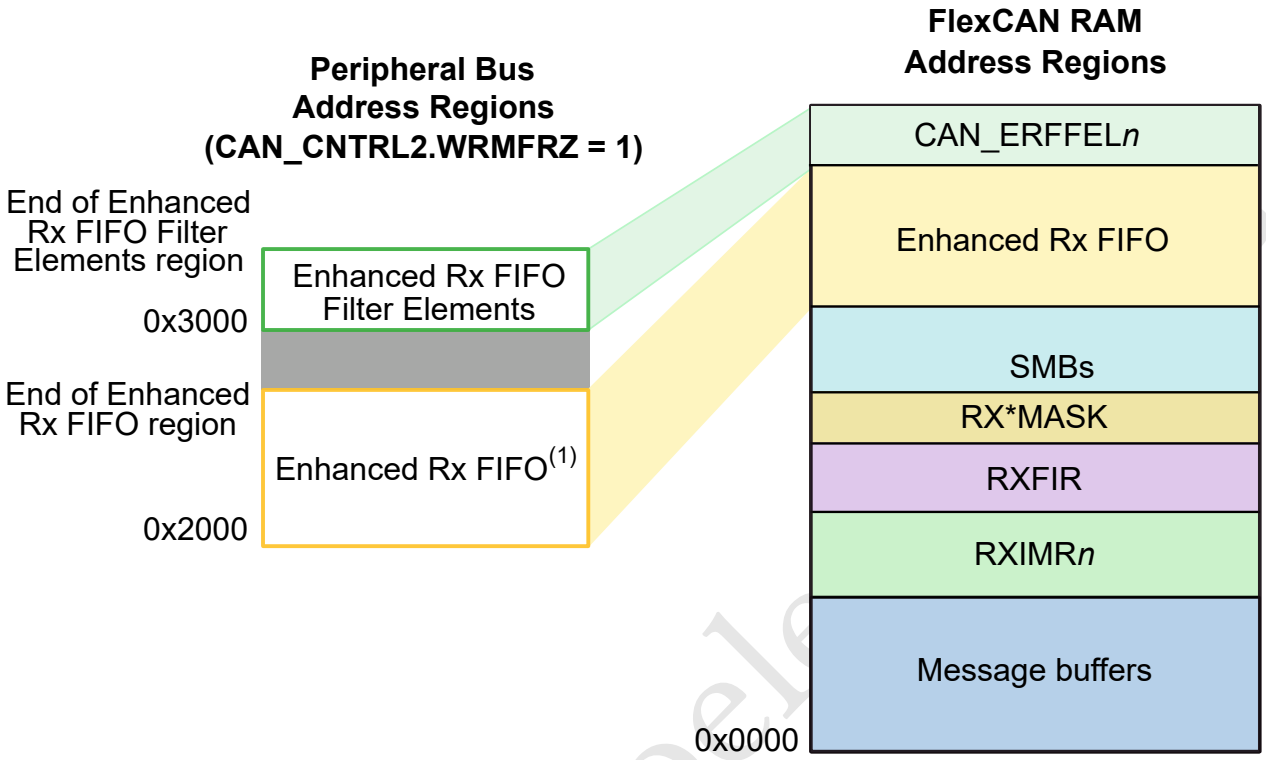
Figure 17.30: Mapping of Peripheral Bus Address for Enhanced MB Memory Region

Below figure shows peripheral bus to RAM address mapping for the Enhanced Rx FIFO and Enhanced Rx FIFO filter elements regions, which are present when ERX_FIFO_EN = 1:

- The size of the Enhanced Rx FIFO region depends on the value of configuration parameter ERX_FIFO_SIZE, which is the number of messages that can be stored in the Enhanced Rx FIFO. There are 80 bytes allocated for each message:
  - In Freeze mode, the entire Enhanced Rx FIFO region is accessible from the peripheral bus for FlexCAN RAM initialization. The start address is 0x2000. The end address (word-aligned) is 0x2000 + (80 x ERX_FIFO_SIZE) - 4. For example, if ERX_FIFO_SIZE = 2, the range of the Enhanced Rx FIFO region is 0x2000–0x209C. If ERX_FIFO_SIZE = 32, the range of the Enhanced Rx FIFO region is 0x2000–0x29FC.
  - In Normal mode, only oldest unread entry in the Enhanced Rx FIFO is accessible from the peripheral bus, at addresses 0x2000–0x204C.
- The size of the Enhanced Rx FIFO filter elements region depends on the value of configuration parameter ERX_FIFO_FLT_ELEM, which is the Enhanced Rx FIFO filter elements implemented. There are 4 bytes allocated for each filter element.

  The Enhanced Rx FIFO filter elements region start address is 0x3000. The end address (word-aligned) is 0x3000 + (4 x ERX_FIFO_FLT_ELEM) - 4. For example, if ERX_FIFO_FLT_ELEM

= 2, the range of the Enhanced Rx FIFO filter elements region is 0x3000–0x3004. If ERX_FIFO_FLT_ELEM = 128, the range of the Enhanced Rx FIFO region is 0x3000–0x31FC.



1. Oldest unread Enhanced Rx FIFO entry is accessible at 0x2000-0x204C in Normal mode.

Figure 17.31: Mapping of Peripheral Bus Address to CAN RAM for Enhanced RX FIFO Regions

Below figure shows peripheral bus to RAM address mapping for the high-resolution time stamp region, which is present when TIME_STAMP_EN = 1. In the RAM, the high-resolution time stamp region starts at the next address above the SMBs region:

- High-resolution time stamps for the Rx SMBs occupy the first two words:
  - Rx_SMB0_TIME_STAMP is mapped to peripheral bus address 0x0C20.
  - Rx_SMB1_TIME_STAMP is mapped to peripheral bus address 0x0C24.
- The CAN_HR_TIME_STAMPn registers are also implemented in RAM starting at peripheral bus address 0x03C0.

  There is one CAN_HR_TIME_STAMPn register for each implemented message buffer. Therefore, the end address of the CAN_HR_TIME_STAMPn registers region is 0x03C0 + (4 x NUMBER_OF_MB) - 4. For example, if NUMBER_OF_MB = 128, the range of the CAN_HR_TIME_STAMPn registers region is 0x03C0–0x0E2C.

Yuntu Microelectronics

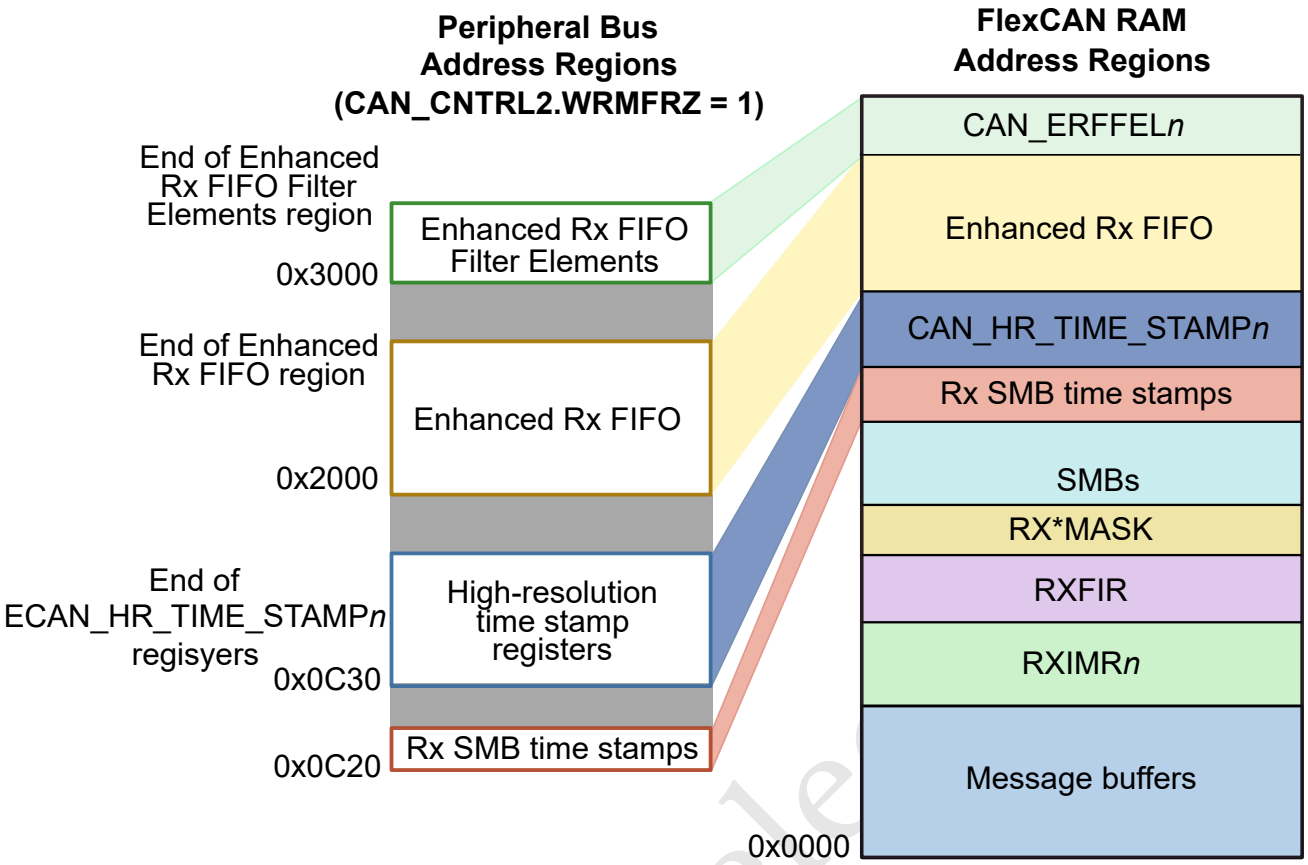


Figure 17.32: Mapping of Peripheral Bus Address to CAN RAM for High-resolution Time Stamp Regions

17.6.3 RAM Initialization Addresses

When ECC is enabled, the entire FlexCAN RAM must be initialized at startup so that each location and its corresponding ECC bits are written before being read.

NOTE: The CAN_MCR.RFEN bit must not be set during RAM initialization.

To initialize the FlexCAN RAM:

1. Set CAN_CTRL2.WRMFRZ to enable Freeze mode write access to all memory positions that require initialization.
2. While in Freeze mode, write to the required locations depending on your FlexCAN configuration as shown in below Table.
3. If enhanced MB memory is supported (ENHANCE_MB_MEM = 1), also write to the following locations:
   - 0x1000 through 0x17FF
4. If the Enhanced Rx FIFO is supported (ERX_FIFO_EN = 1), also write to the following locations:
   - 0x2000 through [0x2000 + (80 x ERX_FIFO_SIZE) - 4]
   - 0x3000 through [0x3000 + (4 x ERX_FIFO_FLT_ELEM) - 4]
5. If the high-resolution time stamp feature is supported (TIME_STAMP_EN = 1), also write to the following locations:
   - 0x0C20 and 0x0C24
   - 0x0C30 through [0x0C30 + (4 x NUMBER_OF_MB) - 4]

Yuntu Microele

Table 17.101: CAN Initialization Address for Basic Features

Number of MBs	Init Address (IRMQ_EN=1)	Init Address (IRMQ_EN=1)
128	0x0080-0x0AAC	0x0080-0x087C 0x0AB0-0x0ADC 0x0A80-0x0AAC 0x0F28-0x0FFC 0x0AB0-0x0ADC 0x0F28-0x0FFC
96	0x0080-0x067C	0x0080-0x067C 0x0880-0x09FC 0x0A80-0x0AAC 0x0A80-0x0AAC 0x0AB0-0x0ADC 0x0AB0-0x0ADC 0x0F28-0x0FFC 0x0F28-0x0FFC
64	0x0080-0x047C	0x0080-0x047C 0x0880-0x097C 0x0A80-0x0AAC 0x0A80-0x0AAC 0x0AB0-0x0ADC 0x0AB0-0x0ADC 0x0F28-0x0FFC 0x0F28-0x0FFC
32	0x080 - 0x27C	0x080 - 0x27C 0x880 - 0x8FC 0x0A80-0x0AAC 0xA80 - 0xAAC 0x0AB0-0x0ADC 0xAB0 - 0xADC 0x0F28-0x0FFC 0xF28 - 0xFFC
16	0x0080-0x017C	0x0080-0x017C 0x0880-0x08BC 0x0A80-0x0AAC 0x0A80-0x0AAC 0x0AB0-0x0ADC 0x0AB0-0x0ADC 0x0F28-0x0FFC 0x0F28-0x0FFC

1. Initialization of addresses 0x0AB0–0x0ADC is required when CAN-FD mode is disabled (CAN_MCR.FDEN = 0) or is not supported.
2. Initialization of addresses 0x0F28–0x0FFC is required when CAN-FD mode is enabled (CAN_MCR.FDEN = 1).

Yuntu Microelectronics

17.6.4 RAM Addresses for Error Injection and Reporting

When ECC is enabled, the memory addresses used for ECC error injection and error reporting correspond to the RAM addresses generated at the FlexCAN can_addr[13:0] output to the FlexCAN RAM. As described in “RAM Usage”, the mapping of peripheral bus address to FlexCAN RAM address depends on the values of FlexCAN hardware configuration parameters.

17.6.4.1 RAM Addresses for FlexCAN Standard Features

The following tables show the RAM address range for each RAM region and its corresponding peripheral (IPS) bus address range for different values of NUMBER_OF_MB and IRMQ_EN. The RAM locations are accessible from the IPS bus addresses shown Table ”CAN RAM address mappings when number of MB 128” through Table “CAN RAM address mappings when number of MB 16” in Freeze mode when CAN_CNTRL2.WRMFRZ = 1.

The RAM ranges and corresponding IPS addresses of the Tx and Rx SMBs differ depending on whether CAN-FD mode is enabled. Therefore, each table shows the SMB RAM address mapping for both cases:

- CAN-FD disabled (CAN_MCR.FDEN = 0)
- CAN-FD enabled (CAN_MCR.FDEN = 1)

Table 17.102: CAN RAM Address Mappings when Number of MB 128

RAM Region	IPS address	RAM address(IRMQ_EN=1)	RAM address(IRMQ_EN=0)
MBs	0x0080-0x087C	0x0000-0x07FC	0x0000-0x07FC
RXIMRs	0x0880-0x0A7C	0x0800-0x09FC	N/A
RXFIR	0x0A80-0x0A94	0x0A00-0x0A14	0x0800-0x0814
RXMGMASK	0x0AA0	0x0A20	0x0820
RXFGMASK	0x0AA4	0x0A24	0x0824
RX14MASK	0x0AA8	0x0A28	0x0828
RX15MASK	0x0AAC	0x0A2C	0x082C
Tx_SMB (FDEN = 0)	0x0AB0-0x0ABC	0x0A30-0x0A3C	0x0830-0x083C
Rx_SMB0 (FDEN = 0)	0x0AC0-0x0ACC	0x0A40-0x0A4C	0x0840-0x084C
Rx_SMB1 (FDEN = 0)	0x0AD0-0x0ADC	0x0A50-0x0A5C	0x0850-0x085C
Tx_SMB (FDEN = 1)	0x0F28-0x0F6C	0x0A30-0x0A74	0x0830-0x0874
Rx_SMB0 (FDEN = 1)	0x0F70-0x0FB4	0x0A78-0x0ABC	0x0878-0x08BC
Rx_SMB1 (FDEN = 1)	0x0FB8-0x0FFC	0x0AC0-0x0B04	0x08C0-0x0904

Table 17.103: CAN RAM Address Mappings when Number of MB 96

RAM Region	IPS address	RAM address(IRMQ_EN=1)	RAM address(IRMQ_EN=0)
MBs	0x0080-0x067C	0x0000-0x05FC	0x0000-0x05FC
RXIMRs	0x0880-0x09FC	0x0600-0x077C	N/A
RXFIR	0x0A80-0x0A94	0x0780-0x0794	0x0600-0x0614
RXMGMASK	0x0AA0	0x07A0	0x0620
RXFGMASK	0x0AA4	0x07A4	0x0624
RX14MASK	0x0AA8	0x07A8	0x0628
RX15MASK	0x0AAC	0x07AC	0x062C
Tx_SMB (FDEN = 0)	0x0AB0-0x0ABC	0x07B0-0x07BC	0x0630-0x063C
Rx_SMB0 (FDEN = 0)	0x0AC0-0x0ACC	0x07C0-0x07CC	0x0640-0x064C
Rx_SMB1 (FDEN = 0)	0x0AD0-0x0ADC	0x07D0-0x07DC	0x0650-0x065C
Tx_SMB (FDEN = 1)	0x0F28-0x0F6C	0x07B0-0x07B4	0x0630-0x0674
Rx_SMB0 (FDEN = 1)	0x0F70-0x0FB4	0x07B8-0x083C	0x0678-0x06BC
Rx_SMB1 (FDEN = 1)	0x0FB8-0x0FFC	0x0840-0x0884	0x06C0-0x0704

Yuntu Microelectronics

Table 17.104: CAN RAM Address Mappings when Number of MB 64

RAM Region	IPS address	RAM address(IRMQ_EN=1)	RAM address(IRMQ_EN=0)
MBs	0x0080-0x047C	0x0000-0x03FC	0x0000-0x03FC
RXIMRs	0x0880-0x097C	0x0400-0x04FC	N/A
RXFIR	0x0A80-0x0A94	0x0500-0x0514	0x0400-0x0414
RXMGMASK	0x0AA0	0x0520	0x0420
RXFGMASK	0x0AA4	0x0524	0x0424
RX14MASK	0x0AA8	0x0528	0x0428
RX15MASK	0x0AAC	0x052C	0x042C
Tx_SMB (FDEN = 0)	0x0AB0-0x0ABC	0x0530-0x053C	0x0430-0x043C
Rx_SMB0 (FDEN = 0)	0x0AC0-0x0ACC	0x0540-0x054C	0x0440-0x044C
Rx_SMB1 (FDEN = 0)	0x0AD0-0x0ADC	0x0550-0x055C	0x0450-0x045C
Tx_SMB (FDEN = 1)	0x0F28-0x0F6C	0x0530-0x0574	0x0430-0x0474
Rx_SMB0 (FDEN = 1)	0x0F70-0x0FB4	0x0578-0x05BC	0x0478-0x04BC
Rx_SMB1 (FDEN = 1)	0x0FB8-0x0FFC	0x05C0-0x0604	0x04C0-0x0504

Table 17.105: CAN RAM Address Mappings when Number of MB 32

RAM Region	IPS address	RAM address(IRMQ_EN=1)	RAM address(IRMQ_EN=0)
MBs	0x0080-0x027C	0x0000-0x01FC	0x0000-0x01FC
RXIMRs	0x0880-0x08FC	0x0200-0x027C	N/A
RXFIR	0x0A80-0x0A94	0x0280-0x0294	0x0200-0x0214
RXMGMASK	0x0AA0	0x02A0	0x0220
RXFGMASK	0x0AA4	0x02A4	0x0224
RX14MASK	0x0AA8	0x02A8	0x0228
RX15MASK	0x0AAC	0x02AC	0x022C
Tx_SMB (FDEN = 0)	0x0AB0-0x0ABC	0x02B0-0x02BC	0x0230-0x023C
Rx_SMB0 (FDEN = 0)	0x0AC0-0x0ACC	0x02C0-0x02CC	0x0240-0x024C
Rx_SMB1 (FDEN = 0)	0x0AD0-0x0ADC	0x02D0-0x02DC	0x0250-0x025C
Tx_SMB (FDEN = 1)	0x0F28-0x0F6C	0x02B0-0x02F4	0x0230-0x0274
Rx_SMB0 (FDEN = 1)	0x0F70-0x0FB4	0x02F8-0x033C	0x0278-0x02BC
Rx_SMB1 (FDEN = 1)	0x0FB8-0x0FFC	0x0340-0x0384	0x02C0-0x0304

Table 17.106: CAN RAM Address Mappings when Number of MB 16

RAM Region	IPS address	RAM address(IRMQ_EN=1)	RAM address(IRMQ_EN=0)
MBs	0x0080-0x017C	0x0000-0x00FC	0x0000-0x00FC
RXIMRs	0x0880-0x08BC	0x0100-0x013C	N/A
RXFIR	0x0A80-0x0A94	0x0140-0x0154	0x0100-0x0114
RXMGMASK	0x0AA0	0x0160	0x0120
RXFGMASK	0x0AA4	0x0164	0x0124
RX14MASK	0x0AA8	0x0168	0x0128
RX15MASK	0x0AAC	0x016C	0x012C
Tx_SMB (FDEN = 0)	0x0AB0-0x0ABC	0x0170-0x017C	0x0130-0x013C
Rx_SMB0 (FDEN = 0)	0x0AC0-0x0ACC	0x0180-0x018C	0x0140-0x014C
Rx_SMB1 (FDEN = 0)	0x0AD0-0x0ADC	0x0190-0x019C	0x0150-0x015C

17.6.4.2 RAM Addresses for FlexCAN Advanced Features

The following advanced features of the FlexCAN Controller also use the FlexCAN RAM and affect the RAM address allocation:

Yuntu Microelectronics

- Enhanced MB memory (implemented when ENHANCE_MB_MEM = 1)
- High-resolution time stamp (implemented when TIME_STAMP_EN = 1)
- Enhanced Rx FIFO (implemented when ERX_FIFO_EN = 1)

17.6.4.2.1 Enhanced MB Memory Region

Enhanced MB memory is implemented when the following configuration parameter value are set:

- NUMBER_OF_MB = 128
- FD_EN = 1
- ENHANCE_MB_MEM = 1

The RAM address region for enhanced MB memory is at RAM addresses 0x0800–0x0FFC and maps to peripheral bus addresses 0x1000–0x17FC.

Below Table shows the RAM address mapping when ENHANCE_MB_MEM = 1.

Table 17.107: CAN RAM Address Mapping when ENHANCE MB MEM 1

RAM Region	IPS address	RAM address(IRMQ_EN=1)	RAM address(IRMQ_EN=0)
MBs	0x0080-0x087C	0x0000-0x07FC	0x0000-0x07FC
Enhanced MB memory	0x1000-0x17FC	0x0800-0x0FFC	0x0800-0x0FFC
RXIMRs	0x0880-0x0A7C	0x1000-0x11FC	N/A
RXFIR	0x0A80-0x0A94	0x1200-0x1214	0x1000-0x1014
RXMGMASK	0x0AA0	0x1220	0x1020
RXFGMASK	0x0AA4	0x1224	0x1024
RX14MASK	0x0AA8	0x1228	0x1028
RX15MASK	0x0AAC	0x122C	0x102C
Tx_SMB (FDEN = 1)	0x0F28-0x0F6C	0x1230-0x1274	0x1030-0x1074
Rx_SMB0 (FDEN = 1)	0x0F70-0x0FB4	0x1278-0x12BC	0x1078-0x10BC
Rx_SMB1 (FDEN = 1)	0x0FB8-0x0FFC	0x12C0-0x1304	0x10C0-0x1104

17.6.4.2.2 High-Resolution Time Stamp Region

The high-resolution time stamp region of RAM begins at the next word address above the last address used for the SMBs, as shown in Below Table.

Within the high-resolution time stamp region, there are three subregions:

- Rx_SMB0_TIME_STAMP – One word at RAM address HrTsStart
- Rx_SMB1_TIME_STAMP – One word at RAM address HrTsStart + 0x4
- CAN_HR_TIME_STAMPn registers – NUMBER_OF_MB words starting at RAM address HrTsStart + 0x8

Table below shows the RAM addresses and associated IPS peripheral bus addresses for the high-resolution time stamp region.

Table 17.108: CAN Start Address of High-resolution Time Stamp RAM Region

NUMBER_OF_MB	IRMQ_EN	FD_EN	Last Address of SMB Region	First Address of High-Resolution Time Stamp Region(HrTsStart)
128 (ENHANCE_MB_MEM = 0)	1	1	0x0B04	0x0B08
		0	0x0A5C	0x0A60

Yuntu Microelectronics

Table 17.108 continued from previous page

NUMBER_OF_MB	IRMQ_EN	FD_EN	Last Address of SMB Region	First Address of High-Resolution Time Stamp Region(HrTsStart)
	0	1	0x0904	0x0908
		0	0x085C	0x0860
128 (ENHANCE_MB_MEM = 1)	1	1	0x1304	0x1308
	0	1	0x1104	0x1108
96	1	1	0x0884	0x0888
		0	0x07DC	0x07E0
	0	1	0x0704	0x0708
		0	0x065C	0x0660
64	1	1	0x0604	0x0608
		0	0x055C	0x0560
	0	1	0x0504	0x0508
		0	0x045C	0x0460
32	1	1	0x0384	0x0388
		0	0x02DC	0x02E0
	0	1	0x0304	0x0308
		0	0x025C	0x0260
16	1	0	0x019C	0x01A0
	0	0	0x015C	0x0160

Table 17.109: CAN RAM Address Mapping for High-resolution Time Stamp Region

High-Resolution Time Stamp Sub-Region	NUMBER_OF_MB	IPS address	RAM Address
Rx_SMB0_TIME_STAMP	x	0x0C20	HrTsStart
Rx_SMB1_TIME_STAMP	x	0x0C24	HrTsStart + 0x4
CAN_HR_TIME_STAMPn	128	0x0C30-0xE2C	(HrTsStart + 0x8) - (HrTsStart + 0x8 + 0x1FC)
	96	0x0C30-0xDAC	(HrTsStart + 0x8) -(HrTsStart + 0x8 + 0x17C)
	64	0x0C30–0xD2C	(HrTsStart + 0x8) –(HrTsStart + 0x8 + 0xFC)
	32	0x0C30–0xCAC	(HrTsStart + 0x8) –(HrTsStart + 0x8 + 0x7C)
	16	0x0C30–0xC6C	(HrTsStart + 0x8) –(HrTsStart + 0x8 + 0x3C)

17.6.4.2.3 Enhanced Rx FIFO Region

The peripheral bus address range for the Enhanced Rx FIFO always starts at peripheral bus address 0x2000. The range of addresses allocated to the Enhanced Rx FIFO depends on:

- Enhanced Rx FIFO depth (ERX_FIFO_SIZE)
- CAN FD support (FD_EN)

Yuntu Microelectronics

The full peripheral bus address range for the Enhanced Rx FIFO is calculated as follows:

- If FD_EN = 1, the Enhanced Rx FIFO peripheral bus address range is 0x2000 through [0x2000 + (80 x ERX_FIFO_SIZE) - 4]
- If FD_EN = 0, the Enhanced Rx FIFO peripheral bus address range is 0x2000 through [0x2000 + (24 x ERX_FIFO_SIZE) - 4]

The RAM region allocated to the Enhanced Rx FIFO begins after the last RAM location used for the:

- SMBs when TIME_STAMP_EN = 0 (see “Mapping of peripheral bus address to CAN RAM for enhanced RX FIFO regions”)
- High-resolution time stamp region when TIME_STAMP_EN = 1 (see “Mapping of peripheral bus address to CAN RAM for high-resolution time stamp regions”)

Below Table shows the first RAM address of Enhanced Rx FIFO region (ErxFifoStart) for different FlexCAN configurations.

Table 17.110: CAN Start Address of Enhanced RX FIFO RAM Region

NUMBER_OF_MB	IRMQ_EN	FD_EN	TIME_STAMP_EN	First Address of Enhanced Rx FIFO Region(ErxFifoStart)
128 (ENHANCE_MB_MEM = 0)	1	1	1	0x0D10
			0	0x0B08
		0	1	0x0C68
			0	0x0A60
	0	1	1	0x0B10
			0	0x0908
		0	1	0x0A68
			0	0x0860
128 (ENHANCE_MB_MEM = 1)	1	1	1	0x1510
		1	0	0x1308
	0	1	1	0x1310
		1	0	0x1108
96	1	1	1	0x0A10
			0	0x0888
		0	1	0x0968
			0	0x07E0
	0	1	1	0x0890
			0	0x0708
		0	1	0x07E8
			0	0x0660
64	1	1	1	0x0710
			0	0x0608
		0	1	0x0668

Yuntu Microelectronics

Table 17.110 continued from previous page

NUMBER_OF_MB	IRMQ_EN	FD_EN	TIME_STAMP_EN	First Address of Enhanced Rx FIFO Region(ErxFifoStart)
			0	0x0560
	0	1	1	0x0610
			0	0x0508
		0	1	0x0568
			0	0x0460
32	1	1	1	0x0410
			0	0x0388
		0	1	0x0368
			0	0x02E0
	0	1	1	0x0390
			0	0x0308
		0	1	0x02E8
			0	0x0260
16	1	0	1	0x01E8
			0	0x01A0
	0	0	1	0x01A8
			0	0x0160

The full RAM address range for the Enhanced Rx FIFO is calculated as follows:

- If FD_EN = 1, the Enhanced Rx FIFO RAM bus address range is ErxFifoStart through [ErxFifoStart + (80 x ERX_FIFO_SIZE) - 4]
- If FD_EN = 0, the Enhanced Rx FIFO peripheral bus address range is ErxFifoStart through [ErxFifoStart + (24 x ERX_FIFO_SIZE) - 4]

“CAN RAM address mapping for enhanced RX FIFO (FD_EN=1)” shows the Enhanced Rx FIFO RAM address MAPPING for each value of ERX_FIFO_SIZE when FD_EN = 1. ”CAN RAM address mapping for enhanced RX FIFO (FD_EN=0)” shows the Enhanced Rx FIFO RAM address MAPPING for each value of ERX_FIFO_SIZE when FD_EN = 0.

NOTE: In Normal mode, only the oldest unread Enhanced Rx FIFO entry is accessible through the peripheral address bus, at peripheral bus addresses 0x2000–0x204C (FD_EN = 1) or 0x2000–0x2014 (FD_EN = 0).

Table 17.111: CAN RAM Address Mapping for Enhanced RX FIFO FD EN 1

ERX_FIFO_SIZE	IPS address	Enhanved Rx FIFO RAM address(Start)	Enhanved Rx FIFO RAM address(End)
2	0x2000-0x209C	ErxFifoStart	ErxFifoStart + 0x009C
3	0x2000-0x20EC	ErxFifoStart	ErxFifoStart + 0x00EC
4	0x2000-0x213C	ErxFifoStart	ErxFifoStart + 0x013C
5	0x2000-0x218C	ErxFifoStart	ErxFifoStart + 0x018C

Yuntu Microelectronics

Table 17.111 continued from previous page

ERX_FIFO_SIZE	IPS address	Enhanved Rx FIFO RAM address(Start)	Enhanved Rx FIFO RAM address(End)
6	0x2000-0x21DC	ErxFifoStart	ErxFifoStart + 0x01DC
7	0x2000-0x222C	ErxFifoStart	ErxFifoStart + 0x022C
8	0x2000-0x227C	ErxFifoStart	ErxFifoStart + 0x027C
9	0x2000-0x22CC	ErxFifoStart	ErxFifoStart + 0x02CC
10	0x2000-0x231C	ErxFifoStart	ErxFifoStart + 0x031C
11	0x2000-0x236C	ErxFifoStart	ErxFifoStart + 0x036C
12	0x2000-0x23BC	ErxFifoStart	ErxFifoStart + 0x03BC
13	0x2000-0x240C	ErxFifoStart	ErxFifoStart + 0x040C
14	0x2000-0x245C	ErxFifoStart	ErxFifoStart + 0x045C
15	0x2000-0x24AC	ErxFifoStart	ErxFifoStart + 0x04AC
16	0x2000-0x24FC	ErxFifoStart	ErxFifoStart + 0x04FC
17	0x2000-0x254C	ErxFifoStart	ErxFifoStart + 0x054C
18	0x2000-0x259C	ErxFifoStart	ErxFifoStart + 0x059C
19	0x2000-0x25EC	ErxFifoStart	ErxFifoStart + 0x05EC
20	0x2000-0x263C	ErxFifoStart	ErxFifoStart + 0x063C
21	0x2000-0x268C	ErxFifoStart	ErxFifoStart + 0x068C
22	0x2000-0x26DC	ErxFifoStart	ErxFifoStart + 0x06DC
23	0x2000-0x272C	ErxFifoStart	ErxFifoStart + 0x072C
24	0x2000-0x277C	ErxFifoStart	ErxFifoStart + 0x077C
25	0x2000-0x277C	ErxFifoStart	ErxFifoStart + 0x07CC
26	0x2000-0x281C	ErxFifoStart	ErxFifoStart + 0x081C
27	0x2000-0x286C	ErxFifoStart	ErxFifoStart + 0x086C
28	0x2000-0x28BC	ErxFifoStart	ErxFifoStart + 0x08BC
29	0x2000-0x290C	ErxFifoStart	ErxFifoStart + 0x090C
30	0x2000-0x295C	ErxFifoStart	ErxFifoStart + 0x095C
31	0x2000-0x29AC	ErxFifoStart	ErxFifoStart + 0x09AC
32	0x2000-0x29FC	ErxFifoStart	ErxFifoStart + 0x09FC

Table 17.112: CAN RAM Address Mapping for Enhanced RX FIFO FD EN 0

ERX_FIFO_SIZE	IPS Address	Enhanved Rx FIFO RAM Address(Start)	Enhanved Rx FIFO RAM Address(End)
2	0x2000-0x202C	ErxFifoStart	ErxFifoStart + 0x002C
3	0x2000-0x2044	ErxFifoStart	ErxFifoStart + 0x0044
4	0x2000-0x205C	ErxFifoStart	ErxFifoStart + 0x005C
5	0x2000-0x2074	ErxFifoStart	ErxFifoStart + 0x0074
6	0x2000-0x208C	ErxFifoStart	ErxFifoStart + 0x008C
7	0x2000-0x20A4	ErxFifoStart	ErxFifoStart + 0x00A4
8	0x2000-0x20BC	ErxFifoStart	ErxFifoStart + 0x00BC
9	0x2000-0x20D4	ErxFifoStart	ErxFifoStart + 0x00D4
10	0x2000-0x20EC	ErxFifoStart	ErxFifoStart + 0x00EC
11	0x2000-0x2104	ErxFifoStart	ErxFifoStart + 0x0104
12	0x2000-0x211C	ErxFifoStart	ErxFifoStart + 0x011C
13	0x2000-0x2134	ErxFifoStart	ErxFifoStart + 0x0134

Yuntu Microelectronics

Table 17.112 continued from previous page

ERX_FIFO_SIZE	IPS Address	Enhanved Rx FIFO RAM Address(Start)	Enhanved Rx FIFO RAM Address(End)
14	0x2000-0x214C	ErxFifoStart	ErxFifoStart + 0x014C
15	0x2000-0x2164	ErxFifoStart	ErxFifoStart + 0x0164
16	0x2000-0x217C	ErxFifoStart	ErxFifoStart + 0x017C
17	0x2000-0x2194	ErxFifoStart	ErxFifoStart + 0x0194
18	0x2000-0x21AC	ErxFifoStart	ErxFifoStart + 0x01AC
19	0x2000-0x21C4	ErxFifoStart	ErxFifoStart + 0x01C4
20	0x2000-0x21DC	ErxFifoStart	ErxFifoStart + 0x01DC
21	0x2000-0x21F4	ErxFifoStart	ErxFifoStart + 0x01F4
22	0x2000-0x220C	ErxFifoStart	ErxFifoStart + 0x020C
23	0x2000-0x2224	ErxFifoStart	ErxFifoStart + 0x0224
24	0x2000-0x223C	ErxFifoStart	ErxFifoStart + 0x023C
25	0x2000-0x2254	ErxFifoStart	ErxFifoStart + 0x0254
26	0x2000-0x226C	ErxFifoStart	ErxFifoStart + 0x026C
27	0x2000-0x2284	ErxFifoStart	ErxFifoStart + 0x0284
28	0x2000-0x229C	ErxFifoStart	ErxFifoStart + 0x029C
29	0x2000-0x22B4	ErxFifoStart	ErxFifoStart + 0x02B4
30	0x2000-0x22CC	ErxFifoStart	ErxFifoStart + 0x02CC
31	0x2000-0x22E4	ErxFifoStart	ErxFifoStart + 0x02E4
32	0x2000-0x22FC	ErxFifoStart	ErxFifoStart + 0x02FC

17.6.4.2.4 Enhanced Rx FIFO Filter Elements Region

The peripheral bus address range for the Enhanced Rx FIFO filter elements always starts at peripheral bus address 0x3000. The range of addresses allocated to the Enhanced Rx FIFO depends on the number of Enhanced Rx FIFO filter elements (ERX_FIFO_FLT_ELEM).

The full peripheral bus address range for the Enhanced Rx FIFO filter elements is: 0x3000 through [0x3000 + (4 x ERX_FIFO_FLT_ELEM) - 4].

As shown in Figure ”Mapping of peripheral bus address to CAN RAM for enhanced RX FIFO regions”, the RAM region allocated to the Enhanced Rx FIFO filter elements begins after the last RAM location used for the Enhanced Rx FIFO.

Example: Full FlexCAN Configuration

For a full FlexCAN configuration with NUMBER_OF_MB = 128, ENHANCE_MB_MEM = 1, IRMQ_EN = 1, FD_EN = 1, TIME_STAMP_EN = 1, ERX_FIFO_SIZE = 32, and ERX_FIFO_FLT_ELEM = 128, the last RAM address used by the Enhanced Rx FIFO is 0x1F0C, as shown in Table “CAN RAM address mapping for enhanced RX FIFO (FD_EN=1)”. The first RAM address allocated to the Enhanced Rx FIFO filter elements (ErxFifoFltElemStart) is therefore 0x1F10.

The full RAM address range for the Enhanced Rx FIFO filter elements is: ErxFifoFltElemStart through [ErxFifoFltElemStart + (4 x ERX_FIFO_FLT_ELEM) - 4].

For the full FlexCAN configuration described above, the Enhanced Rx FIFO filter elements RAM region uses:

- Peripheral bus addresses 0x3000–0x31FC
- RAM addresses 0x1F10–0x210C

Yuntu Microelectronics

Example: Full FlexCAN Configuration minus ENHANCE_MB_MEM and TIME_STAMP_EN

For a FlexCAN configuration with NUMBER_OF_MB = 128, ENHANCE_MB_MEM = 0, IRMQ_EN = 1, FD_EN = 1, TIME_STAMP_EN = 0, ERX_FIFO_SIZE = 32, and ERX_FIFO_FLT_ELEM = 128, the last RAM address used by the Enhanced Rx FIFO is 0x1504, as shown in Table "CAN RAM address mapping for enhanced RX FIFO (FD_EN=1)". The first RAM address allocated to the Enhanced Rx FIFO filter elements (ErxFifoFltElemStart) is therefore 0x1508.

The full RAM address range for the Enhanced Rx FIFO filter elements is: ErxFifoFltElemStart through [ErxFifoFltElemStart + (4 x ERX_FIFO_FLT_ELEM) - 4].

For the full configuration described above and the maximum of 128 Enhanced Rx FIFO filter elements, the Enhanced Rx FIFO filter elements RAM region uses:

- Peripheral bus addresses 0x3000–0x31FC
- RAM addresses 0x1508–0x1704

Yuntu Microelectronics

Chapter 18

Universal Asynchronous Receiver Transmitter (UART)

18.1 Chip-specific UART Information

18.1.1 Instantiation Information

The UART module supports basic UART with DMA interface function and ×4 to ×32 oversampling of baud-rate.

18.1.2 Signal Description

Table 18.1: UART Signal Table

Signal	Description	Direction
TXD	Transmit data output Can swap to data input in line swap mode	IO
RXD	Receive data input Can swap to data output in line swap mode	IO

18.1.3 Clock Description

Table 18.2: UART Clock Table

Clock Name	Description
APB Clock	Clock for master and slave register access
Function Clock	Clock for UART baudrate generation and data sample

18.1.4 Power Mode Description

Yuntu Microelectronics

Table 18.3: UART Power Mode Table

Power Mode	Description
Active	Full function
Sleep	Keeps working when function clock available
Deepsleep	Keeps working when function clock available
Standby	Keeps working when function clock available

Yuntu Microelectronics

18.2 Introduction

18.2.1 Features

Features of the UART module include:

- Support LIN break detect
- Transmit/Receive FIFO
- Support Transmit/Receive via DMA
- Baudrate setting
- 1-bit or 2-bit STOP size
- 7-bit, 8-bit, 9-bit or 10-bit frame size
- Transmit/Receive polarity setting
- Receive data match
- Line idle, address match wakeup
- Support transmit/receive line switch

18.2.2 Block Diagram

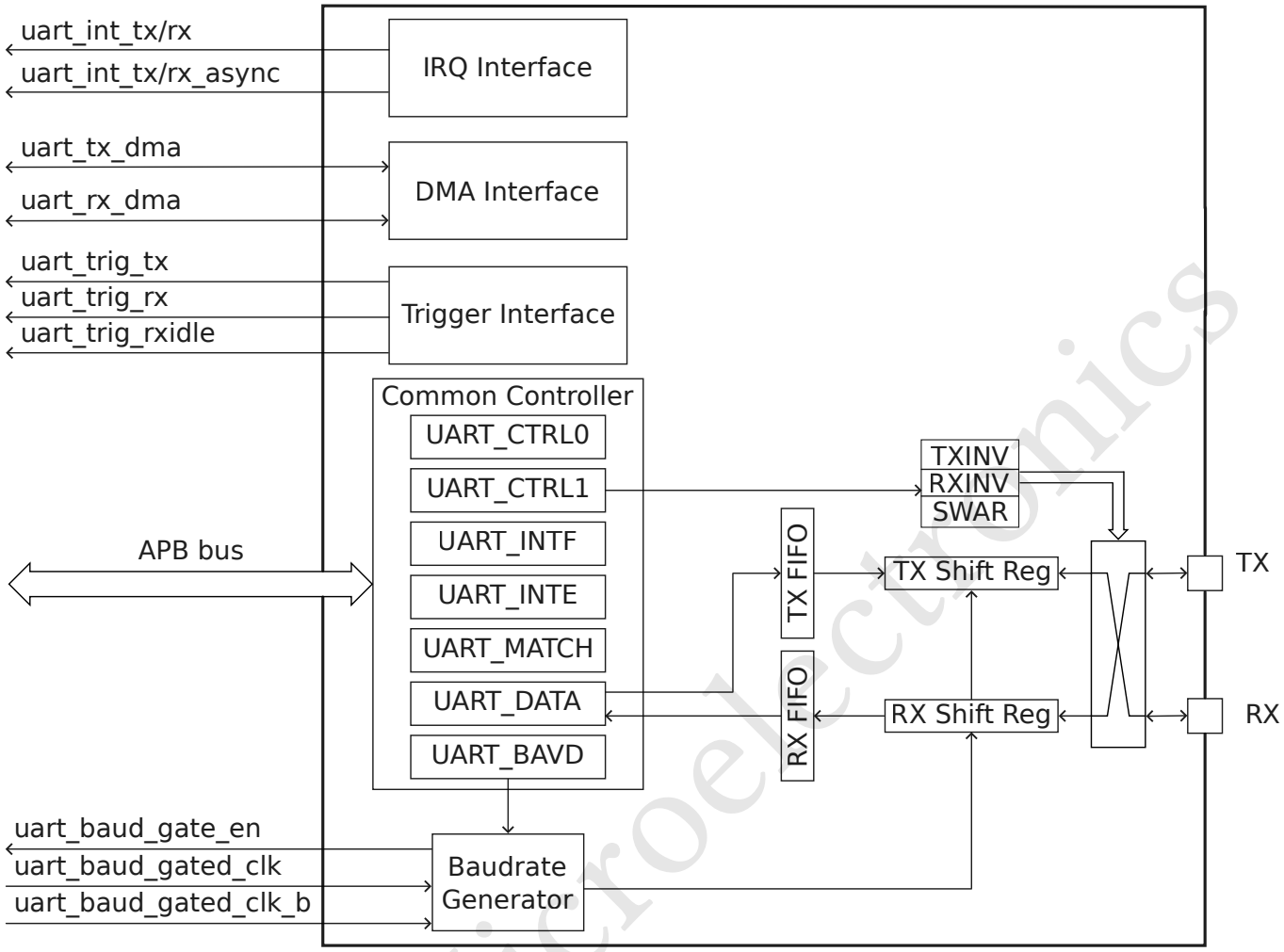


Figure 18.1: UART Block Diagram

18.3 Register Definition

18.3.1 UART Memory Map

UART0 base address 4006a000h
UART1 base address 4006b000h
UART2 base address 4006c000h

Table 18.4: UART Memory Map

Offset	Register	Width	Access	Reset Value
0h	Baud Rate Configuration Register (BAUD)	32	RW	000F0004h
4h	Control 0 Register (CTRL0)	32	RW	00000000h
8h	Control 1 Register (CTRL1)	32	RW	10000000h
ch	Interrupt Flag Register (INTF)	32	RW	00000006h
10h	Interrupt Enable Register (INTE)	32	RW	00000000h
14h	Match Configuration Register (MATCH)	32	RW	00000000h
18h	Data Buffer Register (DATA)	32	RW	00010000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
1ch	TX FIFO Configuration Register (TXFIFO)	32	RW	81000000h
20h	RX FIFO Configuration Register (RXFIFO)	32	RW	81000000h

18.3.1.1 UART BAUD Register

Offset: 0h

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

Table 18.5: UART BAUD Register Description

Field	Function
20 - 16	**Over Sample Value**

Yuntu Microelectronics

Field	Function
OSRVAL	Over sample config, only support update when module disabled 00000b (0x00) - Over sample value is 16 00001b (0x01) - Reserved 00010b (0x02) - Reserved 00011b (0x03) - Over sample value is 4, need to set BOTHEDGE bit 00100b (0x04) - Over sample value is 5, need to set BOTHEDGE bit 00101b (0x05) - Over sample value is 6, need to set BOTHEDGE bit 00110b (0x06) - Over sample value is 7, need to set BOTHEDGE bit 00111b (0x07) - Over sample value is 8. 01000b (0x08) - Over sample value is 9. 01001b (0x09) - Over sample value is 10. 01010b (0x0a) - Over sample value is 11. 01011b (0x0b) - Over sample value is 12. 01100b (0x0c) - Over sample value is 13. 01101b (0x0d) - Over sample value is 14. 01110b (0x0e) - Over sample value is 15. 01111b (0x0f) - Over sample value is 16. 10000b (0x10) - Over sample value is 17. 10001b (0x11) - Over sample value is 18. 10010b (0x12) - Over sample value is 19. 10011b (0x13) - Over sample value is 20. 10100b (0x14) - Over sample value is 21. 10101b (0x15) - Over sample value is 22. 10110b (0x16) - Over sample value is 23. 10111b (0x17) - Over sample value is 24. 11000b (0x18) - Over sample value is 25. 11001b (0x19) - Over sample value is 26. 11010b (0x1a) - Over sample value is 27. 11011b (0x1b) - Over sample value is 28. 11100b (0x1c) - Over sample value is 29. 11101b (0x1d) - Over sample value is 30. 11110b (0x1e) - Over sample value is 31. 11111b (0x1f) - Over sample value is 32.
12 - 0 DIV	**Clock Divider** Function clock divider to generate sample baudrate clock 0000000000000b (0x0000) - Reserved 0000000000001b (0x0001) - Divider value is 1 0000000000010b (0x0002) - Divider value is 2 The clock divider value is DIV value. So clock is Function clock is divided by DIV.

18.3.1.2 UART CTRL0 Register

Offset: 4h

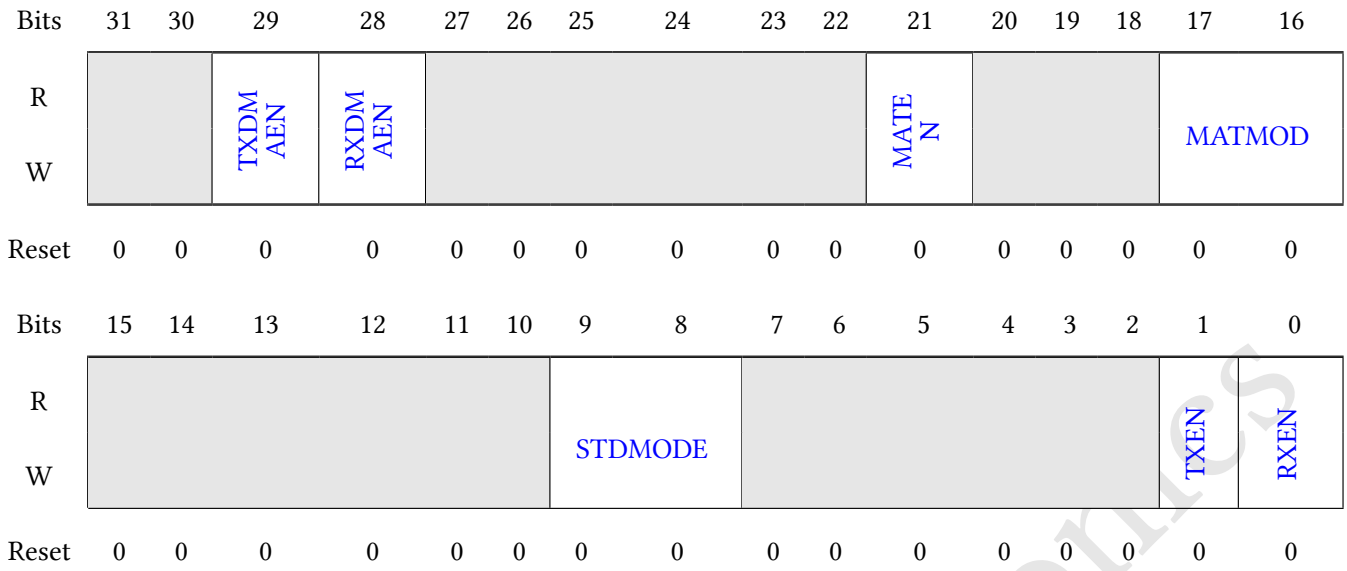


Table 18.6: UART CTRL0 Register Description

Field	Function
29 TXDMAEN	**TX data DMA Enable** TX DATA request DMA enable when TX fifo less than watermark or buff empty 0b - DMA request disabled 1b - DMA request enabled
28 RXDMAEN	**RX data DMA Enable** RX DATA request DMA enable when RX fifo great than watermark or buff full 0b - DMA request disabled 1b - DMA request enabled
21 MATEN	**Match Enable** Setting this bit will enable RX data match logic in UART 0b - UART data match disable 1b - UART data match enable
17 - 16 MATMOD	**Data Match Mode** 00b - Address match wakeup 01b - Idle match wakeup 10b - Match on and match off 11b - Reserved
9 - 8 STDMODE	**Standby Mode Selection** 00b - Normal working mode 01b - standby working mode with idle line wakeup 11b - standby working mode with address match wakeup
1	**TX Enable**

Yuntu Microelectronics

Field	Function
TXEN	0b - TX logic is disabled 1b - TX logic is enabled
0 RXEN	**RX Enable** 0b - RX logic is disabled 1b - RX logic is enabled

18.3.1.3 UART CTRL1 Register

Offset: 8h

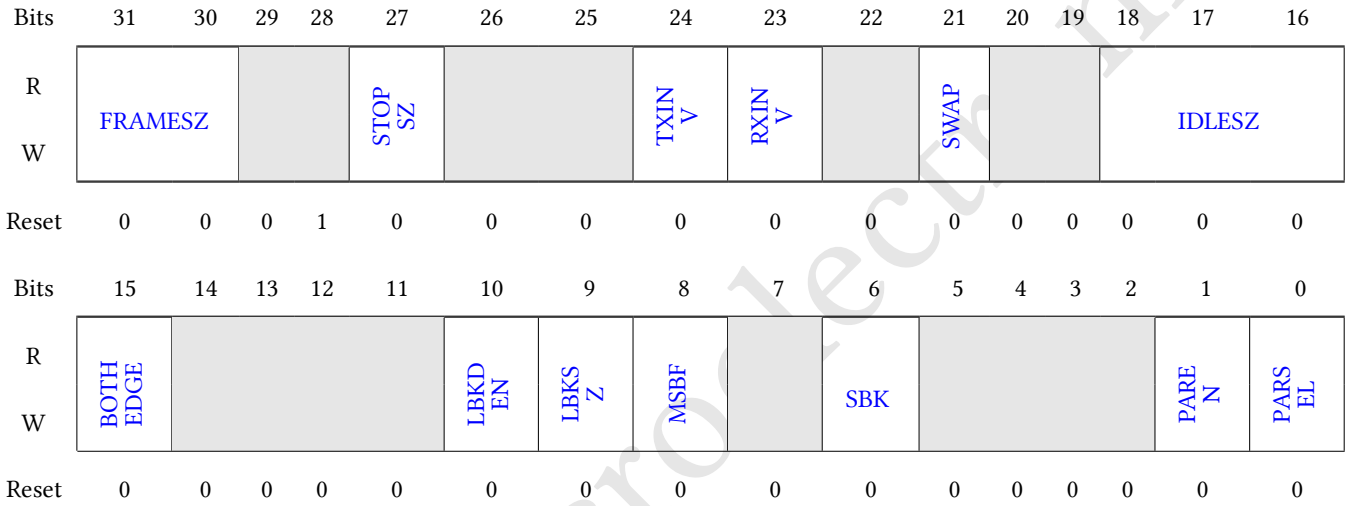


Table 18.7: UART CTRL1 Register Description

Field	Function
31 - 30 FRAMESZ	**Frame Size** Control bit count in one frame transfer 00b - 7-bit data mode 01b - 8-bit data mode 10b - 9-bit data mode 11b - 10-bit data mode
27 STOPSZ	**Stop Size** Stop bit size selection 0b - 1-bit stop 1b - 2-bit stop
24 TXINV	**TX Invert** Invert TX line output, this bit will change data, start, stop, break and idle polarity. 0b - TX data line is not inverted 1b - TX data line is inverted

Yuntu Microelectronics

Field	Function
23 RXINV	**RX Invert** Invert RX line input, this bit will change data, start, stop, break and idle polarity. 0b - RX data line is not inverted 1b - RX data line is inverted
21 SWAP	**RX TX Swap** Swap TX and RX lines 0b - TX RX line under normal mode 1b - Swap TX and RX lines, data input on TX, data output to RX
18 - 16 IDLESZ	**Idle Size** IDLE detection bytes size 000b - 1 idle character 001b - 2 idle character 010b - 4 idle character 011b - 8 idle character 100b - 16 idle character 101b - 32 idle character 110b - 64 idle character 111b - 128 idle character
15 BOTHEDGE	**Both Edge Sample** Setting this bit allows uart sample data on both rising and falling of baudrate clock. This bit need to set when BAUD[OSRVAL] less than 7, only support update when RX logic disabled.
10 LBKDEN	**LIN Break Detect Enable** LBKDEN enable UART detection LIN break, and the break character is not stored in RX FIFO 0b - LIN break detection is disabled 1b - LIN break detection is enabled
9 LBKSZ	**LIN Break Detection Size** 0b - Break character is transmitted with length of 9 to 13 bit times. 1b - Break character is transmitted with length of 12 to 15 bit times.
8 MSBF	**MSB First Mode** Setting this bit will change following data transfer shift order 0b - LSB is shifted out first after start bit 1b - MSB is shifted out first after start bit
6 SBK	**Send Break** Send LIN break out, when SBK bit set, UART will send out LIN break continuously until SBK bit cleared. 0b - Normal mode 1b - Send out continuous LIN break
1	**Parity Enable**

Yuntu Microelectronics

Field	Function
PAREN	Setting this bit will enable UART hardware parity check/generation on both RX and TX 0b - Hardware parity feature disabled 1b - Hardware parity feature enabled on RX and TX
0 PARSEL	**Parity Selection** Valid when PAREN bit set, PARSEL control UART use odd or even parity check/generation 0b - Even parity 1b - Odd parity

18.3.1.4 UART INTF Register

Offset: ch

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R															RXBUSY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R			RFEIF	TFEIF	LBKDIF	RXEDGEIF	MATIF		NOZIF	OVRIF	FEIF	IDLEIF	PARIF	TCIF	TXIF	RXIF
W			W1C	W1C	W1C	W1C	W1C		W1C	W1C	W1C	W1C	W1C			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Table 18.8: UART INTF Register Description

Field	Function
17 RXBUSY	**RX Busy** 0b - RX line idle and wait for data 1b - RX is receiving data
13 RFEIF	**RX FIFO Error Interrupt Flag** RX FIFO underflow happened, read RX FIFO when RX FIFO empty will set this bit 0b - No RX FIFO underflow detected 1b - RX FIFO underflow detected
12	**TX FIFO Error Interrupt Flag**

Yuntu Microelectronics

Field	Function
TFEIF	TX FIFO overflow happened, write TX FIFO when RX FIFO full will set this bit 0b - No TX FIFO overflow detected 1b - TX FIFO overflow detected
11 LBKDIF	**LIN Break Detected Interrupt Flag** 0b - LIN break not detected 1b - LIN break detected
10 RXEDGEIF	**Receive Edge Detected Interrupt Flag** 0b - UART RX active edge not detected 1b - UART RX active edge detected
9 MATIF	**Data Match Interrupt Flag** 0b - UART RX data match not detected 1b - UART RX data match detected
7 NOZIF	**Noise Interrupt Flag** 0b - UART RX noise not detected 1b - UART RX noise detected
6 OVRIF	**Overrun Interrupt Flag** UART received new data when RX FIFO is full will set this bit **NOTE**. User have to clear this bit before receiving new data after RX FIFO not full. 0b - UART RX overrun not detected 1b - UART RX overrun detected, and no new data will be stored
5 FEIF	**Frame Error Interrupt Flag** FEIF set when no valid STOP bit received after data phase 0b - UART RX frame error not detected 1b - UART RX frame error detected
4 IDLEIF	**IDLE Interrupt Flag** IDLE set when UART RX line keeps high time longer than idle condition setting. after bit cleared, idle flag won't set unless new data received. 0b - UART RX idle not detected 1b - UART RX idle detected
3 PARIF	**Parity Check Interrupt Flag** Parity check error happened when hardware calculated parity bit not match received. 0b - UART RX data parity error not detected 1b - UART RX data parity error detected
2	**Transmit Complete Interrupt Flag**

Yuntu Microelectronics

Field	Function
TCIF	TCIF cleared when write TX data and TX start send data, TCIF set when TX line idle 0b - UART transmitter is busy and transmitting data 1b - UART transmitter is idle
1 TXIF	**TX Request Flag** Indicates words in TX FIFO is equal or less than TX FIFO watermark setting 0b - Word in TX FIFO is greater than watermark 1b - Word in TX FIFO is equal or less than watermark, request to write new data
0 RXIF	**RX Request Flag** Indicates words in RX FIFO is greater than RX FIFO watermark setting 0b - Word in RX FIFO is equal or less than watermark 1b - Word in RX FIFO is greater than watermark, should read data register

18.3.1.5 UART INTE Register

Offset: 10h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R			RFEIE	TFIFOEIE	LBKDIE	RXEDGEIE	MATIE		NOZIE	OVRIE	FEIE	IDLEIE	PARIE	TCIE	TXIE	RXIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18.9: UART INTE Register Description

Field	Function
13 RFEIE	**RX FIFO Error Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
12 TFIFOEIE	**TX FIFO Error Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
11	**LIN Break Detected Interrupt Enable**

Yuntu Microelectronics

Field	Function
LBKDIE	0b - Interrupt disabled 1b - Interrupt enabled
10 RXEDGEIE	**Receive Edge Detected Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
9 MATIE	**Data Match Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
7 NOZIE	**Noise Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
6 OVRIE	**Overrun Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
5 FEIE	**Frame Error Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
4 IDLEIE	**IDLE Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
3 PARIE	**Parity Check Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
2 TCIE	**Transmit Complete Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
1 TXIE	**TX Request Data Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
0 RXIE	**RX Request Read Data Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled

18.3.1.6 UART MATCH Register

Offset: 14h

Yuntu Microelectronics

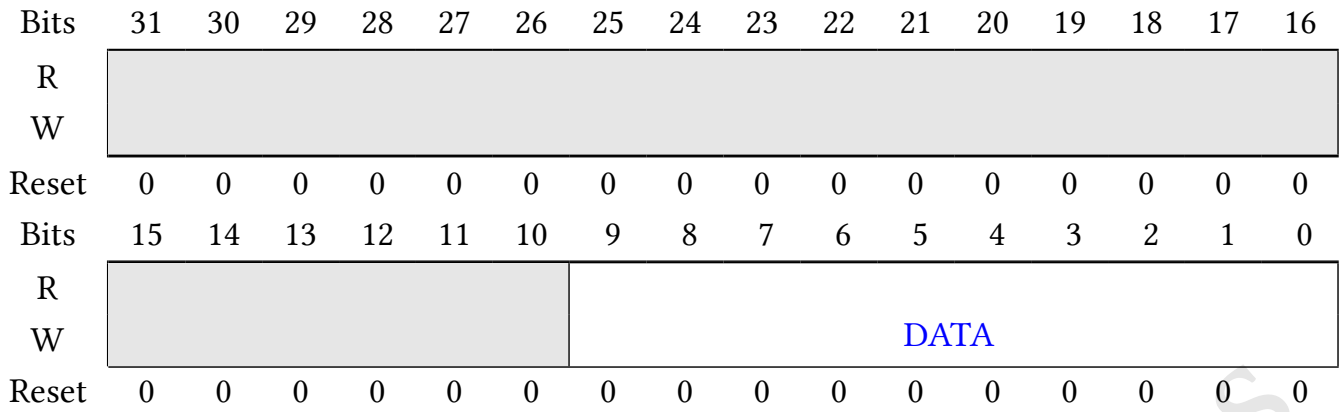


Table 18.10: UART MATCH Register Description

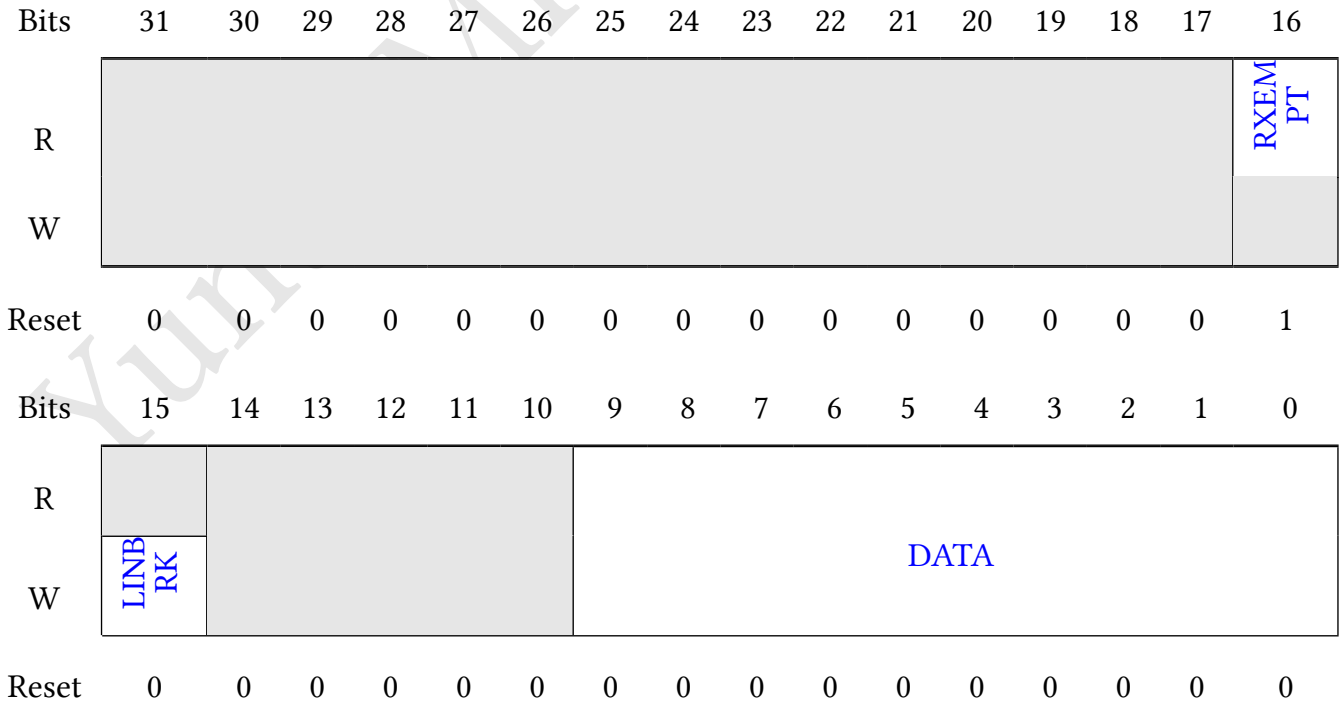
Field	Function
9 - 0 DATA	**Match Data** Data match compare value

18.3.1.7 UART DATA Register

Reading DATA register will return received data in RX FIFO, writing DATA register will push data to TX FIFO and waiting for transfer.

NOTE. This register only support 32-bit access.

Offset: 18h



Yuntu Microelectronics

Table 18.11: UART DATA Register Description

Field	Function
16 RXEMPT	**RX FIFO Empty** 0b - UART RX FIFO is not empty, reading data is valid 1b - UART RX FIFO is empty, reading data is invalid
15 LINBRK	**Send LIN Break** Writing LINBRK bit will let UART send out LIN break instead of DATA
9 - 0 DATA	**DATA Field** Reading DATA field will return data in RX FIFO, writing DATA field will push data to TX FIFO

18.3.1.8 UART TXFIFO Register

Offset: 1ch

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EMPTY	ERROR				SIZE										
W																
Reset	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		EN				COUNT							WATER			
W	RESET															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18.12: UART TXFIFO Register Description

Field	Function
31 EMPTY	**FIFO Empty** TX FIFO empty, asserts when there is no data in the TX FIFO. 1b - TX FIFO is empty
30 ERROR	**TX FIFO Error** Writing TX FIFO with TX FIFO full will set ERROR bit 1b - TX FIFO has overflow error

Yuntu Microelectronics

Field	Function
26 - 24 SIZE	**TX FIFO Size** Available TX FIFO Size is 2^SIZE words
15 RESET	**TX FIFO Reset** Reset TX FIFO address pointer
14 EN	**TX FIFO enable** 0b - FIFO is not enabled, data buffer is used 1b - FIFO is enabled
10 - 8 COUNT	**Word Count in TX FIFO** TX FIFO current data count
2 - 0 WATER	**TX FIFO Watermark** TXIF set when words count in TX FIFO equal or less than watermark

18.3.1.9 UART RXFIFO Register

Offset: 20h

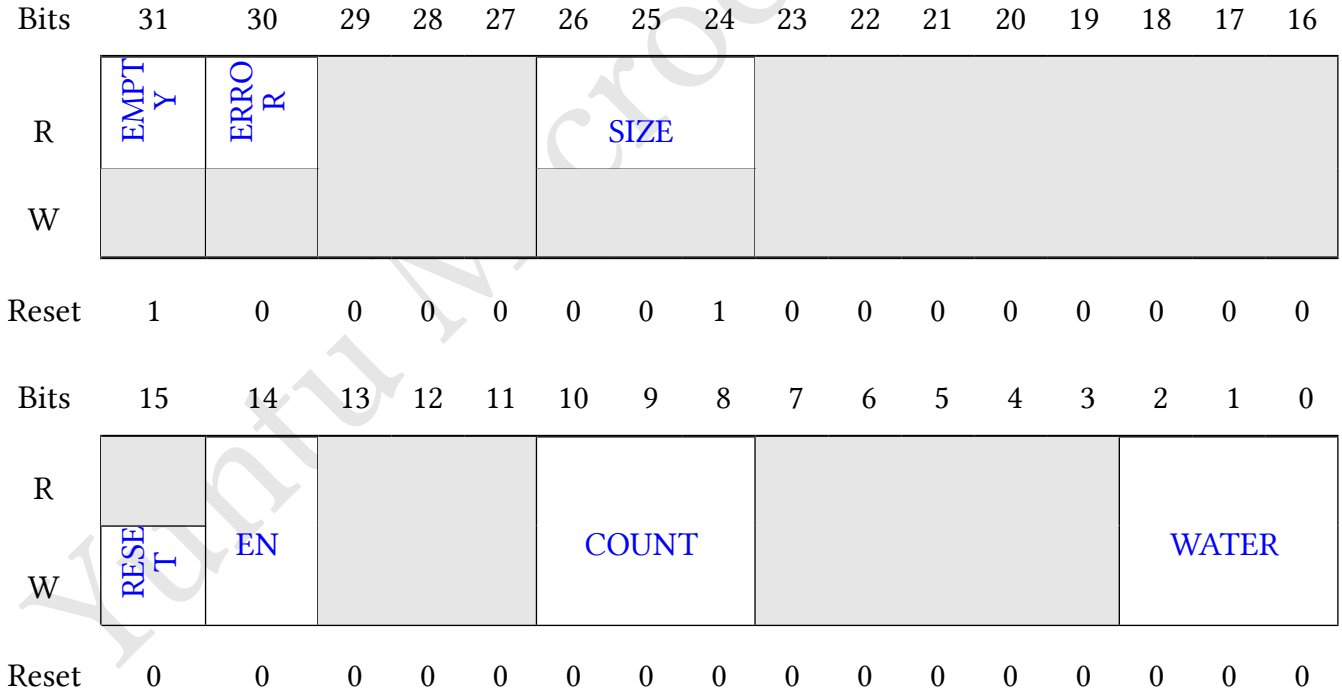


Table 18.13: UART RXFIFO Register Description

Field	Function
31 EMPTY	**FIFO Empty** RX FIFO empty, asserts when there is no data in the RX FIFO.

Yuntu Microelectronics

Field	Function
30 ERROR	**RX FIFO Error** Reading RX FIFO with RX FIFO empty will set ERROR bit 1b - RX FIFO has underflow error
26 - 24 SIZE	**RX FIFO Size** Available RX FIFO Size is 2^SIZE words
15 RESET	**RX FIFO Reset** Reset RX FIFO address pointer
14 EN	**FIFO Enable** 0b - FIFO is not enabled, data buffer is used 1b - FIFO is enabled
10 - 8 COUNT	**Word Count in RX FIFO** RX FIFO current data count
2 - 0 WATER	**RX FIFO Watermark** RXIF set when words count in RX FIFO greater than watermark

18.4 Functional Description

The UART supports full-duplex, asynchronous, NRZ serial communication and is comprised of a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently although they use the same baud rate generator.

18.4.1 Baud Rate Generator

A 13-bit modulus counter in the baud rate generator derives the baud rate for both the receiver and the transmitter. The value from 1 to 8191 written to BAUD[DIV] determines the baud clock divisor for the asynchronous UART baud clock. The baud rate clock drives the receiver, while the transmitter is driven by a bit clock which is generated from baud rate clock divided by the over sampling ratio. Depending on the over sampling ratio, the receiver has an acquisition rate of 4 to 32 samples per bit time.

Yuntu Microelectronics

18.4.1.1 Baud Rate Generator Diagram

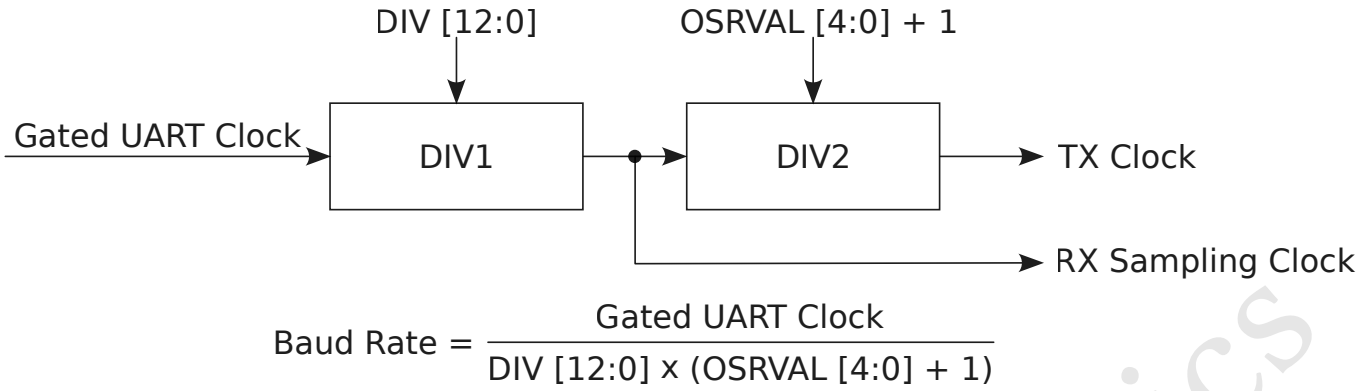


Figure 18.2: Baud Rate Generator Diagram

18.4.2 Transmitter Functional Description

The transmitter output TXD idle state defaults to logic high and will change to logic low if CTRL1[TXINV] is set. The transmitter output is inverted by setting CTRL1[TXINV]. The transmitter is enabled by setting CTRL0[TXEN]. The transmitter then remains idle until data is available in the transmit data buffer. Programs store data into the transmit data buffer by writing to the DATA register.

The transmit shift register is 9-bit to 13-bit long depending on the settings in CTRL1[FRAMESZ] and CTRL1[STOPSZ]. For example, if CTRL1[FRAMESZ] is set to 2'b01, and CTRL1[STOPSZ] is cleared, it is selecting the 8-bit data mode. In the 8-bit data mode, the shift register holds a start bit, eight data bits, and a stop bit. When the transmit shift register is available for a new character, the value waiting in the transmit data register is transferred to the shift register, synchronized with the baud rate clock. And the TX Request (INTF[TXIF]) status flag is set to indicate another character may be written to the transmit data buffer at the DATA register.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out to the TXD pin, the transmitter sets the Transmit Complete (INTF[TCIF]) flag and enters an idle mode, waiting for more character to transmit.

Writing 0 to CTRL0[TXEN] does not immediately disable the transmitter. The current transmit activity in progress will be completed first and then the transmitter will not transmit another character.

18.4.3 Receiver Functional Description

The receiver input is inverted by setting CTRL1[RXINV]. The receiver is enabled by setting CTRL0[RXEN] bit. Character frames consist of a start bit of logic 0, seven to ten data bits (msb or lsb first), and one or two stop bits of logic 1.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the RX request (INTF[RXIF]) status flag is set. If the the RX request (INTF[RXIF]) status flag is already set, the Overrun (INTF[OVRIF]) status flag is set and the last data is lost.

When a program detects that the receive data register is full (INTF[RXIF] = 1), it gets the data from the receive data register by reading the DATA register.

Yuntu Microelectronics

Chapter 19

Serial Port Interface (SPI)

19.1 Chip-specific SPI Information

19.1.1 Instantiation Information

YTM32B1LE0x contains 3 SPI instances.

19.2 Introduction

19.2.1 Features

- Support clock polarity and phase configuration
- Configurable frame size
- Transmit/Receive FIFO
- Support single line mode
- Support master and slave mode
- Support Transmit/Receive via DMA

19.2.2 Signal Description

Table 19.1: SPI Module Signal Description

Signal	Name	Description	Type
SCK	Serial Clock	• Clock output in mater mode • Clock input in slave mode	IO
SOUT	Serial data output	• Serial data output in master and serial data input in slave mode • Could be configured as data input • Used as DATA[0] in parallel data transfer	IO
SIN	Serial data input	• Serial data input in master and serial data output in slave mode • Could be configured as data output • Used as DATA[1] in parallel data transfer	IO

Yuntu Microelectronics

Signal	Name	Description	Type
PCS[3:0]	Peripheral Chip Selection	• Peripheral chip selection in master and slave mode • PCS[3:2] used as DATA[3:2] in 4 line parallel data transfer	IO

19.3 Register Definition

19.3.1 SPI Memory Map

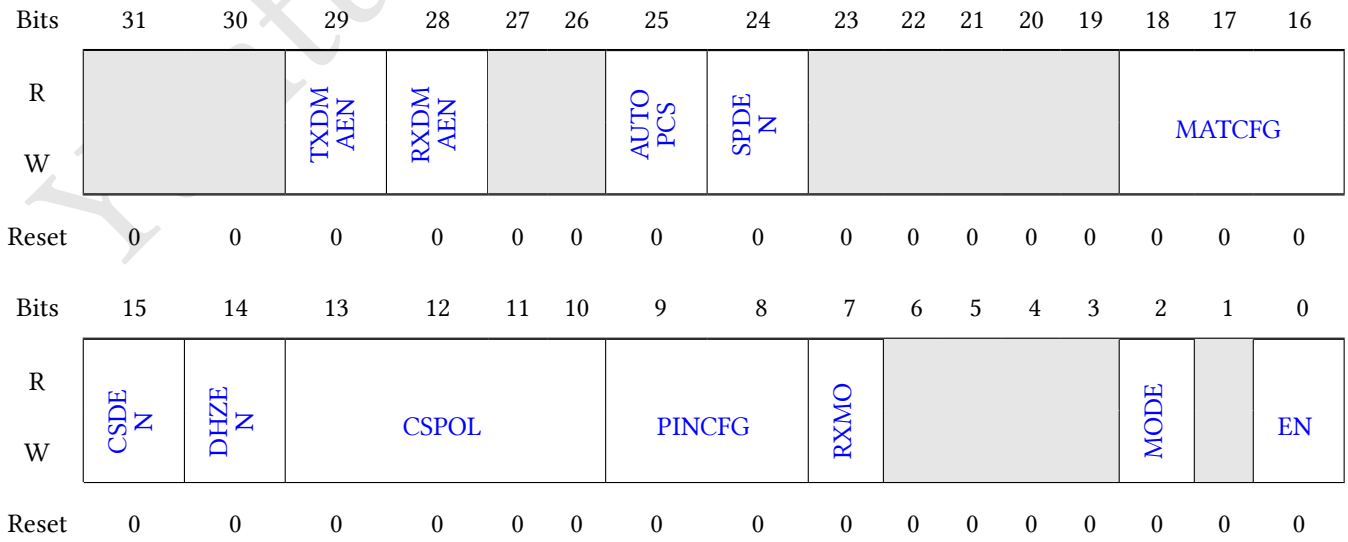
SPI0 base address 4002c000h
SPI1 base address 4002d000h
SPI2 base address 4002e000h

Table 19.2: SPI Memory Map

Offset	Register	Width	Access	Reset Value
0h	Control Register (CTRL)	32	RW	00000000h
4h	Transmit Configuration Register (TXCFG)	32	RW	0000001Fh
8h	Clock Configuration Register (CLK)	32	RW	00000000h
ch	Status Register (STS)	32	RW	00020002h
10h	Interrupt Enable Register (INTE)	32	RW	00000000h
14h	Data Match Register 0 (MATCH0)	32	RW	00000000h
18h	Data Match Register 1 (MATCH1)	32	RW	00000000h
1ch	TX FIFO Configuration Register (TXFIFO)	32	RW	02000000h
20h	RX FIFO Configuration Register (RXFIFO)	32	RW	02000000h
24h	Data Register (DATA)	32	RW	00000000h

19.3.1.1 SPI CTRL Register

Offset: 0h



Yuntu Microelectronics

Table 19.3: SPI CTRL Register Description

Field	Function
29 TXDMAEN	**TX Data DMA Enable** TX DATA request DMA enable when TX fifo less than watermark or buff empty 0b - DMA request disabled 1b - DMA request enabled
28 RXDMAEN	**RX Data DMA Enable** RX DATA request DMA enable when RX fifo great than watermark or buff full 0b - DMA request disabled 1b - DMA request enabled
25 AUTOPCS	**Automatic PCS** When SPI in slave mode, setting this bit will cause SPI slave to automatically generate an internal PCS signal when TXCFG[CPHA]=1 0b - SPI slave use PCS pin directly 1b - SPI slave will regenerate internal PCS signal
24 SPDEN	**Sample Point Delay Enable** Setting this bit will enable sample delay logic before data input under master mode. 0b - SPI sample data on SCK edge 1b - SPI sample data on delayed SCK edge
18 - 16 MATCFG	**Match Configuration** SPI match feature configuration 000b - Match disabled 010b - First data match MATCH0 or MATCH1 011b - Any data match MATCH0 or MATCH1 100b - First data match MATCH0 and second data match MATCH1 101b - Any data match MATCH0 and next data match MATCH1 110b - First data bit AND with MATCH1 match MATCH0 AND MATCH1 111b - Any data bit AND with MATCH1 match MATCH0 AND MATCH1
15 CSDEN	**SPI PCS[3:2] Pin as Data Line Enable** Setting this bit will change SPI PCS[3:2] pin as data line to support 4 data lines transfer 0b - PCS pin works as normal chip selection function 1b - SPI works in half_duplex 4 data lines mode and PCS[3:2] is DATA[3:2]
14 DHZEN	**SPI Data Highz After Transfer Enable** Setting this bit will set output data line to high z when chip select is negated 0b - SPI data output line keeps at last value when chip select is negated 1b - SPI data output line in high z mode when chip select is negated
13 - 10	**SPI Pin Selection Polarity Selection**

Yuntu Microelectronics

Field	Function
CSPOL	Pin selection polarity selection of pcs pin. 0000b - set PCS[n] to active low 0001b - set PCS[n] to active high
9 - 8 PINCFG	**Pin Configuration** Control the data input source and data output target 00b - Data input from SIN and data output to SOUT 01b - Data input from SIN and data output to SIN, spi works under half-duplex serial mode 10b - Data input from SOUT and data output to SOUT, spi works under half-duplex serial mode 11b - Data input from SOUT and data output to SIN
7 RXMO	**Receive Data Match Only** Setting this bit will set SPI only stores data after received matched match configuration 0b - SPI will store all received data 1b - SPI will only store data after data matched
2 MODE	**SPI Mode Selection** SPI Master/slave mode selection 0b - SPI slave mode 1b - SPI master mode
0 EN	**SPI Module Enable** Write to this field will enable/disable SPI module Read this field will reflect current SPI module state **NOTE**. This bit only update when SPI logic is enabled or disabled successfully 0b - SPI module is disabled 1b - SPI module is enabled

19.3.1.2 SPI TXCFG Register

Any data written to TXCFG will be push to TX FIFO, TXCFG will only support 32 bit access, and the configuration in TXCFG will only affect following Data transmissions.

NOTE. TXCFG should only update when SPI is IDLE

NOTE. Read TXCFG should read twice check or read when SPI TX FIFO empty

Offset: 4h

Yuntu Microelectronics

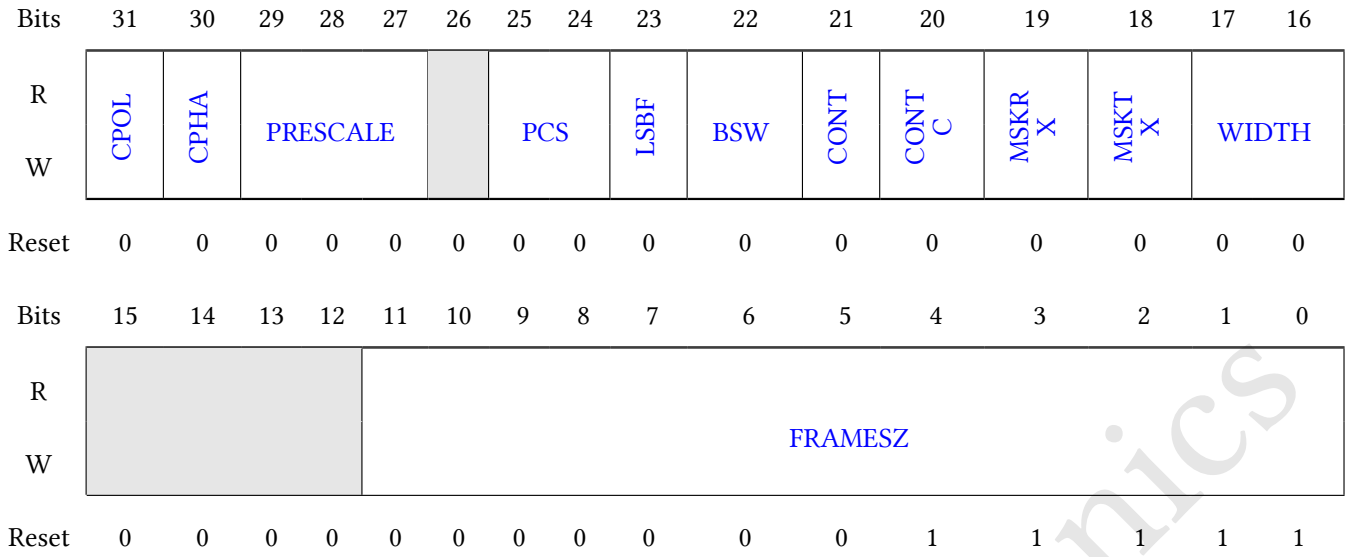


Table 19.4: SPI TXCFG Register Description

Field	Function
31 CPOL	**Clock Polarity** Clock polarity of following data transfer 0b - Clock idles low, leading edge is rising, trailing edge is falling 1b - Clock idles high, leading edge is falling, trailing edge is rising
30 CPHA	**Clock Phase** Clock phase fo following data transfer 0b - Data captured on clock leading edge, Data changed on clock trailing edge 1b - Data captured on clock trailing edge, Data changed on clock leading edge
29 - 27 PRESCALE	**Prescaler Value** Clock prescaler of following SPI clock 000b - SPI clock divided by 1 001b - SPI clock divided by 2 010b - SPI clock divided by 4 011b - SPI clock divided by 8 100b - SPI clock divided by 16 101b - SPI clock divided by 32 110b - SPI clock divided by 64 111b - SPI clock divided by 128
25 - 24 PCS	**Peripheral Chip Selection** Changing this field will change following peripheral selection 00b - Peripheral 0 selected 01b - Peripheral 1 selected 10b - Peripheral 2 selected 11b - Peripheral 3 selected
23	**LSB First Mode**

Yuntu Microelectronics

Field	Function
LSBF	Setting this bit will change following data transfer shift order 0b - MSB is shifted first in data transfer 1b - LSB is shifted first in data transfer
22 BSW	**Byte Swap Mode** Setting this bit will change 32-bit data byte order, this bit controls both TX and RX FIFO 0b - Byte swap mode not enabled 1b - Byte[3] swap with Byte[0], Byte[2] swap with Byte[1]
21 CONT	**Continuous Transfer** In master mode, continuous transfer will keep PCS selected, not space will be inserted between data frame. 0b - Continuous transfer mode is disabled 1b - Continuous transfer mode is enabled
20 CONTC	**Continuous Command** Setting this bit will allow insert command during continuous transfer 0b - Continuous command mode is disabled 1b - Continuous command mode is enabled
19 MSKRX	**Mask RX Data** Setting this bit will disable write received data to RX FIFO 0b - Received data will write to RX FIFO 1b - Received data won't write to RX FIFO
18 MSKTX	**Mask TX Data** In master mode, setting this bit will SPI start a new transfer, SPI data out will be tristated This bit is used to receive data from slave in half-duplex mode. **NOTE**. This bit is cleared by hardware when data transfer ends. 0b - Normal data transfer 1b - Mask TX data and start a data transfer
17 - 16 WIDTH	**Data Width Control** Changing this field allows SPI working in 2 line or 4 line mode **NOTE**. MSKTX or MSKRX must be set to let SPI working in half_duplex mode 00b - 1 data line, normal mode 01b - 2 data line [SDI, SDO], half-duplex mode 10b - 4 data line [PCS[3:2], SDI, SDO], half-duplex mode
11 - 0 FRAMESZ	**Frame Size Control** One frame include FRAMESZ + 1 bits

Yuntu Microelectronics

19.3.1.3 SPI CLK Register

Offset: 8h

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

Table 19.5: SPI CLK Register Description

Field	Function
31 - 24 SCKPCS	**SCK to PCS Delay** Control the delay from last SCK to the PCS negation, only valid in SPI master mode The delay is SCKPCS + 1 divided SPI clock divided by TXCFG[PRESCALE] field
23 - 16 PCSSCK	**PCS to SCK Delay** CS to SCK delay
15 - 8 FMDLY	**Frame Delay** Control SCK delay between frames in continuous mode or the time of PCS negation, only valid in master mode • Delay between frames in continuous mode is FMDLY + 1 divided SPI clock divided by TXCFG[PRESCALE] field • PCS negation time is FMDLY + 1 divided SPI clock divided by TXCFG[PRESCALE] field
7 - 0 DIV	**SPI Clock Divider** SPI clock period is DIV + 2 divided SPI clock divided by TXCFG[PRESCALE] field

Yuntu Microelectronics

19.3.1.4 SPI STS Register

Offset: ch

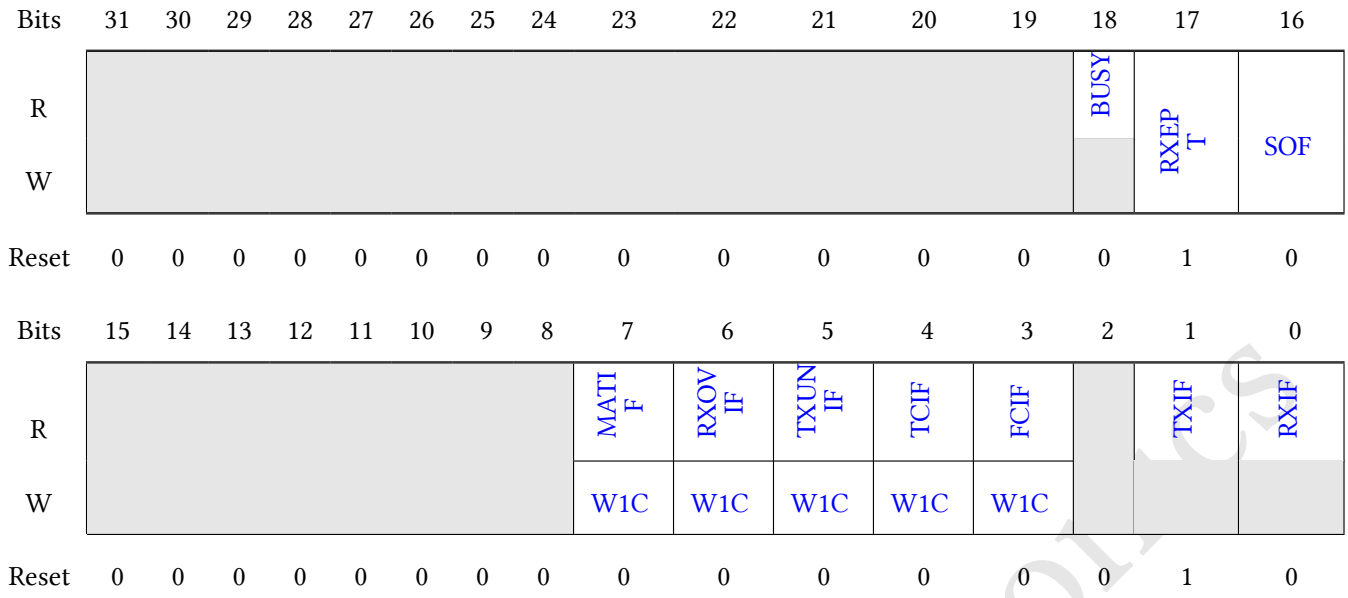


Table 19.6: SPI STS Register Description

Field	Function
18 BUSY	**SPI Module Busy** SPI module is transmitting or receiving data 0b - SPI module is idle 1b - SPI module is busy
17 RXEPT	**SPI RX FIFO Empty Flag** 0b - SPI RX FIFO is not empty 1b - SPI RX FIFO is empty
16 SOF	**Start of Frame** Indicates SPI received first data frame 0b - Received data after first frame or idle 1b - Received first frame data
7 MATIF	**Data Match Interrupt Flag** Indicates SPI received data matched data match configuration 0b - Received data not matched 1b - Received data matched
6 RXOVIF	**RX FIFO Overflow Interrupt Flag** Indicates SPI RX FIFO is overflowed 0b - RX FIFO is not overflowed 1b - RX FIFO is overflowed
5	**TX FIFO Underflow Interrupt Flag**

Yuntu Microelectronics

Field	Function
TXUNIF	Indicates SPI TX FIFO is underflow 0b - TX FIFO is not underflow 1b - TX FIFO is underflow
4 TCIF	**Transmit Complete Interrupt Flag** Indicates SPI has finished data transfer and return to idle 0b - All data transfer not completed 1b - All data transfer completed
3 FCIF	**Frame Transmit Complete Interrupt Flag** Indicates SPI has finished one frame data transfer 0b - One frame data transfer not completed 1b - One frame data transfer completed
1 TXIF	**TX Request Data Flag** Indicates words in TX FIFO is equal or less than TX FIFO watermark setting 0b - Word in TX FIFO is greater than watermark 1b - Word in TX FIFO is equal or less than watermark, request to write new data
0 RXIF	**RX Request Read Data Flag** Indicates words in RX FIFO is greater than RX FIFO watermark setting 0b - Word in RX FIFO is equal or less than watermark 1b - Word in RX FIFO is greater than watermark, should read data register

19.3.1.5 SPI INTE Register

Offset: 10h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									MATIE	RXOVIE	TXUNIE	TCIE	FCIE		TXIE	RXIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Yuntu Microelectronics

Table 19.7: SPI INTE Register Description

Field	Function
7 MATIE	**Data Match Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
6 RXOVIE	**RX FIFO Overflow Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
5 TXUNIE	**TX FIFO Underflow Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
4 TCIE	**SPI TX Complete Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
3 FCIE	**SPI Frame Complete Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
1 TXIE	**TX DATA Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
0 RXIE	**RX DATA Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled

19.3.1.6 SPI MATCH0 Register

Offset: 14h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MATCH0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MATCH0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Yuntu Microelectronics

Table 19.8: SPI MATCH0 Register Description

Field	Function
31 - 0 MATCH0	**SPI Data Match Register 0**

19.3.1.7 SPI MATCH1 Register

Offset: 18h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MATCH1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MATCH1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 19.9: SPI MATCH1 Register Description

Field	Function
31 - 0 MATCH1	**SPI Data Match Register 1**

19.3.1.8 SPI TXFIFO Register

Offset: 1ch

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

Yuntu Microelectronics

Table 19.10: SPI TXFIFO Register Description

Field	Function
28 - 24 SIZE	**SPI TX FIFO Size** Available TX FIFO Size is 2^SIZE words
15 RESET	**SPI TX FIFO Reset** Reset TX FIFO address pointer
12 - 8 COUNT	**Word Count in TX FIFO** SPI TX FIFO current data count
3 - 0 WATER	**TX FIFO Watermark** TXIF set when words count in TX FIFO equal or less than watermark

19.3.1.9 SPI RXFIFO Register

Offset: 20h

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

Table 19.11: SPI RXFIFO Register Description

Field	Function
28 - 24 SIZE	**SPI RX FIFO Size** Available RX FIFO Size is 2^SIZE words
15 RESET	**SPI RX FIFO Reset** Reset SPI RX FIFO address pointer
12 - 8 COUNT	**Word Count in TX FIFO** SPI RX FIFO current data count
3 - 0	**RX FIFO Watermark**

Yuntu Microelectronics

Field	Function
WATER	RXIF set when words count in RX FIFO greater than watermark

19.3.1.10 SPI DATA Register

Offset: 24h

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

Table 19.12: SPI DATA Register Description

Field	Function
31 - 0 DATA	**RX TX Data Register** Read DATA will return SPI received data in RX FIFO Write DATA will push new data into TX FIFO **NOTE**. This register could only access in word

19.4 Functional Description

The module supports full-duplex synchronous serial communications between chips. The SPI module can operate as master or slave. As master, the SPI initiates and controls all data transactions. The SPI has Tx/CMD FIFO and Rx FIFO for data and command buffering.

19.4.1 SPI Functional Clock

There are a few of clock sources for the SPI functional clock, and it can be configured by IPC. It is asynchronous to the bus clock. Whether SPI is able to work in the low power modes depends on whether the SPI functional clock remains enabled. The clock used for timing control in the register CLK is the SPI functional clock.

19.4.2 Master Mode

19.4.2.1 Tx and CMD FIFO

In master mode, the module initiates the serial transfers by controlling the SCK and PCS signals, the Tx FIFO stores the Tx data and command words. A command word should be written between frames and

Yuntu Microelectronics

always takes control in the next frame.

19.4.2.2 Rx FIFO

The Rx FIFO stores the received data during SPI bus transfer. The received data is stored in the least significant bits of a 32-bit word and then is written to the Rx FIFO at the end of the frame. During a continuous transfer, the received data is only written to the Rx FIFO when the Tx FIFO is not empty or the Transmit Configuration Register(TXCFG) is written to end the frame. When the TXCFG[MSKRX] bit(Mask RX Data) is set, the received data is discarded instead of being stored in the Rx FIFO.

19.4.2.3 Data Match

Receive data supports a received data match function.

- Received data that is discarded due to the Mask RX Data TXCFG[MSKRX] bit cannot cause the data match, and will delay the received data match on the first received data word until all discarded data is received.
- The receive data match function can also be configured to discard all received data until a data match is detected, when the Receive Data Match Only CTRL[RXMO] bit is set.
- After a receive data match, to allow all subsequent data to be received, should clear the Receive Data Match Only CTRL[RXMO] bit and the DATA Match Interrupt Flag STS[MATIF] bit first.

19.4.2.4 Pin Configuration

- To swap direction or support half-duplex transfers on the same pin, the SPI_SIN and SPI_SOUT pins can be configured using the Pin Configuration CTRL[PINCFG].
- When performing quad-data transfers, the Chip Selection Polarity CTRL[CSPOL] must be enabled. The Peripheral Chip Selection Polarity CTRL[CSPOL] is also used to disable SPI_PCS[3:2] functions.

19.4.2.5 Clock LoopBack

The SPI master can use the SPI_SCK or the delayed version of the SPI_SCK to sample the input data. The delayed version of the SPI_SCK is delayed by the SPI_SCK pin output delay plus the SPI_SCK pin input delay, and is configured by setting CTRL[SPDEN].

19.4.2.6 Continuous Transfer

When TXCFG[CONT] bit is set, the PCS will keep asserted at the end of the frame size transfer and the SPI will start a new frame size transfer.

19.4.2.6.1 Continuous Command

When TXCFG[CONTC] bit is set, it allows to change the command word during a continuous transfer. When TXCFG[CONTC] bit is cleared, the new command word will terminate the current frame at the the end of the existing word.

19.4.3 Slave Mode

In SPI slave mode, the module responds to transfers initiated by SPI bus master. SPI slave mode uses the same shift register and logic that master mode uses, but does not use the Clock Configuration

Yuntu Microelectronics

Register(CLK). During the SPI transfer, the Transmit Configuration Register (TXCFG) should be unchanging.

19.4.3.1 Tx and CMD FIFO

Before enabling SPI slave mode, the Transmit Configuration Register (TXCFG) should be initialized. If SPI is enabled, the Transmit Configuration Register(TXCFG) should only be updated when SPI is idle. Before the SPI_PCS input asserts, the Tx FIFO must be filled with transmit data, or the Tx FIFO Underflow Interrupt Flag STS[TXUNIF] will set.

19.4.3.2 Receive FIFO

The Rx FIFO is used to store received data during SPI bus transfers. When the Mask RX Data TXCFG[MSKRX] is set, the receive data is discarded instead of being stored in the receive FIFO.

19.4.3.3 Data Match

Receive data supports a receive data match function. - Received data that is discarded due to the Mask RX Data TXCFG[MSKRX] bit cannot cause the data match, and will delay the receive data match on the first received data word until all discarded data is received. - The receive data match function can also be configured to discard all received data until a data match is detected, when the Receive Data Match Only CTRL[RXMO] bit is set. - After a receive data match, to allow all subsequent data to be received, should clear the Receive Data Match Only CTRL[RXMO] bit and the DATA Match Interrupt Flag STS[MATIF] bit first.

19.4.3.4 Internal PCS

When SPI in slave mode, SPI support to generate an internal PCS signal. It requires: - The Clock Phase TXCFG[CPHA] bit is set - The Clock Polity TXCFG[CPOL] bit is set - The Regenerate PCS Enable CTRL[REPCS] bit is set.

19.4.3.5 Continuous Transfer

When TXCFG[CONT] bit is set, the SPI will transmits the received data after the first frame size bits are transmitted until the PCS is inactive.

19.4.3.5.1 Continuous Mode

When TXCFG[CONTC] bit is set, the internal configuration RXMSK and TXMSK of the command word will take effect after the first frame size bits are transmitted until the PCS is inactive.

Yuntu Microelectronics

Chapter 20

Inter-Integrated Circuit (I2C)

20.1 Chip-specific I2C Information

20.1.1 Instantiation Information

YTM32B1LE0x has 2 I2C instances. I2C0 has full function and I2C1 only support master function.

20.1.2 Signal Description

Table 20.1: I2C Signal Table

Signal	Description	Direction
SCL	I2C clock line	IO
SDA	I2C data line	IO

20.1.3 Clock Description

Table 20.2: I2C Clock Table

Clock Name	Description
APB Clock	Clock for master and slave register access
Function Clock	Main clock for I2C master and slave logic, when using clock enabled in sleep mode, I2C could still working under sleep mode
External Clock	Only valid in slave mode, I2C slave can work with external SCL clock even main function clock disabled. **NOTE**. Filter and sample configuration is disable in external clock mode.

20.1.4 Reset Description

Yuntu Microelectronics

Table 20.3: I2C Reset Table

Reset Name	Description
Chip reset	The logic and registers for the I2C master and slave are reset to their default state.
FIFO reset	Setting the TXFIFO[RESET] bit and RXFIFO[RESET] bit will reset the TXFIFO and RXFIFO respectively. After a FIFO is reset, the FIFO is empty.

20.1.5 Power Mode Description

Table 20.4: I2C Power Mode Table

Power Mode	Description
Active	Full function
Sleep	• Keeps working when function clock available • Slave working when function clock available or only external clock available
Deepsleep	• Keeps working when function clock available • Slave working when function clock available or only external clock available

20.2 Introduction

20.2.1 Features

Features of the I2C module include:

- Support standard, fast and ultra fast mode
- Support 7-bit/10-bit address mode with master and slave
- Support SMBus mode
- Support multi-master arbitration and synchronization
- Support Master and slave clock stretching
- Transmit/Receive FIFO (Master only)
- Analog and digital filter on both SCL and SDA pins
- Support Transmit/Receive via DMA

20.3 Register Definition

20.3.1 I2C Memory Map

I2C0 base address 40066000h
I2C1 base address 40067000h

Yuntu Microelectronics

Table 20.5: I2C Memory Map

Offset	Register	Width	Access	Reset Value
0h	Master Control Register (MCTRL)	32	RW	00000000h
4h	Master Clock Configuration Register (MCLKCFG)	32	RW	00000000h
8h	Master Clock Filter Configuration and Divider Register (MFLTCFG)	32	RW	00000000h
ch	Master Status Register (MSTS)	32	RW	00000002h
10h	Master Interrupt Register (MIE)	32	RW	00000000h
14h	Master TX FIFO Control and Status Register (TXFIFO)	32	RW	02000000h
18h	Master RX FIFO Control and Status Register (RXFIFO)	32	RW	02000000h
1ch	Master Data Register (MDATA)	32	RW	00010000h
20h	Data Match Register (MATCH)	32	RW	00000000h
24h	Timeout Configure Register (TOCFG)	32	RW	00000000h
28h	Master clock configuration under high speed mode register (HSCLK)	32	RW	00000000h
100h	Slave Control Register (SCTRL)	32	RW	00000000h
104h	Slave Clock Configuration Register (SCLKCFG)	32	RW	00000000h
108h	Slave Filter Configuration Register (SFLTCFG)	32	RW	00000000h
10ch	Slave Status Register (SSTS)	32	RW	00000000h
110h	Slave Interrupt Enable Register (SIE)	32	RW	00000000h
114h	Slave Data Register (SDATA)	32	RW	00010000h
118h	Slave Ack Register (SACK)	32	RW	00000000h
11ch	Slave Address Match Configuration Register (SADDR)	32	RW	00000000h
120h	Slave Received Address Register (SRADDR)	32	RW	00000000h

20.3.1.1 I2C MCTRL Register

Offset: 0h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W			TXDMAEN	RXDMAEN			RDMO							MATCFG		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IGACK	ASTOP					HSMOD	PPEN								MEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Yuntu Microelectronics

Table 20.6: I2C MCTRL Register Description

Field	Function
29 TXDMAEN	**Master TX DATA DMA Enable** TX DATA request DMA enable when TX fifo less than watermark or buff empty 0b - DMA request disabled 1b - DMA request enabled
28 RXDMAEN	**Master RX data DMA Enable** RX DATA request DMA enable when RX fifo great than watermark or buff full 0b - DMA request disabled 1b - DMA request enabled
25 RDMO	**Receive Data Match Only** Setting this bit will set I2C only stores data after received matched match configuration 0b - I2C will store all received data 1b - I2C will only store data after data matched
18 - 16 MATCFG	**Match Configuration** I2C match feature configuration 000b - Match disabled 010b - First data match MATCH0 or MATCH1 011b - Any data match MATCH0 or MATCH1 100b - First data match MATCH0 and second data match MATCH1 101b - Any data match MATCH0 and next data match MATCH1 110b - First data byte AND with MATCH1 match MATCH0 111b - Any data byte AND with MATCH1 match MATCH0
15 IGACK	**Ignore I2C ACK** Setting this bit will let I2C generate internal ACK even if NACK received. **NOTE**. This bit should be set in Ultra-Fast mode. 0b - I2C will receive ACK and NACK normally 1b - I2C will treat received NACK as ACK
14 ASTOP	**Automatically STOP** Setting this bit will let I2C automatically generate STOP condition when TX FIFO empty 0b - I2C master generated STOP with STOP command 1b - I2C master generated STOP with STOP command or TX FIFO is empty
9 HSMOD	**High Speed Mode Enable** Setting this bit will let I2C working under High-Speed mode(only support data output) 0b - I2C under normal mode 1b - I2C in High-Speed output only mode
8	**Push Pull Mode Enable**

Yuntu Microelectronics

Field	Function
PPEN	Setting this bit will let I2C SCL SDA to push-pull mode 0b - SCL SDA in open drain mode 1b - SCL SDA in push pull mode
0 MEN	**Master Mode Enable** Control all I2C master logic 0b - Master mode is disabled 1b - Master mode is enabled

20.3.1.2 I2C MCLKCFG Register

MCLK register not support update when master logic is enabled

Offset: 4h

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

Table 20.7: I2C MCLKCFG Register Description

Field	Function
29 - 24 HIGH	**SCL High Period** SCL clock high period is (HIGH + 1) * divided clock period
21 - 16 LOW	**SCL Low Period** SCL clock low period is (LOW + 1) * divided clock period
13 - 8 VALID	**SDA Data Valid Period** SDA data valid period is (VALID + 1) * divided clock period
5 - 0 STHD	**SDA Data Setup/Hold Period** SDA data setup/hold period is (STHD + 1) * divided clock period

20.3.1.3 I2C MFLTCFG Register

FLTDIV register not support update when master logic is enabled

Yuntu Microelectronics

Offset: 8h

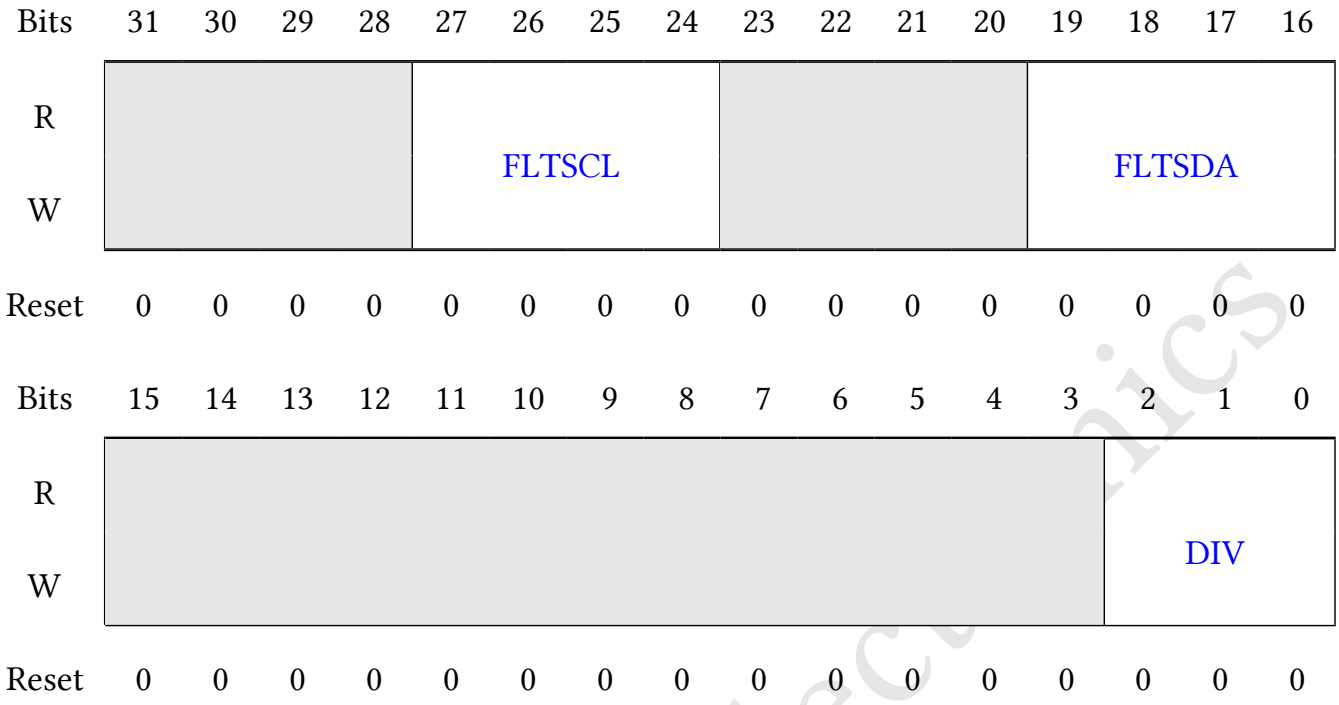


Table 20.8: I2C MFLTCFG Register Description

Field	Function
27 - 24 FLTSCL	**SCL Filter Control** Control digital filter on SCL line SCL digital filter use undivided I2C function clock Noise less or equal FLTSCL * function period will be filtered
19 - 16 FLTSDA	**SDA Filter Control** Control digital filter on SDA line SDA digital filter use undivided I2C function clock Noise less or equal FLTSDA * function period will be filtered
2 - 0 DIV	**Function Clock Divider** Control I2C master logic clock divider 000b - I2C master clock divided by 1 001b - I2C master clock divided by 2 010b - I2C master clock divided by 4 011b - I2C master clock divided by 8 100b - I2C master clock divided by 16 101b - I2C master clock divided by 32 110b - I2C master clock divided by 64 111b - I2C master clock divided by 128

Yuntu Microelectronics

20.3.1.4 I2C MSTS Register

Offset: ch

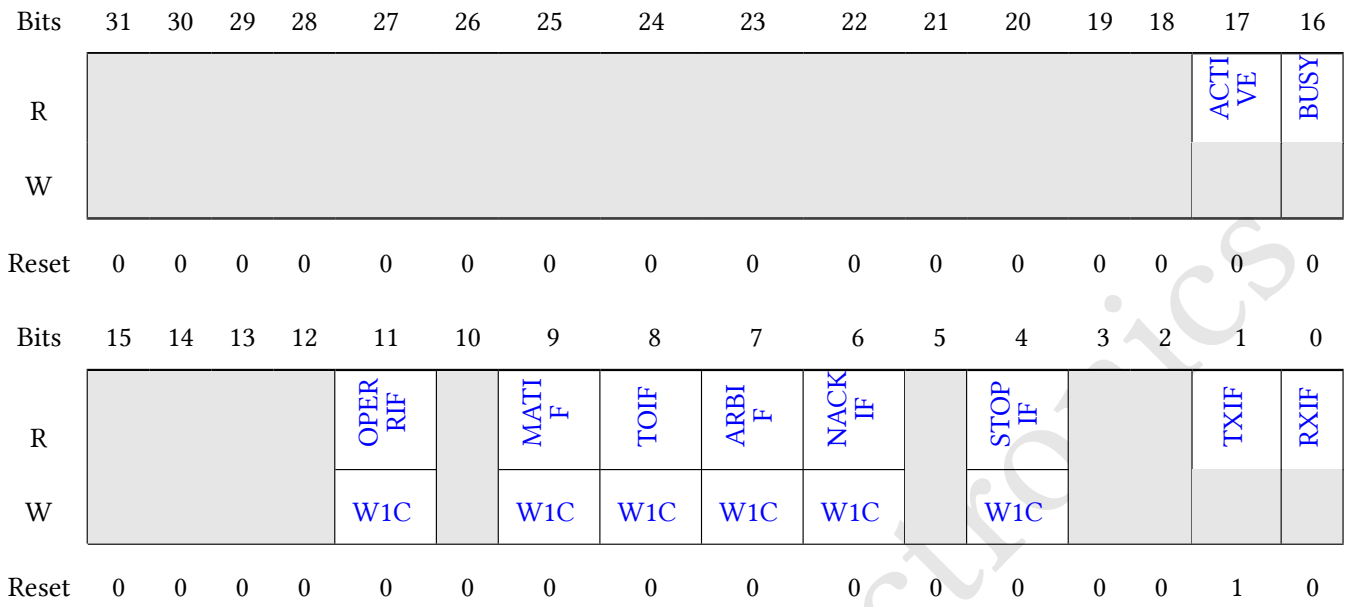


Table 20.9: I2C MSTS Register Description

Field	Function
17 ACTIVE	**I2C Master Active** I2C master is transmitting or receiving data 0b - I2C master is idle 1b - I2C master is busy
16 BUSY	**I2C Bus Busy** I2C Bus is transmitting or receiving data 0b - I2C bus is idle 1b - I2C bus is busy
11 OPERRIF	**Operation Error Interrupt Flag** I2C master operation error, this bit is set when TX FIFO is empty and no automatic STOP generated. User have to clear this bit before start a new data transfer
9 MATIF	**Data Match Interrupt Flag** Indicates I2C master received data matched data match configuration 0b - Received data not matched 1b - Received data matched
8	**Timeout Interrupt Flag**

Yuntu Microelectronics

Field	Function
TOIF	This bit set when line low timeout detection enable and SCL/SDA low timeout This bit only cleared when SCL/SDA line return to 1 User have to solve pin low error before start a new data transfer 0b - Line low timeout not detected 1b - Line low timeout detected
7 ARBIF	**Arbitration Lost Interrupt Flag** This bit is set when Master send out logic 1 but detect logic 0 on SCL or SDA line Master detected START or STOP condition on bus User have to clear this bit before start a new data transfer 0b - I2C master working normal 1b - I2C master lost arbitration
6 NACKIF	**NACK Received Interrupt Flag** If ACK detected logic is enable (IGACK is cleared), I2C master will send out STOP and terminate current transfer, NACKIF bit will set. 0b - I2C master working normal 1b - I2C master detected NACK
4 STOPIF	**STOP Received Interrupt Flag** Set when I2C master send out a STOP condition successfully 0b - I2C master not generated STOP 1b - I2C master generated STOP
1 TXIF	**TX Request Flag** Indicates words in TX FIFO is equal or less than TX FIFO watermark setting 0b - Word in TX FIFO is greater than watermark 1b - Word in TX FIFO is equal or less than watermark, request to write new data
0 RXIF	**RX Request Flag** Indicates words in RX FIFO is greater than RX FIFO watermark setting 0b - Word in RX FIFO is equal or less than watermark 1b - Word in RX FIFO is greater than watermark, should read data register

Yuntu Microelectronics

20.3.1.5 I2C MIE Register

Offset: 10h

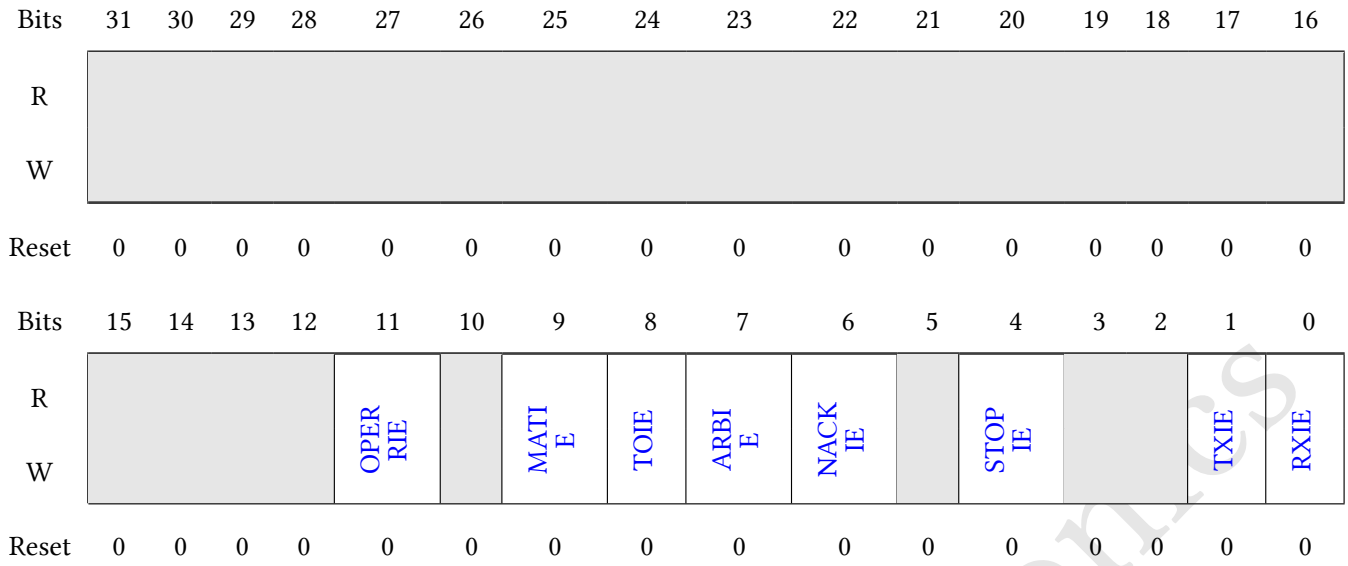


Table 20.10: I2C MIE Register Description

Field	Function
11 OPERRIE	**Operation Error Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
9 MATIE	**Data Match Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
8 TOIE	**Timeout Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
7 ARBIE	**Arbitration Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
6 NACKIE	**NACK Detected Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
4 STOPIE	**STOP Detected Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
1 TXIE	**TX DATA Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
0	**RX DATA Interrupt Enable**

Yuntu Microelectronics

Field	Function
RXIE	0b - Interrupt disabled 1b - Interrupt enabled

20.3.1.6 I2C TXFIFO Register

Offset: 14h

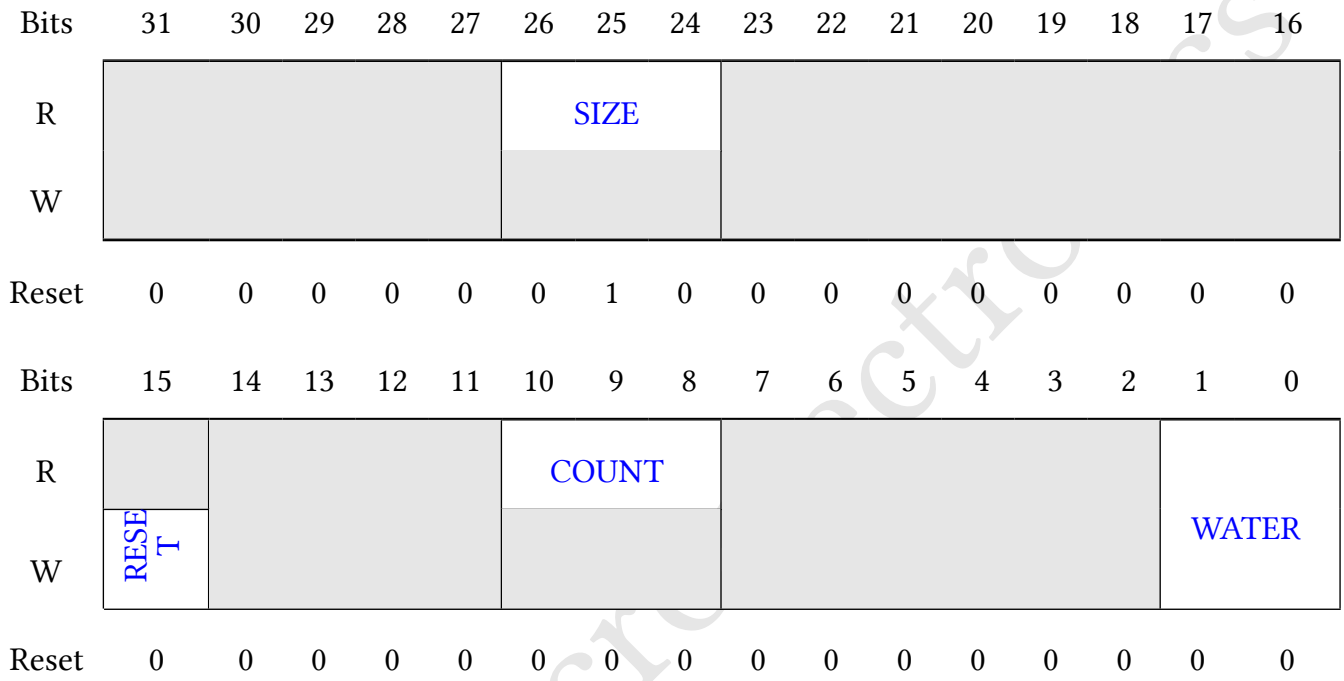


Table 20.11: I2C TXFIFO Register Description

Field	Function
26 - 24 SIZE	**I2C Master TX FIFO Size** Available TX FIFO Size is 2^SIZE words
15 RESET	**I2C Master TX FIFO Reset** Reset TX FIFO address pointer
10 - 8 COUNT	**Word Count in TX FIFO** I2C Master TX FIFO current data count
1 - 0 WATER	**TX FIFO Watermark** TXIF set when words count in TX FIFO equal or less than watermark

20.3.1.7 I2C RXFIFO Register

Offset: 18h

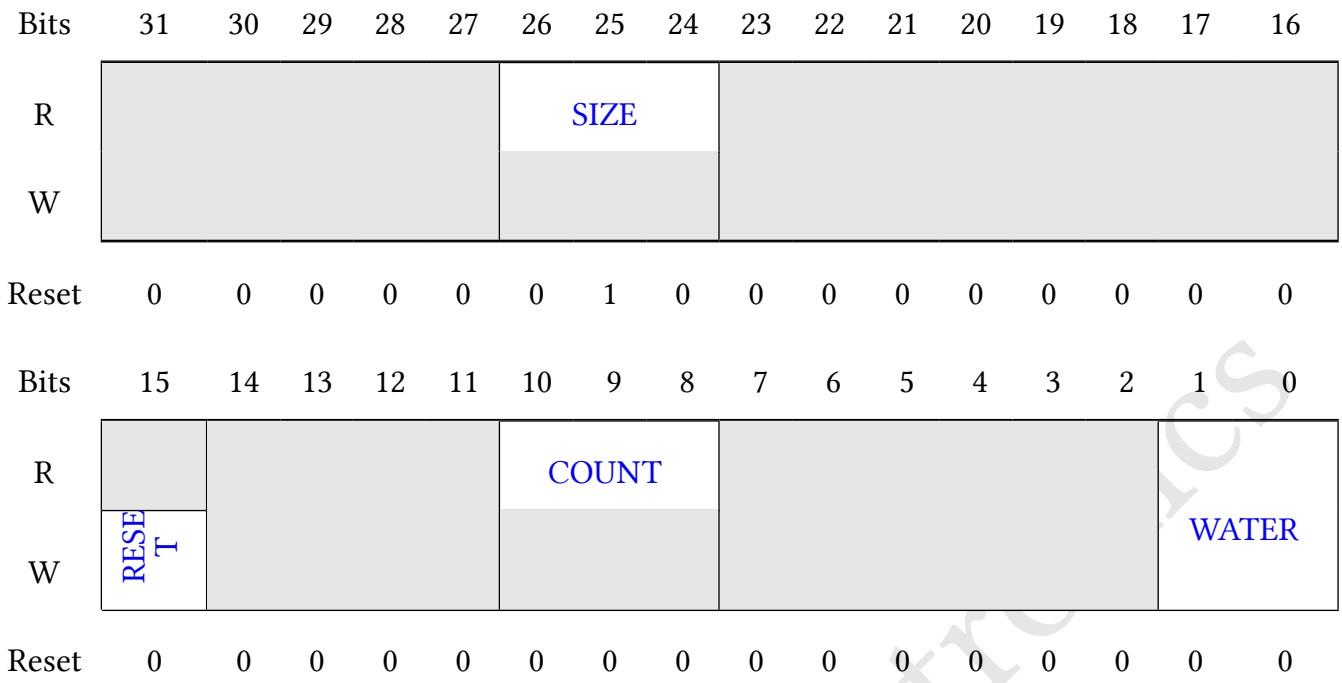


Table 20.12: I2C RXFIFO Register Description

Field	Function
26 - 24 SIZE	**I2C Master RX FIFO Size** Available RX FIFO Size is 2^SIZE words
15 RESET	**I2C Master RX FIFO Reset** Reset I2C master RX FIFO address pointer
10 - 8 COUNT	**Word Count in TX FIFO** Current data count in RX FIFO
1 - 0 WATER	**RX FIFO Watermark** RXIF set when words count in RX FIFO greater than watermark

Yuntu Microelectronics

20.3.1.8 I2C MDATA Register

Write MDATA will let I2C master send out address/data/repeat start/stop to I2C bus and read MDATA will return I2C master received data and status.

Offset: 1ch

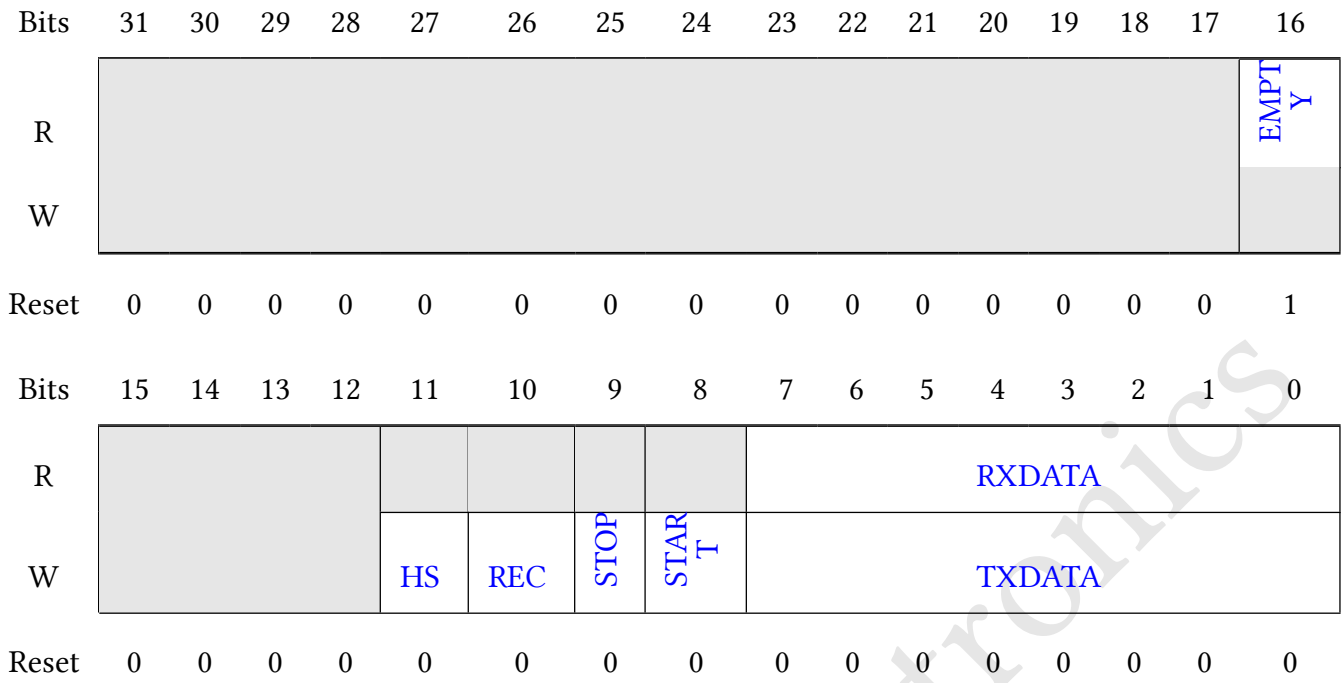


Table 20.13: I2C MDATA Register Description

Field	Function
16 EMPTY	**RX Data Empty** RX FIFO or buffer is empty
11 HS	**High Speed Mode** Setting this bit will let I2C master send data in high speed mode
10 REC	**Receive Data** Setting this bit with target bytes count in DATA field will let I2C receive (DATA + 1) bytes data
9 STOP	**Send STOP**
8 START	**Generate START/repeat START** Setting this bit with slave address in DATA field will let I2C master generate START/repeat START to given slave
7 - 0 DATA	**TX RX DATA** Write DATA will let I2C master send out data or address Read DATA will return I2C master received data in FIFO or buffer

20.3.1.9 I2C MATCH Register

Offset: 20h

Yuntu Microelectronics

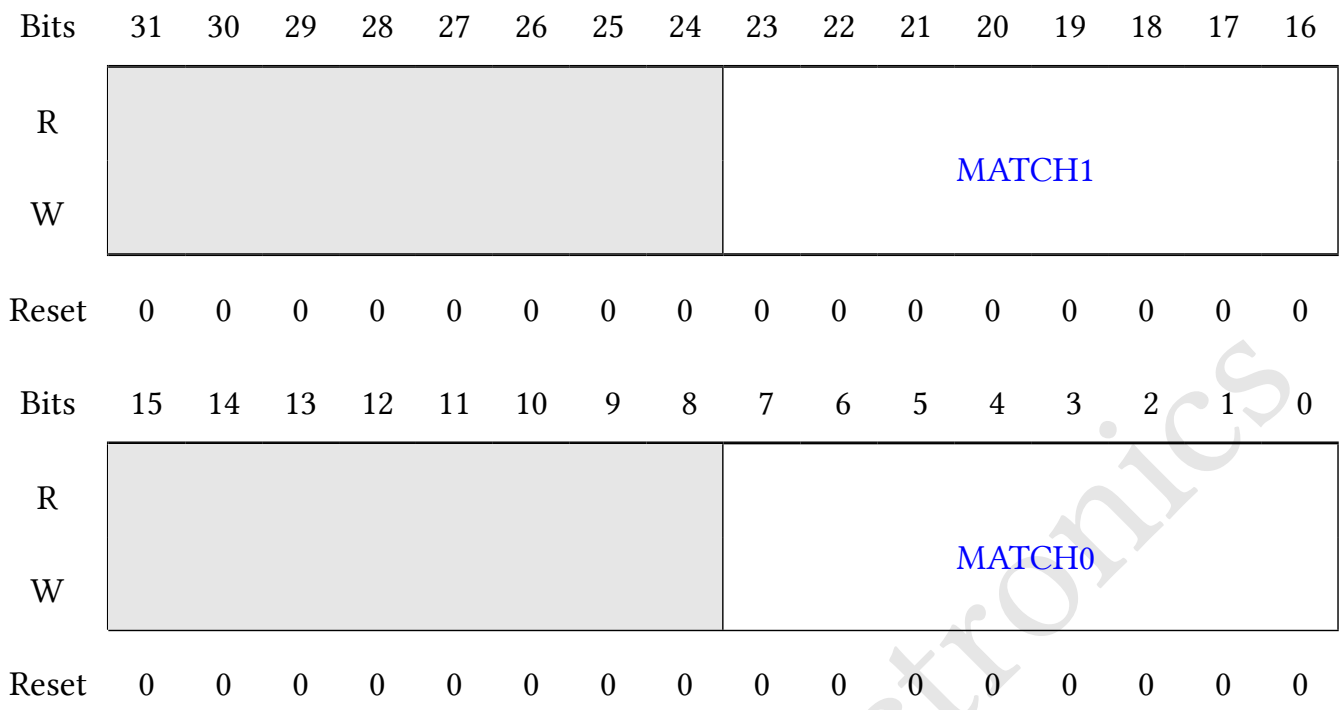


Table 20.14: I2C MATCH Register Description

Field	Function
23 - 16 MATCH1	**Match Data 1**
7 - 0 MATCH0	**Match Data 0**

Yuntu Microelectronics

20.3.1.10 I2C TOCFG Register

Offset: 24h

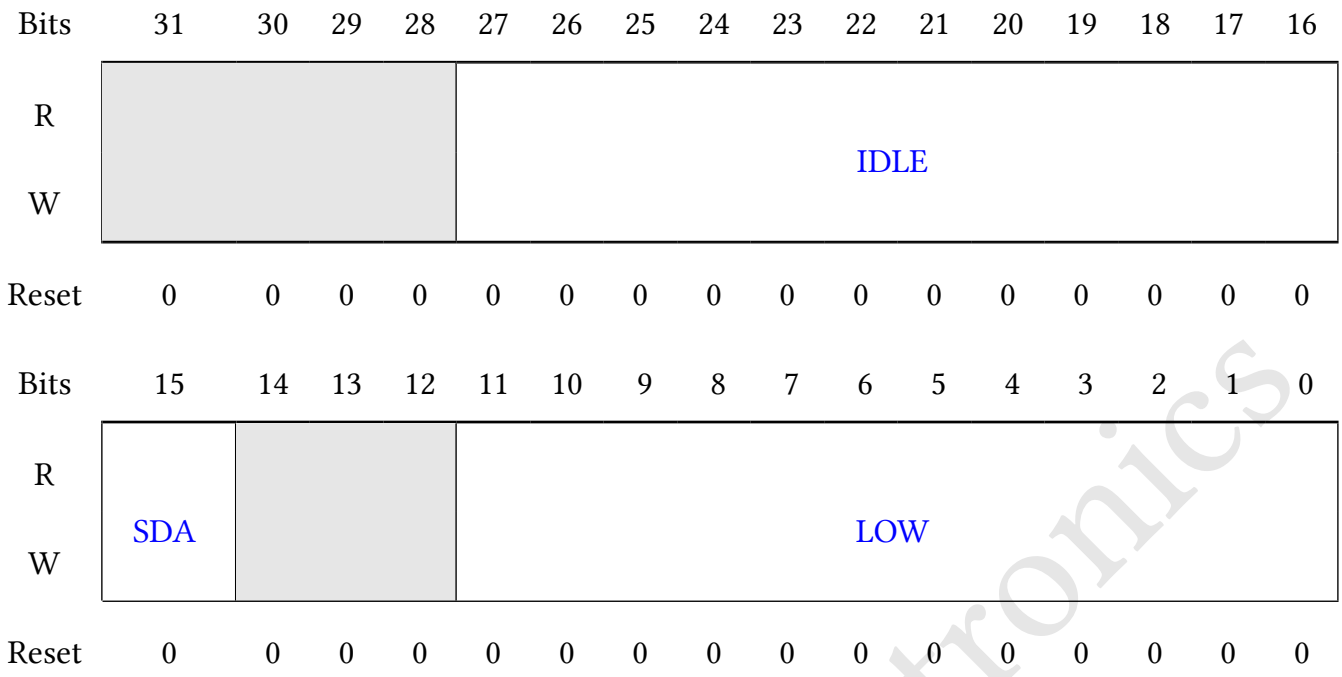


Table 20.15: I2C TOCFG Register Description

Field	Function
27 - 16 IDLE	**Line Idle Timeout Control** Line idle flag set when SDA and SCL line high for (IDLE) * divided function clock, after line idle, master can send out START.
15 SDA	**SDA Line Low Detect** Set this bit to enable SDA line low detection 0b - Line low only detect SCL 1b - Line low detect SDA and SCL
11 - 0 LOW	**Line Low Timeout Control** Line low period is (LOW * 256) * divided function clock

Yuntu Microelectronics

20.3.1.11 I2C HSCLK Register

Offset: 28h

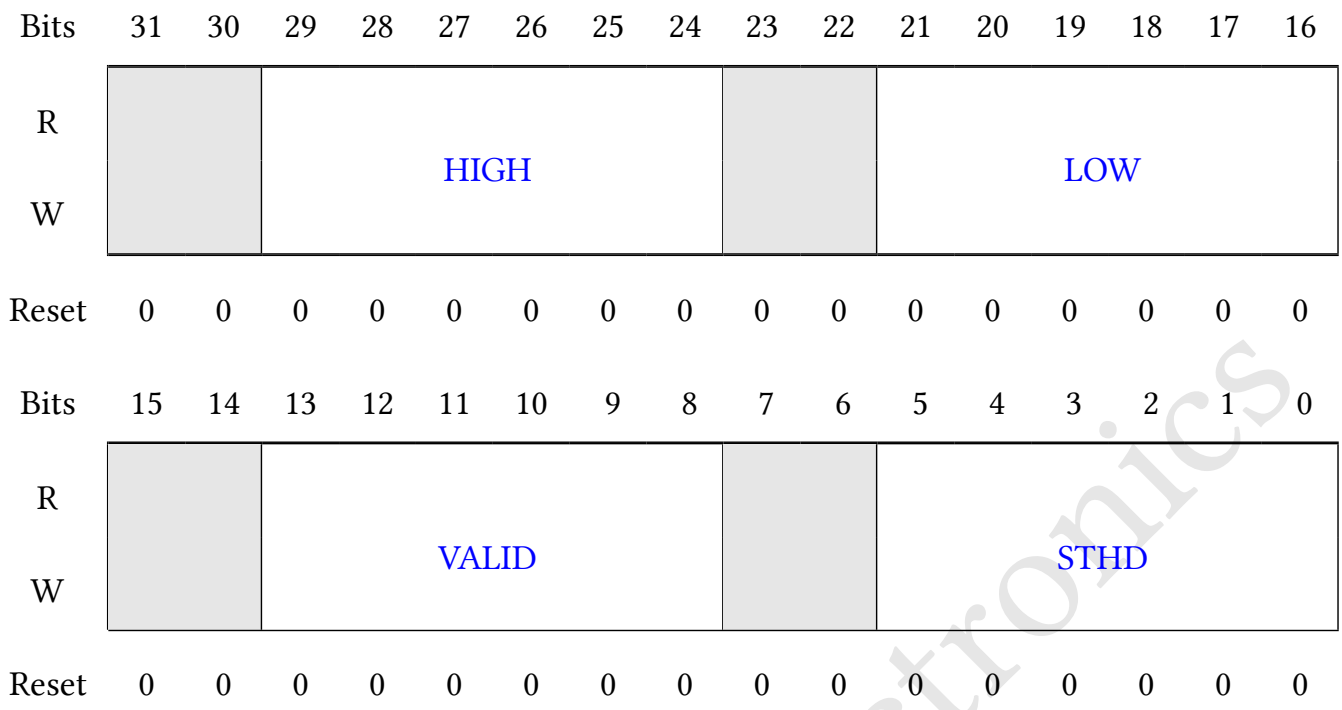


Table 20.16: I2C HSCLK Register Description

Field	Function
29 - 24 HIGH	**SCL High Period** SCL clock high period is (HIGH + 1) * divided clock period
21 - 16 LOW	**SCL Low Period** SCL clock low period is (LOW + 1) * divided clock period
13 - 8 VALID	**SDA Data Valid Period** SDA data valid period is (VALID + 1) * divided clock period
5 - 0 STHD	**SDA Data Setup/Hold Period** SDA data setup/hold period is (STHD + 1) * divided clock period

Yuntu Microelectronics

20.3.1.12 I2C SCTRL Register

Offset: 100h

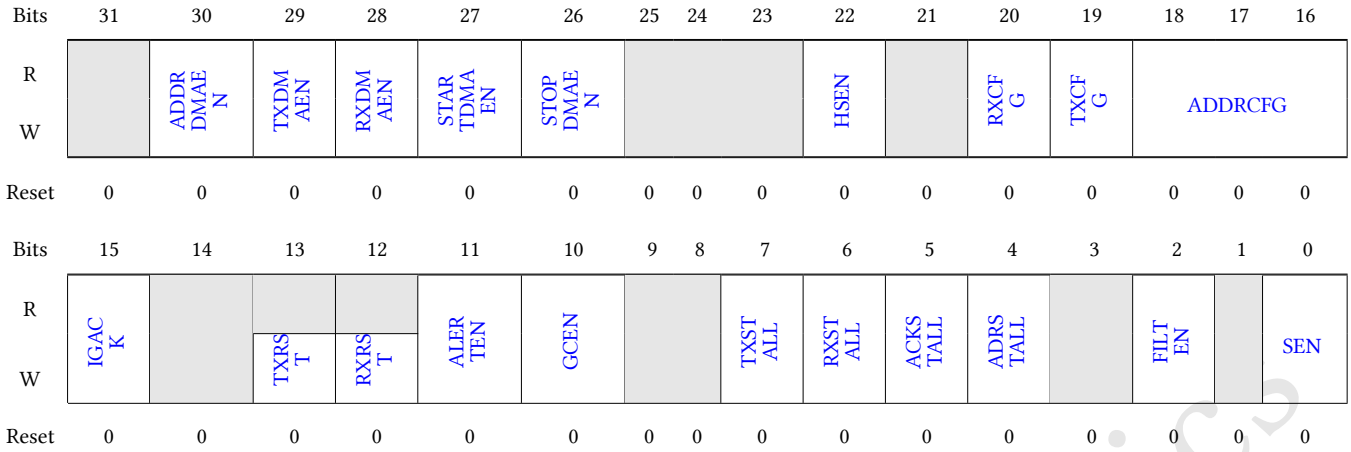


Table 20.17: I2C SCTRL Register Description

Field	Function
30 ADDRDMAEN	**Address Valid DMA Enable** Control weather DMA request generated after a valid address received 0b - DMA request disabled 1b - DMA request enabled
29 TXDMAEN	**TX data DMA Enable** TX DATA request DMA enable when TX buff empty 0b - DMA request disabled 1b - DMA request enabled
28 RXDMAEN	**RX data DMA Enable** RX DATA request DMA enable when RX buff full 0b - DMA request disabled 1b - DMA request enabled
27 STARTDMAEN	**Receive START DMA Enable** Control weather DMA request generated after a START condition received 0b - DMA request disabled 1b - DMA request enabled
26 STOPDMAEN	**Receive STOP DMA Enable** Control weather DMA request generated after a STOP condition received 0b - DMA request disabled 1b - DMA request enabled
22 HSEN	**High Speed Mode Enable** Setting this bit enables I2C salve high speed address match and switch enable 0b - Disable high speed mode switch 1b - Enable high speed mode switch
20	**RX Data Config**

Yuntu Microelectronics

Field	Function
RXCFG	Controls weather address include in reading RX data register 0b - Reading RX data register will only return received data and clear RX data flag 1b - Reading RX data register will only return received address when valid address mode flag set, and clear address valid flag. Reading RX data when address valid flag clear will return received data and clear RX data flag
19 TXCFG	**TX Data Config** Controls weather TX data flag asserted when I2C slave idle 0b - TX data flag only asserted when I2C active and TX buffer empty 1b - TX data flag asserted when TX buffer empty
18 - 16 ADDRCFG	**Slave Address Match Config** Control slave address match configuration 000b - Address match 0 (7-bit) 001b - Address match 0 (10-bit) 010b - Address match 0 (7-bit) or match 1(7-bit) 011b - Address match 0 (10-bit) or match 1(10-bit) 100b - Address match 0 (7-bit) or match 1(10-bit) 101b - Address match 0 (10-bit) or match 1(7-bit) 110b - Address match from match 0 (7-bit) to match 1(7-bit) 111b - Address match from match 0 (10-bit) to match 1(10-bit)
15 IGACK	**Ignore I2C ACK** Setting this bit will let I2C generate internal ACK even if NACK received. **NOTE**. This bit should be set in Ultra-Fast mode. 0b - I2C will receive ACK and NACK normally 1b - I2C will treat received NACK as ACK
13 TXRST	**TX buffer reset** Reset TX buffer pointer 0b - No effect 1b - Reset TX buffer pointer
12 RXRST	**RX buffer reset** Reset RX buffer pointer 0b - No effect 1b - Reset RX buffer pointer
11 ALERTEN	**Alert Address Detect Enable** Setting this bit will let I2C slave set SMBus alert address(0001100xb) as a match condition 0b - Alert address detect is disabled 1b - Alert address detect is enabled
10	**General Call Address Detect Enable**

Yuntu Microelectronics

Field	Function
GCEN	Setting this bit will let I2C slave set general call address(0000000xb) as a match condition 0b - General call address detect is disabled 1b - General call address detect is enabled
7 TXSTALL	**TX SCL Stall Enable** Enables I2C slave stretching SCL line to low when slave TX buffer is empty 0b - Slave stretching is disabled 1b - Slave stretching is enabled
6 RXSTALL	**RX SCL Stall Enable** Enables I2C slave stretching SCL line to low when slave RX buffer is full 0b - Slave stretching is disabled 1b - Slave stretching is enabled
5 ACKSTALL	**ACK SCL Stall Enable** Enables I2C slave stretching SCL line to low when slave waiting for software write ACK register **NOTE**. Setting ACKSTALL will set RXSTALL and ADRSTALL automatically. 0b - Slave stretching is disabled 1b - Slave stretching is enabled
4 ADRSTALL	**Address SCL Stall Enable** Enables I2C slave stretching SCL line to low when valid address received but not read 0b - Slave stretching is disabled 1b - Slave stretching is enabled
2 FILTEN	**Slave Digital Filter Enable** Slave digital filter not update when I2C slave enabled 0b - Digital filter is disabled 1b - Digital filter is enabled
0 SEN	**Slave Module Enable** Setting this bit will enable I2C slave logic 0b - Slave module is disabled 1b - Slave module is enabled

Yuntu Microelectronics

20.3.1.13 I2C SCLKCFG Register

Offset: 104h

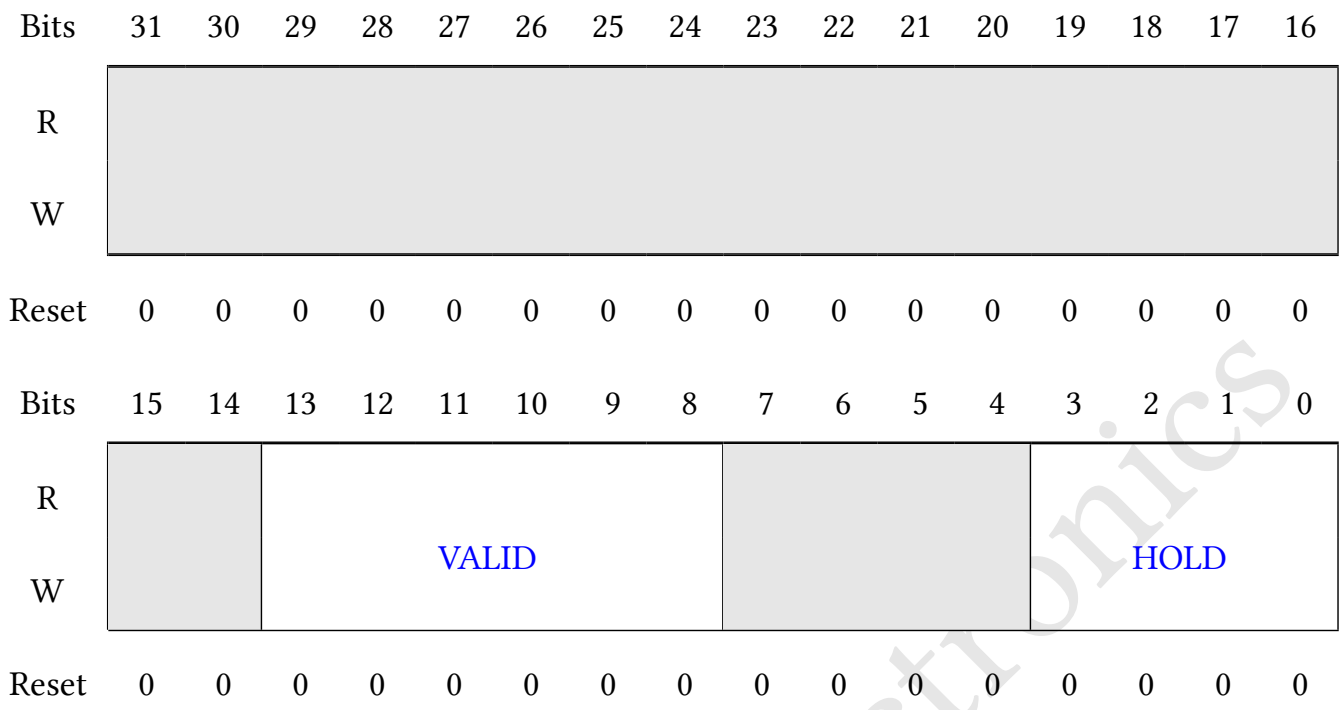


Table 20.18: I2C SCLKCFG Register Description

Field	Function
13 - 8 VALID	**Data Valid Period** SDA data delay is (FILTSCL + VALID + 3) function clock (not divided clock)
3 - 0 HOLD	**Data Hold Period** Data hold delay is (HOLD + 3) function clock (not divided clock)

Yuntu Microelectronics

20.3.1.14 I2C SFLTCFG Register

Offset: 108h

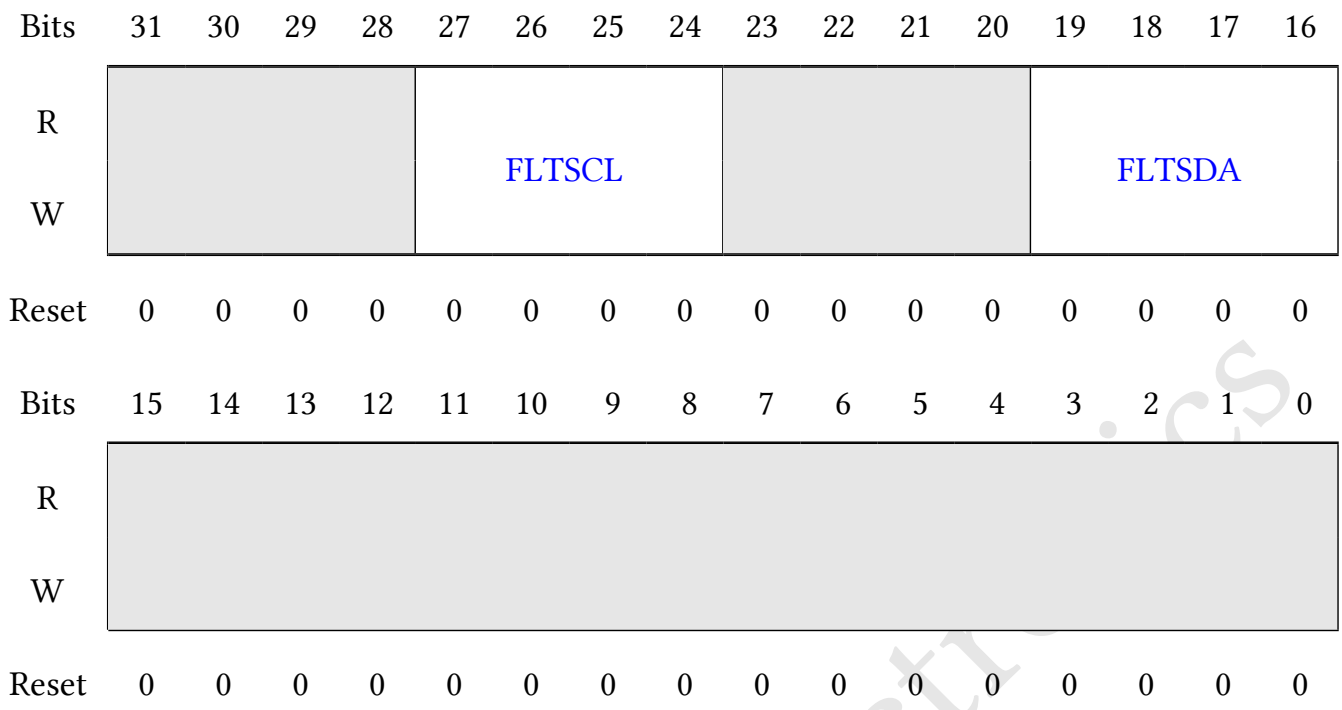


Table 20.19: I2C SFLTCFG Register Description

Field	Function
27 - 24 FLTSCL	**SCL Filter Control** Control digital filter on SCL line SCL digital filter use undivided I2C function clock Noise less or equal FLTSCL * function period will be filtered
19 - 16 FLTSDA	**SDA Filter Control** Control digital filter on SDA line SDA digital filter use undivided I2C function clock Noise less or equal FLTSDA * function period will be filtered

Yuntu Microelectronics

20.3.1.15 I2C SSTS Register

Offset: 10ch

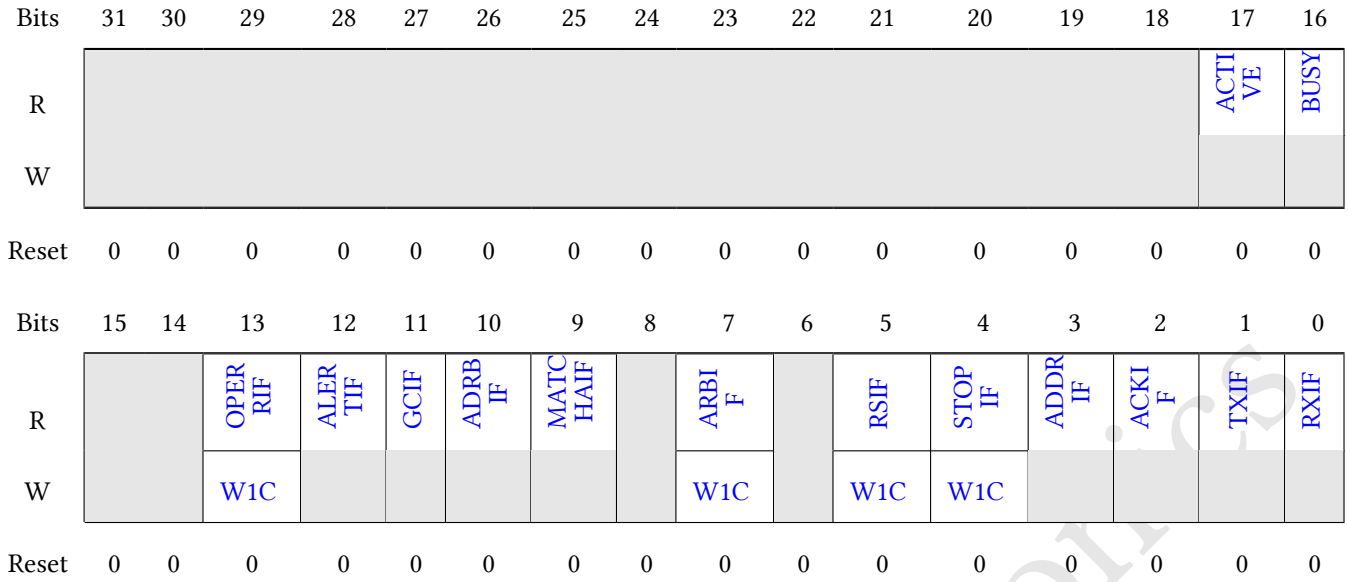


Table 20.20: I2C SSTS Register Description

Field	Function
17 ACTIVE	**I2C Slave Active** I2C slave is transmitting or receiving data 0b - I2C slave is idle 1b - I2C slave is busy
16 BUSY	**I2C Bus Busy** I2C Bus is transmitting or receiving data 0b - I2C bus is idle 1b - I2C bus is busy
13 OPERRIF	**Operation Error Interrupt Flag** I2C slave operation error, this bit is set when • Slave need to send data when TX buffer empty and TXSTALL not enabled • Slave received new data when RX buffer is full and RXSTALL not enabled • Slave received new address when address already valid and ADDRSTALL not enabled
12 ALERTIF	**Alert Address Received Interrupt Flag** 0b - Alert address not received 1b - Alert address received
11 GCIF	**General Call Address Received Interrupt Flag** 0b - General call address not received 1b - General call address received
10 ADRBIF	**Address B Received Interrupt Flag** Slave received address match address B flag 0b - Address B not received 1b - Address B received

Yuntu Microelectronics

Field	Function
9 MATCHAIF	**Address A Received Interrupt Flag** Slave received address match address A flag 0b - Address A not received 1b - Address A received
7 ARBIF	**Arbitration Lost Interrupt Flag** This bit is set when Slave send out logic 1 but detect logic 0 on SCL or SDA line Slave will ignore following transfer until new (repeat) START received 0b - I2C slave working normal 1b - I2C slave lost arbitration
5 RSIF	**Repeat Start Interrupt Flag** Repeat start set when I2C received a repeat START and previous address matched 0b - I2C slave not received repeat start 1b - I2C slave received repeat start
4 STOPIF	**Stop Interrupt Flag** Stop interrupt set when I2C already addressed and received a stop condition 0b - I2C slave not received STOP 1b - I2C slave received STOP
3 ADDRIF	**Address Valid Interrupt Flag** Address valid set when address match received and cleared after read address register or read DATA register when RXCFG bit set 0b - I2C slave not received any valid address 1b - I2C slave received a valid address
2 ACKIF	**Ack Request Interrupt Flag** Ack request set when slave need to send ACK or NACK to master, this bit cleared by write ACK register 0b - I2C slave don't need to send ACK or NACK to master 1b - I2C slave need to send ACK or NACK to master
1 TXIF	**TX Request Flag** Indicates words in TX FIFO is equal or less than TX FIFO watermark setting 0b - Word in TX FIFO is greater than watermark 1b - Word in TX FIFO is equal or less than watermark, request to write new data
0 RXIF	**RX Request Flag** Indicates words in RX FIFO is greater than RX FIFO watermark setting 0b - Word in RX FIFO is equal or less than watermark 1b - Word in RX FIFO is greater than watermark, should read data register

Yuntu Microelectronics

20.3.1.16 I2C SIE Register

Offset: 110h

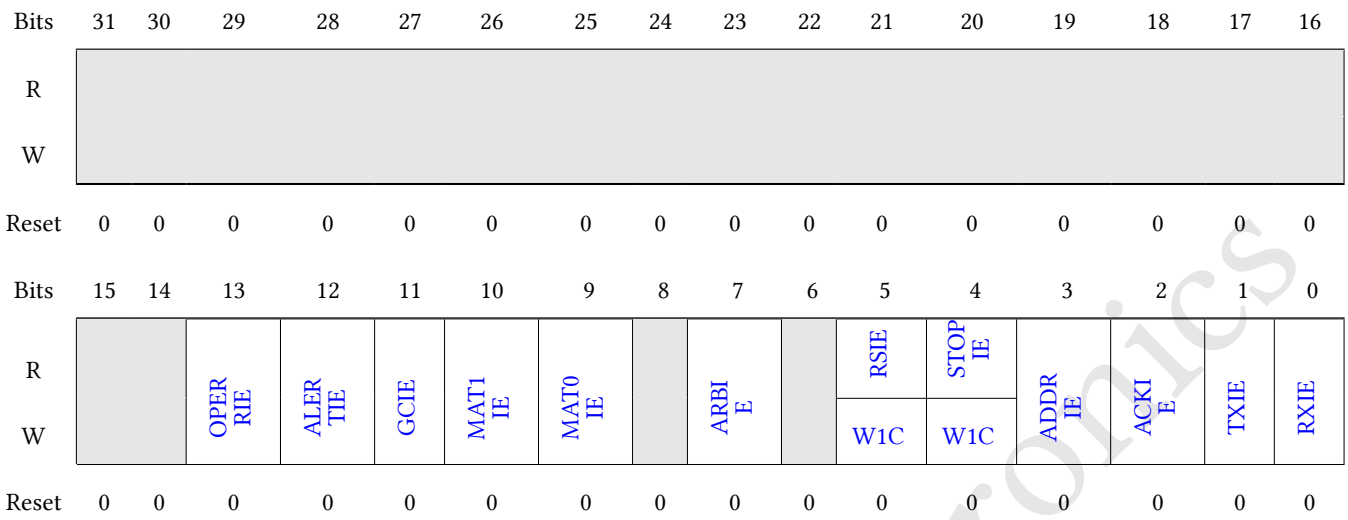


Table 20.21: I2C SIE Register Description

Field	Function
13 OPERRIE	**Slave Operation Error Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
12 ALERTIE	**Alert Address Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
11 GCIE	**General Call Address Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
10 MAT1IE	**Match Address 1 Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
9 MAT0IE	**Match Address 0 Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
7 ARBIE	**Arbitration Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
5	**Repeat Start Interrupt Enable**

Yuntu Microelectronics

Field	Function
RSIE	0b - Interrupt disabled 1b - Interrupt enabled
4 STOPIE	**Stop Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
3 ADDRIE	**Address Valid Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
2 ACKIE	**ACK Request Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
1 TXIE	**TX DATA Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled
0 RXIE	**RX DATA Interrupt Enable** 0b - Interrupt disabled 1b - Interrupt enabled

20.3.1.17 I2C SDATA Register

Offset: 114h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																EMPTY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									RXDATA							
W									TXDATA							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Yuntu Microelectronics

Table 20.22: I2C SDATA Register Description

Field	Function
16 EMPTY	**RX Buffer Empty** 0b - RX buffer is not empty 1b - RX buffer is empty
7 - 0 DATA	**Slave TX RX Data** Write will push data to TX buffer and read will return received data or address

20.3.1.18 I2C SACK Register

Offset: 118h

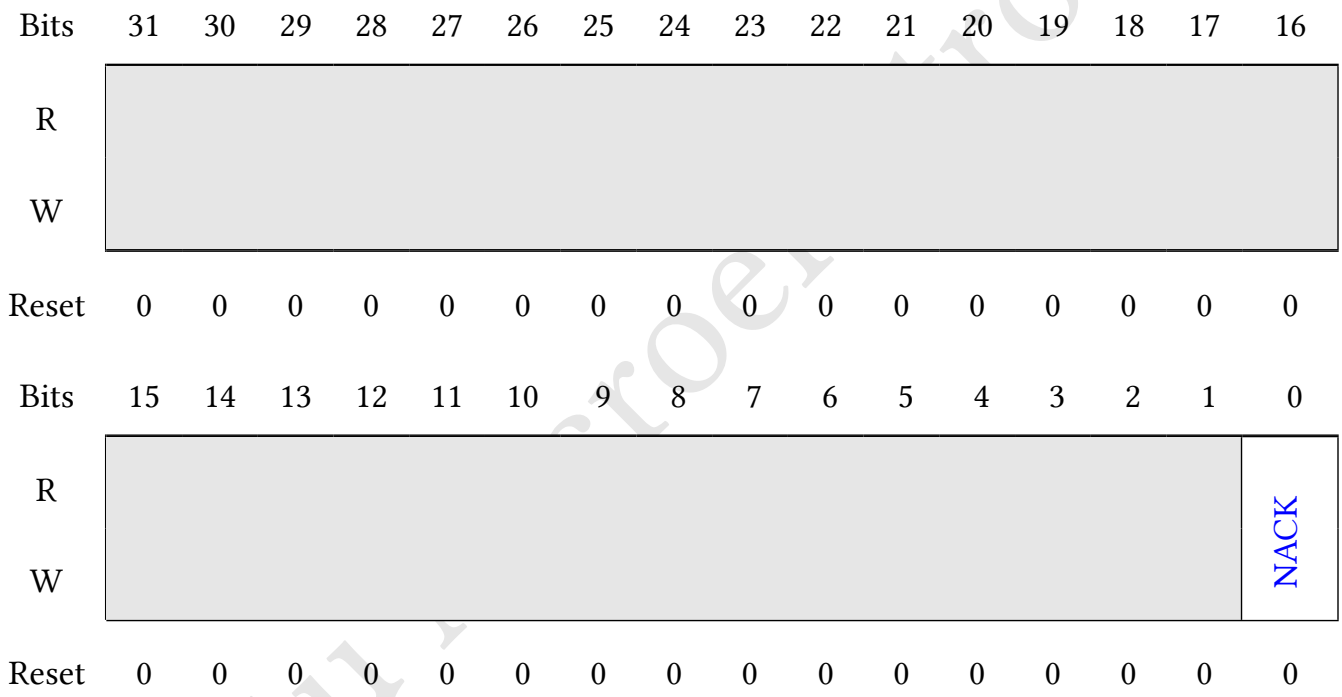


Table 20.23: I2C SACK Register Description

Field	Function
0 NACK	**Sent NACK** After slave received data, software can write NACK with 1 to send NACK or 0 to send ACK 0b - Send ACK 1b - Send NACK

20.3.1.19 I2C SADDR Register

Offset: 11ch

Yuntu Microelectronics

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

Table 20.24: I2C SADDR Register Description

Field	Function
26 - 16 ADDRB	**Slave ADDR B**
10 - 0 ADDRA	**Slave ADDR A**

20.3.1.20 I2C SRADDR Register

Offset: 120h

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

Yuntu Microelectronics

Table 20.25: I2C SRADDR Register Description

Field	Function
15 VALID	**Address Valid** 0b - Slave address is not valid 1b - Slave address is valid
10 - 0 ADDR	**Slave Received Address**

20.4 Functional Description

The I2C module uses two physical lines for communication:

- Serial Data Line (SDA) for data transfer
- Serial Clock Line (SCL) for the bus clock

A transaction starts with the I2C master sending the start condition, followed by address and a direction bit (read or write to/from the slave).

The addressed I2C slave will then acknowledge (ACK) the address, and data packet transactions can begin. Every 9-bit data packet consists of 8 data bits followed by a one-bit reply indicating whether the data was acknowledged or not.

If a data packet is not acknowledged (NACK), whether by the I2C slave or master, the I2C master takes action by either terminating the transaction by sending the stop condition, or by sending a repeated start to transfer more data.

20.4.1 I2C Functional Clock

There are a few clock sources for the I2C functional clock, and it can be configured by IPC. It is asynchronous to the bus clock. Whether I2C is able to work in the low power modes depends on whether the I2C functional clock remains enabled. The clock used for timing control in the register MFILTCFG, SCLKCFG and SFLTCFG is the I2C functional clock, while the clock used for timing control in the register MCLKCFG and HSCLK is the divided I2C functional clock.

20.4.2 Master Mode

20.4.2.1 Tx and CMD FIFO

The Tx and CMD FIFO stores the command and the data together in one word. There are various command and data combination corresponding to different I2C operations.

- START + DATA: Generate START or repeated START and transmit the address in DATA.
- STOP + START + DATA: Generate START or repeated START and transmit the address in DATA. Expect NACK from slave.
- No command + DATA: Transmit the data in DATA.
- REC + DATA: Receive (DATA+1) bytes data.
- REC + STOP + DATA: Receive (DATA+1) bytes data and discard them.
- STOP: Generate STOP.

Yuntu Microelectronics

- HS + START + DATA: Generate START or repeated START and transmit the address in DATA with high speed mode.
- HS + STOP + START + DATA: Generate START or repeated START and transmit the address in DATA with high speed mode. Expect NACK from slave.
- HS + DATA: Transmit the data in DATA with high speed mode.

The transmitted or received commands can be inserted between START and STOP. But it needs repeated START before changing the transmission direction. The I2C master will automatically transmit a NACK on the last byte of a received data command unless the next command in the FIFO is also a received data command. The I2C master supports 10-bit addressing through a (repeated) START condition, followed by a transmitted data byte containing the second address byte, followed by any number of data bytes with the transmitted or received data.

20.4.2.2 Master Operations

If I2C is enabled, the I2C starts transmission when it is idle by checking the MSTS[BUSY] bit. When either SCL or SDA are low, the I2C bus is busy. The I2C bus becomes idle if a STOP condition or a bus idle timeout is detected. After the I2C bus is idle, the I2C will check Tx FIFO. If there is any data in Tx FIFO and the first word is read out, and the legal command will be executed. Below are the steps.

- Detect a STOP condition on the I2C bus or Wait the bus idle time equal to (MCLKCFG[LOW]+1) cycles divided clocks (MFLTCFG[DIV]).
- Transmit a START condition and address byte.
- Perform master-transmit or master-receive transfers, as configured by the Tx FIFO.
- Transmit NACK on the last byte of a master-receive transfer, unless the next command in the Tx FIFO is also a received data command.
- Transmit a Repeated START or STOP condition as configured by the Tx FIFO.

If the I2C master is disabled, the I2C will continue to empty the Tx FIFO or a STOP condition is transmitted.

The I2C master can stall the I2C bus under certain conditions; this will result in SCL pulled low continuously:

- I2C master is enabled and busy, the Tx FIFO is empty, and MCTRL[ASTOP] is clear.
- I2C master is enabled and receiving data, received data is not being discarded, and the Rx FIFO is full.

20.4.2.3 Rx FIFO and Data Matching

In Master mode, the Rx FIFO stores received data during master-receiver transfers. Receive data will be discarded instead of storing in the Rx FIFO when discarded command is configured.

The I2C supports a received data match function that can match received data against one of two bytes or against a masked data byte. The data match function can also be configured to compare only the first one or two received data words since the last (repeated) START condition. The receiver match function can also be configured to discard all receive data until a data match is detected, using the MCTRL[RDMO] control bit.

20.4.3 Slave Mode

20.4.3.1 Address Matching

The I2C slave can be configured with the address match conditions.

Yuntu Microelectronics

- Match one of the two addresses, both 7-bit and 10-bit addressing modes are supported.
- Match a range of addresses, both 7-bit and 10-bit addressing modes are supported.
- Match the General Call Address.
- Match the SMBus Alert Address.
- Detect the high speed mode master code, and then disable the digital filters use the valid delay time until the next STOP condition is detected.

After a valid address is matched, the I2C slave will automatically perform slave-transmit or slave-receive transfers until:

- A NACK is detected (unless SCTRL[IGNACK] is set).
- A bit error is detected (the I2C slave is driving SDA, but a different value is sampled)
- A (repeated) START or STOP condition is detected.

20.4.3.2 Data Processing

- The Tx and Rx Data registers are double-buffered
- The slave address that was received can be read from either the Slave Data register or the Slave Receive Address register.
- The Slave Data register can be configured to only request data whenever the Tx buffer is empty or a slave-transmit transfer is detected when the Tx buffer is empty.
- The Slave Data register should only be written when the SSTS[TXIF] is set.
- The Slave Receive Address register should only be read the received address when the Address Valid flag is set.

20.4.3.3 Clock Stretching

The I2C slave supports a mechanism for clock stretching. The following conditions can be configured to perform clock stretching.

- During the 9th clock pulse of the address byte and the Address Valid flag is set which enabled by SCTRL[ADRSTALL].
- During the 9th clock pulse of a slave-transmit transfer and the Transmit Data flag is set which enabled by SCTRL[TXSTALL].
- During the 9th clock pulse of a slave-receive transfer and the RX Request flag is set which enabled by SCTRL[RXSTALL].
- During the 8th clock pulse of an address byte or a slave-receive transfer and the ACK Request flag is set which enabled by SCTRL[ACKSTALL]. In high speed mode, this is disabled.
- Clock stretching can also be extended for CLKHOLD cycles, to allow additional setup time to sample the SDA pin externally. In high speed mode, this is disabled.

Yuntu Microelectronics

Part VII

Timer

Yuntu Microelectronics

Chapter 21

Periodic Timer (pTMR)

21.1 Chip-specific pTMR Information

21.1.1 Instantiation Information

YTM32B1LE0x contains one pTMR module with four channels of 32bit timers that can be used to count and raise an interrupt.

21.2 Introduction

21.2.1 Block Diagram

The following figure shows the pTMR block diagram.

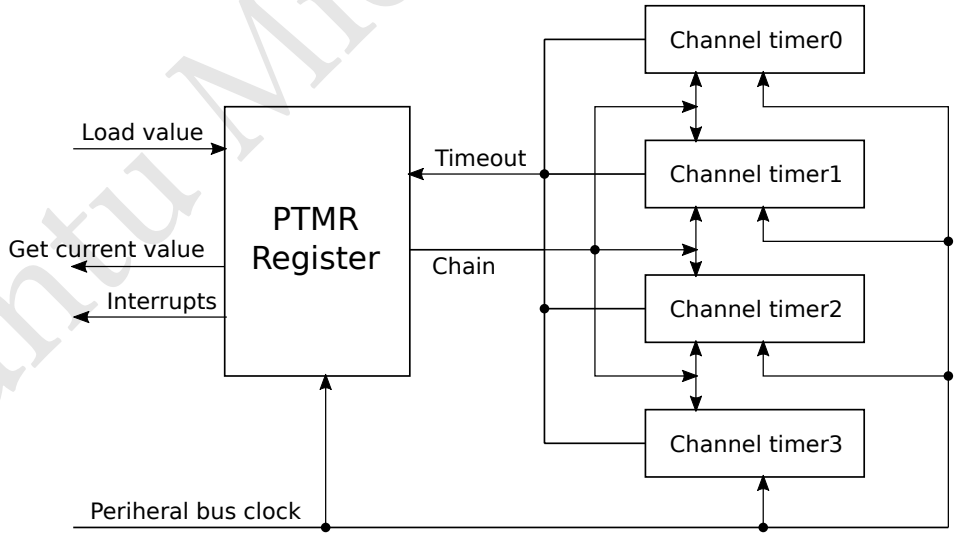


Figure 21.1: Block Diagram of pTMR

21.2.2 Features

- Timers can generate interrupts, and each channel can generate independent interrupt request
- Four channels of 32bit timers, each timer has independent timeout periods

Yuntu Microelectronics

- Ability to stop in debug mode
- Support chain mode to connect multiple timer to a longer timer

21.3 Register Definition

The pTMR includes registers to control pTMR module enable or disable and control each channel timer enable or disable, load start value, get current count value, and enable or disable timer interrupt and flag. And it also can link 2/3/4 channels as a whole.

Accessing a reserved address will result in a bus error.

Register_address = Base_address + Address_offset
For example: as CHn_TSV
Address_offset = 0x100 + n * 0x10 (n=0/1/2/3)

21.3.1 pTMR Memory Map

pTMR0 base address 40037000h

Table 21.1: pTMR Memory Map

Offset	Register	Width	Access	Reset Value
0h	Module Control Register (MCR)	32	RW	00000000h
200h	Timer Start Value Register (CH0_TSV)	32	RW	00000000h
204h	Timer Current Value Register (CH0_TCV)	32	RW	00000000h
208h	Timer Control Register (CH0_TCR)	32	RW	00000000h
20ch	Timer Flag Register (CH0_TFR)	32	RW	00000000h
210h	Timer Start Value Register (CH1_TSV)	32	RW	00000000h
214h	Timer Current Value Register (CH1_TCV)	32	RW	00000000h
218h	Timer Control Register (CH1_TCR)	32	RW	00000000h
21ch	Timer Flag Register (CH1_TFR)	32	RW	00000000h
220h	Timer Start Value Register (CH2_TSV)	32	RW	00000000h
224h	Timer Current Value Register (CH2_TCV)	32	RW	00000000h
228h	Timer Control Register (CH2_TCR)	32	RW	00000000h
22ch	Timer Flag Register (CH2_TFR)	32	RW	00000000h
230h	Timer Start Value Register (CH3_TSV)	32	RW	00000000h
234h	Timer Current Value Register (CH3_TCV)	32	RW	00000000h
238h	Timer Control Register (CH3_TCR)	32	RW	00000000h
23ch	Timer Flag Register (CH3_TFR)	32	RW	00000000h

21.3.1.1 pTMR MCR Register

Offset: 0h

Yuntu Microelectronics

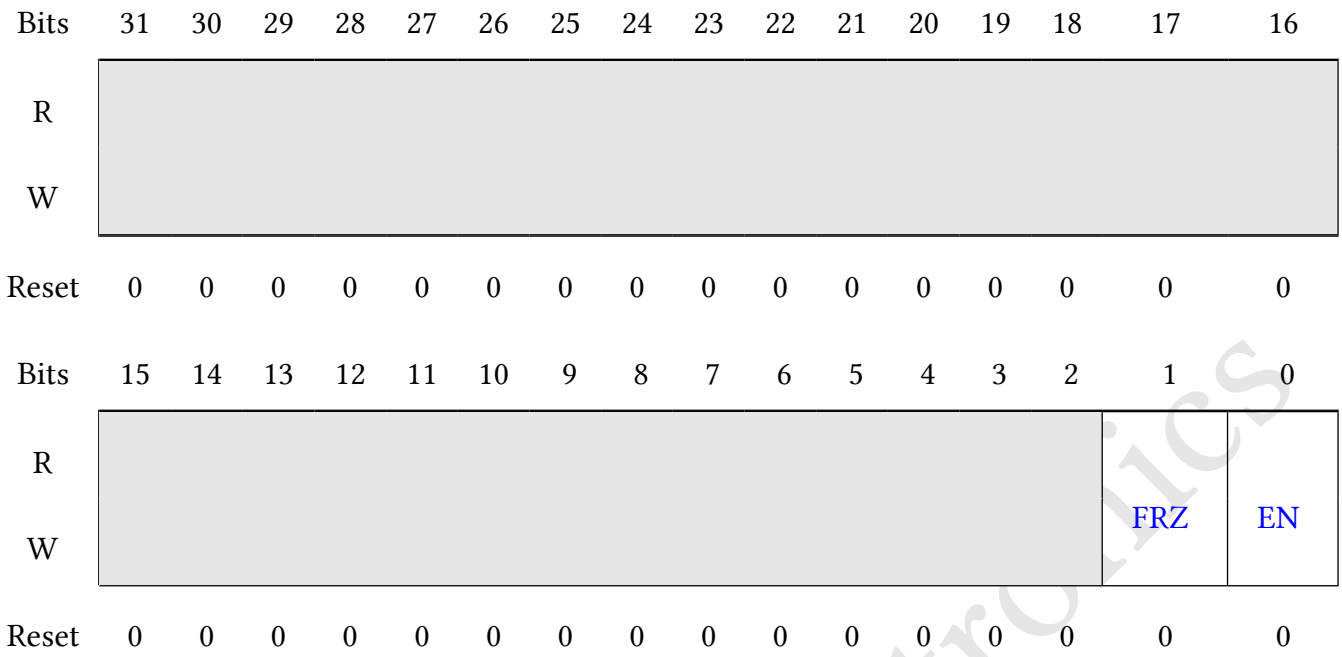


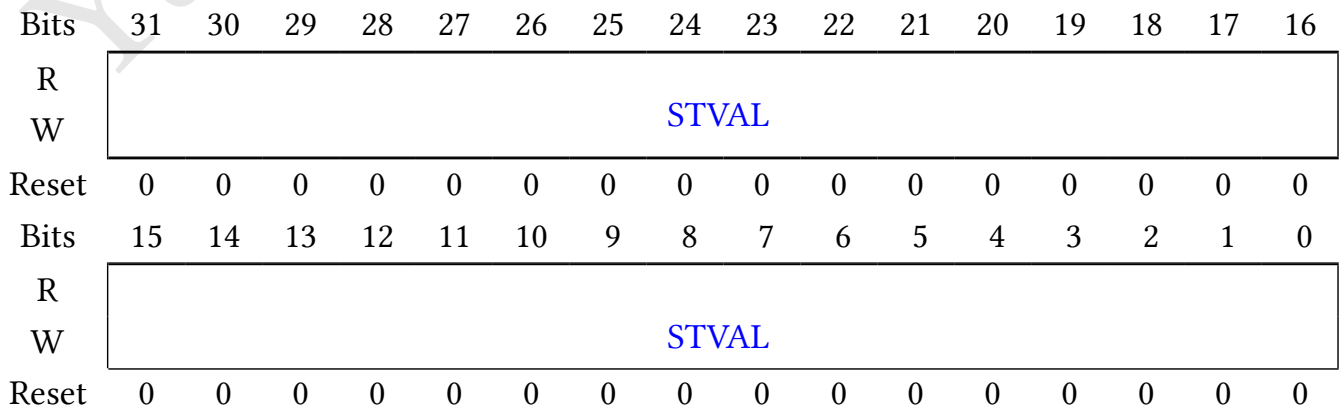
Table 21.2: pTMR MCR Register Description

Field	Function
1 FRZ	**Freeze** Allows the timers to be stopped when the device enters debug mode. 0b - pTMR continues to run in debug mode 1b - pTMR stops in debug mode
0 EN	**pTMR Enable** Enable pTMR module 0b - pTMR is disabled 1b - pTMR is enabled

21.3.1.2 pTMR CH_TSV Register

Timer Start Value Register

Offset: 200h + n * 10h



Yuntu Microelectronics

Table 21.3: pTMR CH_TSV Register Description

Field	Function
31 - 0 STVAL	**Start Value** This register set the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and loads the value again. Writing a new value to this register will not restart the timer, instead the value will be loaded once the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again.

21.3.1.3 pTMR CH_TCV Register

Timer Current Value Register

Offset: 204h + n * 10h

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

Table 21.4: pTMR CH_TCV Register Description

Field	Function
31 - 0 CVAL	**Current Value** This register stands for current timer value when user read this register. The value of this register will be frozen in debug mode if MCR[FRZ] is set.

21.3.1.4 pTMR CH_TCR Register

Timer Control Register

Offset: 208h + n * 10h

Yuntu Microelectronics

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

Table 21.5: pTMR CH_TCR Register Description

Field	Function
2 CHAIN	**Chain Mode** When setting this bit, it will link multi-channel in sequence to combine a new bits counter, it just links the front channel. 0b - Chaining is disabled, channel timer runs independently 1b - Chaining is enabled
1 TIE	**Timer Interrupt Enable** Enable interrupt when setting this bit 0b - Disable timer interrupt of corresponding channel 1b - Enable timer interrupt of corresponding channel
0 TEN	**Timer Enable** Enable the timer of corresponding channel 0b - Disable the timer of corresponding channel 1b - Enable the timer of corresponding channel

21.3.1.5 pTMR CH_TFR Register

Timer Flag Register

Offset: 20ch + n * 10h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W																TIF W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Yuntu Microelectronics

Table 21.6: pTMR CH_TFR Register Description

Field	Function
0 TIF	**Timer Interrupt Flag** Hardware will set this bit as 1 when timer interrupt is enabled, and writing logic 1 to this bit will clear this bit as 0. 0b - Timeout did not occur 1b - Timeout occurred

21.4 Functional Description

21.4.1 Timers

These timers load start values in the CH_TSV register, then count down until it reaches 0, then they load their respective start value. Once one timer reaches 0, it will generate an interrupt and set the interrupt flag.

Interrupt can be enabled or masked by setting TIE bit in the CH_TCR register of each channel. A new interrupt can be generated after the former one is cleared. When user wants to clear the interrupt flag, TIF bit in the CH_TFR register should be set. When it is set to 1, it will be cleared.

The current counter value of each channel timer could be read from CH_TCV register of its own.

The counter period can be restarted by firstly disabling, then enabling the timer with the TEN bit in the CH_TCR register.

The counter period of a running timer can be modified, by firstly disabling the timer, setting a new value to load and then enabling the timer again.

21.4.2 Chain Mode

These timers can be linked as a longer timer, setting CHAIN bit in the CH_TCR register to 1 will enable the chain mode. In this mode, timers only link with the front one, and timeout interrupt of each timer can work independently. For example, if set CHAIN in CH1_TCR, then timer0 and timer1 are linked to a 64-bit timer, timer1 timeout will be longer.

21.4.3 Debug Mode

These timers will be frozen in the debug mode when MCR[FRZ] bit is set to enable. Its purpose is to aid software engineer to halt the processor, investigate the current state of the system and then continue the operation.

21.4.4 Interrupt

pTMR module supports four interrupts generated by four channels, each channel can generate interrupt individually. Please refer to the table of interrupt vectors. Timer interrupt can be disabled by setting the TIE bit to 0. The timer interrupt flag(TIF bit) is set to 1 while timeout occurs, and interrupt flag can be cleared by setting TIF bit from CH_TFR register to 1.

Yuntu Microelectronics

Chapter 22

Low Power Timer (lpTMR)

22.1 Introduction

The low-power timer (lpTMR) can be configured to operate as a time counter with optional prescaler, or as a pulse counter with optional glitch filter, across all power modes.

22.1.1 Features

The features of the lpTMR module include:

- 16-bit time counter or pulse counter with compare
  - Optional interrupt can generate asynchronous wakeup from any low-power mode
  - Hardware trigger output
  - Counter supports free-running mode or reset on compare
- Configurable clock source for prescaler/glitch filter
- Configurable input source for pulse counter
  - Rising-edge or falling-edge

22.2 Register Definition

This subsection provides a description of the memory map and registers.

22.2.1 lpTMR Memory Map

lpTMR0 base address 40040000h

Table 22.1: lpTMR Memory Map

Offset	Register	Width	Access	Reset Value
0h	Control Register (CTRL)	32	RW	00000000h
4h	Prescaler Register (PRS)	32	RW	00000000h
8h	DMA and Interrupt Enable Register (DIE)	32	RW	00000000h
ch	Status Register (STS)	32	RW	00000000h
10h	Compare Value Register (CMP)	32	RW	00000000h
14h	Latch Timer Counter Register (LCNT)	32	WO	00000000h
18h	Timer Current Counter Register (CNT)	32	RW	00000000h

Yuntu Microelectronics

22.2.1.1 lpTMR CTRL Register

Offset: 0h

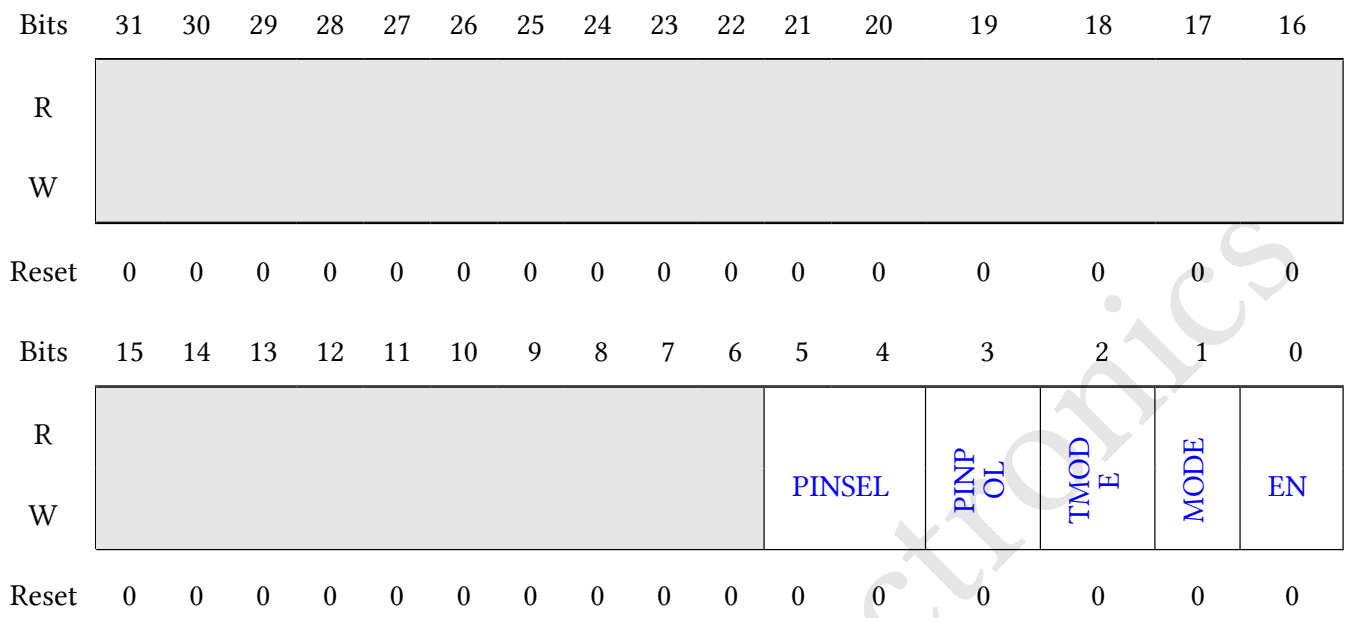


Table 22.2: lpTMR CTRL Register Description

Field	Function
5 - 4 PINSEL	**Timer Pin Select** Configures the input source to be used in Pulse Counter mode. PINSEL must be altered only when the lpTMR is disabled. 00b - Pulse counter input 0 is selected. (It's from lpTMR_ALT0 pad) 01b - Pulse counter input 1 is selected. (It's from lpTMR_ALT1 pad) 10b - Pulse counter input 2 is selected. (It's from lpTMR_ALT2 pad) 11b - Pulse counter input 3 is selected. (It's from lpTMR_ALT3 pad)
3 PINPOL	**Timer Pin Polarity** Configures the polarity of the input source in Pulse Counter mode. PINPOL must be changed only when the lpTMR is disabled. 0b - Pulse Counter input source is active-high, and the CNT will increment on the rising-edge. 1b - Pulse Counter input source is active-low, and the CNT will increment on the falling-edge.
2 TMODE	**Timer Free-Running Mode** When clear, TMODE configures the CNT to reset whenever CCF is set. When set, TMODE configures the CNT to reset on overflow. TMODE must be altered only when the lpTMR is disabled. 0b - CNT is reset whenever CCF is set. 1b - CNT is reset on overflow.
1	**Timer Mode Select**

Yuntu Microelectronics

Field	Function
MODE	Configures the mode of the lpTMR. MODE must be altered only when the lpTMR is disabled. 0b - Time Counter mode. 1b - Pulse Counter mode.
0 EN	**Timer Enable** When EN is clear, it resets the lpTMR internal logic, including the CNT and CCF. When EN is set, the lpTMR is enabled. While writing 1 to this field, CTRL[5:1] must not be altered. 0b - lpTMR is disabled and internal logic is reset. 1b - lpTMR is enabled.

22.2.1.2 lpTMR PRS Register

Offset: 4h

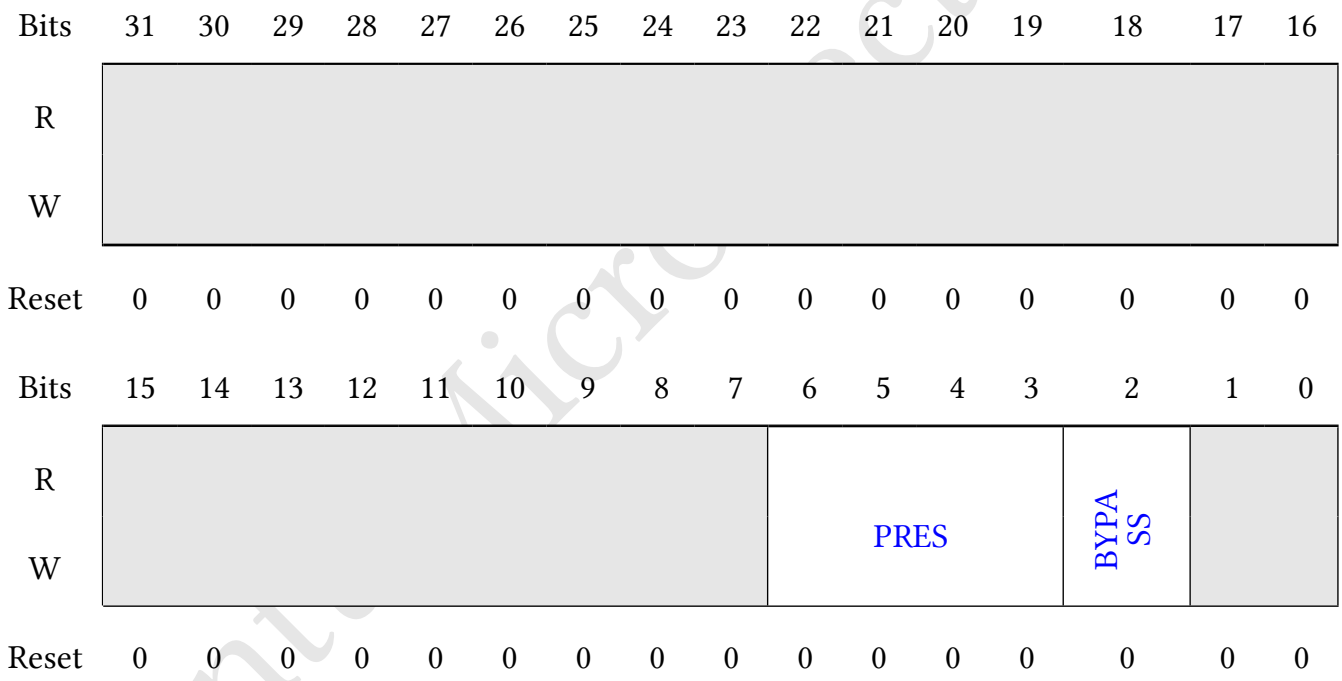


Table 22.3: lpTMR PRS Register Description

Field	Function
6 - 3	**Prescaler Value**

Yuntu Microelectronics

Field	Function
PRES	Configures the size of the Prescaler in Time Counter mode or width of the glitch filter in Pulse Counter mode. The width of the glitch filter can vary by 1 cycle due to synchronization of the pulse counter input. PRESCALE must be altered only when the lpTMR is disabled. 0000b - Prescaler divides the prescaler clock by 2; glitch filter does not support this configuration. 0001b - Prescaler divides the prescaler clock by 4; glitch filter recognizes change on input pin after 2 rising clock edges. 0010b - Prescaler divides the prescaler clock by 8; glitch filter recognizes change on input pin after 4 rising clock edges. 0011b - Prescaler divides the prescaler clock by 16; glitch filter recognizes change on input pin after 8 rising clock edges. 0100b - Prescaler divides the prescaler clock by 32; glitch filter recognizes change on input pin after 16 rising clock edges. 0101b - Prescaler divides the prescaler clock by 64; glitch filter recognizes change on input pin after 32 rising clock edges. 0110b - Prescaler divides the prescaler clock by 128; glitch filter recognizes change on input pin after 64 rising clock edges. 0111b - Prescaler divides the prescaler clock by 256; glitch filter recognizes change on input pin after 128 rising clock edges. 1000b - Prescaler divides the prescaler clock by 512; glitch filter recognizes change on input pin after 256 rising clock edges. 1001b - Prescaler divides the prescaler clock by 1024; glitch filter recognizes change on input pin after 512 rising clock edges. 1010b - Prescaler divides the prescaler clock by 2048; glitch filter recognizes change on input pin after 1024 rising clock edges. 1011b - Prescaler divides the prescaler clock by 4096; glitch filter recognizes change on input pin after 2048 rising clock edges. 1100b - Prescaler divides the prescaler clock by 8192; glitch filter recognizes change on input pin after 4096 rising clock edges. 1101b - Prescaler divides the prescaler clock by 16384; glitch filter recognizes change on input pin after 8192 rising clock edges. 1110b - Prescaler divides the prescaler clock by 32768; glitch filter recognizes change on input pin after 16384 rising clock edges. 1111b - Prescaler divides the prescaler clock by 65536; glitch filter recognizes change on input pin after 32768 rising clock edges.
2 BYPASS	**Prescaler Bypass** When BYPASS is set, the selected prescaler clock in Time Counter mode or selected input source in Pulse Counter mode directly clocks the CNT. When BYPASS is clear, the CNT is clocked by the output of the prescaler/glitch filter. BYPASS must be altered only when the lpTMR is disabled.

22.2.1.3 lpTMR DIE Register

Offset: 8h

Yuntu Microelectronics

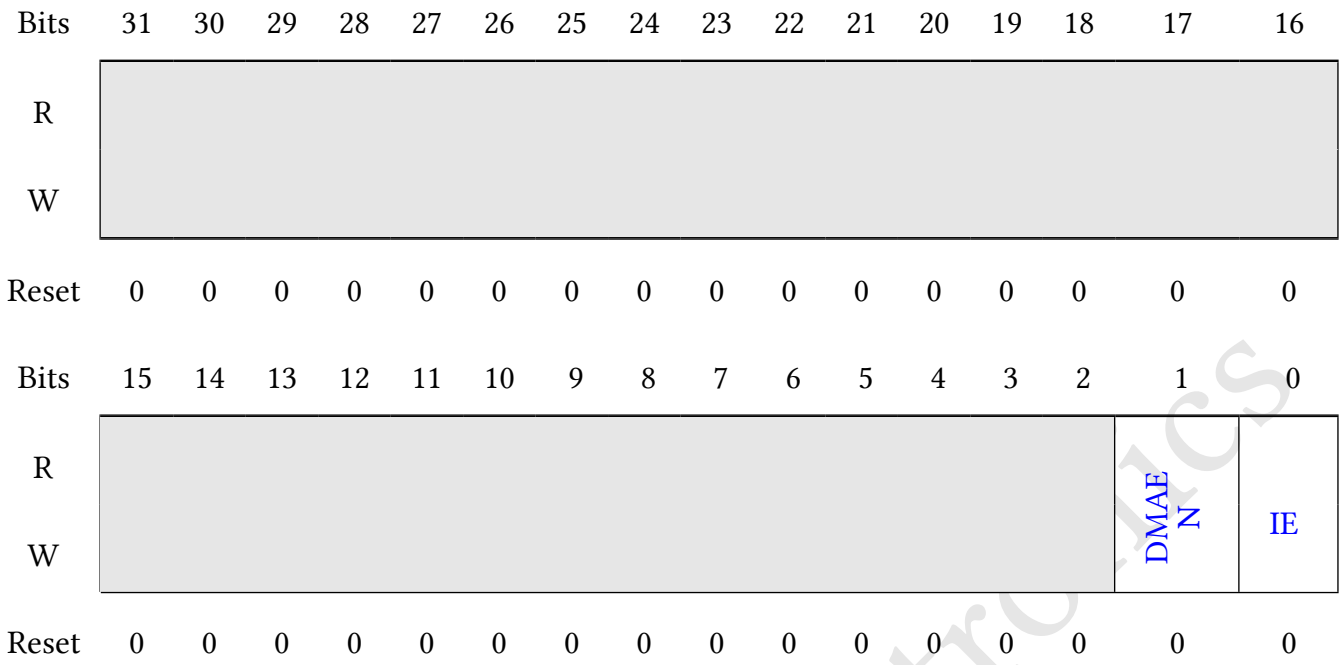


Table 22.4: lpTMR DIE Register Description

Field	Function
1 DMAEN	**Timer DMA Request Enable** When TDRE is set, the lpTMR DMA Request is generated whenever CCF is also set and the CCF is cleared when the DMA Controller is done. 0b - Timer DMA Request disabled. 1b - Timer DMA Request enabled.
0 IE	**Timer Interrupt Enable** When IE is set, the lpTMR Interrupt is generated whenever CCF is also set. 0b - Timer interrupt disabled. 1b - Timer interrupt enabled.

Yuntu Microelectronics

22.2.1.4 lpTMR STS Register

Offset: ch

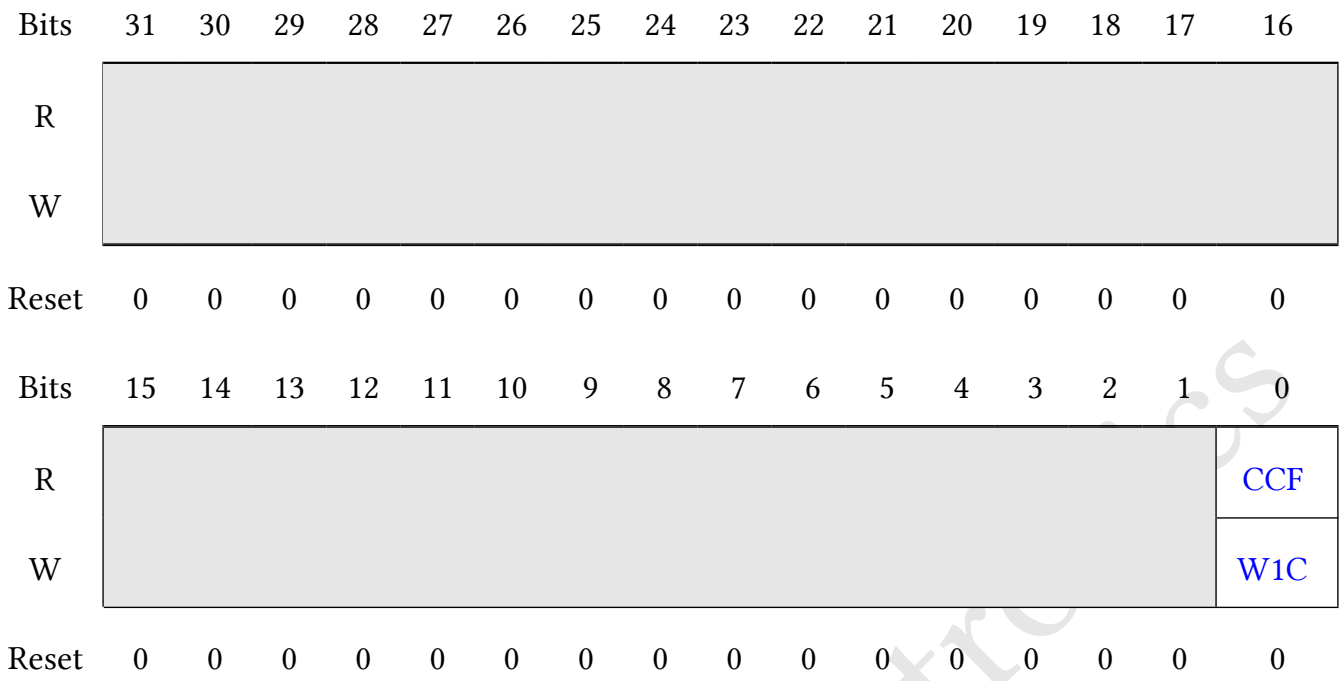


Table 22.5: lpTMR STS Register Description

Field	Function
0 CCF	**Counter Compare Flag** CCF is set when the lpTMR is enabled and the CNT equals the CMP and increments. CCF is cleared when the lpTMR is disabled or writing 1 to it. 0b - The value of CNT is not equal to CMP and increments. 1b - The value of CNT is equal to CMP and increments.

Yuntu Microelectronics

22.2.1.5 lpTMR CMP Register

Offset: 10h

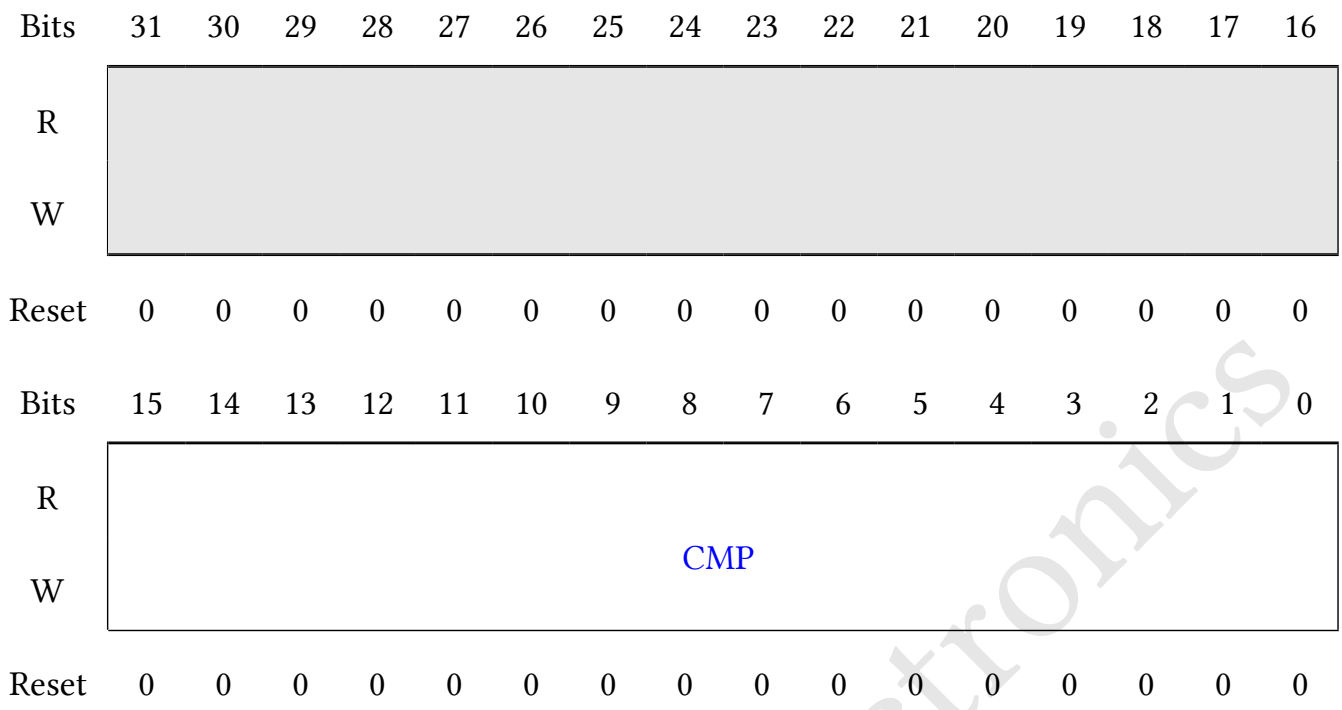


Table 22.6: lpTMR CMP Register Description

Field	Function
15 - 0 CMP	**Compare Value** When the lpTMR is enabled and the CNT equals the value in the CMP and increments, CCF is set and the hardware trigger asserts until the next time the CNT increments. If the CMP is 0, the hardware trigger will remain asserted until the lpTMR is disabled. If the lpTMR is enabled, the CMP must be altered only when CCF is set.

Yuntu Microelectronics

22.2.1.6 lpTMR LCNT Register

Offset: 14h

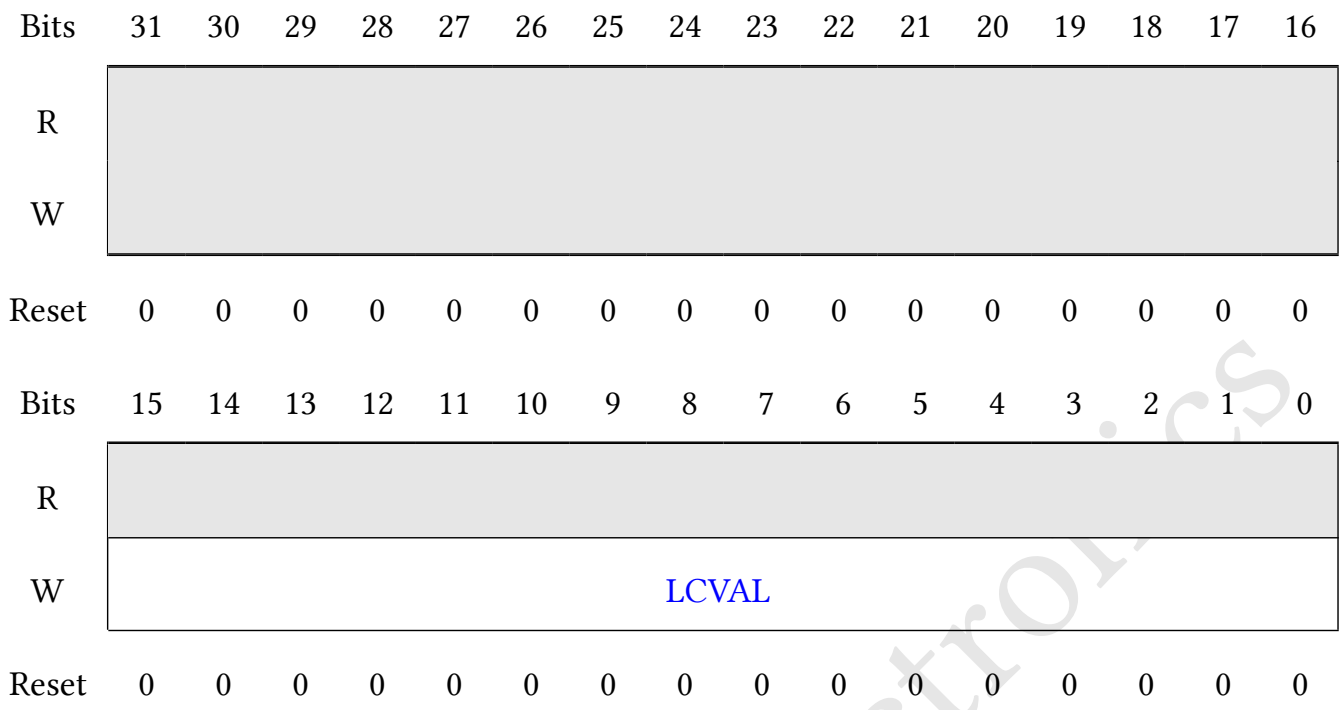


Table 22.7: lpTMR LCNT Register Description

Field	Function
15 - 0 LCVAL	**Latch Timer Counter Value** Write this register to latch timer counter value to CNT register, which is to avoid metastability.

Yuntu Microelectronics

22.2.1.7 lpTMR CNT Register

Offset: 18h

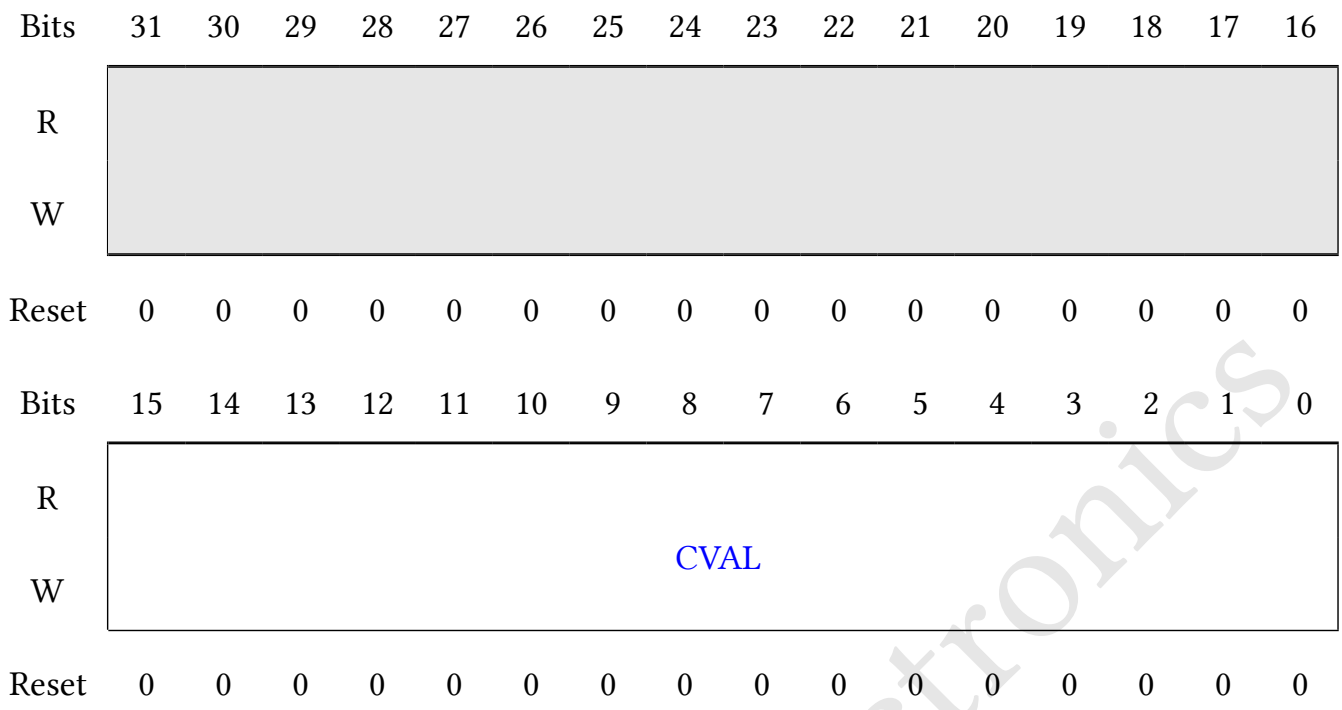


Table 22.8: lpTMR CNT Register Description

Field	Function
15 - 0 CVAL	**Timer Current Counter Value** The CNT returns the current value of the lpTMR counter at the time this LCNT register was last written.

22.3 Functional Description

22.3.1 lpTMR Clocking

The lpTMR prescaler/glitch filter can be clocked by one of the clock sources from IPC module. The clock source must be enabled before the lpTMR is enabled.

The clock source selected by IPC may need to be configured to selected the clock source which remains enabled in low-power modes, otherwise the lpTMR will not operate during low-power modes.

In Pulse Counter mode with the prescaler/glitch filter bypassed, the selected input source directly clocks the CNT and no other clock source is required.

22.3.2 lpTMR Prescaler/Glitch Filter

The lpTMR prescaler and glitch filter share the same logic, it can be configured as either prescaler in Time Counter mode or as a glitch filter in Pulse Counter mode.

NOTE. The prescaler/glitch filter configuration must not be changed when the lpTMR is enabled.

Yuntu Microelectronics

22.3.2.1 Time Counter

In Time Counter mode, when the prescaler and lpTMR is enabled, the CNT will increment with every output of prescaler. When prescaler is bypassed and lpTMR is enabled, the selected prescaler clock increments the CNT on every clock cycle.

22.3.2.2 Pulse Counter

In Pulse Counter mode, when the glitch filter is enabled, the output of the glitch filter directly clocks the CNT. When the lpTMR is first enabled, the output of the glitch filter is asserted, based on the value of CTRL[PINPOL] bit, that is, logic 1 for active-high and logic 0 for active-low.

The lpTMR CNT increments at each time that the glitch filter output asserts in pulse counter mode. The maximum rate for lpTMR CNT increments in pulse mode is 2^2 to 2^{16} prescaler clock edges based on the value of PRS[PRES] register.

In Pulse Counter mode, if the glitch filter is bypassed, the CNT increments on every assertion of the selected input source.

22.3.3 lpTMR Compare

When the CNT equals the value of the CMP and increments, the following events occur:

- lpTMR status bit STS[CCF] is set.
- lpTMR interrupt is generated if DIE[IE] is set.
- lpTMR hardware trigger is generated.
- CNT is reset if CTRL[TMODE] is clear.

When the lpTMR is enabled, the CMP can be changed only when STS[CCF] is set.

22.3.4 lpTMR Counter

The CNT increments by one on every:

- Prescaler clock in Time Counter mode with prescaler bypassed
- Prescaler output in Time Counter mode with prescaler enabled
- Input source assertion in Pulse Counter mode with glitch filter bypassed
- Glitch filter output in Pulse Counter mode with glitch filter enabled

22.3.5 lpTMR Hardware Trigger

The lpTMR hardware trigger asserts at the same time the STS[CCF] is set and can be used to trigger hardware events in other peripherals without software intervention.

- When the CMP is set to 0 with CTRL[TMODE] clear, then the lpTMR hardware trigger will assert on the first compare and does not deassert.
- When the CMP is set to a nonzero value, or, if CTRL[TMODE] is set, then the lpTMR hardware trigger will assert on each compare and deassert on the following increment of the CNT.

22.3.6 lpTMR Interrupt

The lpTMR interrupt is generated whenever DIE[TIE] and STS[CCF] are set. STS[CCF] is cleared by disabling the lpTMR or by writing a logic 1 to it. DIE[TIE] can be changed and STS[CCF] can be cleared while the lpTMR is enabled.

Yuntu Microelectronics

The lpTMR interrupt is generated asynchronously to the system clock and can be used to generate a wake-up signal from any low power mode, provided the lpTMR is enabled as a wake-up source.

22.3.7 Operation Modes

lpTMR supports the chip operation modes of operation as follows:

- Active: The lpTMR operates normally.
- Sleep: The lpTMR continues to operate normally and can be configured to exit the low power mode by generating an interrupt request.
- Deepsleep: The lpTMR continues to operate normally and can be configured to exit the low power mode by generating an interrupt request.
- Standby: The lpTMR continues to operate normally and can be configured to exit the low power mode by generating an interrupt request.
- Debug: The lpTMR operates normally in Pulse Counter mode, but the counter does not increment in Time Counter mode.

Yuntu Microelectronics

Chapter 23

Enhanced Timer (eTMR)

23.1 Chip-specific eTMR Information

23.1.1 Instantiation Information

YTM32B1LE0x contains eTMR instances.

Table 23.1: eTMR Instances and features

Instances	Channels	Fault inputs	counter width	eTMR modulation	Quad Decoder	Hall sensor support	ACMP input support
eTMR0	8	4	16	Yes	No	No	No
eTMR1	8	4	16	No	Yes	No	Yes
eTMR2	2	0	16	No	No	Yes	Yes

23.1.2 Signal Description

Table 23.2: eTMR Signal Table

Signal Name	Description	Direction
TCLK_IN0	Optional external clock source 0 for eTMR counter	I
TCLK_IN1	Optional external clock source 1 for eTMR counter	I
TCLK_IN2	Optional external clock source 2 for eTMR counter	I
eTMR_CH0	Input signal used for input capture mode. Output signal used for PWM mode and output compare mode.	IO
eTMR_CH1	Input signal used for input capture mode. Output signal used for PWM mode and output compare mode.	IO
eTMR_CH2	Input signal used for input capture mode. Output signal used for PWM mode and output compare mode.	IO
eTMR_CH3	Input signal used for input capture mode. Output signal used for PWM mode and output compare mode.	IO

Yuntu Microelectronics

Table 23.2 continued from previous page

Signal Name	Description	Direction
eTMR_CH4	Input signal used for input capture mode. Output signal used for PWM mode and output compare mode.	IO
eTMR_CH5	Input signal used for input capture mode. Output signal used for PWM mode and output compare mode.	IO
eTMR_CH6	Input signal used for input capture mode. Output signal used for PWM mode and output compare mode.	IO
eTMR_CH7	Input signal used for input capture mode. Output signal used for PWM mode and output compare mode.	IO
eTMR_FLT0	Used for fault event detection.	I
eTMR_FLT1	Used for fault event detection.	I
eTMR_FLT2	Used for fault event detection.	I
eTMR_FLT3	Used for fault event detection.	I
eTMR_QD_PHA	Phase A used for quadrature decoder.	I
eTMR_QD_PHB	Phase B used for quadrature decoder.	I

23.1.3 Clock Description

Table 23.3: eTMR Clock Table

Clock Name	Description
Bus clock	Bus clock used for register access. Configurable for eTMR counter.
External clock	Configurable for eTMR counter sync to bus clock, from PIN.

23.1.4 DMA Description

Table 23.4: eTMR DMA Table

Instances	No.	Request Source	Description
eTMR0	36	Compare event or capture event	'OR' of channels compare event or capture event generates request source
eTMR1	20-27	Compare event or capture event	channel 0-7 compare event or capture event generates request source independently
eTMR2	28-29	Compare event or capture event	channel 0-1 compare event or capture event generates request source independently

23.1.5 Interrupt Description

Yuntu Microelectronics

Table 23.5: eTMR Interrupt Table

Vector	IRQn	Request Source	Description	Wakeup
28	12	eTMR0 channel 0~7	The interrupt can be generated by STS[CHxF] (x=0,1,...,7)	Y
29	13	eTMR0 fault event	The interrupt can be generated by STS[FxF] (x=0,1,2,3)	Y
30	14	eTMR0 timeout event	The interrupt is generated by STS[TOF]	Y
31	15	eTMR1 channel 0~7, eTMR1 fault event, eTMR1 timeout event	The interrupt can be generated by STS[CHxF] (x=0,1,...,7), STS[FxF] (x=0,1,2,3), STS[TOF], and STS[QDTOF]	Y
32	16	eTMR2 channel 0~1, eTMR2 timeout event	The interrupt can be generated by STS[CHxF] (x=0,1) and STS[TOF]	Y

23.1.6 Power Mode Description

Table 23.6: eTMR Power Mode Table

Power Mode	Description
Active	Full function
Sleep	Keeps working
Deepsleep	Disable
Standby	Disable

23.1.7 Channel Pair Description

Table 23.7: eTMR Channel Pair Table

Pair number	Description
0	Channel 0 and channel 1
1	Channel 2 and channel 3
2	Channel 4 and channel 5
3	Channel 6 and channel 7

23.2 Introduction

23.2.1 Overview

The eTMR contains a variety of functions, such as PWM output, input capture, output compare, and so on. There is little difference between the functions of each eTMR except for some special functions. The eTMR module works under the bus clock, it's disabled when the bus clock disappears except for the fault function.

Yuntu Microelectronics

23.2.2 Features

This chip contains 3 eTMRs (eTMR0, eTMR1, eTMR2), their features can divide into common features and individual features.

The common features contained in all eTMRs are listed below:

- Contain a 16-bit counter
- Configurable final counter value
- Support two clock sources
- Bus clock
- External clock
- Support 7-bits clock prescaler
- Support three channel modes
- PWM mode
  - Independent mode for each channel
  - Complementary mode for each pair of channels
  - Odd channels and even channels support independent deadtime insertion
  - Channel output control (initialization, software control, mask control)
  - Relevant registers have buffer registers and support loading mechanism
- Output Compare mode
  - The output can be configured to set, clear or toggle on match point
- Input Capture mode
  - Support rising edges, falling edges or dual edges capture
  - Support input filter with a prescaler
  - Support capture test mode
- Support generating triggers
- Output triggers on match point
- Output pulse with adjustable width by PWM
- Polarity control is available for each channel
- Support GTB (Global Time Base)
- Support several interrupts
- Channel interrupt (capture interrupt and compare interrupt)
- Counter overflow interrupt
- Support DMA
- Support counter running under debug mode

The individual features are listed below:

- Channel number
- eTMR0 contains 8 channels
- eTMR1 contains 8 channels
- eTMR2 contains 2 channels
- eTMR0 and eTMR1 support fault inputs for global fault control in PWM mode
- Support 4 fault input sources
- Support fault input from TMU or pad
- Support fault input polarity control
- Support fault input filter
- Support fault event interrupt
- Support fault input stretch

Yuntu Microelectronics

- Support fault event generated by combinational logic
- eTMR1 supports quadrature decoder mode
- Contain a independent 16-bit counter with a clock prescaler
- Support 4 up-down counting modes
- Support phase A and phase B input filter
- Support quadrature decoder counter overflow interrupt
- eTMR0 supports modulated output
- eTMR2 supports hall sensor input
- eTMR1 and eTMR2 support input from ACMP

23.2.3 Block Diagram

The following figure shows the eTMR block diagram.

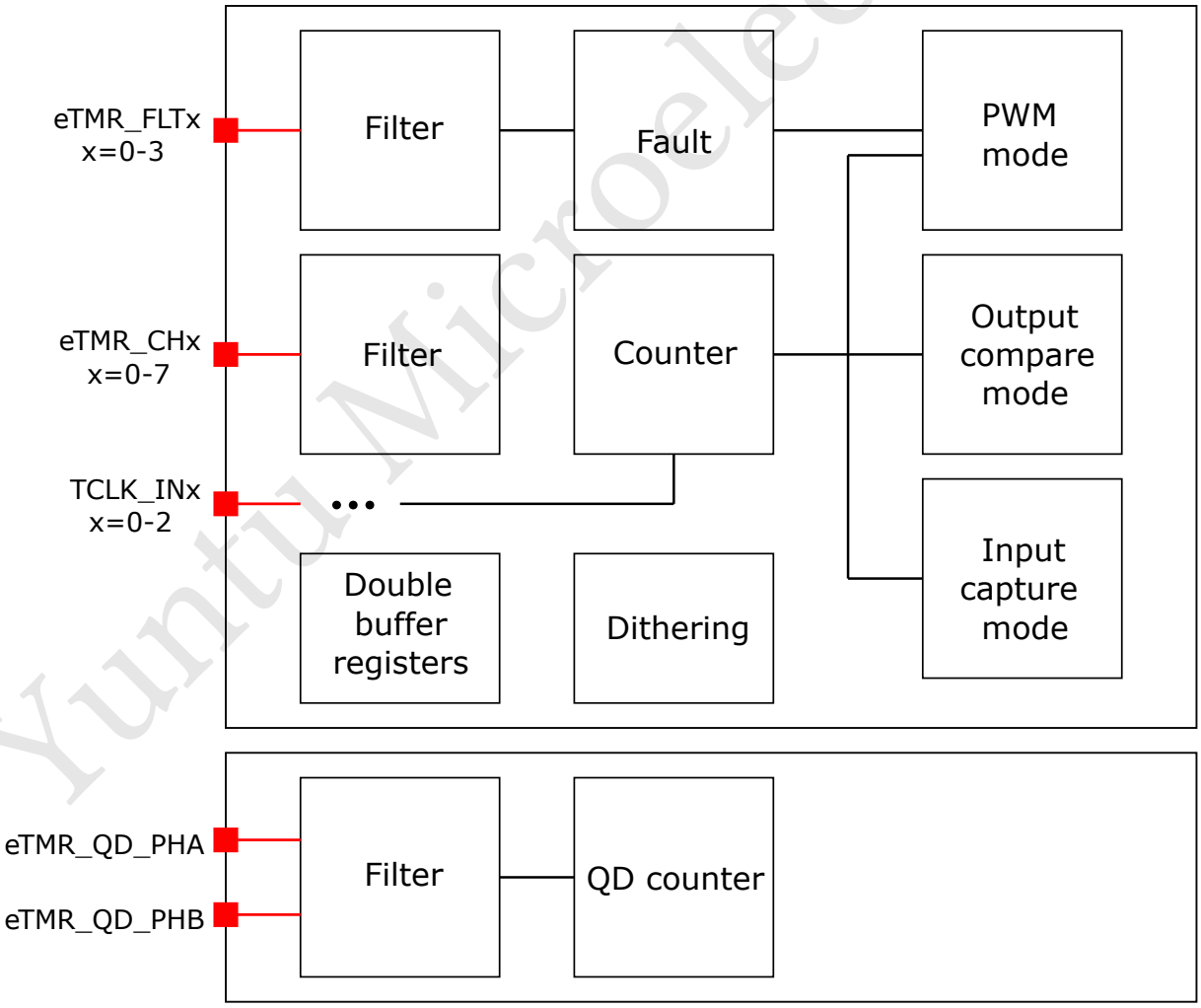


Figure 23.1: eTMR Block Diagram

Yuntu Microelectronics

23.3 Register Definition

The eTMR registers are used to control eTMR module function and display the current working status. Some registers are accessible, but they don't work because there are no corresponding functions.

Reading reserved registers is always 0, the write operation is not valid.

Register_address = Base_address + Address_offset.

23.3.1 eTMR Memory Map

eTMR0 base address 40038000h
eTMR1 base address 40039000h
eTMR2 base address 4003a000h

Table 23.8: eTMR Memory Map

Offset	Register	Width	Access	Reset Value
0h	Status Register (STS)	32	RW	00000000h
4h	Input/Output Status Register (IOSTS)	32	RW	00000000h
8h	Counter Value Register (CNT)	32	RW	00000000h
ch	Quadrature Decoder Counter Value Register (QDCNT)	32	RW	00000000h
10h	Control Register (CTRL)	32	RW	00000000h
14h	Synchronous Register (SYNC)	32	RW	00000000h
18h	Output Trigger Register (OTRIG)	32	RW	00000000h
1ch	Interrupt Enable Register (INTE)	32	RW	00000000h
20h	Channel Output Initial Value Register (CHINIT)	32	RW	00000000h
24h	Channel Software Value Register (CHSWV)	32	RW	00000000h
28h	Channel Output Mask Value Register (CHMASK)	32	RW	00000000h
2ch	Channel Fault Value Register (CHFV)	32	RW	00000000h
30h	Counter Modulus Value Register (MOD)	32	RW	0000FFFFh
34h	Deadtime Register (DT)	32	RW	00000000h
38h	Quadrature Decoder Control Register (QDCTRL)	32	RW	00000000h
3ch	Fault Control Register (FAULT)	32	RW	00000000h
a0h	Channel Control Register (CH0_CTRL)	32	RW	00000000h
a4h	Value 0 Register (CH0_VAL0)	32	RW	00000000h
a8h	Value 1 Register (CH0_VAL1)	32	RW	00000000h
ach	Capture Value Register (CH0_CVAL)	32	RO	00000000h
c0h	Channel Control Register (CH1_CTRL)	32	RW	00000000h
c4h	Value 0 Register (CH1_VAL0)	32	RW	00000000h
c8h	Value 1 Register (CH1_VAL1)	32	RW	00000000h
cch	Capture Value Register (CH1_CVAL)	32	RO	00000000h
e0h	Channel Control Register (CH2_CTRL)	32	RW	00000000h
e4h	Value 0 Register (CH2_VAL0)	32	RW	00000000h
e8h	Value 1 Register (CH2_VAL1)	32	RW	00000000h
ech	Capture Value Register (CH2_CVAL)	32	RO	00000000h
100h	Channel Control Register (CH3_CTRL)	32	RW	00000000h
104h	Value 0 Register (CH3_VAL0)	32	RW	00000000h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
108h	Value 1 Register (CH3_VAL1)	32	RW	00000000h
10ch	Capture Value Register (CH3_CVAL)	32	RO	00000000h
120h	Channel Control Register (CH4_CTRL)	32	RW	00000000h
124h	Value 0 Register (CH4_VAL0)	32	RW	00000000h
128h	Value 1 Register (CH4_VAL1)	32	RW	00000000h
12ch	Capture Value Register (CH4_CVAL)	32	RO	00000000h
140h	Channel Control Register (CH5_CTRL)	32	RW	00000000h
144h	Value 0 Register (CH5_VAL0)	32	RW	00000000h
148h	Value 1 Register (CH5_VAL1)	32	RW	00000000h
14ch	Capture Value Register (CH5_CVAL)	32	RO	00000000h
160h	Channel Control Register (CH6_CTRL)	32	RW	00000000h
164h	Value 0 Register (CH6_VAL0)	32	RW	00000000h
168h	Value 1 Register (CH6_VAL1)	32	RW	00000000h
16ch	Capture Value Register (CH6_CVAL)	32	RO	00000000h
180h	Channel Control Register (CH7_CTRL)	32	RW	00000000h
184h	Value 0 Register (CH7_VAL0)	32	RW	00000000h
188h	Value 1 Register (CH7_VAL1)	32	RW	00000000h
18ch	Capture Value Register (CH7_CVAL)	32	RO	00000000h

23.3.1.1 eTMR STS Register

Offset: 0h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	QDDIR	QDTOF	TOF	RF	F3F	F2F	F1F	F0F	CH7F	CH6F	CH5F	CH4F	CH3F	CH2F	CH1F	CH0F
W		W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23.9: eTMR STS Register Description

Field	Function
15 QDDIR	**Quadrature Decoder Direction** Indicate quadrature decoder counter direction. 0b - Quadrature counter increase. 1b - Quadrature counter decrease.
14	**Quadrature Decoder Timer Overflow Flag**

Yuntu Microelectronics

Field	Function
QDTOF	When STS[QDDIR]=0, STS[QDTOF]=1 means quadrature decoder counter increase from 0xFFFF to 0. When STS[QDDIR]=1, STS[QDTOF]=1 means quadrature decoder counter decrease from 0 to oxFFFF. Write 1 to clear the bit.
13 TOF	**Timer Overflow Flag** Timer overflow flag sets when counter increase from MOD[MOD] to 0. Write 1 to clear the bit.
12 RF	**Reload Flag** Reload flag sets when registers loading occurs. write 1 to clear the bit.
11 F3F	**Fault 3 Flag** Fault event 3 occurs flag. Write 1 to clear the bit.
10 F2F	**Fault 2 Flag** Fault event 2 occurs flag. Write 1 to clear the bit.
9 F1F	**Fault 1 Flag** Fault event 1 occurs flag. Write 1 to clear the bit.
8 F0F	**Fault 0 Flag** Fault event 0 occurs flag. Write 1 to clear the bit.
7 CH7F	**Channel 7 Flag** Channel 7 flag is used for capture mode and compare mode. In capture mode, this flag sets when capture event occurs. In compare mode, this flag sets when counter matches the values. Write 1 to clear the bit.
6 CH6F	**Channel 6 Flag** Channel 6 flag is used for capture mode and compare mode. In capture mode, this flag sets when capture event occurs. In compare mode, this flag sets when counter matches the values. Write 1 to clear the bit.
5 CH5F	**Channel 5 Flag** Channel 5 flag is used for capture mode and compare mode. In capture mode, this flag sets when capture event occurs. In compare mode, this flag sets when counter matches the values. Write 1 to clear the bit.
4 CH4F	**Channel 4 Flag** Channel 4 flag is used for capture mode and compare mode. In capture mode, this flag sets when capture event occurs. In compare mode, this flag sets when counter matches the values. Write 1 to clear the bit.
3	**Channel 3 Flag**

Yuntu Microelectronics

Field	Function
CH3F	Channel 3 flag is used for capture mode and compare mode. In capture mode, this flag sets when capture event occurs. In compare mode, this flag sets when counter matches the values. Write 1 to clear the bit.
2 CH2F	**Channel 2 Flag** Channel 2 flag is used for capture mode and compare mode. In capture mode, this flag sets when capture event occurs. In compare mode, this flag sets when counter matches the values. Write 1 to clear the bit.
1 CH1F	**Channel 1 Flag** Channel 1 flag is used for capture mode and compare mode. In capture mode, this flag sets when capture event occurs. In compare mode, this flag sets when counter matches the values. Write 1 to clear the bit.
0 CH0F	**Channel 0 Flag** Channel 0 flag is used for capture mode and compare mode. In capture mode, this flag sets when capture event occurs. In compare mode, this flag sets when counter matches the values. Write 1 to clear the bit.

23.3.1.2 eTMR IOSTS Register

Offset: 4h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R							PHB	PHA					F3	F2	F1	F0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									CH7IO	CH6IO	CH5IO	CH4IO	CH3IO	CH2IO	CH1IO	CH0IO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23.10: eTMR IOSTS Register Description

Field	Function
25 PHB	**Phase B Status** Phase B input status is used for Quadrature Decoder. Read-only.

Yuntu Microelectronics

Field	Function
24 PHA	**Phase A Status** Phase A input status is used for Quadrature Decoder. Read-only.
19 F3	**Fault 3 Status** Fault 3 input status is used for fault logic. Read-only.
18 F2	**Fault 2 Status** Fault 2 input status is used for fault logic. Read-only.
17 F1	**Fault 1 Status** Fault 1 input status is used for fault logic. Read-only.
16 F0	**Fault 0 Status** Fault 0 input status is used for fault logic. Read-only.
7 CH7IO	**Channel 7 I/O Status** In capture mode, indicate channel 7 input status. In compare mode or PWM mode, indicate channel 7 output status. Read-only.
6 CH6IO	**Channel 6 I/O Status** In capture mode, indicate channel 6 input status. In compare mode or PWM mode, indicate channel 6 output status. Read-only.
5 CH5IO	**Channel 5 I/O Status** In capture mode, indicate channel 5 input status. In compare mode or PWM mode, indicate channel 5 output status. Read-only.
4 CH4IO	**Channel 4 I/O Status** In capture mode, indicate channel 4 input status. In compare mode or PWM mode, indicate channel 4 output status. Read-only.
3 CH3IO	**Channel 3 I/O Status** In capture mode, indicate channel 3 input status. In compare mode or PWM mode, indicate channel 3 output status. Read-only.
2 CH2IO	**Channel 2 I/O Status** In capture mode, indicate channel 2 input status. In compare mode or PWM mode, indicate channel 2 output status. Read-only.
1 CH1IO	**Channel 1 I/O Status** In capture mode, indicate channel 1 input status. In compare mode or PWM mode, indicate channel 1 output status. Read-only.
0	**Channel 0 I/O Status**

Yuntu Microelectronics

Field	Function
CH0IO	In capture mode, indicate channel 0 input status. In compare mode or PWM mode, indicate channel 0 output status. Read-only.

23.3.1.3 eTMR CNT Register

Offset: 8h

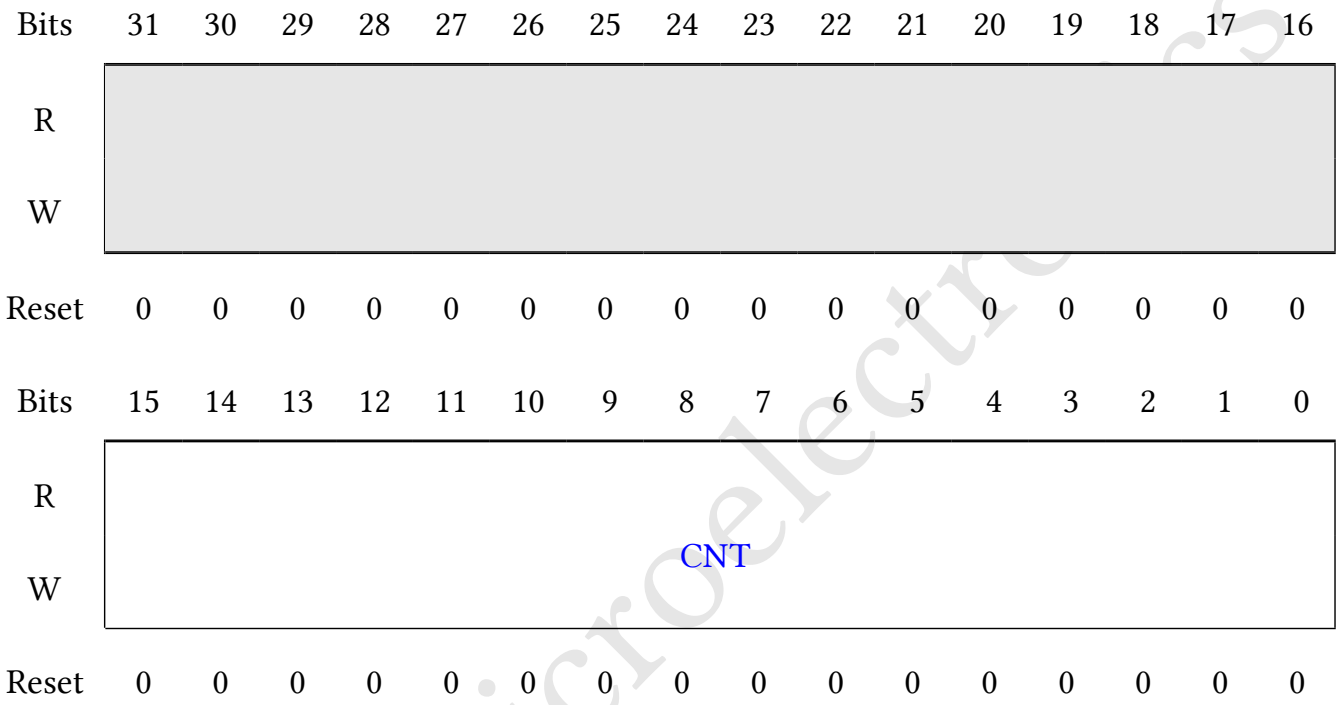


Table 23.11: eTMR CNT Register Description

Field	Function
15 - 0 CNT	**Counter Initial or Value Register** When CTRL[EN]=0, writing to this register initializes counter. When CTRL[EN]=1, reading this register gets the counter value, and writing to this register reloads value 0.

23.3.1.4 eTMR QDCNT Register

Offset: ch

Yuntu Microelectronics

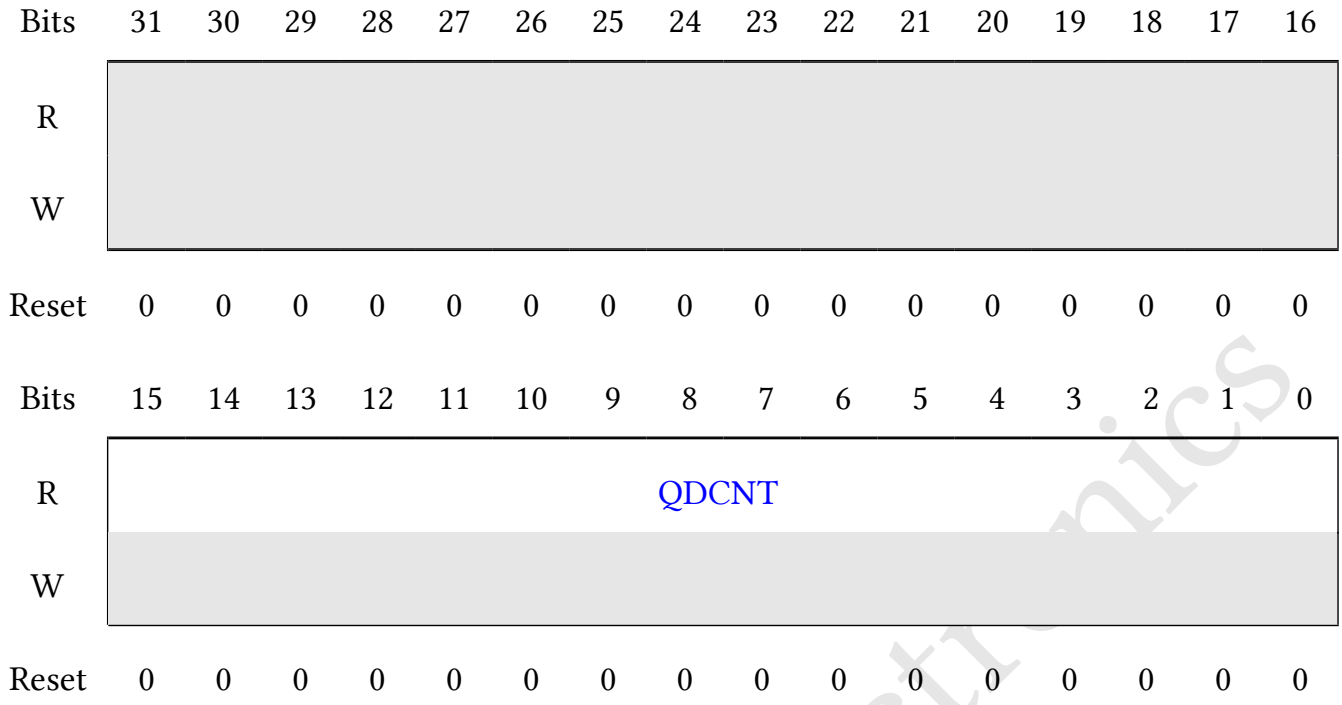
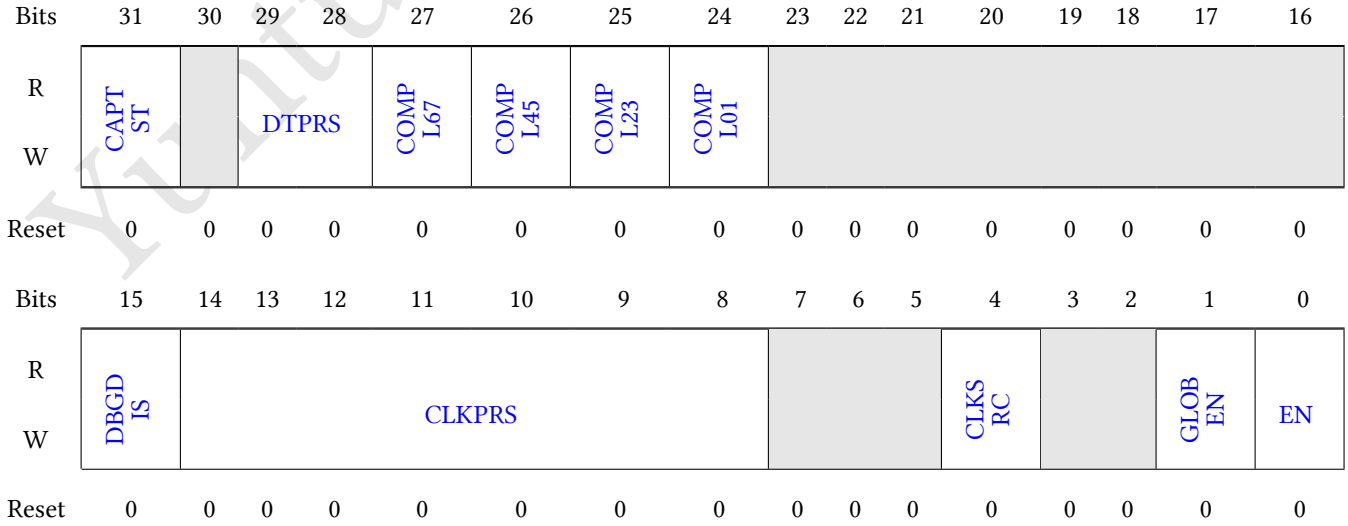


Table 23.12: eTMR QDCNT Register Description

Field	Function
15 - 0 QDCNT	**Quadrature Decoder Counter Value** Reading this register gets quadrature decoder counter value. The write operation is not valid.

23.3.1.5 eTMR CTRL Register

Offset: 10h



Yuntu Microelectronics

Table 23.13: eTMR CTRL Register Description

Field	Function
31 CAPTST	**Capture Test** This bit is used for the capture test. 0b - Capture test is disabled. 1b - Capture test is enabled.
29 - 28 DTPRS	**Deadtime Prescaler** Clock prescaler for deadtime counter, from 1 to 4.
27 COMPL67	**Complementary Mode Enable for Channel 6 and Channel 7** Channel 6 and 7 output independently or complementary in PWM mode. 0b - Independent mode. 1b - Complementary mode.
26 COMPL45	**Complementary Mode Enable for Channel 4 and Channel 5** Channel 4 and 5 output independently or complementary in PWM mode. 0b - Independent mode. 1b - Complementary mode.
25 COMPL23	**Complementary Mode Enable for Channel 2 and Channel 3** Channel 2 and 3 output independently or complementary in PWM mode. 0b - Independent mode. 1b - Complementary mode.
24 COMPL01	**Complementary Mode Enable for Channel 0 and Channel 1** Channel 0 and 1 output independently or complementary in PWM mode. 0b - Independent mode. 1b - Complementary mode.
15 DBGDIS	**Debug Mode Disabled** Enable/disable the debug mode control. 0b - Counter stops under debug mode. 1b - Counter keeps running under debug mode.
14 - 8 CLKPRS	**Clock Prescaler** Clock prescaler for counter, from 1 to 128.
4 CLKSRC	**Clock Source** Clock source for counter, bus clock or external clock. 0b - Select bus clock. 1b - Select external clock.
1	**Global Enable**

Yuntu Microelectronics

Field	Function
GLOBEN	eTMRs counter synchronization mechanisms. 0b - Global logic is disabled. 1b - Global logic is enabled. When SYNC[GLOB]=1 from any eTMR includes itself, the counter starts running. When all eTMRs includes itself SYNC[GLOB]=0, the counter stops.
0 EN	**eTMR Enable** eTMR enable/disable. 0b - The eTMR is disabled. 1b - The eTMR is enabled.

23.3.1.6 eTMR SYNC Register

Offset: 14h

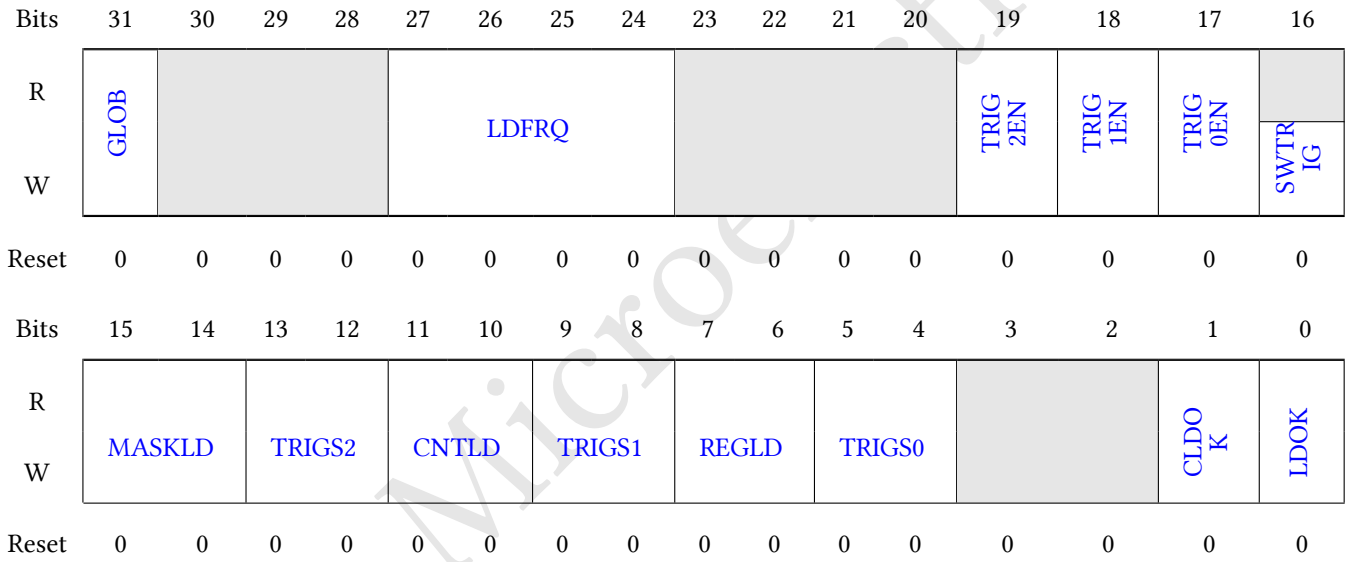


Table 23.14: eTMR SYNC Register Description

Field	Function
31 GLOB	**Global Base** Generate global base signal. This bit is used to synchronize with other eTMRs which CTRL[GLOBEN]=1. 0b - Not generate global base signal. 1b - Generate global base signal.
27 - 24 LDFRQ	**Registers Loading Frequency** Registers Loading occurs at the reloading opportunity that meets this specified frequency.
19	**Hardware Trigger 2 Enable**

Yuntu Microelectronics

Field	Function
TRIG2EN	Enable hardware trigger 2. 0b - Hardware trigger 2 is disabled. 1b - Hardware trigger 2 is enabled.
18 TRIG1EN	**Hardware Trigger 1 Enable** Enable hardware trigger 1. 0b - Hardware trigger 1 is disabled. 1b - Hardware trigger 1 is enabled.
17 TRIG0EN	**Hardware Trigger 0 Enable** Enable hardware trigger 0. 0b - Hardware trigger 0 is disabled. 1b - Hardware trigger 0 is enabled.
16 SWTRIG	**Software Trigger** Software trigger is used for relevant registers loading. Writing 1 to this bit generates one bus clock cycle pulse trigger. Write-only.
15 - 14 MASKLD	**CHMASK Loading Opportunity** Select loading opportunity for CHMASK register. 00b - loading together with registers (MOD, CH_VAL0 and CH_VAL1). 01b - Loading occurs when counter matches with MOD[MOD]. 10b - Loading occurs with trigger event which selected by SYNC[TRIGS2]. Any valid hardware trigger x will generate a trigger event (x=0,1,2). 11b - Reserved.
13 - 12 TRIGS2	**Trigger Source 2** Select the loading trigger source for CHMASK register. 00b - Disable trigger. 01b - Select software trigger. 10b - Select hardware trigger. 11b - Software and hardware trigger are both valid.
11 - 10 CNTLD	**Counter Loading Opportunity** Select loading opportunity for counter, the counter loads with value 0. 00b - loading together with registers (MOD, CH_VAL0 and CH_VAL1). 01b - Loading occurs with trigger event which selected by SYNC[TRIGS1]. Any valid hardware trigger x will generate a trigger event (x=0,1,2). 10b - Reserved. 11b - Reserved.
9 - 8 TRIGS1	**Trigger Source 1** Select the loading trigger source for counter. 00b - Disable trigger. 01b - Select software trigger. 10b - Select hardware trigger. 11b - Software and hardware trigger are both valid.

Yuntu Microelectronics

Field	Function
7 - 6 REGLD	**Registers Loading Opportunity** Select loading opportunity for registers (MOD, CH_VAL0, CH_VAL1). 00b - Loading is disabled. 01b - Loading occurs when counter matches with MOD[MOD]. 10b - Loading occurs with trigger event which selected by SYNC[TRIGS0]. Any valid hardware trigger x will generate a trigger event (x=0,1,2). 11b - Reserved.
5 - 4 TRIGS0	**Trigger Source 0** Select the loading trigger source for registers (MOD, CH_VAL0, CH_VAL1). 00b - Disable trigger. 01b - Select software trigger. 10b - Select hardware trigger. 11b - Software and hardware trigger are both valid.
1 CLDOK	**Clear LDOK** Write 1 to this bit to clear SYNC[LDOK], and it automatically clear when SYNC[LDOK]=0.
0 LDOK	**Registers Loading OK** Set LDOK=1 to indicate registers (MOD, CH_VAL0, CH_VAL1) loading is ready, it automatically clear when registers loading successfully. 0b - Registers loading is not ready. 1b - Registers loading is ready and waits for loading opportunity.

23.3.1.7 eTMR OTRIG Register

Offset: 18h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TFRQ					PULSESRC						TSRC		MODT EN		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Yuntu Microelectronics

Table 23.15: eTMR OTRIG Register Description

Field	Function
15 - 12 TFRQ	**Trigger Frequency** Trigger output occurs at the opportunity that meets the specified frequency.
10 - 8 PULSESRC	**Pulse-out Source** Select PWM as the pulse-out signal. 000b - Select channel 0 PWM as the pulse-out signal. 001b - Select channel 1 PWM as the pulse-out signal. 010b - Select channel 2 PWM as the pulse-out signal. 011b - Select channel 3 PWM as the pulse-out signal. 100b - Select channel 4 PWM as the pulse-out signal. 101b - Select channel 5 PWM as the pulse-out signal. 110b - Select channel 6 PWM as the pulse-out signal. 111b - select channel 7 PWM as the pulse-out signal.
4 TSRC	**Trigger Source** Trigger output source: matching event occurs or PWM output. 0b - Select trigger that generated by matching event. 1b - Select PWM output as the trigger.
2 MODTEN	**MOD Matching Trigger Enable** Enable/disable the MOD[MOD] matching trigger. 0b - Not generate trigger when counter matches with MOD[MOD]. 1b - Generate trigger when counter matches with MOD[MOD].

23.3.1.8 eTMR INTE Register

Offset: 1ch

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		QDTOIE	TOIE					FIE	CH7IE	CH6IE	CH5IE	CH4IE	CH3IE	CH2IE	CH1IE	CH0IE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Yuntu Microelectronics

Table 23.16: eTMR INTE Register Description

Field	Function
14 QDTOIE	**Quadrature Decoder Timer Overflow Interrupt Enable** Quadrature decoder timer overflow interrupt enable/disable. 0b - Quadrature decoder timer overflow interrupt is disabled. 1b - Quadrature decoder timer overflow interrupt is enabled.
13 TOIE	**Timer Overflow Interrupt Enable** Timer overflow interrupt enable/disable. 0b - Timer overflow interrupt is disabled. 1b - Timer overflow interrupt is enabled.
8 FIE	**Fault Interrupt Enable** Fault interrupt enable/disable. 0b - Fault interrupt is disabled. 1b - Fault interrupt is enabled.
7 CH7IE	**Channel 7 Interrupt Enable** Channel 7 capture/compare interrupt enable/disable. 0b - Channel 7 interrupt is disabled. 1b - Channel 7 interrupt is enabled.
6 CH6IE	**Channel 6 Interrupt Enable** Channel 6 capture/compare interrupt enable/disable. 0b - Channel 6 interrupt is disabled. 1b - Channel 6 interrupt is enabled.
5 CH5IE	**Channel 5 Interrupt Enable** Channel 5 capture/compare interrupt enable/disable. 0b - Channel 5 interrupt is disabled. 1b - Channel 5 interrupt is enabled.
4 CH4IE	**Channel 4 Interrupt Enable** Channel 4 capture/compare interrupt enable/disable. 0b - Channel 4 interrupt is disabled. 1b - Channel 4 interrupt is enabled.
3 CH3IE	**Channel 3 Interrupt Enable** Channel 3 capture/compare interrupt enable/disable. 0b - Channel 3 interrupt is disabled. 1b - Channel 3 interrupt is enabled.
2 CH2IE	**Channel 2 Interrupt Enable** Channel 2 capture/compare interrupt enable/disable. 0b - Channel 2 interrupt is disabled. 1b - Channel 2 interrupt is enabled.
1	**Channel 1 Interrupt Enable**

Yuntu Microelectronics

Field	Function
CH1IE	Channel 1 capture/compare interrupt enable/disable. 0b - Channel 1 interrupt is disabled. 1b - Channel 1 interrupt is enabled.
0 CH0IE	**Channel 0 Interrupt Enable** Channel 0 capture/compare interrupt enable/disable. 0b - Channel 0 interrupt is disabled. 1b - Channel 0 interrupt is enabled.

23.3.1.9 eTMR CHINIT Register

Offset: 20h

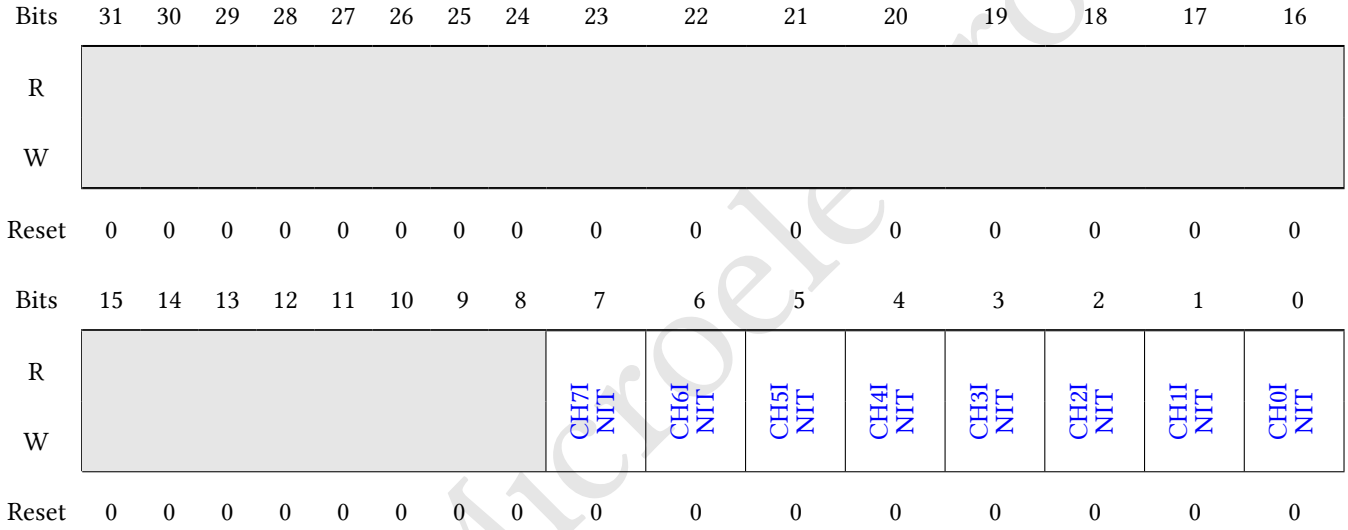


Table 23.17: eTMR CHINIT Register Description

Field	Function
7 CH7INIT	**Channel 7 Output Initial Value** Channel 7 output initial value. 0b - Channel 7 output initial value is 0. 1b - Channel 7 output initial value is 1.
6 CH6INIT	**Channel 6 Output Initial Value** Channel 6 output initial value. 0b - Channel 6 output initial value is 0. 1b - Channel 6 output initial value is 1.
5 CH5INIT	**Channel 5 Output Initial Value** Channel 5 output initial value. 0b - Channel 5 output initial value is 0. 1b - Channel 5 output initial value is 1.

Yuntu Microelectronics

Field	Function
4 CH4INIT	**Channel 4 Output Initial Value** Channel 4 output initial value. 0b - Channel 4 output initial value is 0. 1b - Channel 4 output initial value is 1.
3 CH3INIT	**Channel 3 Output Initial Value** Channel 3 output initial value. 0b - Channel 3 output initial value is 0. 1b - Channel 3 output initial value is 1.
2 CH2INIT	**Channel 2 Output Initial Value** Channel 2 output initial value. 0b - Channel 2 output initial value is 0. 1b - Channel 2 output initial value is 1.
1 CH1INIT	**Channel 1 Output Initial Value** Channel 1 output initial value. 0b - Channel 1 output initial value is 0. 1b - Channel 1 output initial value is 1.
0 CH0INIT	**Channel 0 Output Initial Value** Channel 0 output initial value. 0b - Channel 0 output initial value is 0. 1b - Channel 0 output initial value is 1.

23.3.1.10 eTMR CHSWV Register

Offset: 24h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W									CH7SWV	CH6SWV	CH5SWV	CH4SWV	CH3SWV	CH2SWV	CH1SWV	CH0SWV
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Yuntu Microelectronics

Table 23.18: eTMR CHSWV Register Description

Field	Function
7 CH7SWV	**Channel 7 Software Value** Channel 7 output value configured by software. 0b - Channel 7 output value forced to 0 by software. 1b - Channel 7 output value forced to 1 by software.
6 CH6SWV	**Channel 6 Software Value** Channel 6 output value configured by software. 0b - Channel 6 output value forced to 0 by software. 1b - Channel 6 output value forced to 1 by software.
5 CH5SWV	**Channel 5 Software Value** Channel 5 output value configured by software. 0b - Channel 5 output value forced to 0 by software. 1b - Channel 5 output value forced to 1 by software.
4 CH4SWV	**Channel 4 Software Value** Channel 4 output value configured by software. 0b - Channel 4 output value forced to 0 by software. 1b - Channel 4 output value forced to 1 by software.
3 CH3SWV	**Channel 3 Software Value** Channel 3 output value configured by software. 0b - Channel 3 output value forced to 0 by software. 1b - Channel 3 output value forced to 1 by software.
2 CH2SWV	**Channel 2 Software Value** Channel 2 output value configured by software. 0b - Channel 2 output value forced to 0 by software. 1b - Channel 2 output value forced to 1 by software.
1 CH1SWV	**Channel 1 Software Value** Channel 1 output value configured by software. 0b - Channel 1 output value forced to 0 by software. 1b - Channel 1 output value forced to 1 by software.
0 CH0SWV	**Channel 0 Software Value** Channel 0 forced value by software. 0b - Channel 0 output value forced to 0 by software. 1b - Channel 0 output value forced to 1 by software.

23.3.1.11 eTMR CHMASK Register

Offset: 28h

Yuntu Microelectronics

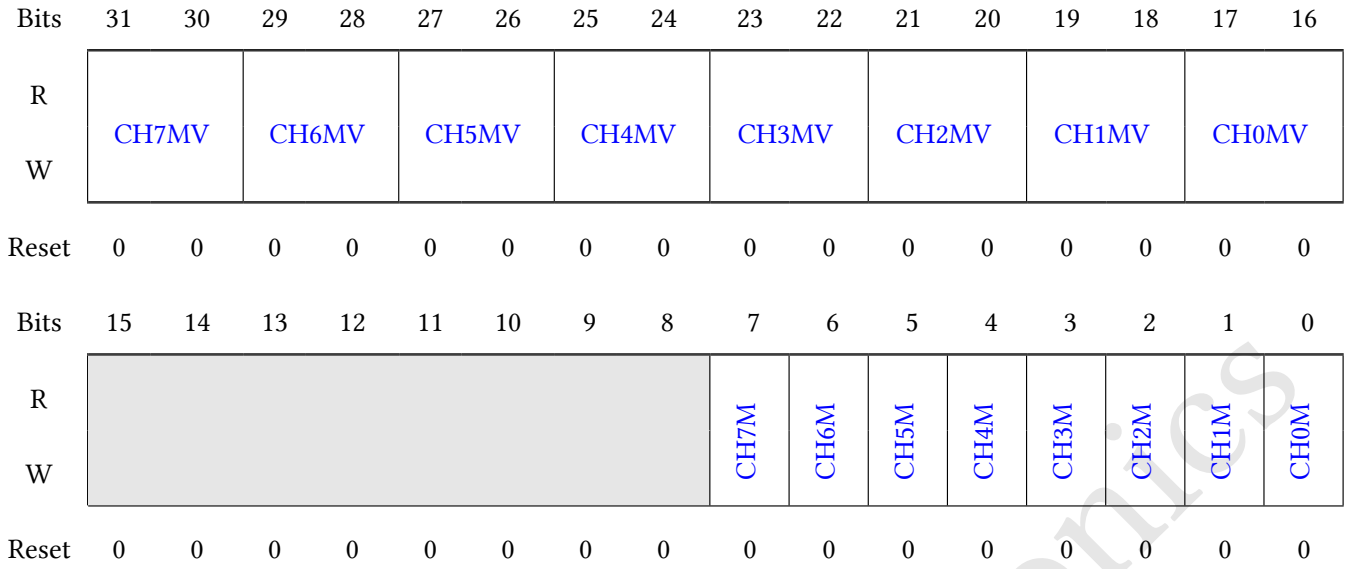


Table 23.19: eTMR CHMASK Register Description

Field	Function
31 - 30 CH7MV	**Channel 7 Output Mask Value** Define channel 7 output mask value, it's enabled when CHMASK[CH7M]=1. 00b - Channel 7 output mask value is 0. 01b - Channel 7 output mask value is 1. 10b - Channel 7 output mask value is tri-state. 11b - Channel 7 output mask value is tri-state.
29 - 28 CH6MV	**Channel 6 Output Mask Value** Define channel 6 output mask value, it's enabled when CHMASK[CH6M]=1. 00b - Channel 6 output mask value is 0. 01b - Channel 6 output mask value is 1. 10b - Channel 6 output mask value is tri-state. 11b - Channel 6 output mask value is tri-state.
27 - 26 CH5MV	**Channel 5 Output Mask Value** Define channel 5 output mask value, it's enabled when CHMASK[CH5M]=1. 00b - Channel 5 output mask value is 0. 01b - Channel 5 output mask value is 1. 10b - Channel 5 output mask value is tri-state. 11b - Channel 5 output mask value is tri-state.
25 - 24 CH4MV	**Channel 4 Output Mask Value** Define channel 4 output mask value, it's enabled when CHMASK[CH4M]=1. 00b - Channel 4 output mask value is 0. 01b - Channel 4 output mask value is 1. 10b - Channel 4 output mask value is tri-state. 11b - Channel 4 output mask value is tri-state.
23 - 22	**Channel 3 Output Mask Value**

Yuntu Microelectronics

Field	Function
CH3MV	Define channel 3 output mask value, it's enabled when CHMASK[CH3M]=1. 00b - Channel 3 output mask value is 0. 01b - Channel 3 output mask value is 1. 10b - Channel 3 output mask value is tri-state. 11b - Channel 3 output mask value is tri-state.
21 - 20 CH2MV	**Channel 2 Output Mask Value** Define channel 2 output mask value, it's enabled when CHMASK[CH2M]=1. 00b - Channel 2 output mask value is 0. 01b - Channel 2 output mask value is 1. 10b - Channel 2 output mask value is tri-state. 11b - Channel 2 output mask value is tri-state.
19 - 18 CH1MV	**Channel 1 Output Mask Value** Define channel 1 output mask value, it's enabled when CHMASK[CH1M]=1. 00b - Channel 1 output mask value is 0. 01b - Channel 1 output mask value is 1. 10b - Channel 1 output mask value is tri-state. 11b - Channel 1 output mask value is tri-state.
17 - 16 CH0MV	**Channel 0 Output Mask Value** Define channel 0 output mask value, it's enabled when CHMASK[CH0M]=1. 00b - Channel 0 output mask value is 0. 01b - Channel 0 output mask value is 1. 10b - Channel 0 output mask value is tri-state. 11b - Channel 0 output mask value is tri-state.
7 CH7M	**Channel 7 Output Mask** Channel 7 output mask control used in PWM mode. 0b - Channel 7 output PWM normally. 1b - Channel 7 output masked by CHMASK[CH7MV].
6 CH6M	**Channel 6 Output Mask** Channel 6 output mask control used in PWM mode. 0b - Channel 6 output PWM normally. 1b - Channel 6 output masked by CHMASK[CH6MV].
5 CH5M	**Channel 5 Output Mask** Channel 5 output mask control used in PWM mode. 0b - Channel 5 output PWM normally. 1b - Channel 5 output masked by CHMASK[CH5MV].
4 CH4M	**Channel 4 Output Mask** Channel 4 output mask control used in PWM mode. 0b - Channel 4 output PWM normally. 1b - Channel 4 output masked by CHMASK[CH4MV].
3	**Channel 3 Output Mask**

Yuntu Microelectronics

Field	Function
CH3M	Channel 3 output mask control used in PWM mode. 0b - Channel 3 output PWM normally. 1b - Channel 3 output masked by CHMASK[CH3MV].
2 CH2M	**Channel 2 Output Mask** Channel 2 output mask control used in PWM mode. 0b - Channel 2 output PWM normally. 1b - Channel 2 output masked by CHMASK[CH2MV].
1 CH1M	**Channel 1 Output Mask** Channel 1 output mask control used in PWM mode. 0b - Channel 1 output PWM normally. 1b - Channel 1 output masked by CHMASK[CH1MV].
0 CH0M	**Channel 0 Output Mask** Channel 0 output mask control used in PWM mode. 0b - Channel 0 output PWM normally. 1b - Channel 0 output masked by CHMASK[CH0MV].

23.3.1.12 eTMR CHFV Register

Offset: 2ch

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CH7FV		CH6FV		CH5FV		CH4FV		CH3FV		CH2FV		CH1FV		CH0FV	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23.20: eTMR CHFV Register Description

Field	Function
15 - 14	**Channel 7 Fault Value**

Yuntu Microelectronics

Field	Function
CH7FV	Channel 7 forced value when fault event occurs. 00b - Channel 7 output value forced to 0. 01b - Channel 7 output value forced to 1. 10b - Channel 7 output value forced to tri-state. 11b - Channel 7 output value forced to tri-state.
13 - 12 CH6FV	**Channel 6 Fault Value** Channel 6 forced value when fault event occurs. 00b - Channel 6 output value forced to 0. 01b - Channel 6 output value forced to 1. 10b - Channel 6 output value forced to tri-state. 11b - Channel 6 output value forced to tri-state.
11 - 10 CH5FV	**Channel 5 Fault Value** Channel 5 forced value when fault event occurs. 00b - Channel 5 output value forced to 0. 01b - Channel 5 output value forced to 1. 10b - Channel 5 output value forced to tri-state. 11b - Channel 5 output value forced to tri-state.
9 - 8 CH4FV	**Channel 4 Fault Value** Channel 4 forced value when fault event occurs. 00b - Channel 4 output value forced to 0. 01b - Channel 4 output value forced to 1. 10b - Channel 4 output value forced to tri-state. 11b - Channel 4 output value forced to tri-state.
7 - 6 CH3FV	**Channel 3 Fault Value** Channel 3 forced value when fault event occurs. 00b - Channel 3 output value forced to 0. 01b - Channel 3 output value forced to 1. 10b - Channel 3 output value forced to tri-state. 11b - Channel 3 output value forced to tri-state.
5 - 4 CH2FV	**Channel 2 Fault Value** Channel 2 forced value when fault event occurs. 00b - Channel 2 output value forced to 0. 01b - Channel 2 output value forced to 1. 10b - Channel 2 output value forced to tri-state. 11b - Channel 2 output value forced to tri-state.
3 - 2 CH1FV	**Channel 1 Fault Value** Channel 1 forced value when fault event occurs. 00b - Channel 1 output value forced to 0. 01b - Channel 1 output value forced to 1. 10b - Channel 1 output value forced to tri-state. 11b - Channel 1 output value forced to tri-state.

Yuntu Microelectronics

Field	Function
1 - 0 CH0FV	**Channel 0 Fault Value** Channel 0 forced value when fault event occurs. 00b - Channel 0 output value forced to 0. 01b - Channel 0 output value forced to 1. 10b - Channel 0 output value forced to tri-state. 11b - Channel 0 output value forced to tri-state.

23.3.1.13 eTMR MOD Register

Offset: 30h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MOD															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 23.21: eTMR MOD Register Description

Field	Function
15 - 0 MOD	**Counter Modulus Value** Counter modulus value.

23.3.1.14 eTMR DT Register

Offset: 34h

Yuntu Microelectronics

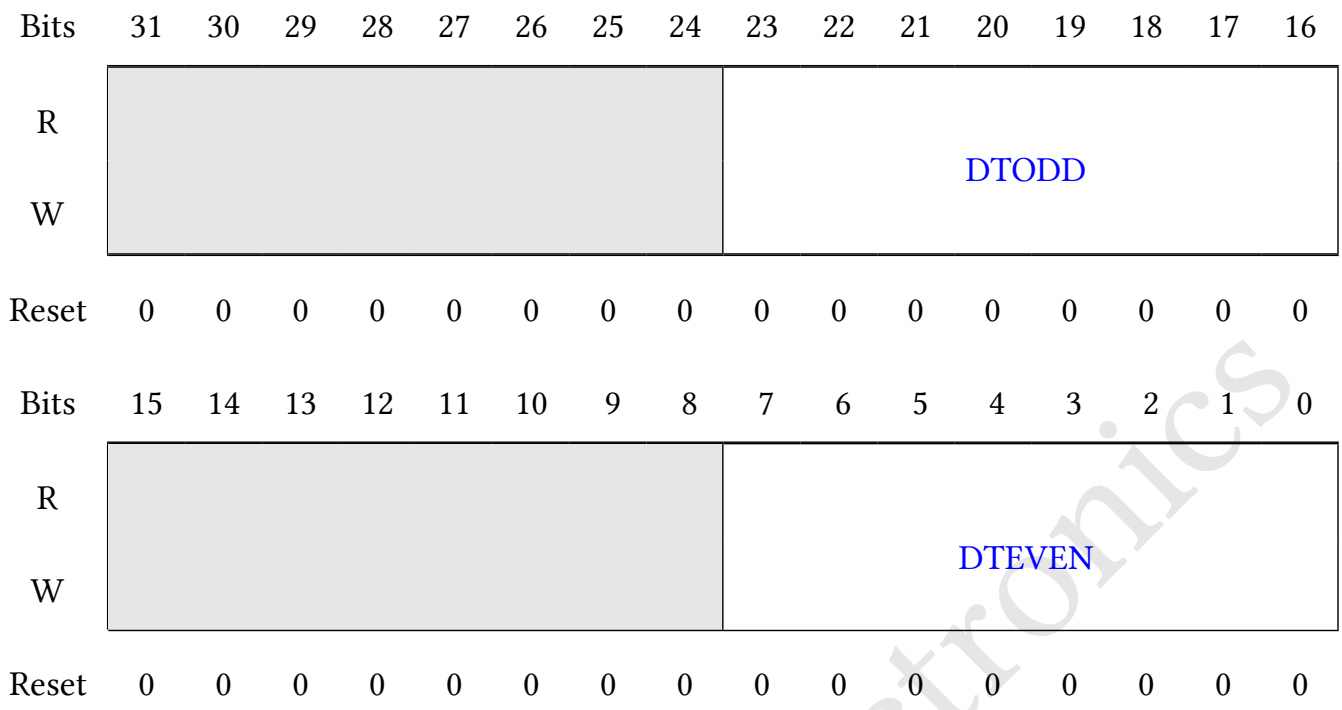


Table 23.22: eTMR DT Register Description

Field	Function
23 - 16 DTODD	**Deadtime in Odd Channels** Define the deadtime value in odd channels output PWM.
7 - 0 DTEVEN	**Deadtime in Even Channels** Define the deadtime value in even channels output PWM.

Yuntu Microelectronics

23.3.1.15 eTMR QDCTRL Register

Offset: 38h

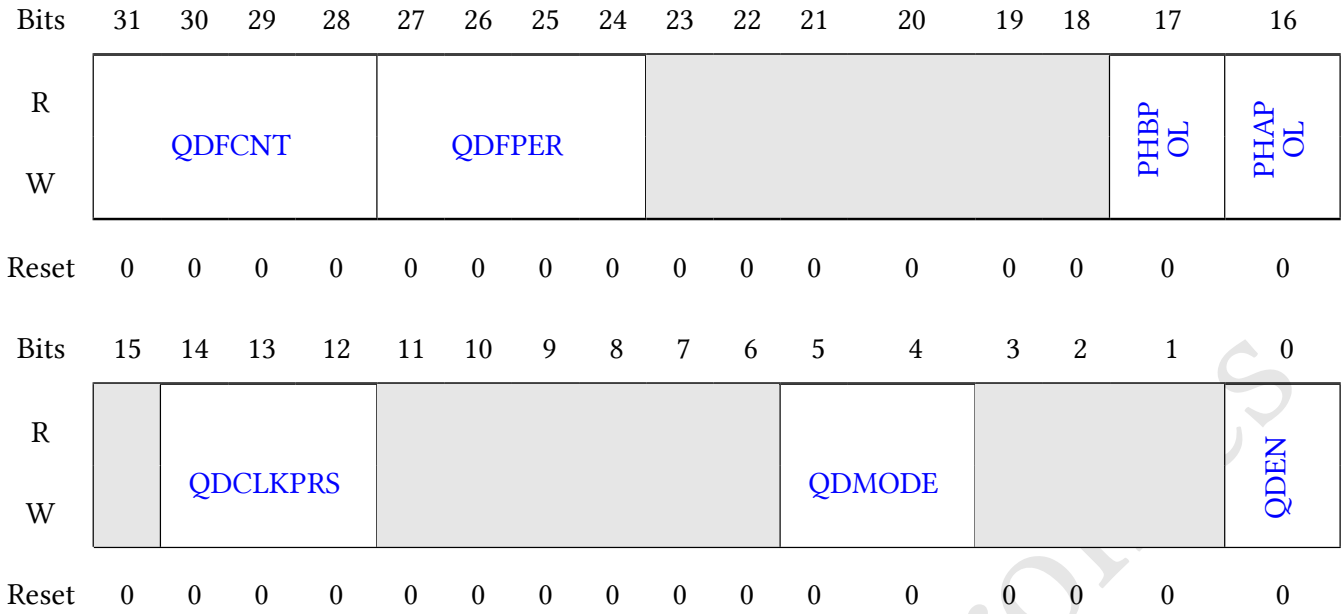


Table 23.23: eTMR QDCTRL Register Description

Field	Function
31 - 28 QDFCNT	**Quadrature Decoder Filter Sample Count** Quadrature decoder phase A and phase B input filter sample count, the input is ignored when its valid width was less than QDFCNT x QDFPER.
27 - 24 QDFPER	**Quadrature Decoder Filter Sample Period** Quadrature decoder phase A and phase B input filter sample period, the input is ignored when its valid width was less than QDFCNT x QDFPER.
17 PHBPOL	**Phase B Polarity** Phase B polarity selection. 0b - Not invert phase B input. 1b - Invert phase B input.
16 PHAPOL	**Phase A Polarity** Phase A polarity selection. 0b - Not invert phase A input. 1b - Invert phase A input.
14 - 12	**Quadrature Decoder Clock Prescaler**

Yuntu Microelectronics

Field	Function
QDCLKPRS	Clock prescaler for quadrature decoder counter. 000b - Divided by 1. 001b - Divided by 2. 010b - Divided by 4. 011b - Divided by 8. 100b - Divided by 16. 101b - Divided by 32. 110b - Divided by 64. 111b - Divided by 128.
5 - 4 QDMODE	**Quadrature Decoder Mode** Select decoding mode in the quadrature decoder mode. 00b - Phase A and phase B decoding mode. When phase B changs follow phase A, the counter increases. When phase A changs follow phase B, the counter decreases. 01b - Phase A and phase B decoding mode. When phase A changs follow phase B, the counter increases. When phase B changs follow phase A, the counter decreases. 10b - Counting and direction decoding mode. Phase B input controls direction. When phase B input is high, the counter increases with phase A pulse. When phase B input is low, the counter decreases with phase A pulse. 11b - Counting and direction decoding mode. Phase A input controls direction. When phase A input is high, the counter increases with phase B pulse. When phase A input is low, the counter decreases with phase B pulse.
0 QDEN	**Quadrature Decoder Enable** Quadrature decoder enable/disable. 0b - Quadrature decoder is disabled. 1b - Quadrature decoder is enabled.

23.3.1.16 eTMR FAULT Register

Offset: 3ch

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FFCNT				FFPER						STRETCH	FCOMB	RCVRY		FAUTO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W					F3POL	F2POL	F1POL	F0POL					F3EN	F2EN	F1EN	F0EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Yuntu Microelectronics

Table 23.24: eTMR FAULT Register Description

Field	Function
31 - 28 FFCNT	**Fault Filter Sample Count** Fault input filter sample count, the input is ignored when its valid width was less than FFCNT x FFPER.
27 - 24 FFPER	**Fault Filter Sample Period** Fault input filter sample period, the input is ignored when its valid width was less than FFCNT x FFPER.
21 STRETCH	**Fault Input Stretch** Fault input stretch to 2 bus clock cycles width at least.
20 FCOMB	**Fault Combinational Logic** When there is no bus clock, the fault event is generated by the combinational logic circuit. 0b - Fault combinational logic disable. 1b - Fault combinational logic enable.
19 - 18 RCVRY	**PWM Recovery** Select PWM recovery opportunity after fault recovery. 00b - PWM recovery is disabled. 01b - PWM recovers when counter matches with MOD[MOD]. 10b - PWM recovers immediately. 11b - Reserved.
17 - 16 FAUTO	**Fault Automatic Recovery Mode** Select fault automatic recovery mode after the fault event end. 00b - After clearing STS[FxF] flag manually, the fault recovers automatically. 01b - The fault recovers immediately, not care STS[FxF] flag. 10b - The fault recovers immediately and STS[FxF] flag clear automatically. 11b - Reserved.
11 F3POL	**Fault 3 Input Polarity** Fault 3 input polarity selection. 0b - Not invert fault 3 input. 1b - Invert fault 3 input.
10 F2POL	**Fault 2 Input Polarity** Fault 2 input polarity selection. 0b - Not invert fault 2 input. 1b - Invert fault 2 input.
9	**Fault 1 Input Polarity**

Yuntu Microelectronics

Field	Function
F1POL	Fault 1 input polarity selection. 0b - Not invert fault 1 input. 1b - Invert fault 1 input.
8 F0POL	**Fault 0 Input Polarity** Fault 0 input polarity selection. 0b - Not invert fault 0 input. 1b - Invert fault 0 input.
3 F3EN	**Fault 3 Enable** Fault 3 enable. 0b - Fault 3 disable. 1b - Fault 3 enbale.
2 F2EN	**Fault 2 Enable** Fault 2 enable. 0b - Fault 2 disable. 1b - Fault 2 enbale.
1 F1EN	**Fault 1 Enable** Fault 1 enable. 0b - Fault 1 disable. 1b - Fault 1 enbale.
0 F0EN	**Fault 0 Enable** Fault 0 enable. 0b - Fault 0 disable. 1b - Fault 0 enbale.

23.3.1.17 eTMR CH_CTRL Register

Offset: a0h + n * 20h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W		DMAEN		LDEN			CAPEDGE		CFCNT				CFPER			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VAL1CMP		VAL0CMP				PWMSRC		VAL1TEN	VAL0TEN	CHPOL	CHINIT			CHMODE	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Yuntu Microelectronics

Table 23.25: eTMR CH_CTRL Register Description

Field	Function
30 DMAEN	**DMA Enable** DMA enable/disable for compare/capture mode. 0b - DMA is disabled. 1b - DMA is enabled.
28 LDEN	**Registers Loading Enable** Current channel registers (CH_VAL0/CH_VAL1) loading enable/disable. 0b - Registers Loading is disabled. 1b - Registers Loading is enabled.
25 - 24 CAPEDGE	**Capture Edge** Select the capture edge in the input capture mode. 00b - Capture is disabled. 01b - Capture rising edge. 10b - Capture falling edge. 11b - Capture both rising edge and falling edge.
23 - 20 CFCNT	**Capture Filter Sample Counter** Capture input filter sample counter, the input is ignored when its valid width was less than CFCNT x CFPER.
19 - 16 CFPER	**Capture Filter Sample Period** Capture input filter sample period, the input is ignored when its valid width was less than CFCNT x CFPER.
15 - 14 VAL1CMP	**VAL1 Compare Mode** Select the output value when counter matches with CH_VAL1[VAL1] in compare mode. 00b - Output value 0 when counter matches with CH_VAL1[VAL1]. 01b - Output value 1 when counter matches with CH_VAL1[VAL1]. 10b - Output toggle when counter matches with CH_VAL1[VAL1]. 11b - Reserved.
13 - 12 VAL0CMP	**VAL0 Compare Mode** Select the output value when counter matches with CH_VAL0[VAL0] in compare mode. 00b - Output value 0 when counter matches with CH_VAL0[VAL0]. 01b - Output value 1 when counter matches with CH_VAL0[VAL0]. 10b - Output toggle when counter matches with CH_VAL0[VAL0]. 11b - Reserved.
9 - 8	**PWM Source**

Yuntu Microelectronics

Field	Function
PWMSRC	Select PWM source in PWM mode. 00b - Normal mode, PWM is generated by CH_VAL0[VAL0] and CH_VAL1[VAL1]. 01b - Invert mode, invert the PWM in normal mode. 10b - Software mode, PWM is generated by CHSWV register. 11b - Reserved.
7 VAL1TEN	**VAL1 Trigger Enable** Enable/disable generating trigger when counter matches with CH_VAL1[VAL1]. 0b - Not generate trigger when counter matches with CH_VAL1[VAL1]. 1b - Generate trigger when counter matches with CH_VAL1[VAL1].
6 VAL0TEN	**VAL0 Trigger Enable** Enable/disable generating trigger when counter matches with CH_VAL0[VAL0]. 0b - Not generate trigger when counter matches with CH_VAL0[VAL0]. 1b - Generate trigger when counter matches with CH_VAL0[VAL0].
5 CHPOL	**Channel Output Polarity** Channel output polarity control. 0b - Not invert the channel output. 1b - Invert the channel output.
4 CHINIT	**Channel Output Initialization** Write 1 to this bit to initialize channel output with CHINIT[CHxINIT], channel output change with a counter matching event.
1 - 0 CHMODE	**Channel Mode** Select channel mode. 00b - Channel disable. 01b - PWM mode. 10b - Compare mode. 11b - Capture mode.

23.3.1.18 eTMR CH_VAL0 Register

Offset: a4h + n * 20h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VAL0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Yuntu Microelectronics

Table 23.26: eTMR CH_VAL0 Register Description

Field	Function
15 - 0 VAL0	**Value 0** Define the value 0.

23.3.1.19 eTMR CH_VAL1 Register

Offset: a8h + n * 20h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VAL1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23.27: eTMR CH_VAL1 Register Description

Field	Function
15 - 0 VAL1	**Value 1** Define the value 1.

23.3.1.20 eTMR CH_CVAL Register

Offset: ach + n * 20h

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

Yuntu Microelectronics

Table 23.28: eTMR CH_CVAL Register Description

Field	Function
15 - 0 CVAL	**Capture Value** In capture mode, store counter value when a capture event occurs.

23.4 Functional Description

23.4.1 Clock

The eTMR module works in bus clock domain. Two clock sources can be selected via CTRL[CLKSRC] to drive the 16-bit counter: fast bus clock and external clock. The external clock signal is from the pad and synchronized by bus clock. So, the bus clock frequency should be at least two times the external clock frequency. CIM_ETMROPT0[ETMRxECS] defines three external clock sources that can be selected, see ETMR Option 0 Register for details. In addition, the eTMR module supports a 7-bit clock prescaler used for dividing the counter clock. The prescaler supports frequency division from 1 to 128.

Counter clock CLK_{CNT} configuration:

- Select clock source CLK_{SRC} by CTRL[CLKSRC]
- Divide the clock source CLK_{SRC} by CTRL[CLKPRS]

$$CLK_{CNT} = \frac{CLK_{SRC}}{CTRL[CLKPRS] + 1}$$

23.4.2 Counter

The eTMRx module has a 16-bit counter, the counter supports up counting only.

Usually, the counter counts from 0 to MOD[MOD] after the module is enabled. The initial value of the counter is only configurable when the module is disabled, write the initial value to CNT and enable the module, then the counter will count from the initial value to MOD[MOD].

Usually, when the module is enabled, any value written to CNT will load 0 to the counter.

Anyway, when the counter reaches MOD[MOD], the next value is always 0. As shown in Figure "eTMR Counter Overflow". If the counter increases from MOD[MOD] to 0, the timeout flag (STS[TOF]) will be set. The timeout flag (STS[TOF]) will be cleared by writing 1 to this bit.

Yuntu Microelectronics

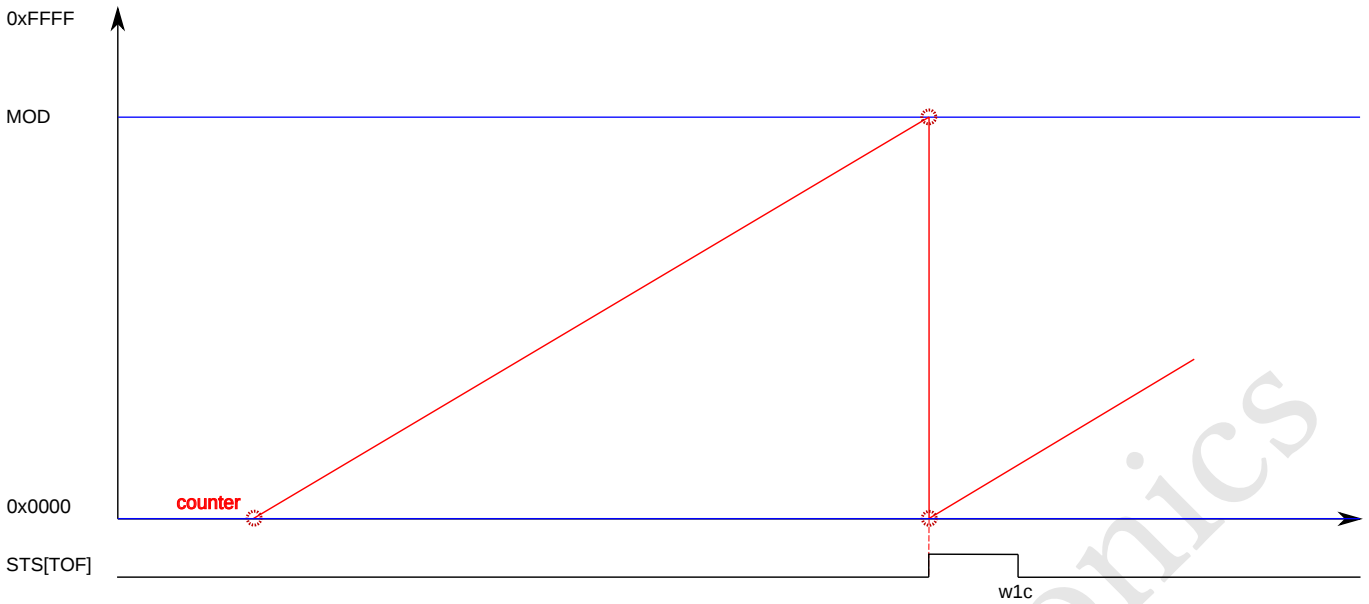


Figure 23.2: eTMR Counter Overflow

23.4.2.1 Global Time Base

Different eTMRs' counter can counts synchronously on the same time base. One eTMR acts as the master, other eTMRs are as slaves. CTRL[GLOBEN] = 1 means current eTMR works in master-slave mode. SYNC[GLOB] = 1 means current eTMR acts as master and other eTMRs act as slaves.

23.4.3 PWM Mode

When CH_CTRL[CHMODE]=0x1, the channel x is in PWM mode. Each channel is capable of generating center aligned PWM, edge aligned PWM and asymmetrical PWM. All channels support independent mode, specific channel pairs support complementary mode. In addition, the eTMR supports output fault control and mask control in PWM mode. As shown in Figure "eTMR PWM Generation Block Diagram".

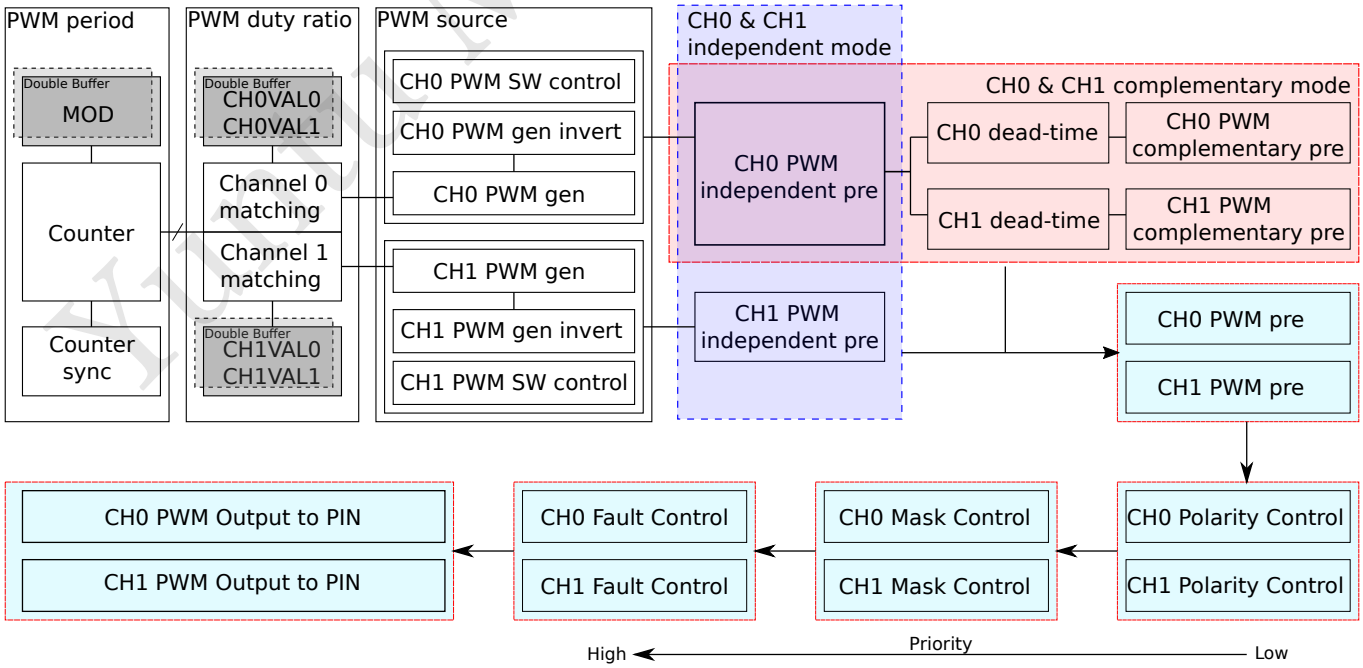


Figure 23.3: eTMR PWM Generation Block Diagram

Yuntu Microelectronics

The period of output PWM is up to counter clock and MOD[MOD]:

$$PER_{PWM} = \frac{MOD[MOD] + 1}{CLK_{CNT}}$$

The duty cycle of output PWM is up to CH_VAL0[VAL0] and CH_VAL1[VAL1], PWM set when the counter matches CH_VAL0[VAL0], PWM clear when the counter matches CH_VAL1[VAL1]:

- CH_VAL0[VAL0] < CH_VAL1[VAL1]:

$$DUTY_{PWM} = \frac{CH_VAL1[VAL1] - CH_VAL0[VAL0]}{PER_{PWM}} \times 100\%$$

- CH_VAL0[VAL0] > CH_VAL1[VAL1]:

$$DUTY_{PWM} = 1 - \frac{CH_VAL0[VAL0] - CH_VAL1[VAL1]}{PER_{PWM}} \times 100\%$$

23.4.3.1 Center Aligned PWM

Center aligned PWM shown in Figure "eTMR PWM Generation Block Diagram". The configuration steps are as follows:

- Determine MOD[MOD] and $DUTY_{PWM}$ as needed
- Calculate CH_VAL0[VAL0] and CH_VAL1[VAL1]

$$CH_VAL0[VAL0] = \frac{(1 - DUTY_{PWM}) \times (MOD[MOD] - INIT[INIT])}{2}$$

$$CH_VAL1[VAL1] = \frac{(1 + DUTY_{PWM}) \times (MOD[MOD] - INIT[INIT])}{2}$$

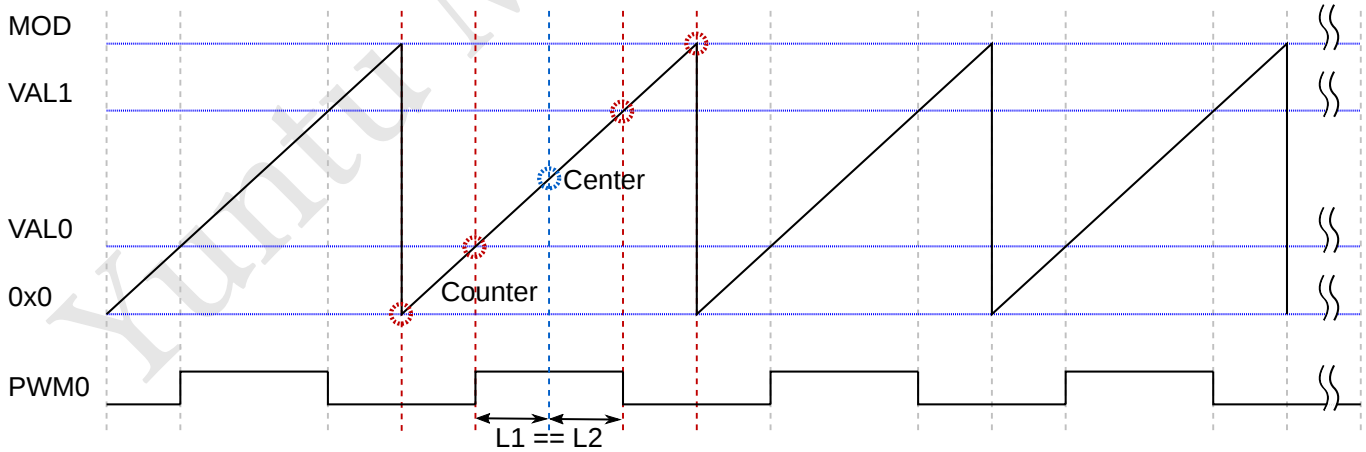


Figure 23.4: eTMR Center Aligned PWM

23.4.3.2 Edge Aligned PWM

Edge aligned PWM contains left edge aligned PWM and right edge aligned PWM. As shown in Figures "eTMR Left Edge Aligned PWM" and "eTMR Right Edge Aligned PWM"

Yuntu Microelectronics

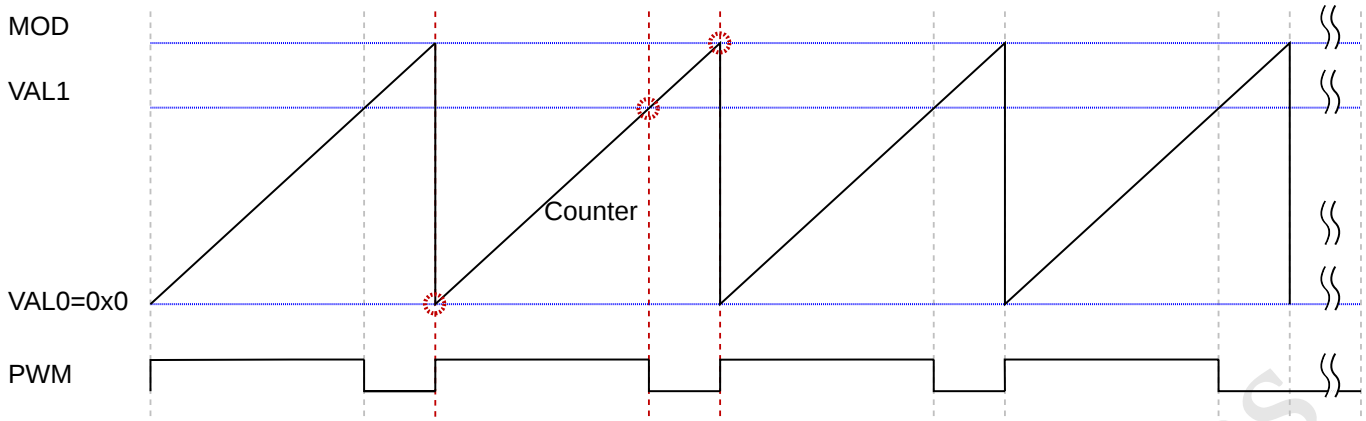


Figure 23.5: eTMR Left Edge Aligned PWM

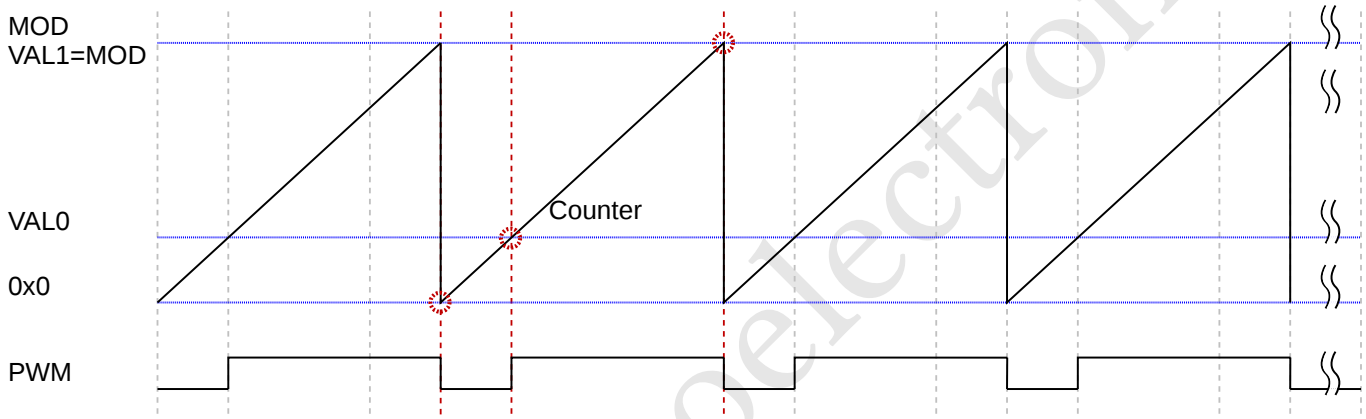


Figure 23.6: eTMR Right Edge Aligned PWM

Left edge aligned PWM configuration:

- Determine MOD[MOD] and $DUTY_{PWM}$ as needed
- Configure CH_VAL0[VAL0]=0x0000
- Calculate CH_VAL1[VAL1]

$$CH_VAL1[VAL1] = DUTY_{PWM} \times MOD[MOD]$$

Right edge aligned PWM configuration:

- Determine MOD[MOD] and $DUTY_{PWM}$ as needed
- Configure CH_VAL1[VAL1]=MOD[MOD]
- Calculate CH_VAL0[VAL0]

$$CH_VAL0[VAL0] = ((1 - DUTY_{PWM}) \times MOD[MOD])$$

23.4.3.3 Asymmetrical PWM

Because the rising edge and falling edge of PWM is both configurable, it's easy to generate asymmetrical PWM by configuring CH_VAL0[VAL0] and CH_VAL1[VAL1]. As shown in Figure "eTMR Asymmetrical PWM".

Asymmetrical PWM configuration:

Yuntu Microelectronics

- Determine MOD[MOD], and $DUTY_{PWM}$ as needed
- Configure CH_VAL0[VAL0] and CH_VAL1[VAL1] as needed

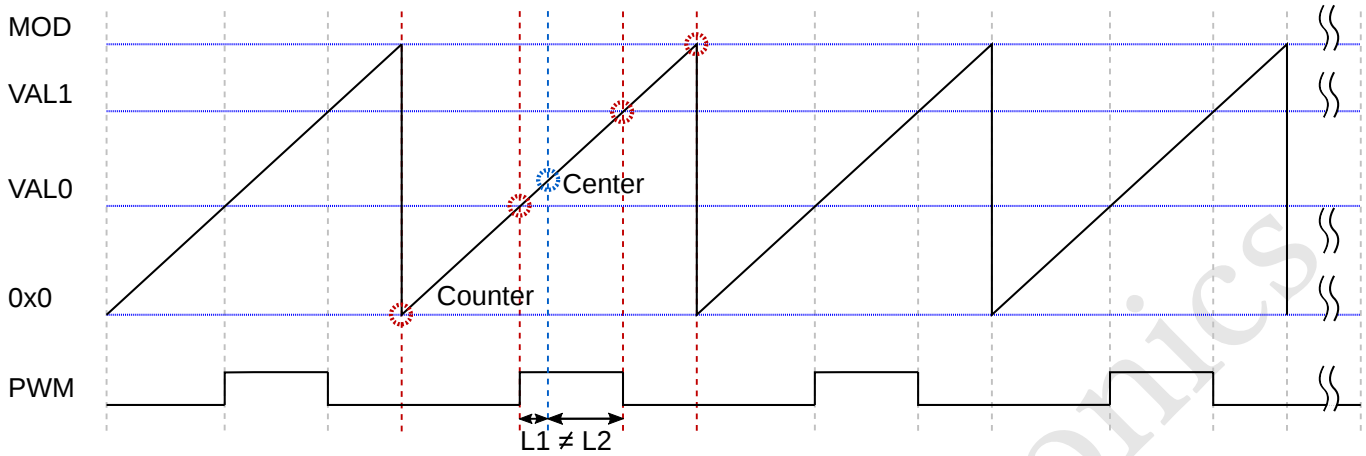


Figure 23.7: eTMR Asymmetrical PWM

23.4.3.4 Software-generated PWM

Software-generated PWM is generated by register CHSWV[CHxSWV] writing operations. Software-generated PWM sets when writing 1 to CHSWV[CHxSWV] and clears when writing 0 to CHSWV[CHxSWV].

23.4.3.5 PWM Source

The source of PWM is selected by CH_CTRL[PWMSRC]:

- CH_CTRL[PWMSRC]=0: select the raw PWM.
- CH_CTRL[PWMSRC]=1: select the inverted PWM, the raw PWM is inverted.
- CH_CTRL[PWMSRC]=2: select the software controlled PWM, the raw PWM is covered by CHSWV[CHxSWV].

23.4.3.6 Independent Mode

Each channel of the eTMR can output PWM independently, which is the independent mode. Channel 0/1 output PWM0/1 independently as shown in Figure eTMR Independent PWM.

Every channel has it's own control register, so it can use specific configurations to generate the independent PWM.

Independent PWM is selected when:

- CH_CTRL[CHMODE]=0x1
- CTRL[COMPLxx]=0

Yuntu Microelectronics

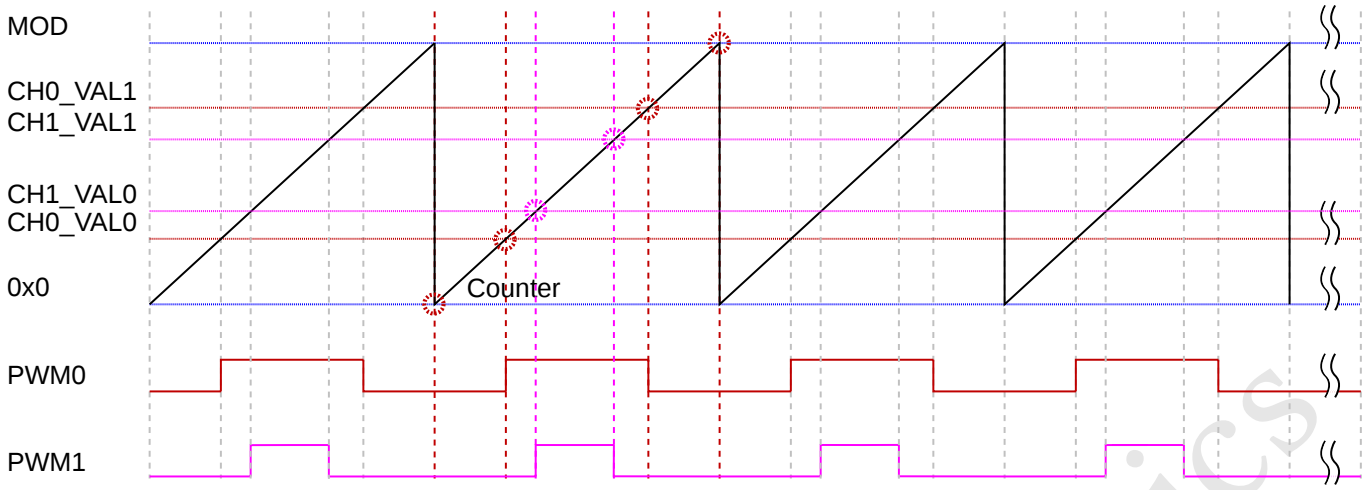


Figure 23.8: eTMR Independent PWM

23.4.3.7 Complementary Mode

Each channel pair of the eTMR can output complementary PWM, which is the complementary mode. Channel 0/1 output complementary PWM0/1 as shown in Figure "eTMR Complementary PWM".

Table 23.29: eTMR Channel Pair Table

Pair number	Description
0	Channel 0 and channel 1
1	Channel 2 and channel 3
2	Channel 4 and channel 5
3	Channel 6 and channel 7

When enabling complementary mode, the channel n+1 output PWM is complementary with the channel n output. The CH_VAL0[VAL0] and CH_VAL1[VAL1] of channel n+1 don't work in complementary mode.

Complementary PWM is selected when:

- CH_CTRL[CHMODE]=0x1
- CTRL[COMPLxx]=1

Yuntu Microelectronics

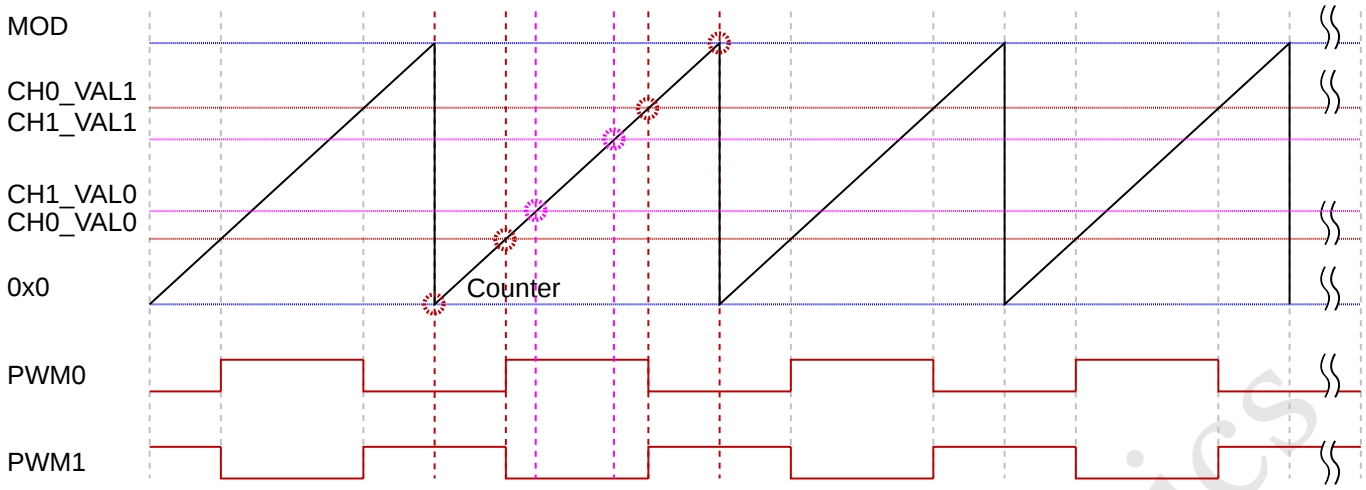


Figure 23.9: eTMR Complementary PWM

23.4.3.8 Complementary Mode with Dead-time Insertion

All channel pairs support deadtime insertion to stagger the edges of PWM in complementary mode.

Deadtime of odd channels is defined by DT[DTODD], deadtime of even channels is defined by DT[DTEVEN], the DT[DTODD] and DT[DTEVEN] work independently. Writing value 0 to DT[DTODD]/DT[DTEVEN] to disable the deadtime insertion. The deadtime counts by bus clock, and its final value is up to CTRL[DTPRS] and DT register. Channel 0/1 output complementary PWM0/1 as shown in Figure "eTMR Complementary PWM Deadtime Insertion".

Odd channels deadtime value :

$$VAL_{DTODD} = (CTRL[DTPRS] + 1) \times (DT[DTODD] + 1)$$

Even channels deadtime value :

$$VAL_{DTEVEN} = (CTRL[DTPRS] + 1) \times (DT[DTEVEN] + 1)$$

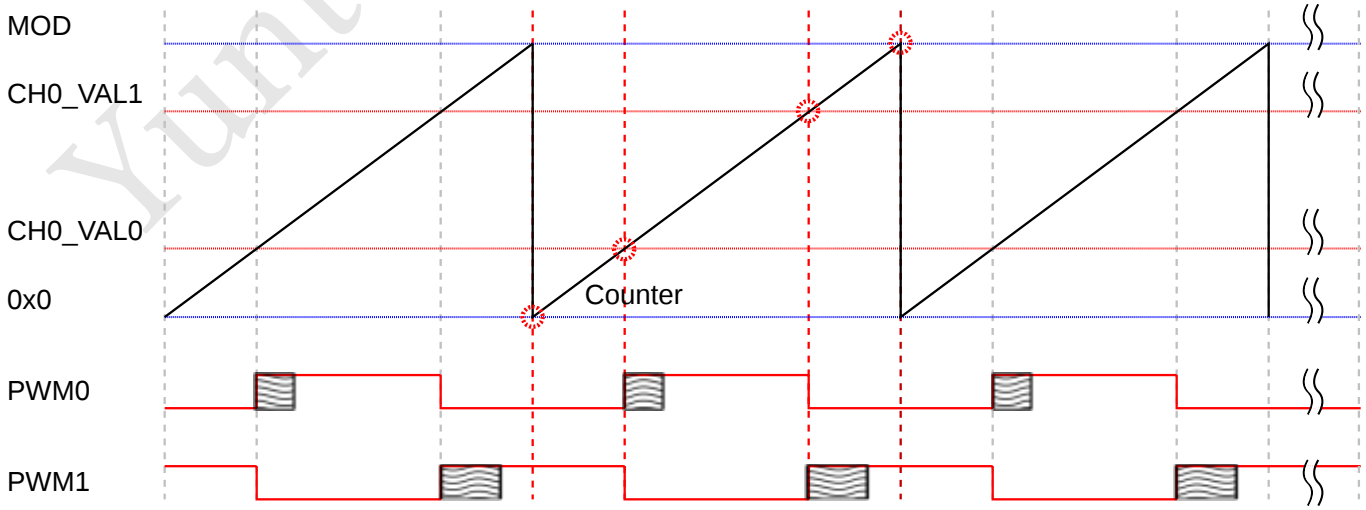


Figure 23.10: eTMR Complementary PWM Deadtime Insertion

Yuntu Microelectronics

23.4.3.9 Fault Detection and Protection

The eTMR0 and eTMR1 support fault input control, they each have four independent fault sources from TMU or pad. If a fault event is detected, the PWM output will force to the safe value. As shown in Figure "eTMR Fault Block Diagram".

There are four available fault sources, which is fault 0, fault 1, fault 2, and fault 3. Fault 0, fault 1 and fault 2 are optional from TMU or pad, fault 3 is only from pad. Fault function is disabled when FAULT[FxEN]=0, please configure the specific function mode first before enabling it.

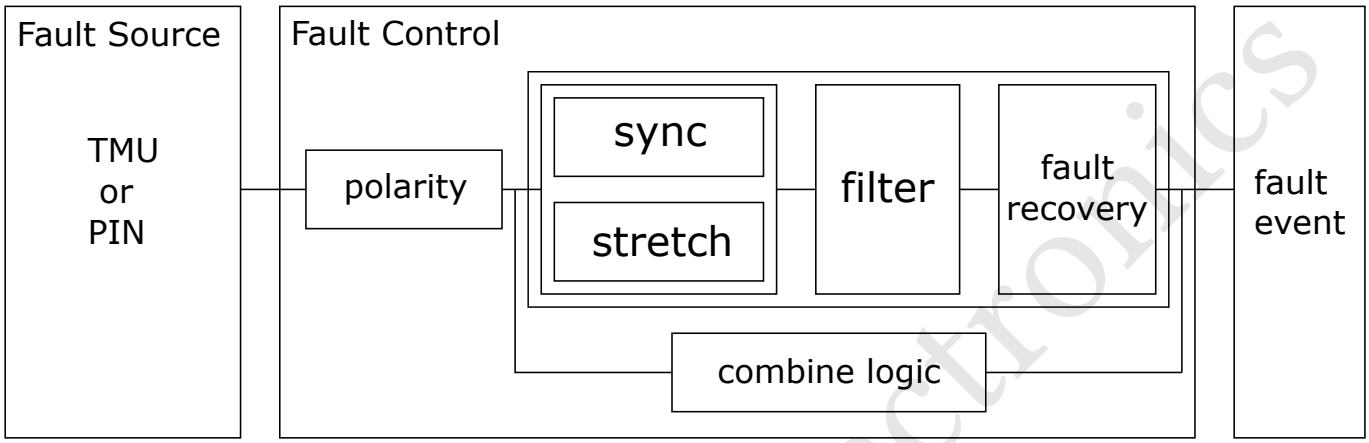


Figure 23.11: eTMR Fault Block Diagram

23.4.3.9.1 Safe Value

The fault function is available for all channels, and the fault event can force all channels to output the safe value. The safe value of each channel can be configured independently by the CHFV register. The safe value includes value 0, value 1, and tri-state.

23.4.3.9.2 Fault Source Selection

The fault source is defined by the CIM register, CIM_ETMROPT0[ETMR0FS] defines the source of eTMR0, CIM_ETMROPT0[ETMR1FS] defines the source of eTMR1. See eTMR0 fault selection and eTMR1 fault selection for details.

23.4.3.9.3 Fault Input Polarity

The polarity of the fault input is defined by FAULT[FxPOL]. When FAULT[FxPOL]=0, fault input 1 is valid. When FAULT[FxPOL]=1, fault input 0 is valid.

23.4.3.9.4 Fault Filter

The fault inputs are filtered by the digital filter. The valid input pulse width is defined by FAULT[FFCNT] and FAULT[FFPER], the input will be ignored when its valid width is less than FAULT[FFCNT] × FAULT[FFPER].

23.4.3.9.5 Input Stretch

To generate a fault event successfully, the fault input pulse should be bigger than one bus clock cycle. If the input signal is less than one bus clock cycle from other modules, the stretch function can stretch the

Yuntu Microelectronics

input signal and generate a fault event. Set FAULT[STRETCH]=1 to enable the stretch function. The input signal after being stretched is at least two bus clock cycles. As shown in Figure "eTMR Fault Stretch".

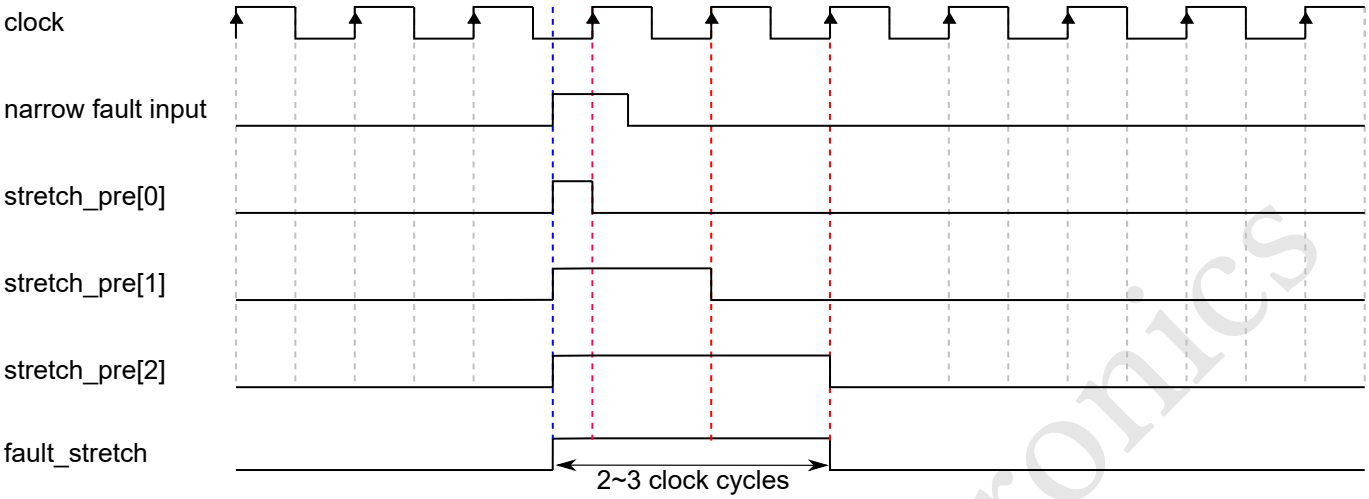


Figure 23.12: eTMR Fault Stretch

23.4.3.9.6 Combinational Logic Fault

When there is no bus clock, the fault event can only be generated by the combinational logic. Set FAULT[FCOMB]=1 to enable the combinational logic fault input. Any valid input signal results in a fault event when the combinational logic fault is enabled. The fault event generated by the combinational logic will keep until the bus clock recovery.

NOTE: The fault flag won't be set while a fault event is generated by the combinational logic without bus clock.

23.4.3.9.7 Fault Recovery

The fault event clear opportunity is selectable after removing the fault input. The fault event clear opportunity can be immediate or when the counter matched with the MOD value, and it can also keep until disabling the fault. Configure FAULT[RCVRY] to define the fault event clear opportunity.

23.4.3.9.8 Fault Flag Clear

The fault flag (STS[FxF]) will be set when a fault event occurs. The fault flag and the fault recovery work together after the fault input is removed. When the fault event clear opportunity comes, it can be selected that the fault event is cleared immediately or cleared after clearing the fault flag. Configure FAULT[FAUTO] to choose the fault flag clearing mode.

23.4.3.10 Mask Control

The mask function is available for all channels in PWM mode. It's able to mask the PWM output by value 0 or value 1, and it can also disable the PWM output by masking it to tri-state. This function is configured by the CHMASK register. See Channel Output Mask Value Register for details.

Yuntu Microelectronics

23.4.4 Output Compare Mode

When CH_CTRL[CHMODE]=0x2, the channel x is in output compare mode. In this mode, each channel can set, clear, or toggle the output when the counter reaches the specific value defined by CH_VAL0[VAL0] and CH_VAL1[VAL1]. Configure CH_CTRL[VAL0CMP] and CH_CTRL[VAL1CMP] to select the action of the corresponding channel when the counter reaches the CH_VAL0[VAL0] and CH_VAL1[VAL1]. As shown in Figure eTMR Output Compare. The channel flag (STS[CHxF]) will be set when a compare event occurs.

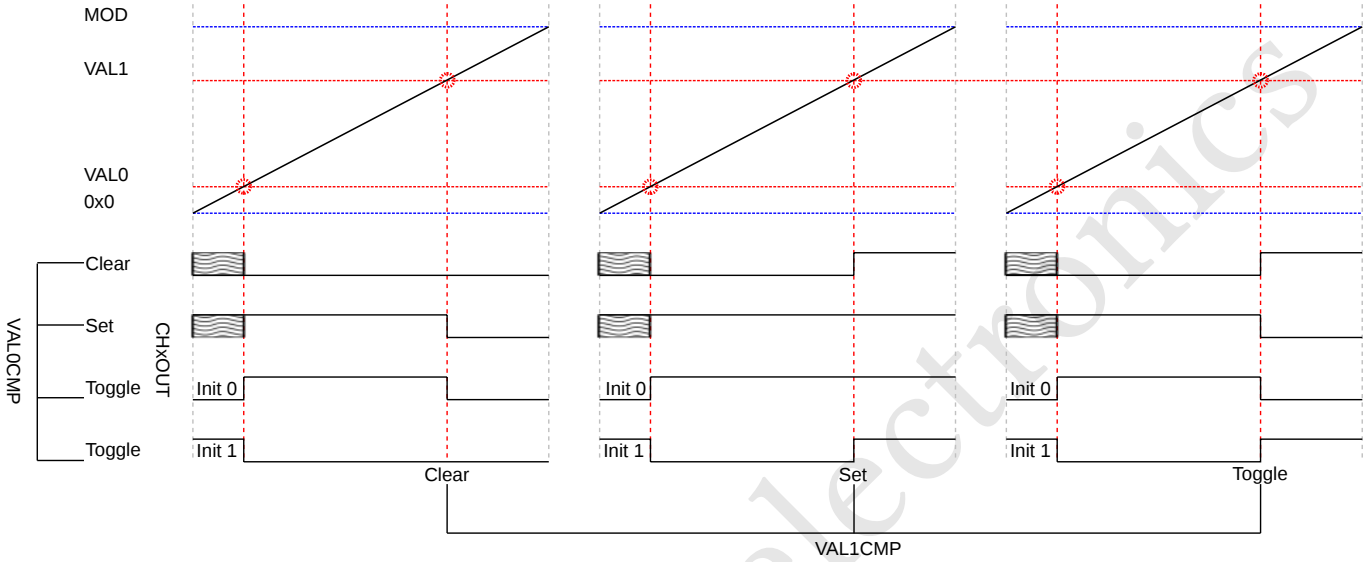


Figure 23.13: eTMR Output Compare

23.4.5 Input Capture Mode

When CH_CTRL[CHMODE]=0x3, the channel x is in input capture mode. In this mode, each channel can detect the input signal change defined by CH_CTRL[CAPEDGE]. When the capture event occurs, the current counter value is saved to the CVAL register. As shown in Figure eTMR Input Capture. The channel flag (STS[CHxF]) will be set when a capture event occurs.

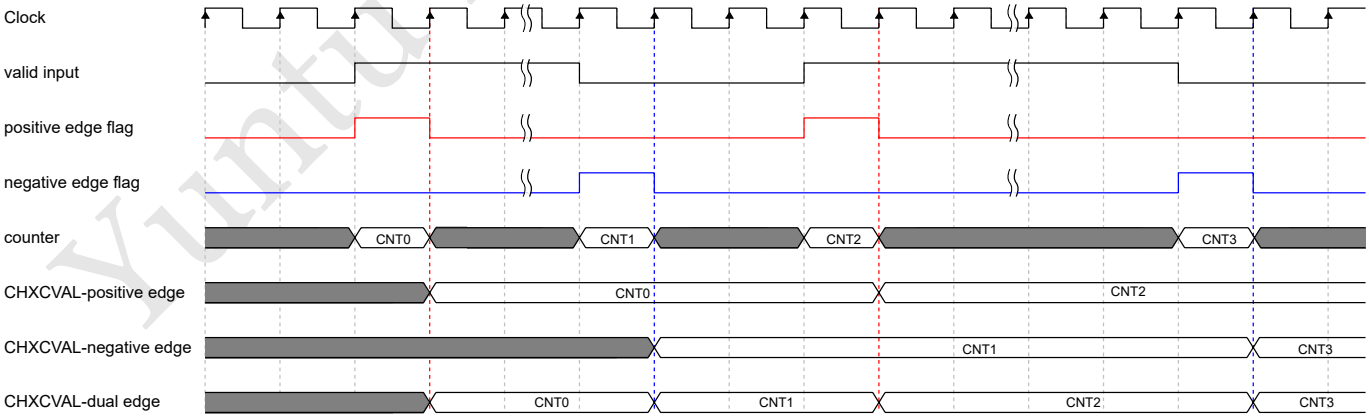


Figure 23.14: eTMR Input Capture

23.4.5.1 Channel Filter

The channel inputs are filtered by the digital filter. The valid input pulse width is defined by CH_CTRL[CFCNT] and CH_CTRL[CFPER], the input will be ignored when its valid width is less

than CH_CTRL[CFCNT] × CH_CTRL[CFPER].

23.4.5.2 Capture Test

When CTRL[CAPTST]=1 and all channels are in input capture mode, the capture test mode is enabled. In this mode, the counter stops and can write a new value to the counter. After the capture mode is disabled, the counter will continue counting based on the new value. When writing the counter in capture test mode, the channel flag (STS[CHxF]) will be set, and the current counter value is saved to the CVAL register.

23.4.6 Output Polarity

In PWM mode or output compare mode, the channel output is controlled by the polarity defined by CH_CTRL[CHPOL]. When CH_CTRL[CHPOL]=0, the corresponding channel outputs the raw signal; When CH_CTRL[CHPOL]=1, the corresponding channel outputs the inverted signal.

23.4.7 Channels Initialization

In PWM mode or output compare mode, when the eTMR is disabled, the channels output the initial value defined by the CHINIT register. When the eTMR is enabled, write value 1 to CH_CTRL[CHINIT] also results in the corresponding channel outputting the initial value.

23.4.8 Loading Mechanism

Some registers are designed with buffer registers, which are CHMASK, MOD, CH_VAL0, and CH_VAL1. When the eTMR is enabled, these registers' value is locked and they can only change at the appropriate opportunity. The loading mechanism is applied to manage the update of these special registers. As shown in Figure eTMR Loading Mechanism.

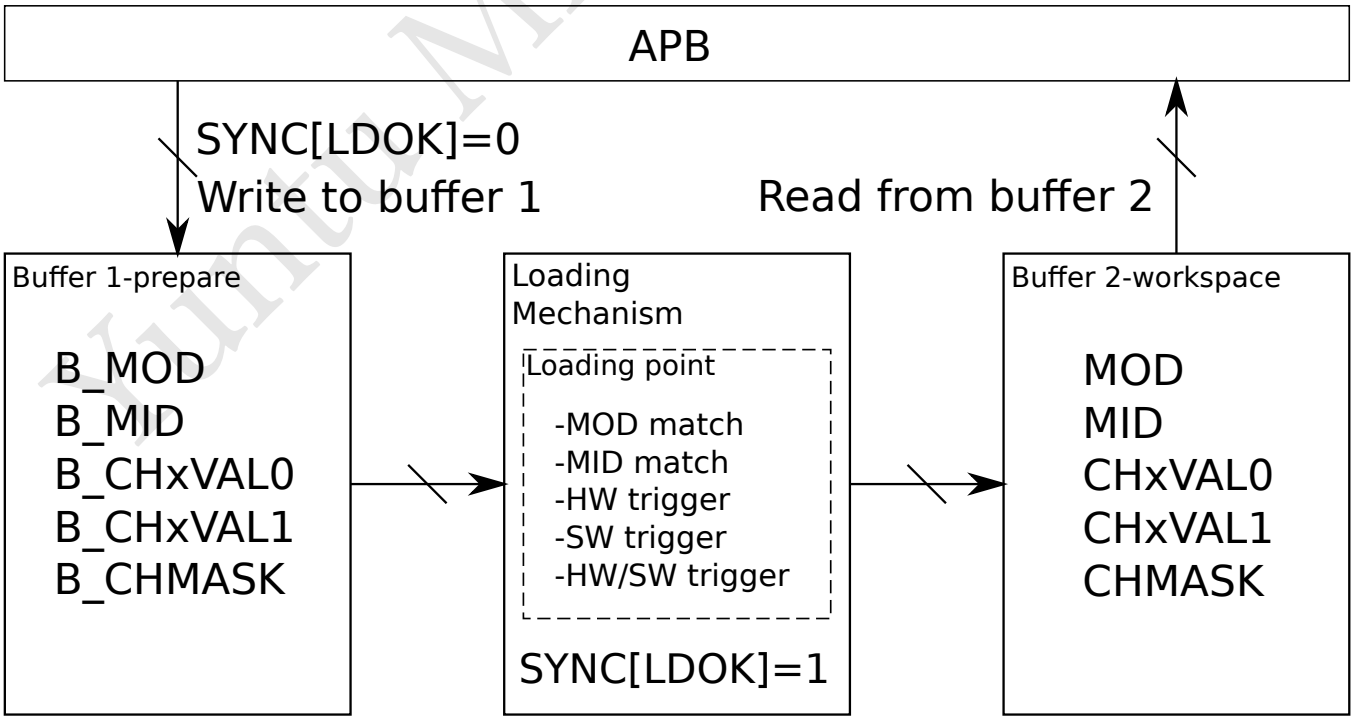


Figure 23.15: eTMR Loading Mechanism

Yuntu Microelectronics

Registers loading configuration:

- Select loading opportunity
- Configure trigger if need
- Configure loading frequency if need
- Write buffer registers
- Set SYNC[LDOK]=1

The reload flag (STS[RF]) will be set if registers load successfully.

23.4.8.1 General Registers Loading

The general registers include MOD, CH_VAL0, and CH_VAL1, and they load with respective buffer registers at the same time point. These registers can update when the counter matches with MOD[MOD]. As shown in Figure eTMR Loading Mechanism Timing. In addition, These registers can update when a loading trigger occurs. As shown in Figure eTMR Loading by Trigger Timing. Configure SYNC[REGLD] to select the loading opportunity of the general registers. Set CH_CTRL[LDEN]=1 to enable the loading of CH_VAL0 and CH_VAL1.

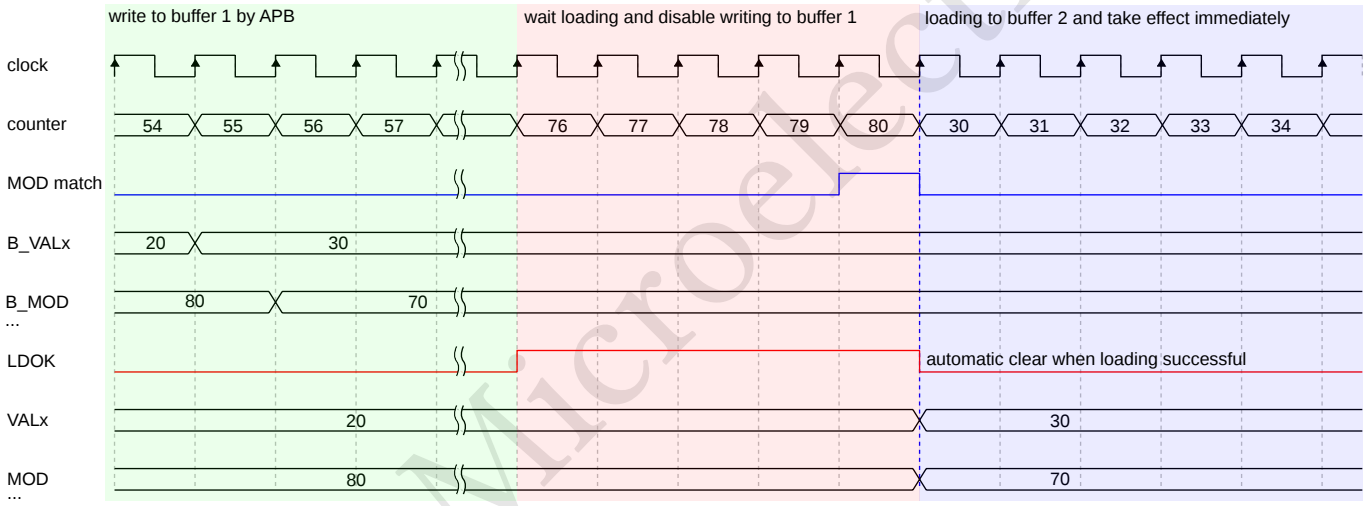


Figure 23.16: eTMR Loading Mechanism Timing

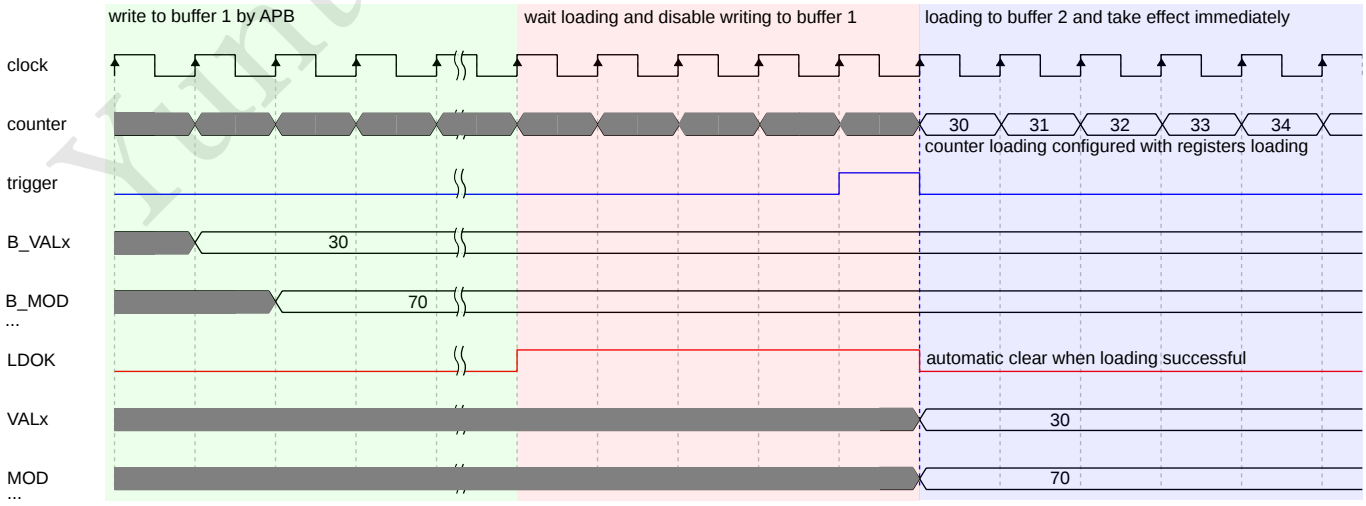


Figure 23.17: eTMR Loading by Trigger Timing

Yuntu Microelectronics

23.4.8.2 Loading Frequency

The loading frequency is applied for executing registers' loading when the loading opportunity comes at specific times. If the loading opportunity counts less than SYNC[LDFREQ], the registers' loading won't be executed.

23.4.8.3 Special Registers loading

The special registers include CHMASK and CNT, and they update at the configurable time point. The CHMASK register can update when the counter matches with MOD[MOD] or a loading trigger occurs, and it can also update together with the general registers. Configure SYNC[MASKLD] to select the loading opportunity of the CHMASK register. The CNT register can update when a loading trigger occurs, and it can also update together with the general registers. Configure SYNC[CNTLD] to select the loading opportunity of the CNT register.

NOTE: The CNT register always load value 0 when a loading event occurs.

23.4.8.4 Loading Trigger

There are three hardware triggers and one software trigger used for registers' loading. Configure SYNC[TRIGSx] to select the trigger source which is used for registers' loading.

The hardware triggers are from the TMU (trigger 0), CIM (trigger 1), and pad (trigger 2), and they are enabled by setting SYNC[TRIGxEN]=1. Any valid hardware trigger x will generate a trigger event (x=0,1,2).

Writing value 1 to SYNC[SWTRIG] can generate a software trigger.

23.4.9 Quadrature Decoder

The eTMR1 supports the quadrature decoder mode, the mode is enabled when QDCTRL[QDEN]=1. In this mode, the independent 16-bit counter will increase or decrease between 0x0 and 0xFFFF according to the input signals from the pad. As shown in Figure "eTMR Quadrature Decoder". The STS[QDDIR] flag indicates the counting direction of the counter. The STS[QDTOF] flag will be set if the counter overflow occurs.

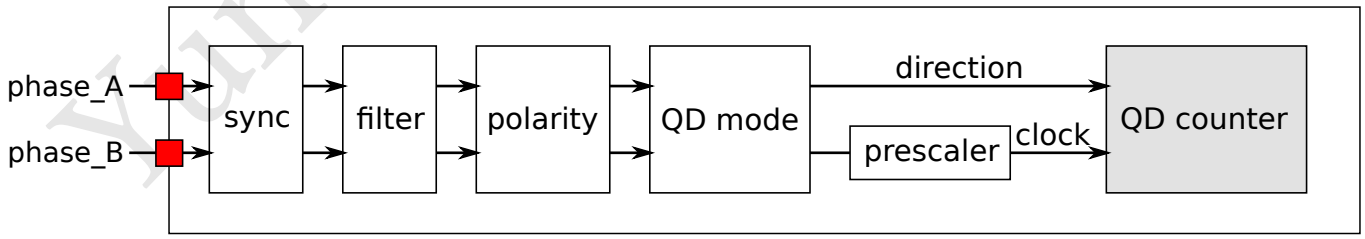


Figure 23.18: eTMR Quadrature Decoder

23.4.9.1 Decoder Prescaler

In the quadrature decoder mode, a 3-bit prescaler is available for the independent counter. The prescaler can be configured by QDCTRL[QDCLKPRS].

23.4.9.2 Count and Direction Decoder Mode

Configure QDCTRL[QDMD]=2 or QDCTRL[QDMD]=3 to enable the direction decoding mode. In this mode, one of the input indicates the counting direction, the other input defines the counting rate. As shown in Figure [eTMR-Quadrature-Decoder-Count-and-Direction-Mode]{eTMR Quadrature Decoder Count and Direction Mode}.

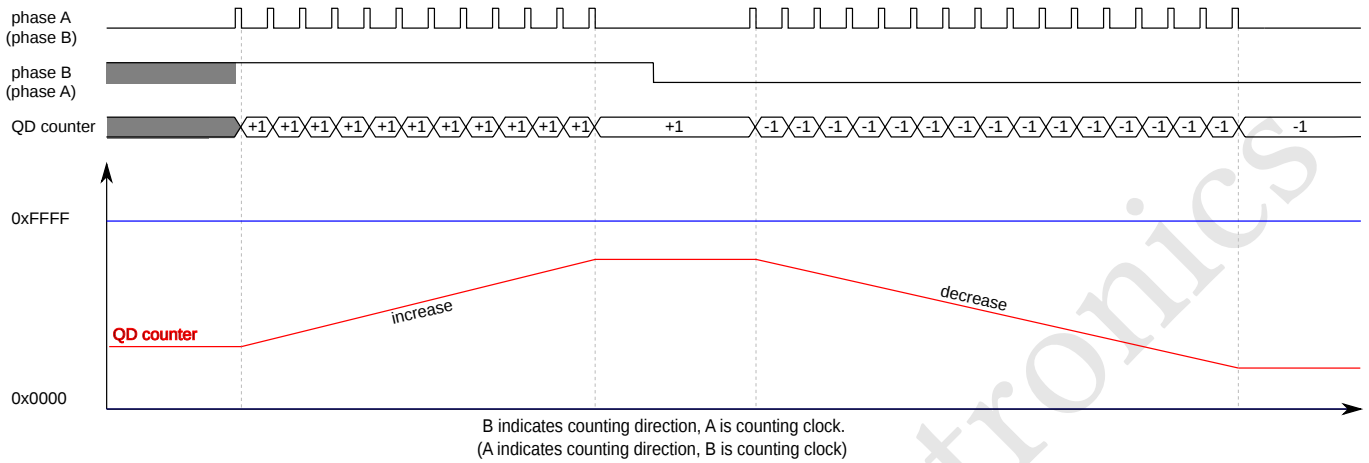


Figure 23.19: eTMR Quadrature Decoder Count and Direction Mode

23.4.9.3 Phase A and Phase B Decoder Mode

Configure QDCTRL[QDMD]=0 or QDCTRL[QDMD]=1 to enable the phase decoding mode. In this mode, the relationship between two input signals indicates the counting direction, the two input signals define the counting rate. As shown in Figure ”eTMR Quadrature Decoder Count and Direction Mode”.

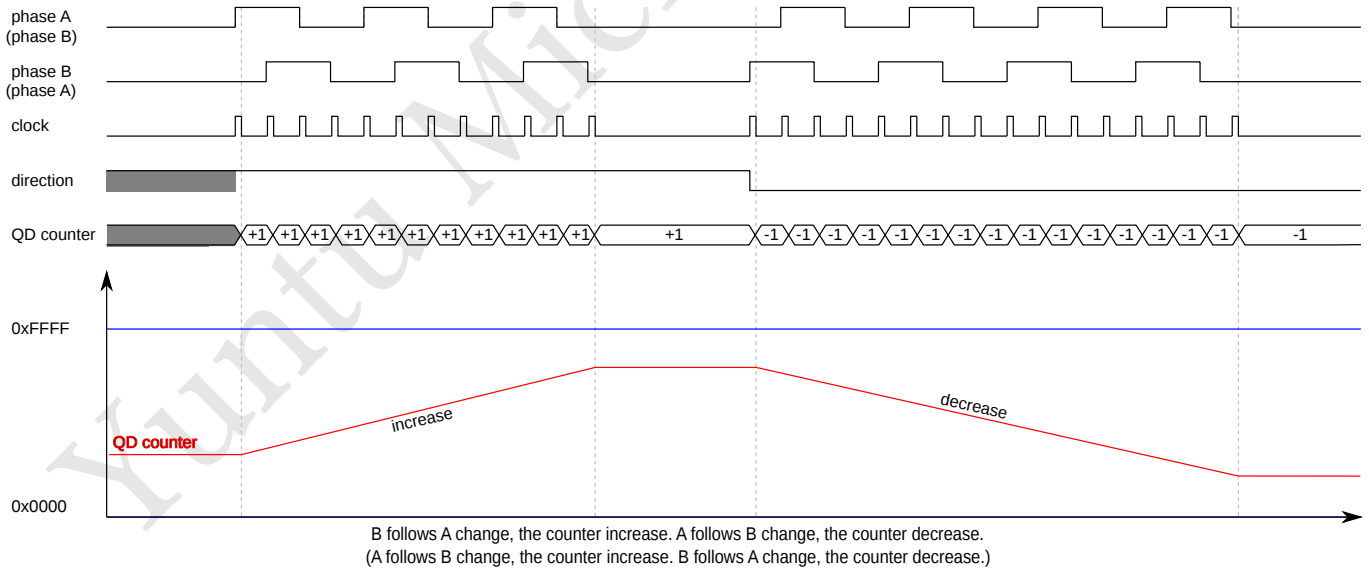


Figure 23.20: eTMR Quadrature Decoder PhaseA and PhaseB Mode

23.4.9.4 Decoder Input Polarity

The polarity of the decoder input is defined by QDCTRL[PHxPOL]. When QDCTRL[PHxPOL]=0, the decoder input is the raw input signal. When QDCTRL[PHxPOL]=1, the decoder input is the inverted input signal.

Yuntu Microelectronics

23.4.9.5 Decoder Filter

The quadrature decoder inputs are filtered by the digital filter. The valid input pulse width is defined by QDCTRL[QDFCNT] and QDCTRL[QDFPER], the input will be ignored when its valid width is less than QDCTRL[QDFCNT] × QDCTRL[QDFPER].

23.4.9.6 Decoder Counter Overflow

Quadrature Decoder counter contains up counting overflow and down counting overflow. As shown in Figure eTMR Quadrature Decoder Counter Overflow.

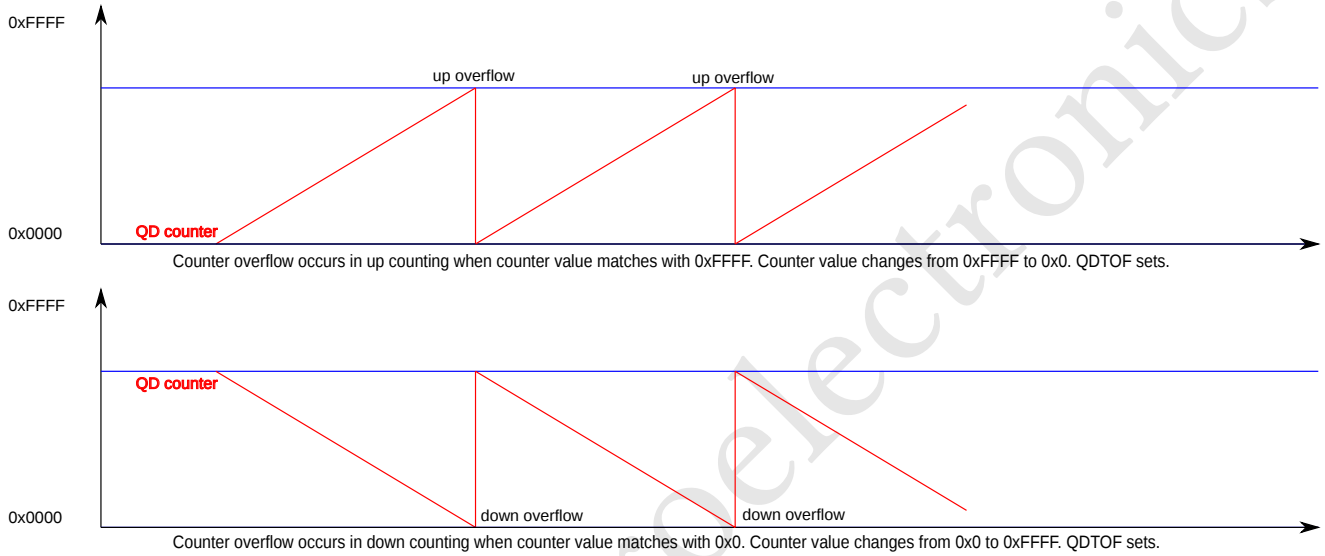


Figure 23.21: eTMR Quadrature Decoder Counter Overflow

23.4.10 Filter

Filter used for input capture mode, fault input and phase A/phase B in quadrature decoder mode. As shown in Figure eTMR Filter Timing.

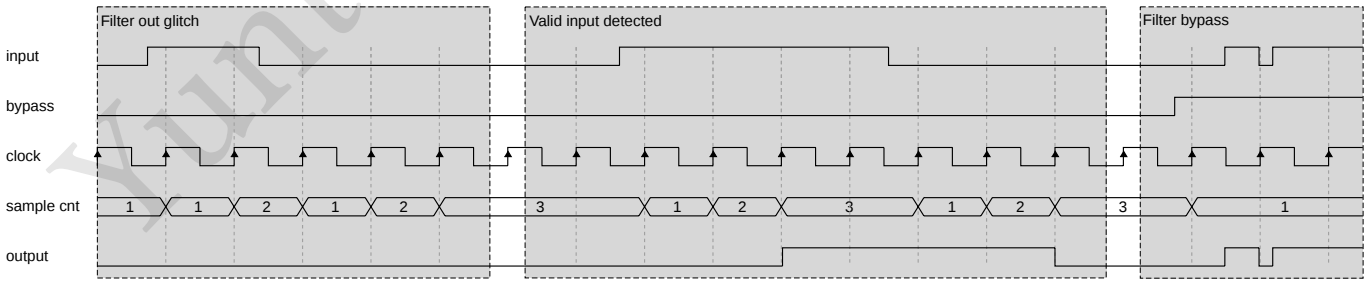


Figure 23.22: eTMR Filter Timing

23.4.11 Output Trigger

The eTMR can output triggers that send to the TMU module. The output trigger can be width fixed pulse or width adjustable pulse.

Yuntu Microelectronics

23.4.11.1 Width Fixed Pulse

Configure OTRIG[TSRC]=0 to enable the generating of width fixed pulse. The width fixed pulse is generated by counter matching, and there are three available match points: MOD, VAL0, and VAL1. The generated pulses are all one bus clock cycle width. When the counter reaches MOD[MOD] and OTRIG[MODTEN]=1, the MOD match pulse generates; When the counter reaches CH_VAL0[VAL0] and CH_CTRL[VAL0TEN]=1, the VAL0 match pulse generates; When the counter reaches CH_VAL1[VAL1] and CH_CTRL[VAL1TEN]=1, the VAL1 match pulse generates.

As shown in Figure eTMR Output Trigger.

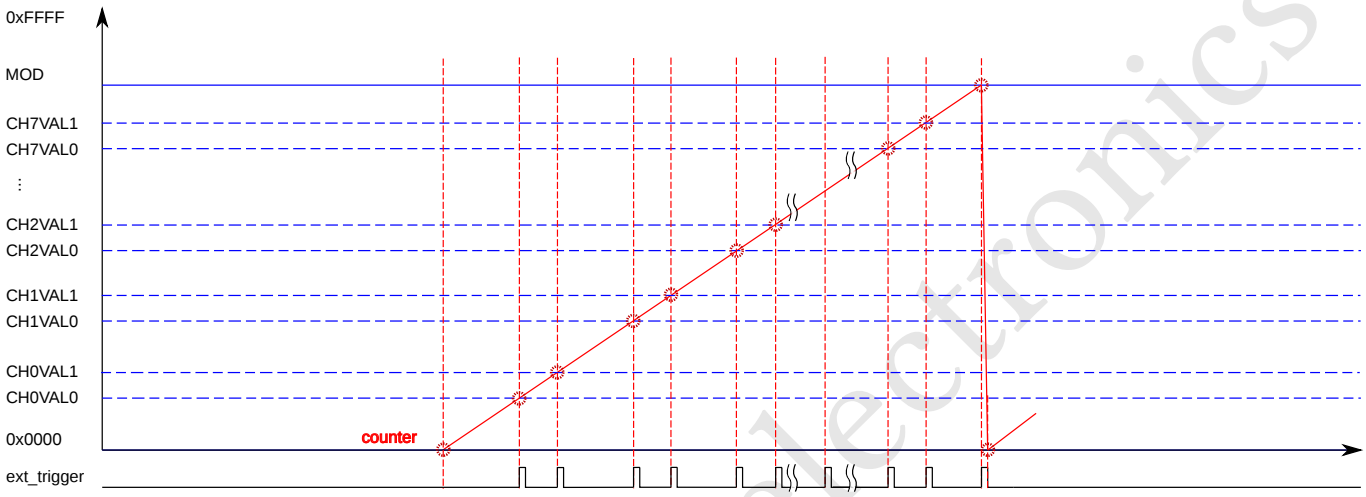


Figure 23.23: eTMR Output Trigger

23.4.11.2 Width Adjustable Pulse

Configure OTRIG[TSRC]=1 to enable the generating of width adjustable pulse. The width adjustable pulse is generated by channel in PWM mode, the PWM output can be used as the output trigger. Configure OTRIG[PULSESRC] to select the PWM source.

23.4.11.3 Output Frequency

The output frequency is applied for outputting pulse when the output opportunity comes at specific times. If the output opportunity counts less than OTRIG[TFRQ], the pulse won't be generate. The output frequency is only used to generate width fixed pulse.

23.4.12 Input Capture Source

The eTMR1 channel 0 and eTMR2 channel 0 input capture sources are selected by CIM_ETMROPT1[ETMRxCH0S]. Their input capture signal can be from the ACMP output. See ETMR1 Channel 0 Select and ETMR2 Channel 0 Select for details.

23.4.13 Hall Sensor Support

To simplify the calculations required by the CPU on each hall sensor's input, the eTMR2 channel 1 supports hall input which from the inputs of eTMR channel 1, eTMR2 channel 0, and eTMR2 channel 1. This function is enabled by setting CIM_ETMROPT1[ETMR2CH1S]=1. See ETMR2 Channel 1 Select for details.

Yuntu Microelectronics

23.4.14 Modulation Implementation

The eTMR0 supports channel outputs modulated by eTMR1 channel 1 output. Any of the 8 channels of eTMR0 can be configured to support this modulation function. This function is enabled by configuring CIM_ETMROPT1[ETMR0OMS]. See ETMR0 Output Modulation Select.

23.4.15 Power Mode

The eTMRs can work in active mode and sleep mode.

23.4.16 Interrupts

The timeout interrupt and channel interrupt are available in all eTMRs. The timeout interrupt is generated by STS[TOF] when INTE[TOIE]=1. Write value 1 to STS[TOF] to clear the timeout interrupt. The channel interrupt is generated by STS[CHxF] when INTE[CHxIE]=1. Write value 1 to STS[CHxF] to clear the channel interrupt.

The fault interrupt is only available in eTMR0 and eTMR1. The fault interrupt is generated by STS[FxF] when INTE[FIE]=1. Write value 1 to STS[FxF] to clear the fault interrupt.

The quadrature decoder counter timeout interrupt is only available in eTMR1. The quadrature decoder counter timeout interrupt is generated by STS[QDTOF] when INTE[QDTOIE]=1. Write value 1 to STS[QDTOF] to clear the quadrature decoder counter timeout interrupt.

23.4.17 DMA

All eTMRs support DMA, this function is enabled by setting CH_CTRL[DMAEN]=1. The DMA request is generated when STS[CHxF]=1. The DMA request is cleared by clearing STS[CHxF]. The DMA priority is higher than the interrupt.

23.4.18 Reset

During a reset, the eTMRs are in the default mode and disabled.

Yuntu Microelectronics

Chapter 24

Real Time Clock (RTC)

24.1 Introduction

The Real Time Clock (RTC) provides a real time counter which is present in always on domain, hence available in all power modes of MCU. RTC generate interrupts when RTC counter reaches the values of ALM register.

24.1.1 Features

- Register write protection for RTC enable register
- 32-bit seconds counter with overflow flag and optional interrupt
- Configurable 32-bit alarm
- Configurable 1, 2, 4, 8, 16, 32, 64 or 128 Hz square wave output with optional interrupt

24.2 Register Definition

This subsection provides a description of the memory map and registers.

24.2.1 RTC Memory Map

RTC base address 4003d000h

Table 24.1: RTC Memory Map

Offset	Register	Width	Access	Reset Value
0h	Second Register (SEC)	32	RW	00000000h
8h	Compensation Register (CMP)	32	RW	00000000h
ch	Alarm Register (ALM)	32	RW	00000000h
10h	Enable Register (EN)	32	RW	00000000h
14h	Control Register (CTRL)	32	RW	00000000h
18h	Interrupt Flag Register (INTF)	32	RW	00000000h
1ch	Interrupt Enable Register (INTE)	32	RW	00000003h

24.2.1.1 RTC SEC Register

RTC Second Register

Yuntu Microelectronics

Offset: 0h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SEC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SEC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24.2: RTC SEC Register Description

Field	Function
31 - 0 SEC	**Second Value** SEC is read-only when RTC is enabled. The SEC increments once a second provided INTF[OVFIF] is not set. The time counter will read as zero when INTF[OVFIF] is set. When RTC is disabled, SEC can be read and written. Writing to this bit will clear the INTF[OVFIF]. Writing to SEC with zero is not recommended because SEC will read as zero when INTF[OVFIF] is set.

Yuntu Microelectronics

24.2.1.2 RTC CMP Register

RTC Compensation Register

Offset: 8h

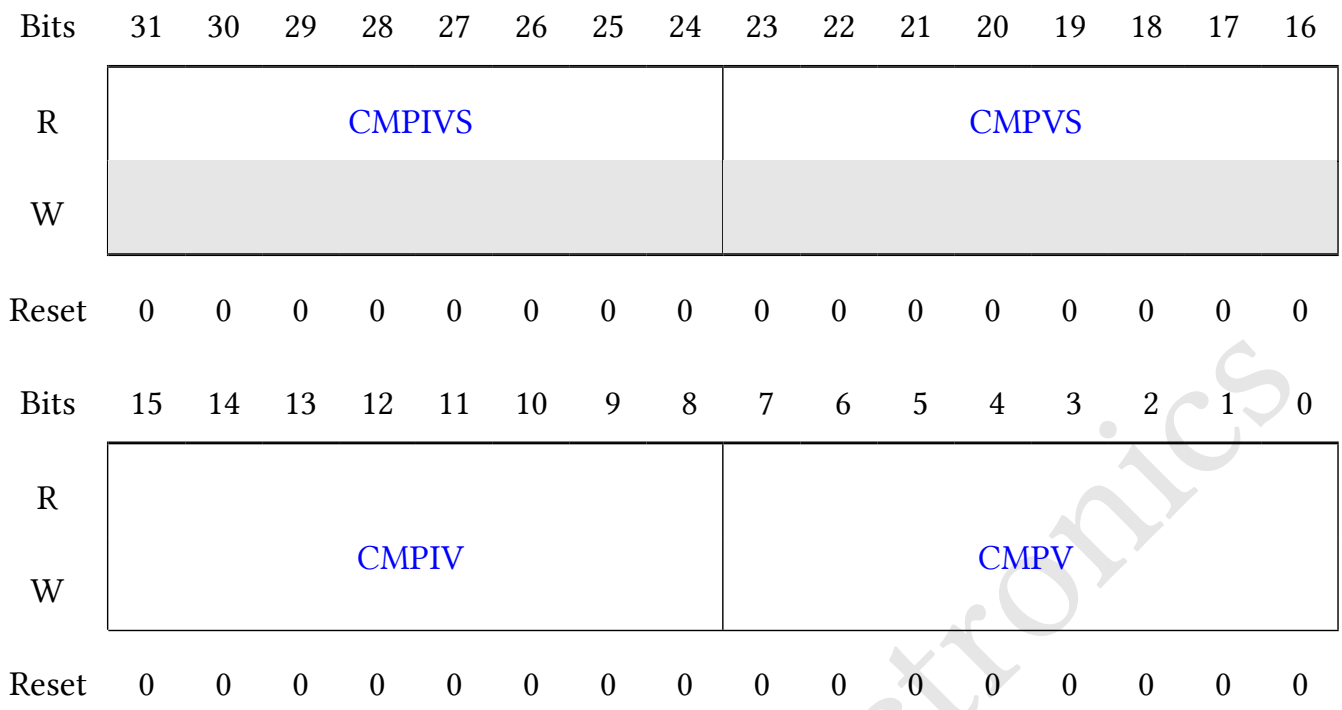


Table 24.3: RTC CMP Register Description

Field	Function
31 - 24 CMPIVS	**Compensation Interval Status** Current value of the compensation interval counter. If the compensation interval counter equals zero then it is loaded with the contents of the CMPIV. If the CMPV does not equal zero then it is decremented once a second. Reading this register at the same time as the seconds counter is incrementing can result in an incorrect value being read.
23 - 16 CMPVS	**Compensation Value Status** Current value used by the compensation logic for the present second interval. Updated once a second if the CMPIVS equals 0 with the contents of the CMPV field. If the CMPIVS does not equal zero then it is loaded with zero (compensation is not enabled for that second increment). Reading this register at the same time as the seconds counter is incrementing can result in an incorrect value being read.
15 - 8 CMPIV	**Compensation Interval Value** Configures the compensation interval in seconds from 1 to 256 to control how frequently the CMPV should adjust the number of 32.768 kHz cycles in each second. The value which is written should be one less than the number of seconds. For example, write zero to configure for a compensation interval of one second. This register is double-buffered and writes do not take effect until the end of the current compensation interval.
7 - 0	**Compensation Value**

Yuntu Microelectronics

Field	Function
CMPV	Configures the number of 32.768 kHz clock cycles in each second, equal to 32,768 CMPV.CMPV is a twos complement sign-extended value. This register is double-buffered and writes do not take effect until the end of the current compensation interval. For example: 00000000b - The number of clock cycles in each second is 32768. 00000001b - The number of clock cycles in each second is 32767. 01111110b - The number of clock cycles in each second is 32642. 01111111b - The number of clock cycles in each second is 32641. 11111111b - The number of clock cycles in each second is 32769. 10000000b - The number of clock cycles in each second is 32896. 10000001b - The number of clock cycles in each second is 32895.”

24.2.1.3 RTC ALM Register

RTC Alarm Register

Offset: ch

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ALM															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ALM															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24.4: RTC ALM Register Description

Field	Function
31 - 0 ALM	**Alarm Value** If RTC is enabled, when ALM equals the SEC[SEC] and the SEC[SEC] increments, INTF[ALMIF] will set.

Yuntu Microelectronics

24.2.1.4 RTC EN Register

RTC Enable Register

Offset: 10h

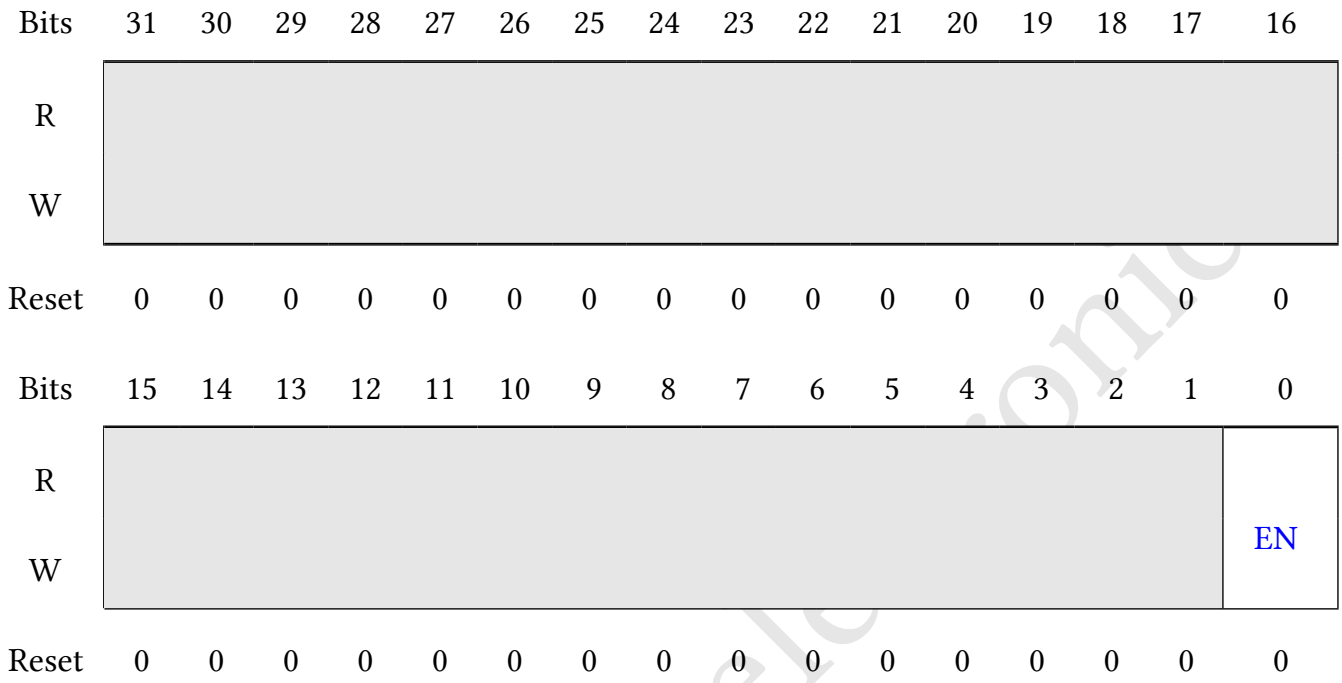


Table 24.5: RTC EN Register Description

Field	Function
0 EN	**RTC Module Enable** When RTC is disabled, SEC is writable. When RTC is enabled, SEC is not writeable. 0b - RTC disable 1b - RTC enable

24.2.1.5 RTC CTRL Register

RTC Control Register

Offset: 14h

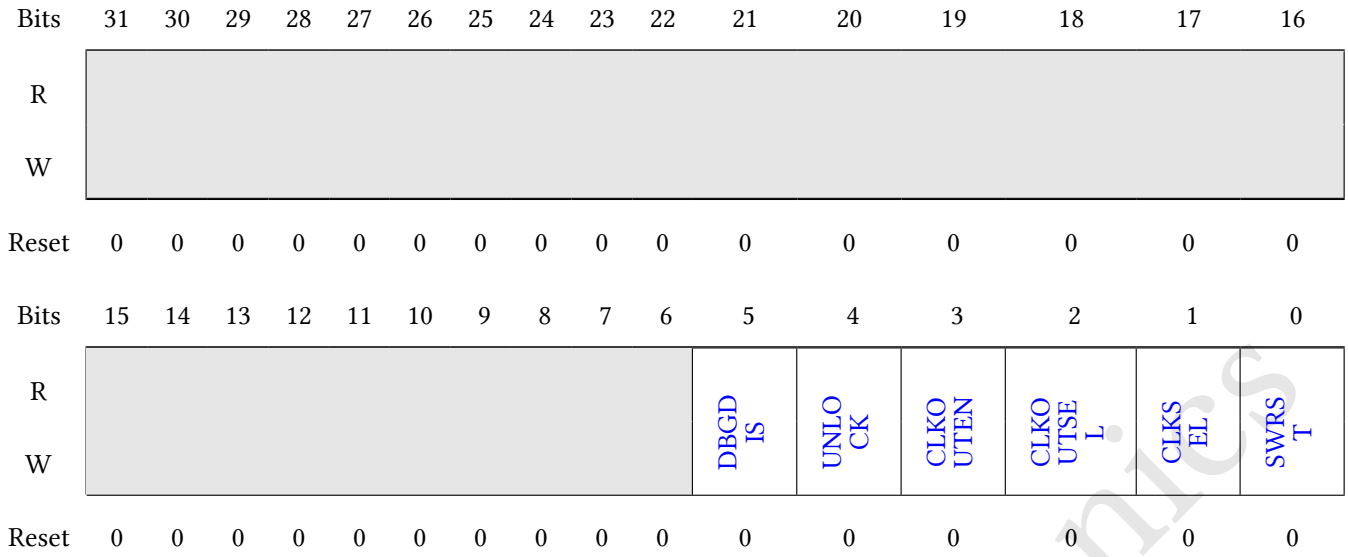


Table 24.6: RTC CTRL Register Description

Field	Function
5 DBGDIS	**Debug Mode Disable** Default 0, run at debug mode. 0b - RTC keeps running in debug mode 1b - RTC stops in debug mode
4 UNLOCK	**Unlock Mode** Unlock register access bit, allows RTC_EN register to be written 0b - Don't allow RTC_EN register to be written 1b - Allow RTC_EN register to be written
3 CLKOUTEN	**RTC Clock Output Enable** Enable the RTC output clock, when CLKOUTEN is not set, rtc output clock is zero. 0b - RTC clock output disable 1b - RTC clock output enable
2 CLKOUTSEL	**RTC Clock Output Select** 0b - RTC clock output is selected by configuration of CTRL[CLKSEL], if CTRL[CLKSEL] is 0, clock output is 32.768KHz, if CTRL[CLKSEL] is 1, clock output is RTC_CLKIN 1b - RTC clock output source is 32.768KHz, and actual output clock is determined by INTE[SECOCS], please see INTE[SECOCS]
1 CLKSEL	**RTC Clock Select** 0b - RTC clock source is SXOSC(32.768KHz) 1b - RTC clock source is RTC_CLKIN pad
0	**Software Reset**

Yuntu Microelectronics

Field	Function
SWRST	0b - No effect 1b - Resets all RTC registers except SWRST which only cleared by POR and by software

24.2.1.6 RTC INTF Register

RTC Interrupt Flag Register

Offset: 18h

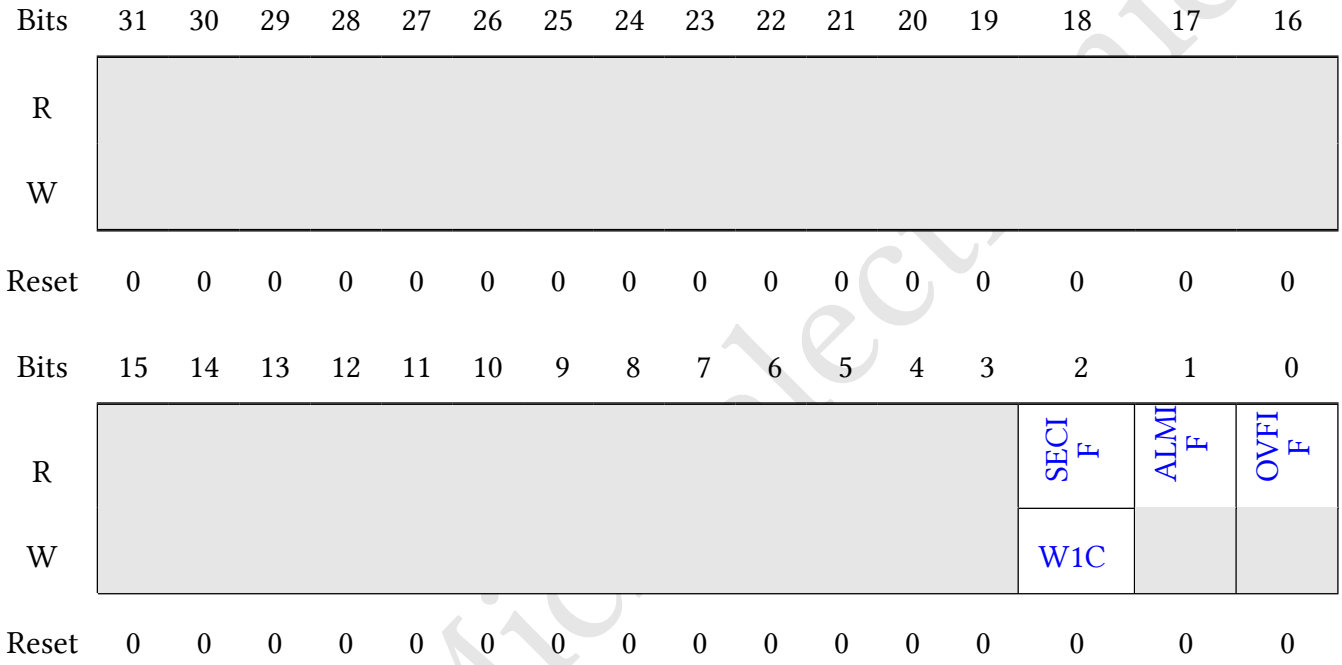


Table 24.7: RTC INTF Register Description

Field	Function
2 SECIF	**Second Interrupt Flag** RTC second interrupt flag. Write 1 and then write 0 to clear this flag. 0b - No second interrupt is pending 1b - The Second interrupt is pending
1 ALMIF	**Alarm Interrupt Flag.** RTC alarm interrupt flag. Write ALM register to clear this flag. 0b - No alarm interrupt is pending 1b - Alarm interrupt is pending
0 OVFIF	**Overflow Interrupt Flag** RTC overflow interrupt flag. Write 1 and then write 0 to clear this flag. 0b - No overflow interrupt is pending 1b - Overflow interrupt is pending

Yuntu Microelectronics

24.2.1.7 RTC INTE Register

RTC Interrupt Enable Register

Offset: 1ch

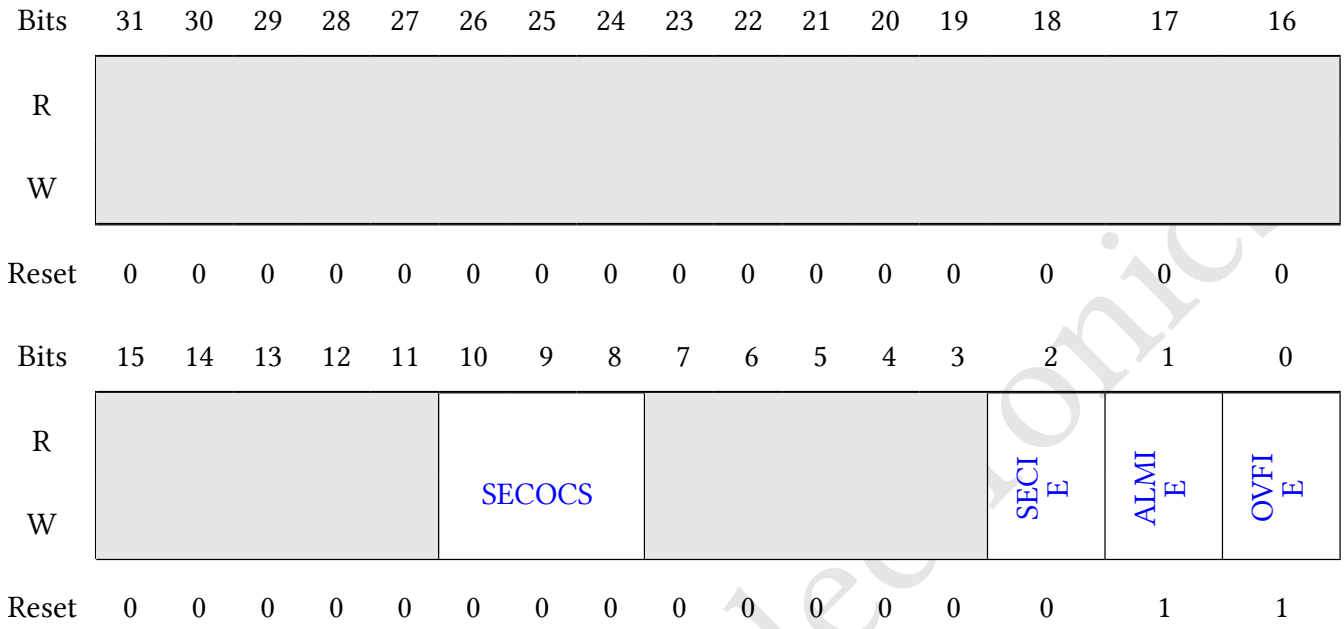


Table 24.8: RTC INTE Register Description

Field	Function
10 - 8 SECOCS	**Second Output Clock Select** RTC Second output clock select 000b - Second output at 1Hz 001b - Second output at 2Hz 010b - Second output at 4Hz 011b - Second output at 8Hz 100b - Second output at 16Hz 101b - Second output at 32Hz 110b - Second output at 64Hz 111b - Second output at 128Hz
2 SECIE	**Second Interrupt Enable** RTC second interrupt enable bit 0b - Second interrupt disable 1b - Second interrupt enable
1 ALMIE	**Alarm Interrupt Enable** 0b - Alarm interrupt disable 1b - Alarm interrupt enable
0	**Overflow Interrupt Enable**

Yuntu Microelectronics

Field	Function
OVFIE	0b - Overflow interrupt disable 1b - Overflow interrupt enable

24.3 Functional Description

24.3.1 General

This section gives detailed information on the internal operation of the module. The RTC is active in all low power modes. The time counter within the RTC is clocked by default from a 32.768KHz clock.

24.3.1.1 Software Reset

Writing 1 to CTRL[SWRST] forces the equivalent of a reset of the RTC module. CTRL[SWRST] is not affected by the software reset and must be cleared by software.

24.3.1.2 Time Counter

The RTC module contains a 32-bit second counter that increments one every second and a 16-bit prescaler register that increments once every 32.768KHz clock. If the SEC register overflow the the INTF[OVFIF] will set and the prescaler register will stop incrementing.

24.3.1.3 Compensation

The CMP register provide a way to accurate correct the time errors. The compensation factor is provided and calculated before write to CMP register. The RTC itself does not have the function to calculate the value of compensation factors.

The compensation factor is calculated with information from 32.768KHz crystal by firmware for operation situations like temperature and voltage, etc.

The compensation logic alters the number of 32.768KHz clock cycles it takes for the prescaler register to overflow and increment the time seconds counter. The time compensation value is used to adjust the number of clock cycles between -127 and +128. Cycles are added or subtracted from the prescaler register when the prescaler register equals 0x3FFF and then increments. The compensation interval is used to adjust the frequency at which the time compensation value is used, that is, from once a second to once every 256 seconds.

Updates to the time compensation register will not take effect until the next time the time seconds register increments and provided the previous compensation interval has expired. When the compensation interval is set to other than once a second then the compensation is applied in the first second interval and the remaining second intervals receive no compensation.

Compensation is disabled by configuring the time compensation register to zero.

24.3.1.4 Register Lock

When CTRL[UNLOCK] bit is cleared, Enable Register cannot be written.

Yuntu Microelectronics

24.3.2 Interrupts

RTC module supports three interrupt. Please refer to the table of interrupt vectors. RTC interrupts is asserted whenever a status flag and the corresponding interrupt enable bit are both set. The RTC interrupt can be used to wakeup the chip from any low-power mode.

If RTC is enabled, when ALMR[ALM] equals the RTCSEC[SEC] and the RTCSEC[SEC] increments, RTCS[ALMIF] will set and Alarm interrupt is generated. When the RTC is enabled and overflows, Overflow interrupt is generated.

If RTC is enabled, when INTE[SECIE] is set,Second interrupt is generated once a second.The frequency of the second Interrupt defaults to 1 Hz, but can be configured by INTE[SECIE].

24.3.3 Operation Modes

RTC is implemented in always-on domain, it keeps working if 32.768kHZ crystal oscillator is enabled even in Deepsleep and Standby modes.

If RTC is required to keep working, it needs to configure RTC and clock sources before entering low power modes.

Yuntu Microelectronics

Part VIII

Analog

Yuntu Microelectronics

Chapter 25

Analog-to-Digital Converter (ADC)

25.1 Chip-specific ADC Information

25.1.1 Instantiation Information

YTM32B1LE0x contains one 12-bit ADC. The 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 17 channels allowing it to measure analog signals. ADC module can operate on an asynchronous clock that is independent from the register read/write access clock and it can be configured as needed to produce asynchronous interrupts. The results of the ADC will store in a 32bit * 16 depth FIFO.

25.1.2 Signal Description

Table 25.1: ADC Signal Table

Signal Name	Description	Direction
ADC_SE0	The analog channel 0 for external signal	I
ADC_SE1	The analog channel 1 for external signal	I
ADC_SE2	The analog channel 2 for external signal	I
ADC_SE3	The analog channel 3 for external signal	I
ADC_SE4	The analog channel 4 for external signal	I
ADC_SE5	The analog channel 5 for external signal	I
ADC_SE6	The analog channel 6 for external signal	I
ADC_SE7	The analog channel 7 for external signal	I
ADC_SE8	The analog channel 8 for external signal	I
ADC_SE9	The analog channel 9 for external signal	I
ADC_SE10	The analog channel 10 for external signal	I
ADC_SE11	The analog channel 11 for external signal	I
ADC_SE12	The analog channel 12 for external signal	I
ADC_SE13	The analog channel 13 for external signal	I
ADC_SE14	The analog channel 14 for external signal	I
ADC_SE15	The analog channel 15 for external signal	I
Temp Sensor	The analog channel 16 for internal temp sensor	I

Yuntu Microelectronics

25.1.3 Clock Description

Table 25.2: ADC Clock Table

Clock Name	Description
Bus Clock	Bus clock used for registers access
Function Clock	Function clock used for function logic

25.1.4 DMA Description

Table 25.3: ADC DMA Table

No.	Request Source	Description
42	The amount of data in FIFO exceeds the threshold CFG0[WM]	Generate DMA request to read DATA and avoid DATA overflow when FIFO is full

25.1.5 Interrupt Description

Table 25.4: ADC Interrupts

Interrupt event	Event flag	Enable Control Bit
ADC ready	STS[ADRDY]	INTE[ADRDYIE]
End of conversion	STS[EOC]	INTE[EOCIE]
End of sequence of conversions	STS[EOSEQ]	INTE[EOSEQIE]
Watchdog status bit is set	STS[WD]	INTE[WDIE]
End of sampling phase	STS[EOSMP]	INTE[EOSMPIE]
Overrun	STS[OVR]	INTE[OVRIE]

25.1.6 Power Mode Description

Table 25.5: ADC Power Mode Table

Power Mode	Description
Active	Full function
Sleep	Keeps working when function clock available
Deepsleep	Keeps working when function clock available

25.2 Introduction

25.2.1 Overview

The 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 17 channels allowing it to measure analog signals. A/D conversion of the various channels can be performed in single, continuous or discontinuous mode. The results of the ADC are stored in a 32bit x 16 depth FIFO, and the data format is right-aligned.

The watchdog feature allows the application to detect if the input voltage goes inside or outside the

Yuntu Microelectronics

user-defined thresholds.

An efficient low power mode is implemented to allow very low consumption at low frequency.

25.2.2 Features

The features of the Analog-to-Digital Converter (ADC) module are listed below:

- Support 12-bit, 10-bit, 8-bit and 6-bit single-ended configurable resolution
- Up to 1Msps for 12-bit resolution conversion performance
- Support DMA and conversion result FIFO with watermark
- Support up to 17 input channels
  - 16 channels to measure external analog signals from pad
  - 1 channel to measure internal temperature sensor
- Support multiple conversion modes
  - Single mode
  - Continuous mode
  - Discontinuous mode
- Support software/hardware trigger for ADC start conversion
- Support two low power modes
  - Wait mode: prevent ADC overrun when FIFO is full
  - Auto off mode: automatic control ADC power off
- Support watchdog for conversion result monitoring
- Support interrupt generate
  - Ready for conversion
  - End of sampling
  - End of conversion
  - End of sequence conversion
  - Overrun event
  - Watchdog event
- Support work and wake up when the chip under low power mode

25.2.3 Block Diagram

The following figure shows the ADC block diagram.

Yuntu Microelectronics

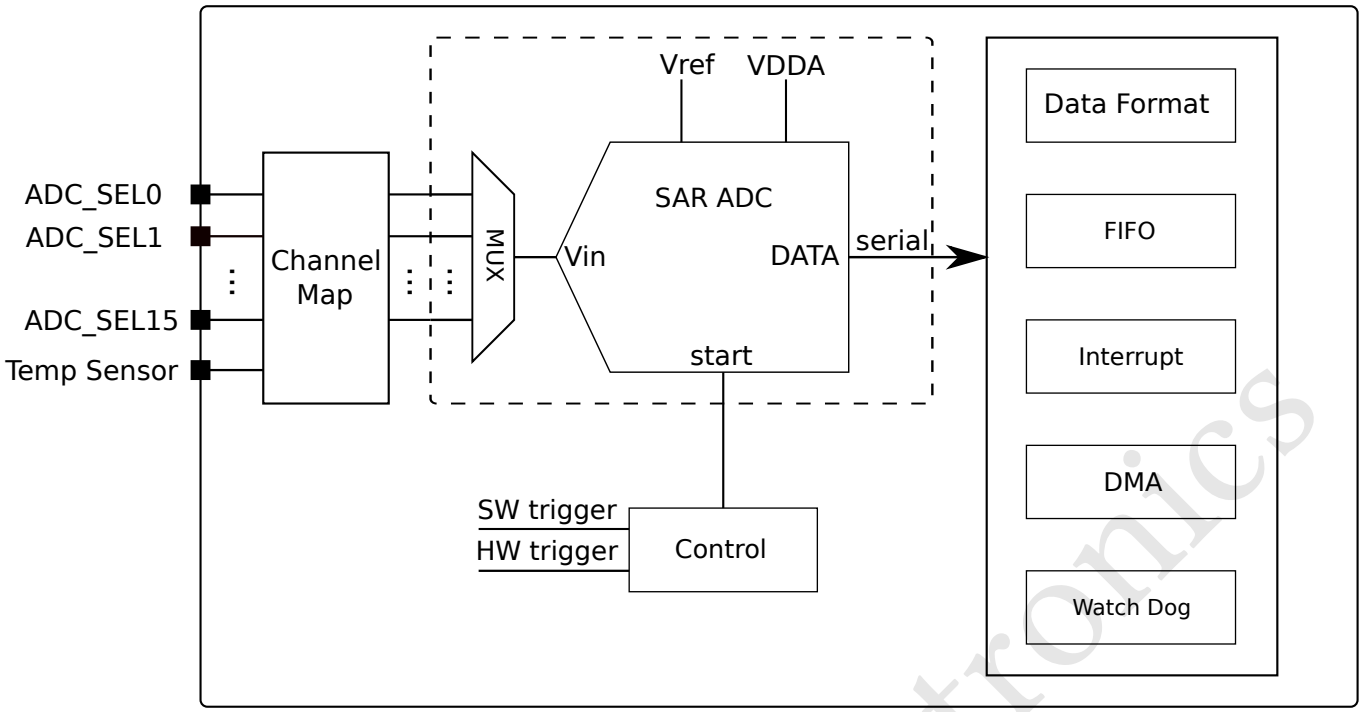


Figure 25.1: Block Diagram of ADC

25.3 Register Definition

The ADC registers are used to control ADC module and display the current working status. In addition, registers are used to configure functional modes, such as single, continuous, discontinuous modes.

Reading reserved registers is always 0, the write operation is not valid.

Register_address = Base_address + Address_offset.

25.3.1 ADC Memory Map

ADC0 base address 4003b000h

Table 25.6: ADC Memory Map

Offset	Register	Width	Access	Reset Value
0h	Status Register (STS)	32	RW	00000040h
4h	Interrupt Enable Register (INTE)	32	RW	00000000h
8h	Control Register (CTRL)	32	RW	00000000h
ch	Configuration 0 Register (CFG0)	32	RW	07000008h
10h	Configuration 1 Register (CFG1)	32	RW	00000220h
14h	Sampling Register (SMP)	32	RW	00000002h
18h	Watchdog Control Register (WDCTRL)	32	RW	00000000h
1ch	Watchdog Threshold Register (WDTH)	32	RW	1FFF0000h
2ch	Channel Configure 0 Register (CHSEL0)	8	RW	00h
2dh	Channel Configure 0 Register (CHSEL1)	8	RW	00h
2eh	Channel Configure 0 Register (CHSEL2)	8	RW	00h
2fh	Channel Configure 0 Register (CHSEL3)	8	RW	00h

Yuntu Microelectronics

Offset	Register	Width	Access	Reset Value
30h	Channel Configure 0 Register (CHSEL4)	8	RW	00h
31h	Channel Configure 0 Register (CHSEL5)	8	RW	00h
32h	Channel Configure 0 Register (CHSEL6)	8	RW	00h
33h	Channel Configure 0 Register (CHSEL7)	8	RW	00h
4ch	FIFO Access Register (FIFO)	32	RW	see description

25.3.1.1 ADC STS Register

Offset: 0h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									WD	EMPTY	FULL	OVR	EOSEQ	EOC	EOSMP	ADRDY
W									W1C			W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Table 25.7: ADC STS Register Description

Field	Function
7 WD	**Watchdog Flag** This bit is set by hardware when the converted voltage crosses the values defined in the WDTH register. It is cleared by software writing 1 to it. 0b - No watchdog event occurred (or the flag event was already acknowledged and cleared by software). 1b - Watchdog event occurred.
6 EMPTY	**FIFO Empty Status** This bit is set by hardware when the FIFO is empty. It is cleared by hardware when FIFO is not empty. 0b - FIFO is not empty. 1b - FIFO is empty.
5	**FIFO Full Status**

Yuntu Microelectronics

Field	Function
FULL	This bit is set by hardware when the FIFO is full. It is cleared by hardware when FIFO is not full. 0b - FIFO is not full. 1b - FIFO is full.
4 OVR	**ADC Overrun** This bit is set by hardware when an overrun occurs, meaning that a new conversion has completed while the FULL flag was already set. It is cleared by software writing 1 to it. 0b - No overrun occurred (or the flag event was already acknowledged and cleared by software). 1b - Overrun has occurred.
3 EOSEQ	**End of Sequence Flag** This bit is set by hardware at the end of the conversion of a sequence. It is cleared by software writing 1 to it. 0b - Conversion sequence not complete (or the flag event was already acknowledged and cleared by software). 1b - Conversion sequence complete.
2 EOC	**End of Conversion Flag** This bit is set by hardware at the end of each conversion of a channel when a new data result is available in the FIFO register. It is cleared by software writing 1 to it or by reading the FIFO register. 0b - Channel conversion not complete (or the flag event was already acknowledged and cleared by software). 1b - Channel conversion complete.
1 EOSMP	**End of Sampling Flag** This bit is set by hardware during the conversion at the end of the sampling phase. 0b - Not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software). 1b - End of sampling phase reached.
0 ADRDY	**ADC Ready Flag** 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. 0b - ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software). 1b - ADC is ready to start conversion.

25.3.1.2 ADC INTE Register

Offset: 4h

Yuntu Microelectronics

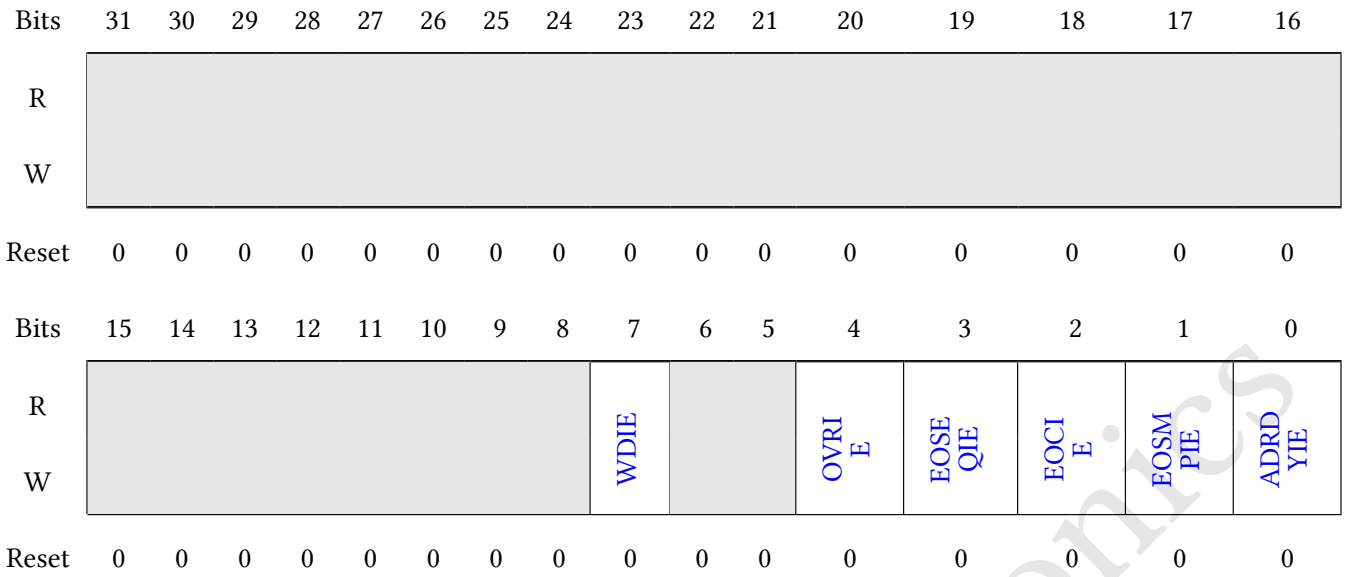


Table 25.8: ADC INTE Register Description

Field	Function
7 WDIE	**Watchdog Interrupt Enable** This bit is set and cleared by software to enable/disable the analog watchdog interrupt. 0b - Watchdog interrupt disabled. 1b - Watchdog interrupt enabled. An interrupt is generated when the WD bit is set.
4 OVRIE	**Overrun Interrupt Enable** This bit is set and cleared by software to enable/disable the overrun interrupt. 0b - Overrun interrupt disabled. 1b - Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.
3 EOSEQIE	**End of Conversion Sequence Interrupt Enable** This bit is set and cleared by software to enable/disable the end of sequence of conversions interrupt. 0b - EOSEQ interrupt disabled. 1b - EOSEQ interrupt enabled. An interrupt is generated when the EOSEQ bit is set.
2 EOCIE	**End of Conversion Interrupt Enable** This bit is set and cleared by software to enable/disable the end of conversion interrupt. 0b - EOC interrupt disabled. 1b - EOC interrupt enabled. An interrupt is generated when the EOC bit is set.
1 EOSMPIE	**End of Sampling Flag Interrupt Enable** This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt. 0b - EOSMP interrupt disabled. 1b - EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.

Yuntu Microelectronics

Field	Function
0 ADRDYIE	**ADC Ready Interrupt Enable** This bit is set and cleared by software to enable/disable the ADC Ready interrupt. 0b - ADRDY interrupt disabled. 1b - ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.

25.3.1.3 ADC CTRL Register

Offset: 8h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R												TSEN	ADSTOP	ADSTART	ADDIS	ADEN
W													rs	rs	rs	rs
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25.9: ADC CTRL Register Description

Field	Function
4 TSEN	**Tempsenser Enable** This bit is to control external tempsenser enable/disable. 0b - Tempsenser is disabled. 1b - Tempsenser is enabled.
3 ADSTOP	**ADC Stop Conversion Command** This bit is set by software to stop and discard an ongoing conversion (ADSTOP Command). It is cleared by hardware when the conversion is effectively discarded and the ADC is ready to accept a new start conversion command. Software is allowed to set ADSTOP only when ADSTART=1 and ADDIS=0 (ADC is enabled and may be converting and there is no pending request to disable the ADC). 0b - No ADC stop conversion command ongoing. 1b - Write 1 to stop the ADC. Read 1 means that an ADSTOP command is in progress.

Yuntu Microelectronics

Field	Function
2 ADSTART	**ADC Start Conversion Command** This bit is set by software to start ADC conversion. Depending on the TRIGMD configuration bits, conversion either starts immediately (software trigger configuration) or once a hardware trigger event occurs (hardware trigger configuration). 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). 0b - No ADC conversion is ongoing. 1b - Write 1 to start the ADC. Read 1 means that the ADC is operating and may be converting.
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). 0b - No ADDIS command ongoing. 1b - Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.
0 ADEN	**ADC Enable Command** This bit is set by software to enable the ADC. The ADC will be effectively ready to operate once the ADRDY flag has been set. It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command. 0b - ADC is disabled (OFF state). 1b - Write 1 to enable the ADC. Read 1 means that tha ADC is enabled.

25.3.1.4 ADC CFG0 Register

Offset: ch

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W		OVRMD				SEQLEN					AUTOOFF	WAIT			CONVMD	
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W					TRIGMD		RES		WM							DMAEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Yuntu Microelectronics

Table 25.10: ADC CFG0 Register Description

Field	Function
30 OVRMD	**Overrun Management Mode** This bit is set and cleared by software and configure the way data overruns. 0b - FIFO register is preserved with the old data when an overrun is detected. 1b - The last data is overwritten with the conversion result when an overrun is detected.
26 - 24 SEQLEN	**Sequence Length** These bits define the length of sequence. The sequence length is SEQLEN+1. For example, SEQLEN=7 means sequence length is 8; SEQLEN=0 means sequence length is 1.
21 AUTOOFF	**Auto-off Mode** This bit is set and cleared by software to enable/disable auto-off mode. 0b - Auto-off mode disabled. 1b - Auto-off mode enabled.
20 WAIT	**Wait Mode** This bit is set and cleared by software to enable/disable wait mode. 0b - Wait mode off. 1b - Wait mode on.
17 - 16 CONVMD	**Conversion Mode Select** These bits are written by software to select the mode of the conversion. 00b - Single conversion mode. 01b - Continuous conversion mode. 10b - Discontinuous conversion mode. 11b - Reserved.
11 TRIGMD	**Trigger Mode Select** The bit is used to select software trigger mode or external hardware trigger. See ”ADC Trigger Polarity”. 0b - Select software trigger as the trigger source. 1b - Select hardware trigger as the trigger source.
9 - 8 RES	**Data Resolution** These bits are written by software to select the resolution of the conversion. 00b - 12-bit resolution. 01b - 10-bit resolution. 10b - 8-bit resolution. 11b - 6-bit resolution.
7 - 4 WM	**Conversion Data Water Mark** When conversion data number is more than WM value and DMA enabled, trigger a DMA request.
0	**Direct Memory Access Enable**

Yuntu Microelectronics

Field	Function
DMAEN	This bit is set and cleared by software to enable the generation of DMA requests. This allows to use the DMA controller to manage conversion data automatically. 0b - DMA disable. 1b - DMA enable.

25.3.1.5 ADC CFG1 Register

Offset: 10h

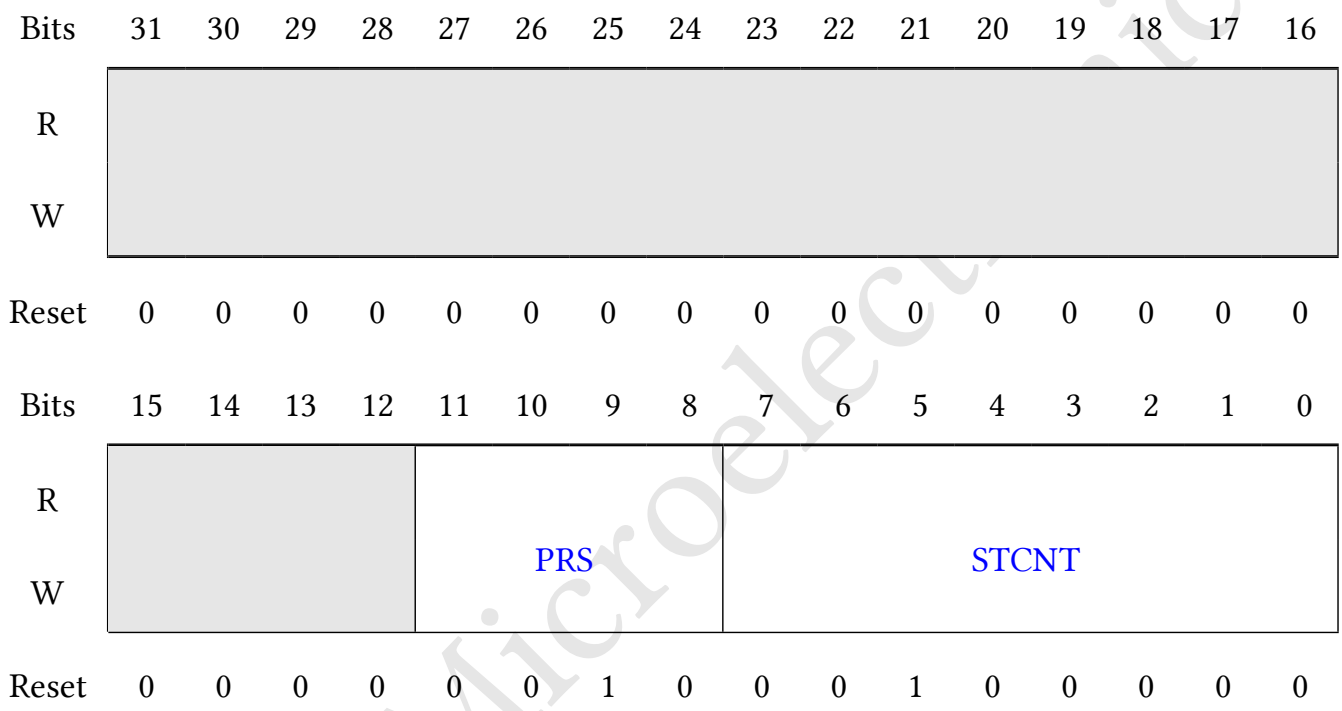


Table 25.11: ADC CFG1 Register Description

Field	Function
11 - 8 PRS	**Function Clock Prescaler** These bits are used to generate the function clock F_{func_clk}. The prescaler value is n+1, so the range of prescaler value is 1 16.
7 - 0 STCNT	**ADC Startup Counter** The ADC needs a stabilization time of $t_{STAB}(2\mu s)$ before it starts converting accurately. This time is calculated by counting clock cycles(which is divided from function clock) until the internal counter reaches the STCNT. So user should set these bits before ADC enable. For example, if the $F_{func_clk} = 16MHz$, then should set the $STCNT = 2000/(1000/16) = 32$.

Yuntu Microelectronics

25.3.1.6 ADC SMP Register

Offset: 14h

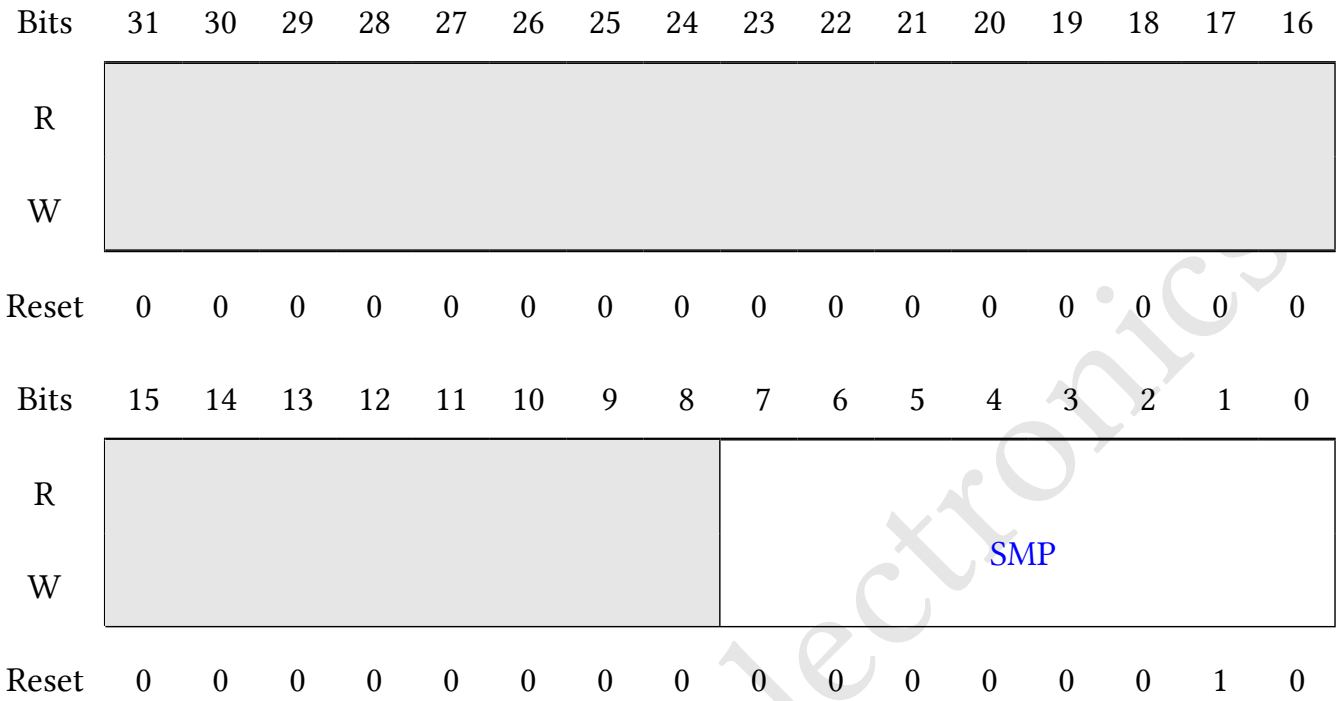


Table 25.12: ADC SMP Register Description

Field	Function
7 - 0 SMP	**Sampling Time Selection** These bits are written by software to select the sampling time that applies to all channels. The sample time is calculated as $(SMP + 2) * T_{func_clk}$.

Yuntu Microelectronics

25.3.1.7 ADC WDCTRL Register

Offset: 18h

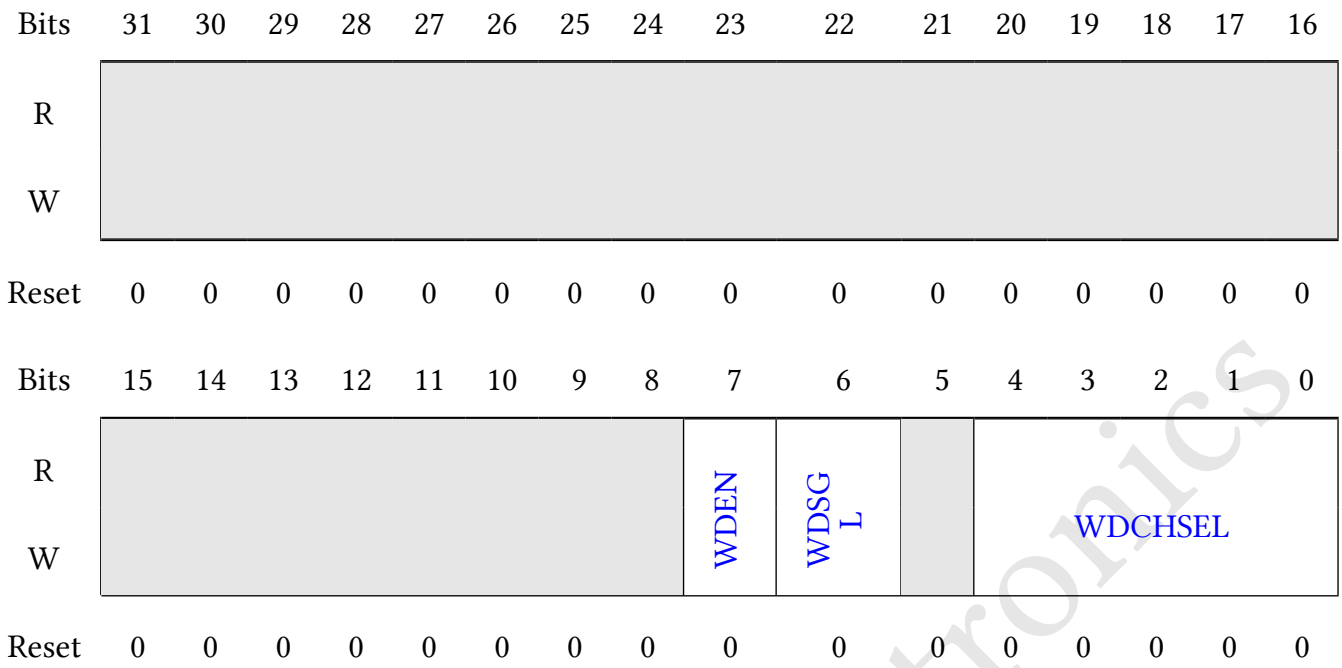


Table 25.13: ADC WDCTRL Register Description

Field	Function
7 WDEN	**Watchdog Enable** This bit is set and cleared by software to enable/disable watchdog. 0b - Watchdog is disabled. 1b - Watchdog is enabled.
6 WDSGL	**Apply the Watchdog on a Single Channel** This bit is set and cleared by software to apply the watchdog on the channel identified by the WDCHSEL or all channels. 0b - Watchdog enabled on a single channel. 1b - Watchdog enabled on all channels.
4 - 0 WDCHSEL	**Watchdog Channel Select** These bits are set and cleared by software. They select the input channel to be guarded by the watchdog.

Yuntu Microelectronics

25.3.1.8 ADC WDTH Register

Offset: 1ch

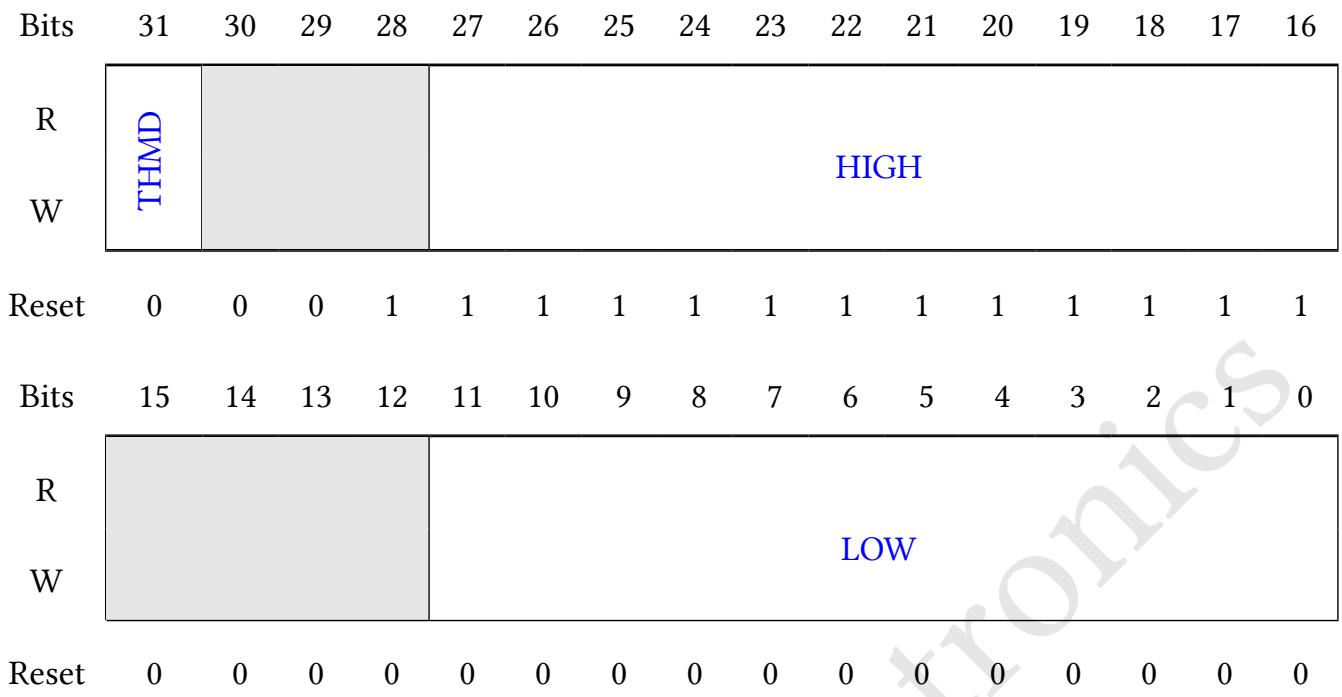


Table 25.14: ADC WDTH Register Description

Field	Function
31 THMD	**Watchdog Threshold Effective Mode** This bit is set and cleared by software to select the watchdog threshold effective mode. 0b - Watchdog output validly when the converted voltage is outside the threshold defined by HIGH and LOW. 1b - Watchdog output validly when the converted voltage is inside the threshold defined by HIGH and LOW.
27 - 16 HIGH	**Watchdog Higher Threshold** These bits are written by software to define the higher threshold for the watchdog. The watchdog threshold scope is from LOW to HIGH.
11 - 0 LOW	**Watchdog Lower Threshold** These bits are written by software to define the lower threshold for the watchdog. The watchdog threshold scope is from LOW to HIGH.

Yuntu Microelectronics

25.3.1.9 ADC CHSEL Register

Offset: 2ch + n * 1h

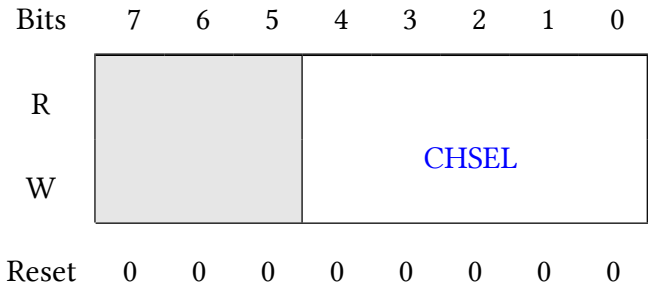


Table 25.15: ADC CHSEL Register Description

Field	Function
4 - 0 CHSEL	**Conversion Channel Select** These bits are configured by software to select the conversion channel.

25.3.1.10 ADC FIFO Register

Offset: 4ch

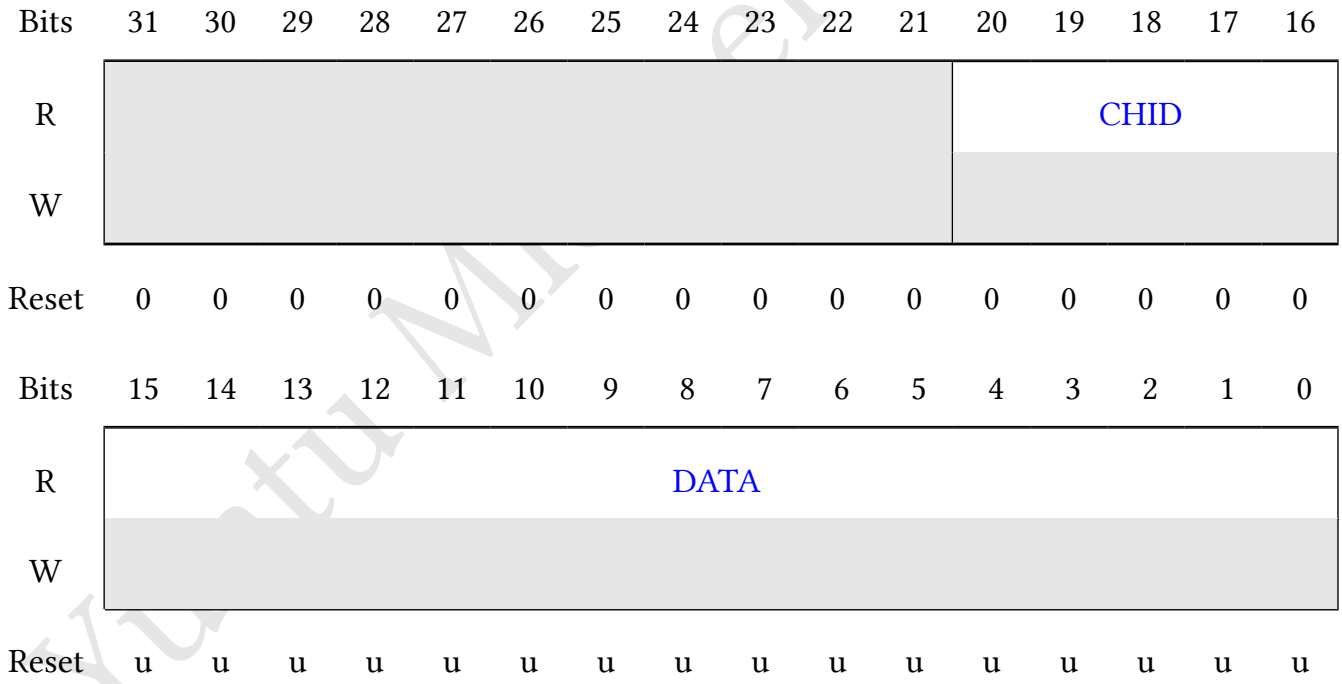


Table 25.16: ADC FIFO Register Description

Field	Function
20 - 16 CHID	**Channel ID** These bits indicate the channel identification that corresponds with the conversion data.

Yuntu Microelectronics

Field	Function
15 - 0 DATA	**Converted Data** Converted data

25.4 Functional Description

ADC module Contains dual clock-domain architecture: bus clock and functional clock. It supports multiple conversion mode and asynchronous interrupt generation. The conversion results are stored in the FIFO. In addition, ADC module supports analog window watchdog.

25.4.1 Basic Configuration

25.4.1.1 ADC On-off Control

After MCU power-up, the ADC is disabled and under the 'Off' state (power-down mode). As shown in Figure "ADC Convert Flow", the ADC needs a stabilization time $t_{startup}(\sim 2\mu s)$ before it starts converting accurately.

Two control bits CTRL[ADEN] and CTRL[ADDIS] are used to enable or disable the ADC:

- Set CTRL[ADEN]=1 to enable the ADC. ADC will enter the 'Start up' state from the 'Off' state, after a stabilization time $t_{startup}$ ADC will enter the 'Ready' state and wait for a conversion to start. The STS[ADRDY] flag is set as soon as the ADC is ready for operation.
- Set CTRL[ADDIS]=1 to disable the ADC. ADC will switch to the 'Off' state. The CTRL[ADEN] and CTRL[ADDIS] bits are then automatically cleared by hardware as soon as the ADC is fully disabled.

Two control bits CTRL[ADSTART] and CTRL[ADSTOP] are used to start or stop the conversion:

- When ADC is under the 'Ready' state, if software trigger is selected (CFG0[TRIGMD]=0), set CTRL[ADSTART]=1 will start conversion immediately. If hardware trigger is selected (CFG0[TRIGMD]=1), conversion doesn't start until getting an external trigger after setting CTRL[ADSTART]=1. The CTRL[ADSTART] bit will be cleared by hardware automatically after a sequence conversion finish or the ADC is fully disabled.
- When a conversion is ongoing, set CTRL[ADSTOP]=1 to stop the conversion, then ADC enters the 'Ready' state again. The CTRL[ADSTOP] bit is cleared by hardware automatically after conversion stops or the ADC is fully disabled.

Follow this procedure to enable the ADC:

- Set CTRL[ADEN]=1.
- Wait until STS[ADRDY]=1 (ADRDY is set after the ADC startup time $t_{startup}$). This can be handled by interrupt if the ADC ready interrupt is enabled by setting the INTE[ADRDYIE] bit.

Follow this procedure to disable the ADC:

- Check that CTRL[ADSTART]=0 to ensure that no conversion is ongoing. If required, stop any ongoing conversion by writing 1 to the CTRL[ADSTOP] bit and waiting until this bit reads 0.
- Set CTRL[ADDIS]=1.
- If required by the application, wait until CTRL[ADEN]=0, indicating that the ADC is fully disabled (CTRL[ADDIS] is automatically reset once ADEN=0).

Yuntu Microelectronics

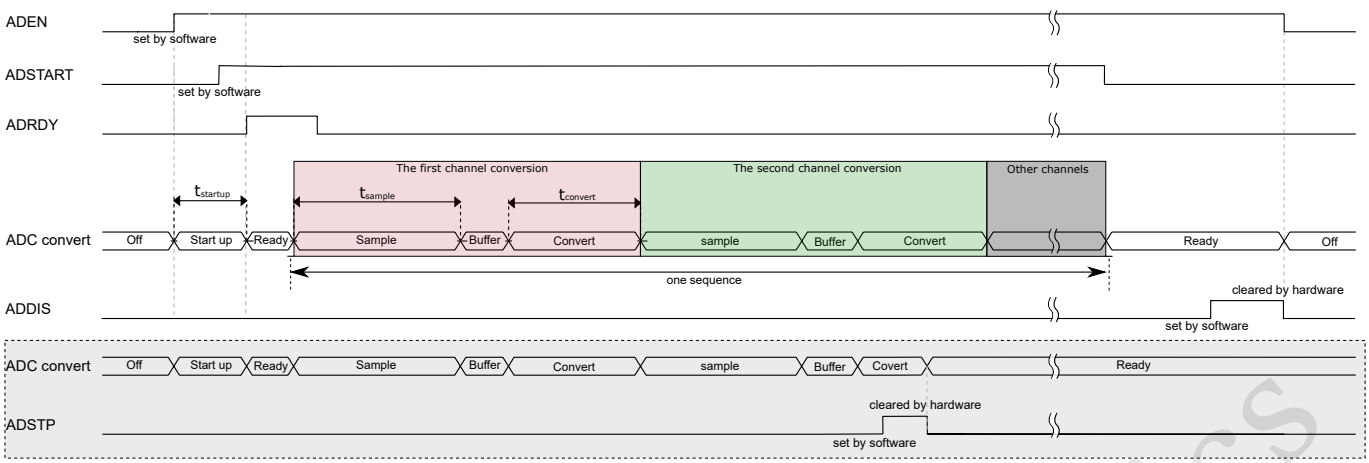


Figure 25.2: ADC Convert Flow

NOTE: In auto-off mode (CFG0[AUTOOFF]=1), the power-on/off phases are performed by hardware automatically, and the ADRDY flag won't set.

25.4.1.2 ADC Clock

The ADC has a dual clock-domain architecture, as show in Figure "ADC Clock Domain", it has the advantage that to reaching the maximum ADC clock frequency whatever the bus clock scheme selected.

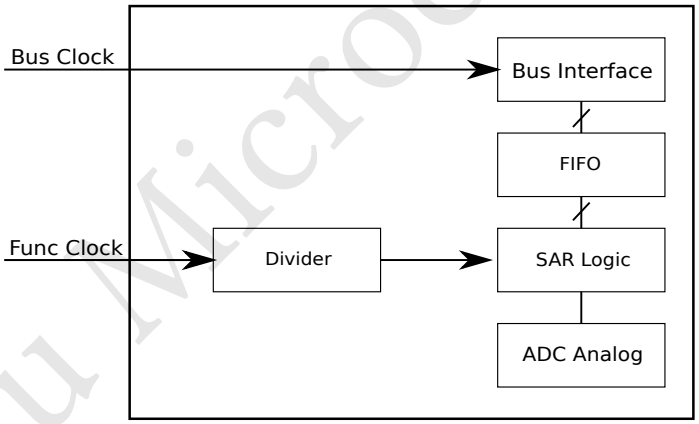


Figure 25.3: ADC Clock Domain

25.4.1.3 Channel Selection

There are up to 16 multiplexed channels from pins (ADC_SE0, ADC_SE1...ADC_SE15) and 1 channel from internal temperature sensor.

It is possible to convert a single channel or multiple channels automatically.

If need to convert multiple channels, they will be converted in sequence, organized as CHSEL0, then CHSEL1, ..., then CHSEL7. The sequence length is configured in CFG0[SEQLEN]. For example, if sequence length is set as 3, then the sequence is organized as CHSEL0, then CHSEL1, then CHSEL2.

The channel decode is shown as bellow.

Yuntu Microelectronics

Table 25.17: ADC Channel Decode

CHSELn[4:0]	Channel Select
5'b00000	ADC_SE0
5'b00001	ADC_SE1
5'b00010	ADC_SE2
5'b00011	ADC_SE3
5'b00100	ADC_SE4
5'b00101	ADC_SE5
5'b00110	ADC_SE6
5'b00111	ADC_SE7
5'b01000	ADC_SE8
5'b01001	ADC_SE9
5'b01010	ADC_SE10
5'b01011	ADC_SE11
5'b01100	ADC_SE12
5'b01101	ADC_SE13
5'b01110	ADC_SE14
5'b01111	ADC_SE15
5'b10000	Internal temperature sensor
Others	Reserved

25.4.1.4 Configuring the ADC

Software must write to the CTRL[ADEN] bit if the ADC is disabled.

Software must only write to the CTRL[ADSTART] and CTRL[ADDIS] bits only if the ADC is enabled and there is no pending request to disable the ADC (CTRL[ADEN] = 1 and CTRL[ADDIS] = 0).

For all the other control bits in the INTE, CFG0/1, SMP, WDCTRL, WDTH, and CHSEL0-7 registers, software must only write to the configuration control bits if there is no conversion ongoing (CTRL[ADSTART] = 0).

Software must only write to the CTRL[ADSTOP] bit if the ADC is enabled (and possibly converting) and there is no pending request to disable the ADC (CTRL[ADSTART] = 1 and CTRL[ADDIS] = 0) .

NOTE: There is no hardware protection preventing software from making write operations forbidden by the above rules. If such a forbidden write access occurs, the ADC may enter an undefined state. To recover correct operation in this case, the ADC must be disabled(CTRL[ADDIS] = 1).

25.4.1.5 Starting Conversions

Software starts ADC conversions by setting CTRL[ADSTART]=1.When CTRL[ADSTART] is set, the conversion:

- Starts immediately if CFG0[TRIGMD] = 0 (software trigger)
- At the next active edge of the selected hardware trigger if CFG0[TRIGMD] = 1

The CTRL[ADSTART] bit is also used to indicate whether an ADC operation is currently ongoing. It is possible to re-configure the ADC while CTRL[ADSTART]=0, indicating that the ADC is idle.

The CTRL[ADSTART] bit is cleared by hardware:

Yuntu Microelectronics

- In single mode with software trigger
  - At any end of conversion sequence (STS[EOSEQ]=1)
- In discontinuous mode with software trigger
  - At any end of conversion
- In all cases
  - After execution of the CTRL[ADSTOP] procedure invoked by software

NOTE:

- In continuous mode (CFG0[CONVMD]=0x2), the CTRL[ADSTART] bit is not cleared by hardware when the STS[EOSEQ] flag is set because the sequence is automatically relaunched.
- When hardware trigger is selected in single mode, CTRL[ADSTART] is not cleared by hardware when the STS[EOSEQ] flag is set. This avoids the need for software having to set the CTRL[ADSTART] bit again and ensures the next trigger event is not missed.

25.4.1.6 Stopping an Ongoing Conversion

The software can decide to stop any ongoing conversions by setting CTRL[ADSTOP]=1.

This will reset the ADC operation and the ADC will be idle, ready for a new operation.

When the CTRL[ADSTOP] bit is set by software, any ongoing conversion is aborted and the result is discarded (FIFO is not updated with the current conversion). The conversion sequence is also aborted and reset (meaning that restarting the ADC would re-start a new sequence).

Once this procedure is complete, the CTRL[ADSTOP] and CTRL[ADSTART] bits are both cleared by hardware and the software must wait until CTRL[ADSTART]=0 before starting new conversions.

NOTE: The flags in STS register are not cleared by STOP command, and the data in FIFO are not lost.

25.4.2 Timings

The ADC conversion logic uses function clock F_{func_clk} with a divider configured by CFG1[PRS] bits.

$$F_{func_clk_div} = F_{func_clk}/(CFG1[PRS] + 1)$$

The ADC needs a stabilization time $t_{startup}(\sim 2\mu s)$ before it starts sampling and converting accurately. The stabilization time configured by CFG1[STCNT] bits. As shown "ADC Convert Time"

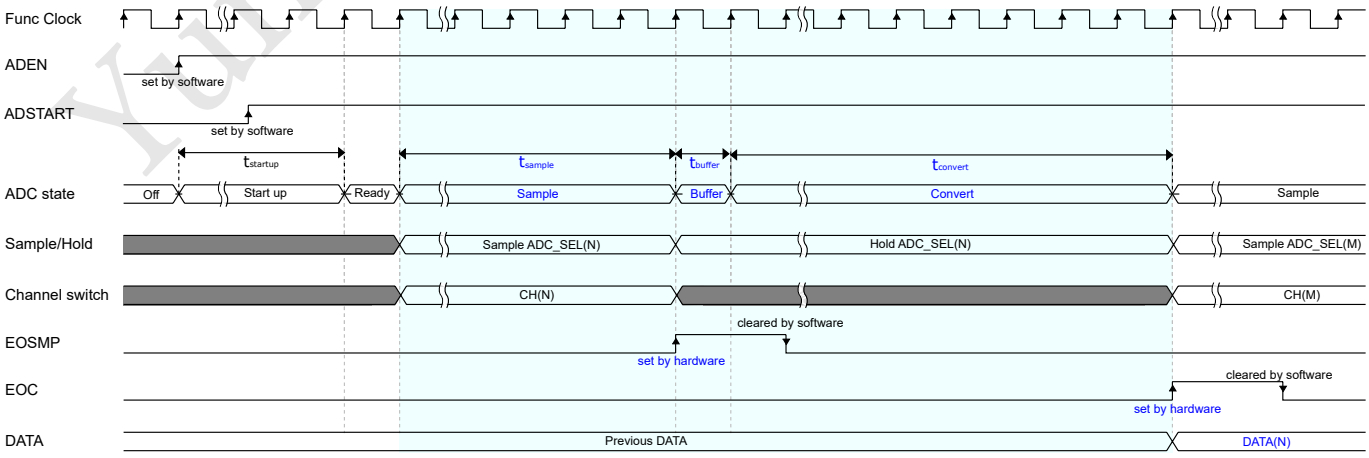


Figure 25.4: ADC Convert Time

Yuntu Microelectronics

$$t_{startup} = CFG1[STCNT]/F_{func_clk_div}$$

Before starting a conversion, the ADC needs to establish a direct connection between the voltage source to be measured and the embedded sampling capacitor of the ADC. This sampling time t_{sample} must be enough for the input voltage source to charge the sample and hold capacitor to the input voltage level. Having a programmable sampling time allows to trim the conversion speed according to the input resistance of the input voltage source. The ADC samples the input voltage for a number of ADC clock cycles that can be modified using the SMP[7:0] bits. This programmable sampling time is common to all channels. The ADC indicates the end of the sampling phase by setting the EOSMP flag. The 'Buffer' state take up a clock cycle and count as part of sample.

$$t_{sample} = (SMP[7:0] + 2)/F_{func_clk_div}$$

After sampling, the ADC starts the conversion and the conversion time varies depending on C_{res} (data resolution configured by CFG0[RES]). For 12-bit resolution, C_{res}=12.

$$t_{convert} = C_{res}/F_{func_clk_div}$$

The ADC conversion rate t_{adc}:

$$t_{adc} = t_{sample} + t_{convert}$$

For example, the ADC conversion rate is 1MHz, $F_{func_clk_div}$ =16MHz, SMP[7:0]=0x2,

CFG0[RES]=0x0

$$t_{adc} = [(2 + 2)/16 + 12/16]\mu s = 1\mu s$$

25.4.3 Functional Mode

25.4.3.1 Single Conversion Mode

In Single conversion mode, the ADC does a conversion of the selected sequence channels. This mode is selected when CFG0[CONVMD]=0. Conversion starts by either:

- Setting the CTRL[ADSTART] bit
- Hardware trigger event

Inside the sequence, after each conversion is complete:

- The converted data is stored into the FIFO
- The STS[EOC] (end of conversion) flag is set
- An interrupt is generated if the INTE[EOCIE] bit is set

After the sequence of conversions is complete:

- The STS[EOSEQ] (end of sequence) flag is set
- An interrupt is generated if the INTE[EOSEQIE] bit is set

Then the ADC stops until a new external trigger event occurs or the CTRL[ADSTART] bit is set again.

NOTE: To convert a single channel once, program the sequence length (CFG0[SEQLEN]) to 0.

Yuntu Microelectronics

25.4.3.2 Continuous Conversion Mode

In continuous conversion mode, when a software or hardware trigger event occurs, the ADC performs a sequence of conversions, after the sequence conversion finish, the ADC will re-start automatically and performs the same sequence of conversions continuously. This mode is selected when CFG0[CONVMD]=1. Conversion is started by either:

- Setting the CTRL[ADSTART] bit
- Hardware trigger event

Inside the sequence, after each conversion is complete:

- The converted data is stored into the FIFO
- The STS[EOC] (end of conversion) flag is set
- An interrupt is generated if the INTE[EOCIE] bit is set

After the sequence of conversions is complete:

- The STS[EOSEQ] (end of sequence) flag is set
- An interrupt is generated if the INE[EOSEQIE] bit is set

Then a new sequence restarts immediately and the ADC repeats the conversion continuously.

25.4.3.3 Discontinuous Mode

In discontinuous conversion mode, a hardware or software trigger event is required to start each conversion defined in the sequence. This mode is selected when CFG0[CONVMD]=2.

Example:

- Channels to be converted = 0, 3, 7, 10
  - 1st trigger: channel 0 is converted and an STS[EOC] event is generated
  - 2nd trigger: channel 3 is converted and an STS[EOC] event is generated
  - 3rd trigger: channel 7 is converted and an STS[EOC] event is generated
  - 4th trigger: channel 10 is converted and both STS[EOC] and STS[EOSEQ] events are generated.
  - 5th trigger: channel 0 is converted an STS[EOC] event is generated
  - 6th trigger: channel 3 is converted and an STS[EOC] event is generated
  - ...

25.4.3.4 Conversion on External Trigger

A conversion or a sequence of conversions can be triggered either by software or by an external event. If the CFG0[TRIGMD] control bit is equal to “1”, then external event is able to trigger a conversion. The trigger selection is effective once software set CTRL[ADSTART]=1.

Any hardware trigger which occurs while a conversion is ongoing will be ignored.

When CTRL[ADSTART]=0, any hardware trigger which occurs will be ignored.

The hardware trigger is only from the TMU module, and users can config the CIM register to select the trigger type. There are three types of triggers to be chosen:

- Raw signal which from the TMU module (CIM->CTRL[ADCTSS]=0)
- Signal from the TMU module and synchronized by bus clock (CIM->CTRL[ADCTSS]=1)
- Signal from the TMU module and synchronized by function clock (CIM->CTRL[ADCTSS]=2)

Yuntu Microelectronics

NOTE: Users should select the prepare trigger type according to the different applications.

25.4.3.5 Programmable Resolution

It is possible to obtain shorter conversion time $t_{convert}$ by reducing the ADC resolution. The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the CFG0[RES] bits. Lower resolution allows faster conversion times for applications where high data precision is not required.

NOTE:

- The CFG0[RES] bits must only be changed when the CTRL[ADEN] bit is “0”.
- The result of the conversion is always right aligned.
- Lower resolution reduces the conversion time needed for the successive approximation steps.

25.4.3.6 End of Sampling/Conversion

The ADC indicates the end of sampling phase by setting the STS[EOSMP] flag. An interrupt can be generated if the INTE[EOSMPIE] bit is set. The STS[EOSMP] flag is cleared by software by writing 1 to it.

The ADC indicates each end of conversion by setting the STS[EOC] flag. The ADC sets the flag as soon as a new conversion data result is available. An interrupt can be generated if the INTE[EOCIE] bit is set. The STS[EOC] flag is cleared by software by either writing 1 to it or reading the FIFO.

25.4.3.7 End of Conversion Sequence

The ADC notifies the application of each end of sequence event by setting the STS[EOSEQ] flag.

The ADC sets the STS[EOSEQ] flag once the last data result of a conversion sequence is available in the FIFO. An interrupt can be generated if the INTE[EOSEQIE] bit is set. The STS[EOSEQ] flag is cleared by software by writing 1 to it.

25.4.4 Data Management

25.4.4.1 Data FIFO

At the end of each conversion (when an STS[EOC] event occurs), the result of the converted data is stored in the FIFO which is 32bit width x 16 depth, the FIFO data is consists of converted results (FIFO[DATA]) and conversion channel identifications (FIFO[CHID]).

The format of the read-out converted data depends on the configured data resolution.

The FIFO supports byte, half-word, and word access, but the address offset should always be 0x4c.

For different data resolutions, see ”ADC DATA Resolution”, users should note:

- If read by word, read once to get data[31:0], the next data is ready after reading data[31:0]
- If read by half word, read twice to get data[31:0], the next data is ready after reading data[15:0]
- If read by byte, read four times to get data[31:0], the next data is ready after reading data[7:0]

Table 25.18: ADC DATA Resolution

RES[1:0]	31	30	...	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			...	chid[4:0]									data[11:0]											
1			...	chid[4:0]											data[9:0]									

Yuntu Microelectronics

Table 25.18 continued from previous page

RES[1:0]	31	30	...	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
2			...	chid[4:0]													data[7:0]							
3			...	chid[4:0]															data[5:0]					

NOTE: If read by half word or by byte. Users should read high significant bytes first and then read low significant bits.

25.4.4.2 ADC Overrun

The overrun flag (STS[OVR]) indicates a data overrun event, when the converted data was not read in time by the CPU or the DMA before the FIFO is full.

The STS[OVR] flag is set if the STS[FULL] flag is still at '1' at the time when a new conversion completes. An interrupt can be generated if the INTE[OVRIE] bit is set.

When an overrun condition occurs, the ADC keeps operating and can continue to convert unless the software decides to stop and reset the sequence by setting the CTRL[ADSTOP] bit.

The STS[OVR] flag is cleared by software by writing 1 to it.

It is possible to configure if the data is preserved or overwritten when an overrun event occurs by programming the CFG0[OVRMD] bit:

- CFG0[OVRMD]=0

An overrun event preserves the data register from being overwritten: the old data is maintained and the new converted data is discarded. If STS[OVR] remains at 1, further conversions can be performed but the resulting data will be discarded. - CFG0[OVRMD]=1

The last data in the FIFO is overwritten with the conversion result. If STS[OVR] remains at 1, further conversions can be performed and the FIFO always contains the data from the latest conversion.

25.4.4.3 Managing a Sequence of Data Converted Without Using the DMA

If the conversions are slow enough, the conversion sequence can be handled by software. In this case the software can use the interrupt that generated by STS[EOC] flag to handle each data result. Each time a conversion is complete, the STS[EOC] bit will be set and the FIFO register can be read.

Software can also use FIFO STS[EMPTY] flag to handle each data result. If STS[EMPTY] is not "0", it means that the FIFO has new data.

The CFG0[OVRMD] bit should be configured to 0 to manage overrun events as an error.

25.4.4.4 Managing Converted Data Without Using the DMA and Overrun

It may be useful to let the ADC convert one or more channels without reading the data after each conversion. In this case, the CFG0[OVRMD] bit must be configured at 1 and the STS[OVR] flag should be ignored by the software. When CFG0[OVRMD]=1, an overrun event does not prevent the ADC from continuing to convert and the FIFO always contains the latest conversion data.

Yuntu Microelectronics

25.4.4.5 Managing Converted Data Using the DMA

Once the data number in the FIFO is not empty and bit CFG0[DMAEN] is set, the ADC will send request to the DMA. This allows the transfer of the converted data from the FIFO to the destination location selected by the software.

Despite this, if an overrun event occurs (STS[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 CFG0[OVRMD] bit, the data is either preserved or overwritten.

The DMA transfer requests are blocked until the software clears the CFG0[OVR] bit.

25.4.5 Low Power Mode

25.4.5.1 Wait Mode

Wait mode conversion can be used to simplify the software as well as optimizing the performance of applications clocked at low frequency where there might be a risk of ADC overrun occurring. When the CFG0[WAIT] bit is set to 1, a new conversion can start only if the FIFO is not full.

This is a way to automatically adapt the ADC speed with the system speed that reads the data.

NOTE: Any hardware trigger which occurs while a conversion is ongoing will be ignored.

25.4.5.2 Auto-off Mode

The ADC has an automatic power management feature which is called auto-off mode, and is enabled by setting CFG0[AUTOOFF]=1.

When CFG0[AUTOOFF]=1, the ADC is always powered off when not converting and automatically wakes-up when a conversion is started by a software or hardware trigger. A startup-time is automatically inserted between the trigger event which starts the sampling and the conversion time of the ADC. The ADC is then automatically disabled once the sequence of conversions is complete. The STS[ADRDY] flag won't set under auto-off mode.

The auto-off mode can cause a dramatic reduction in the power consumption of applications that need relatively few conversions or when conversion requests are timed far enough apart (for example with a low-frequency hardware trigger) to justify the extra power and extra time used for switching the ADC on and off.

The auto-off mode can be combined with the wait mode (CFG0[WAIT]=1) for applications clocked at low frequency. This combination can provide significant power savings if the ADC is automatically powered-off during the wait phase and restarted as soon as the FIFO is read by the application.

25.4.6 Analog Window Watchdog

The analog window watchdog feature is enabled by setting the WDCTRL[WDEN] bit. It is used to monitor that either one selected channel or all enabled channels remain within a configured voltage range (window) as shown "Analog Watchdog Guarded Area".

Yuntu Microelectronics

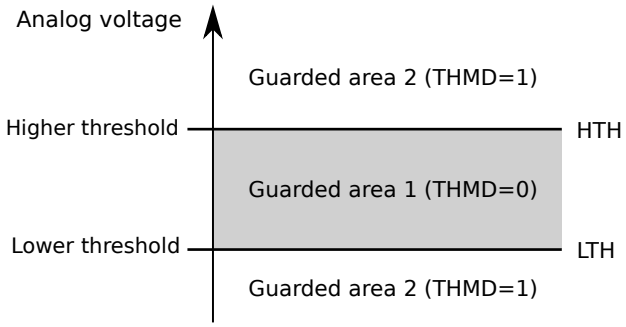


Figure 25.5: Analog Watchdog Guarded Area

It's able to select different threshold monitor modes by configuring WDTH[THMD]. When configure WDTH[THMD]=0, the analog watchdog status bit (STS[WD]) is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold. When configure WDTH[THMD] = 1, the analog watchdog status bit (STS[WD]) is set if the analog voltage converted by the ADC is above a lower threshold and below a higher threshold. These thresholds are programmed in the WDTH register. An interrupt can be enabled by setting the WDIE bit in the INTE register.The WD flag is cleared by software by writing 1 to it.

When converting a data with a resolution that less than 12-bit (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.

Table "Analog Watchdog Channel Selection" shows how to configure the WDSGL and WDEN bits in the WDCTRL register to enable the analog watchdog on one or more channels. When configure WDSGL=1, the monitor channel number is selected by WDCTRL[WDCHSEL].

Table 25.19: Analog Watchdog Channel Selection

WDSGL bit	WDEN bit	Guarded Channels
x	0	None
0	1	All channels
1	1	Single channel

25.4.7 Temperature Sensor

The temperature sensor is internally connected to the 17th channel which is used to convert the sensor's output voltage to a digital value.

Yuntu Microelectronics

Chapter 26

Analog Comparator (ACMP)

26.1 Chip-specific ACMP Information

26.1.1 Instantiation Information

YTM32B1LE0x contains one ACMP module. The input sources of the ACMP come from external analog signals or internal DAC output. ACMP module has two 8-to-1 analog MUXs to select external analog signals. The positive and negative ports of the comparator share the common eight external analog input signals. ACMP module can compare two analog signals and generate an asynchronous interrupt. This module can also operate on function clock which is independent from the register access clock.

26.1.2 Signal Description

Table 26.1: ACMP Signal Table

Signal Name	Description	Direction
ACMP_IN0	Input analog voltage, to the positive/negative port of the comparator	I
ACMP_IN1	Input analog voltage, to the positive/negative port of the comparator	I
ACMP_IN2	Input analog voltage, to the positive/negative port of the comparator	I
ACMP_IN3	Input analog voltage, to the positive/negative port of the comparator	I
ACMP_IN4	Input analog voltage, to the positive/negative port of the comparator	I
ACMP_IN5	Input analog voltage, to the positive/negative port of the comparator	I
ACMP_IN6	Input analog voltage, to the positive/negative port of the comparator	I
ACMP_IN7	Input analog voltage, to the positive/negative port of the comparator	I
ACMP_OUT	Output comparison result	O
VREFH	External reference voltage	I

Yuntu Microelectronics

26.1.3 Clock Description

Table 26.2: ACMP Clock Table

Clock Name	Description
Bus Clock	Bus clock used for registers access
Function Clock	Function clock used for function logic

26.1.4 DMA Description

Table 26.3: ACMP DMA Table

No.	Request Source	Description
48	Edge changes of ACMP out	Combine CTRL[EDGESEL] and STS[OUTNF/OUTPF] to generate DMA request

26.1.5 Interrupt Description

Table 26.4: ACMP Interrupt Table

Vector	IRQn	Request Source	Description	Wakeup
45	29	Result changes of ACMP in continuous mode or edge changes of ACMP out in other modes	Support asynchronous interrupt generation regardless of the existence of the bus clock	Y

26.1.6 Power Mode Description

Table 26.5: ACMP Power Mode Table

Power Mode	Description
Active	Full function
Sleep	Keeps working when function clock available
Deepsleep	Keeps working when function clock available
Standby	Keeps working when function clock available

26.2 Introduction

26.2.1 Overview

The Analog Comparator (ACMP) provides a circuit for comparing two analog input voltages and outputting comparison results. The comparator circuit is designed to operate across the full range of the supply voltage, known as rail-to-rail operation. The analog comparator module has two 8-to-1 analog MUXs. Each analog MUX provides a circuit for selecting an analog input signal from eight channels. The MUX circuit is designed to operate across the full range of the supply voltage.

Yuntu Microelectronics

26.2.2 Features

- Up to 8 channels
- Operational over the entire supply range
- Inputs may range from rail to rail
- Programmable hysteresis control
- Selectable inversion on comparator output
- Function mode:
  - Common mode
  - Sample mode
  - Window mode
  - Continuous mode
    - One-shot mode
    - Loop mode
- All channels can be used to execute automatic comparison
- Support digital filter, the filter can be bypassed
- Two software selectable performance levels
  - Shorter propagation delay at the expense of higher power
  - Low power with longer propagation delay
- Functional in all power mode
- Support independent 8-bit DAC output to the comparator
- Support several interrupts
  - For common/sample/window mode
    - Generate interrupt on rising-edge, falling-edge or both edges of the comparator output
  - For continuous mode
    - Generate interrupt when the comparison results don't match with expectations
- Interrupt can generate without any clock except in continuous mode
- A comparison event can be selected to trigger DMA transfer

26.2.3 Block Diagram

The following figure shows the ACMP block diagram.

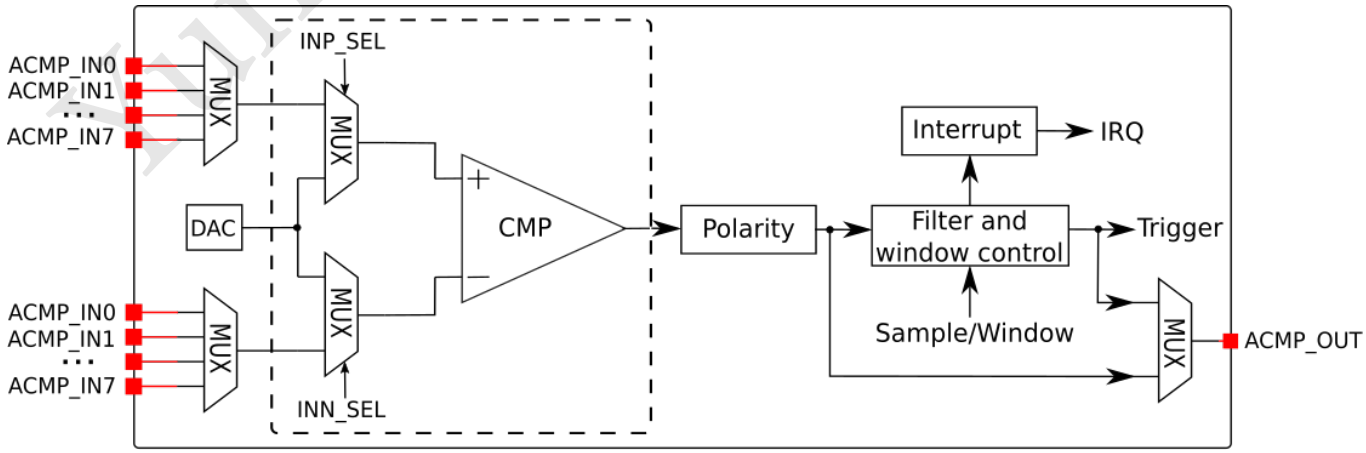


Figure 26.1: Block Diagram of ACMP

Yuntu Microelectronics

26.3 Register Definition

The ACMP registers are used to control ACMP module and display the current working status. In addition, registers are used to configure functional modes, such as continuous mode.

Reading reserved registers is always 0, the write operation is not valid.

Register_address = Base_address + Address_offset.

26.3.1 ACMP Memory Map

ACMP0 base address 40073000h

Table 26.6: ACMP Memory Map

Offset	Register	Width	Access	Reset Value
0h	Status Register (STS)	32	RW	00000000h
4h	Interrupt Enable Register (INTE)	32	RW	00000000h
8h	DAC Register (DAC)	32	RW	00000000h
ch	Channels Multiplexer (MUX)	32	RW	00000000h
10h	Filter Register (FILT)	32	RW	00000000h
14h	Continuous Mode Configuration Register (CONT)	32	RW	18300000h
18h	Control Register (CTRL)	32	RW	00000000h
1ch	Enable Register (EN)	32	RW	00000000h
20h	Direct Memory Access Register (DMACR)	32	RW	00000000h

26.3.1.1 ACMP STS Register

Offset: 0h

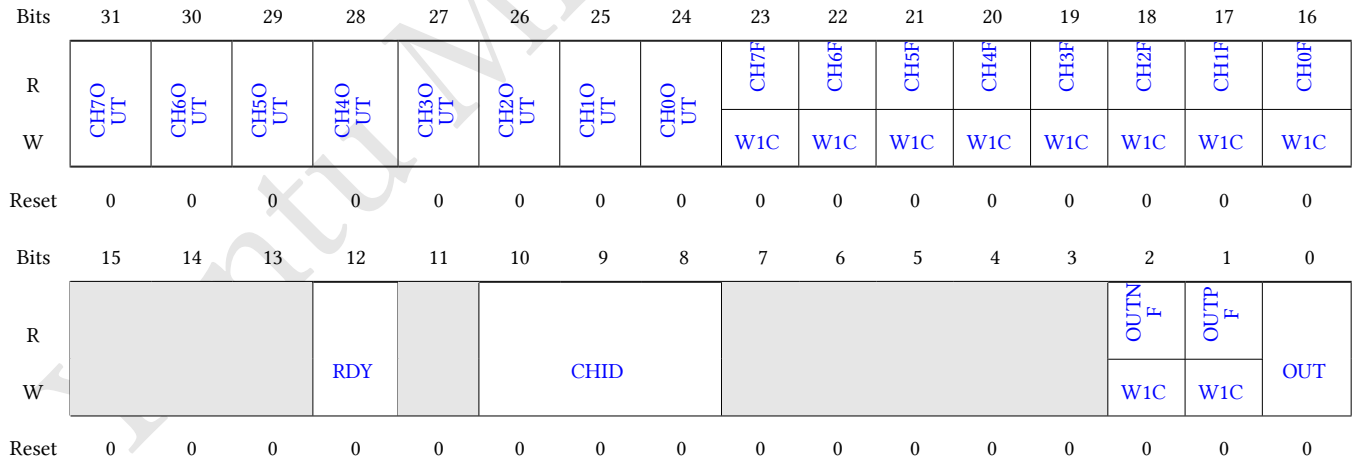


Table 26.7: ACMP STS Register Description

Field	Function
31	**Channel 7 Output In Continuous Mode**

Field	Function
CH7OUT	Channel 7 expected value configuration when ACMP disabled and indicate the channel 7 result of comparison when continuous mode enabled 0b - Indicate the channel 7 result of comparison is 0 when continuous mode enabled 1b - Indicate the channel 7 result of comparison is 1 when continuous mode enabled
30 CH6OUT	**Channel 6 Output In Continuous Mode** Channel 6 expected value configuration when ACMP disabled and indicate the channel 6 result of comparison when continuous mode enabled 0b - Indicate the channel 6 result of comparison is 0 when continuous mode enabled 1b - Indicate the channel 6 result of comparison is 1 when continuous mode enabled
29 CH5OUT	**Channel 5 Output In Continuous Mode** Channel 5 expected value configuration when ACMP disabled and indicate the channel 5 result of comparison when continuous mode enabled 0b - Indicate the channel 5 result of comparison is 0 when continuous mode enabled 1b - Indicate the channel 5 result of comparison is 1 when continuous mode enabled
28 CH4OUT	**Channel 4 Output In Continuous Mode** Channel 4 expected value configuration when ACMP disabled and indicate the channel 4 result of comparison when continuous mode enabled 0b - Indicate the channel 4 result of comparison is 0 when continuous mode enabled 1b - Indicate the channel 4 result of comparison is 1 when continuous mode enabled
27 CH3OUT	**Channel 3 Output In Continuous Mode** Channel 3 expected value configuration when ACMP disabled and indicate the channel 3 result of comparison when continuous mode enabled 0b - Indicate the channel 3 result of comparison is 0 when continuous mode enabled 1b - Indicate the channel 3 result of comparison is 1 when continuous mode enabled
26 CH2OUT	**Channel 2 Output In Continuous Mode** Channel 2 expected value configuration when ACMP disabled and indicate the channel 2 result of comparison when continuous mode enabled 0b - Indicate the channel 2 result of comparison is 0 when continuous mode enabled 1b - Indicate the channel 2 result of comparison is 1 when continuous mode enabled
25 CH1OUT	**Channel 1 Output In Continuous Mode** Channel 1 expected value configuration when ACMP disabled and indicate the channel 1 result of comparison when continuous mode enabled 0b - Indicate the channel 1 result of comparison is 0 when continuous mode enabled 1b - Indicate the channel 1 result of comparison is 1 when continuous mode enabled
24 CH0OUT	**Channel 0 Output In Continuous Mode** Channel 0 expected value configuration when ACMP disabled and indicate the channel 0 result of comparison when continuous mode enabled 0b - Indicate the channel 0 result of comparison is 0 when continuous mode enabled 1b - Indicate the channel 0 result of comparison is 1 when continuous mode enabled
23	**Channel 7 Mismatch Flag**

Yuntu Microelectronics

Field	Function
CH7F	In continuous mode, the flag sets when channel 7 result of comparison doesn't match with it's expected value. Write 1 to clear the flag 0b - Channel 7 result of comparison match with it's expected value 1b - Channel 7 result of comparison doesn't match with it's expected value
22 CH6F	**Channel 6 Mismatch Flag** In continuous mode, the flag sets when channel 6 result of comparison doesn't match with it's expected value. Write 1 to clear the flag 0b - Channel 6 result of comparison match with it's expected value 1b - Channel 6 result of comparison doesn't match with it's expected value
21 CH5F	**Channel 5 Mismatch Flag** In continuous mode, the flag sets when channel 5 result of comparison doesn't match with it's expected value. Write 1 to clear the flag 0b - Channel 5 result of comparison match with it's expected value 1b - Channel 5 result of comparison doesn't match with it's expected value
20 CH4F	**Channel 4 Mismatch Flag** In continuous mode, the flag sets when channel 4 result of comparison doesn't match with it's expected value. Write 1 to clear the flag 0b - Channel 4 result of comparison match with it's expected value 1b - Channel 4 result of comparison doesn't match with it's expected value
19 CH3F	**Channel 3 Mismatch Flag** In continuous mode, the flag sets when channel 3 result of comparison doesn't match with it's expected value. Write 1 to clear the flag 0b - Channel 3 result of comparison match with it's expected value 1b - Channel 3 result of comparison doesn't match with it's expected value
18 CH2F	**Channel 2 Mismatch Flag** In continuous mode, the flag sets when channel 2 result of comparison doesn't match with it's expected value. Write 1 to clear the flag 0b - Channel 2 result of comparison match with it's expected value 1b - Channel 2 result of comparison doesn't match with it's expected value
17 CH1F	**Channel 1 Mismatch Flag** In continuous mode, the flag sets when channel 1 result of comparison doesn't match with it's expected value. Write 1 to clear the flag 0b - Channel 1 result of comparison match with it's expected value 1b - Channel 1 result of comparison doesn't match with it's expected value
16 CH0F	**Channel 0 Mismatch Flag** In continuous mode, the flag sets when channel 0 result of comparison doesn't match with it's expected value. Write 1 to clear the flag 0b - Channel 0 result of comparison match with it's expected value 1b - Channel 0 result of comparison doesn't match with it's expected value
12	**Ready Flag**

Yuntu Microelectronics

Field	Function
RDY	In one-shot mode, STS[RDY]=1 indicates waiting for hard trigger. When valid hard trigger got, STS[RDY] auto cleared.
10 - 8 CHID	**Current Channel ID** Indicate current comparing channel in continuous mode
2 OUTNF	**Comparator Output Negative Edge Flag** Comparator output negative edge flag
1 OUTPF	**Comparator Output Positive Edge Flag** Comparator output positive edge flag
0 OUT	**Current Comparator Output** Current comparator output

26.3.1.2 ACMP INTE Register

Offset: 4h

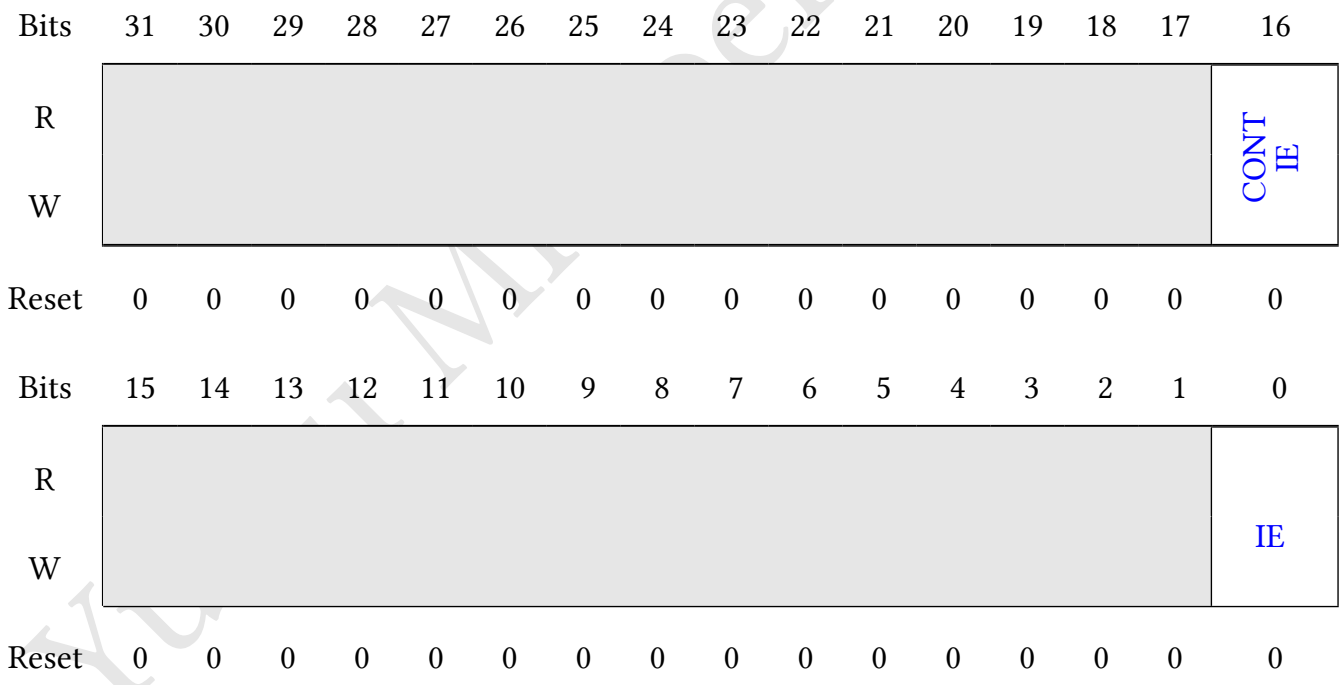


Table 26.8: ACMP INTE Register Description

Field	Function
16	**Continuous Mode Interrupt Enable**

Yuntu Microelectronics

Field	Function
CONTIE	In continuous mode, interrupt occurs when INTE[CONTIE]=1 and STS[CHxF]=1 0b - Continuous mode interrupt is disabled 1b - Continuous mode interrupt is enabled
0 IE	**Interrupt Enable** Interrupt occurs when INTE[IE]=1 and STS[OUTNF]=1/STS[OUTPF]=1(depending on CTRL[EDGESEL]) 0b - Interrupt is disabled 1b - Interrupt is enabled

26.3.1.3 ACMP DAC Register

Offset: 8h

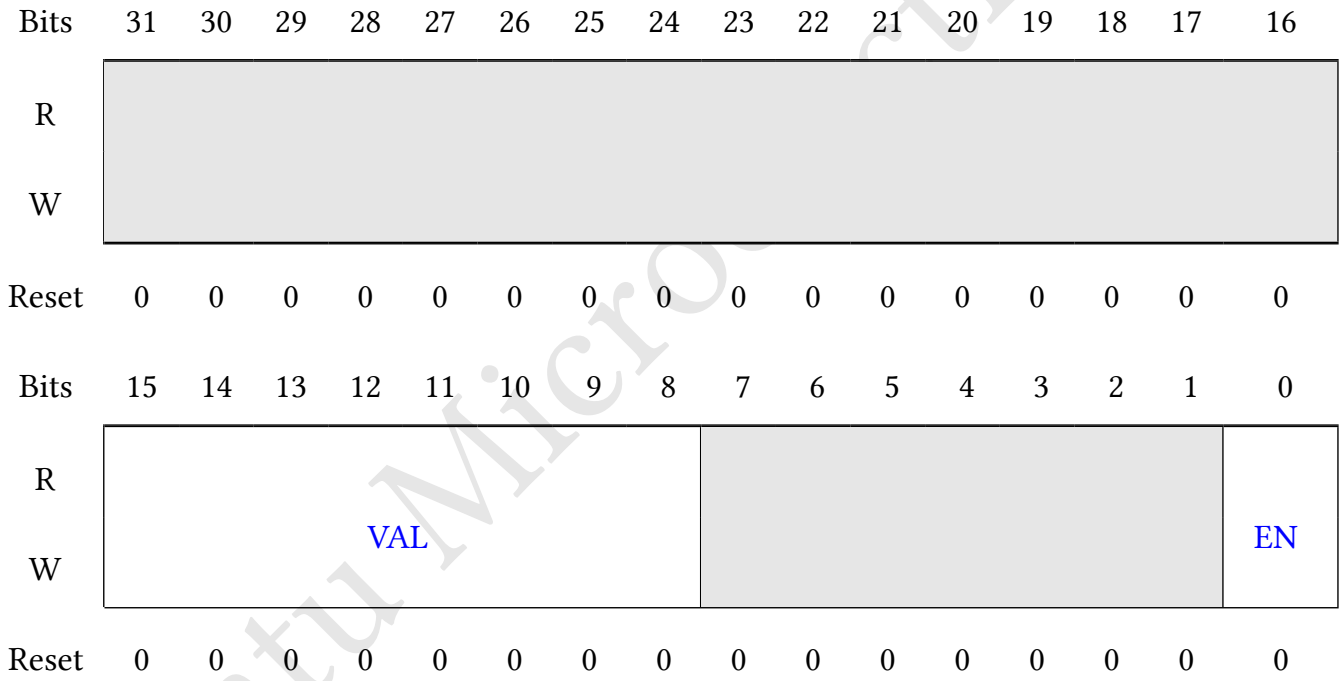


Table 26.9: ACMP DAC Register Description

Field	Function
15 - 8 VAL	**DAC Value** DAC output voltage level selection from 0 to 255, voltage is $(DAC[VAL]/255) * VREFH$ when DAC[VREF] = 0
0 EN	**DAC Enable** DAC enable 0b - DAC is disabled 1b - DAC is enabled

Yuntu Microelectronics

26.3.1.4 ACMP MUX Register

Offset: ch

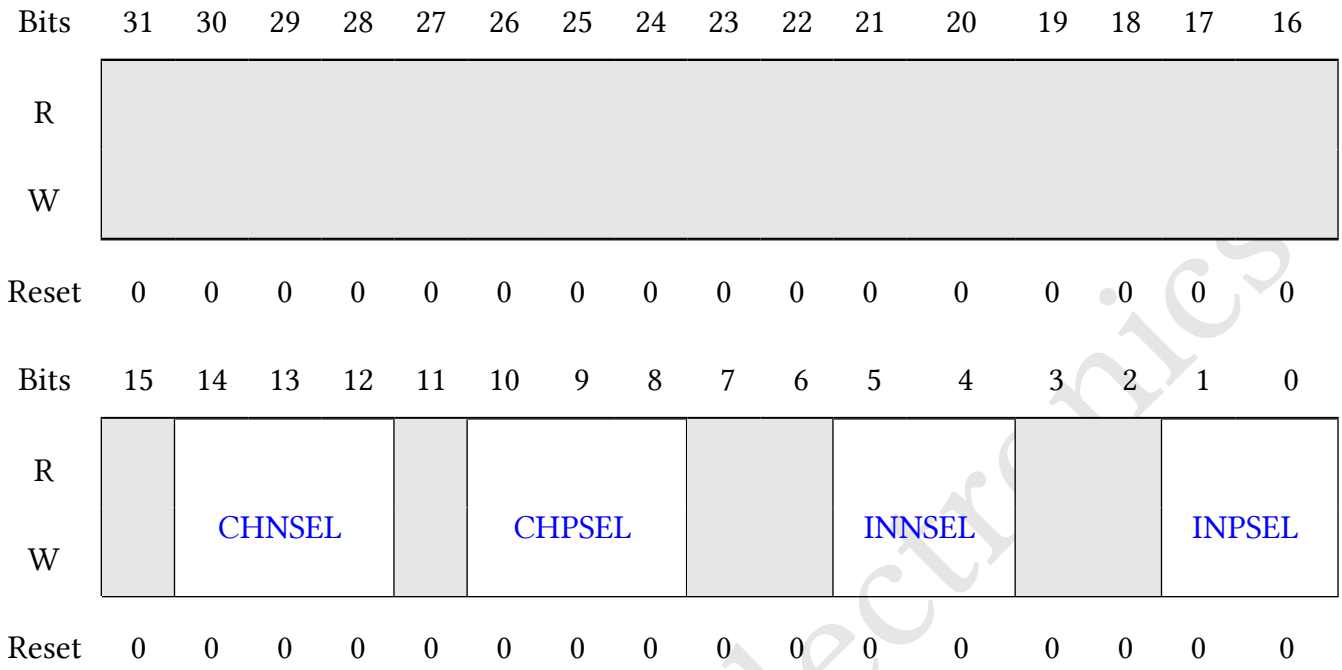


Table 26.10: ACMP MUX Register Description

Field	Function
14 - 12 CHNSEL	**Negative Channel Selection** Select one channel as negative input of comparator 000b - Select channel 0 001b - Select channel 1 010b - Select channel 2 011b - Select channel 3 100b - Select channel 4 101b - Select channel 5 110b - Select channel 6 111b - Select channel 7
10 - 8 CHPSEL	**Positive Channel Selection** Select one channel as positive input of comparator 000b - Select channel 0 001b - Select channel 1 010b - Select channel 2 011b - Select channel 3 100b - Select channel 4 101b - Select channel 5 110b - Select channel 6 111b - Select channel 7
5 - 4	**Negative Input of ACMP Selection**

Yuntu Microelectronics

Field	Function
INNSEL	Negative input of ACMP selection 00b - Select DAC output as comparator negative input 01b - Select one channel by configuring MUX[CHNSEL] as comparator negative input 10b - Reserved 11b - Reserved
1 - 0 INPSEL	**Positive Input of ACMP Selection** Positive input of ACMP selection 00b - Select DAC output as comparator positive input 01b - Select one channel by configuring MUX[CHPSEL] as comparator positive input 10b - Reserved 11b - Reserved

26.3.1.5 ACMP FILT Register

Offset: 10h

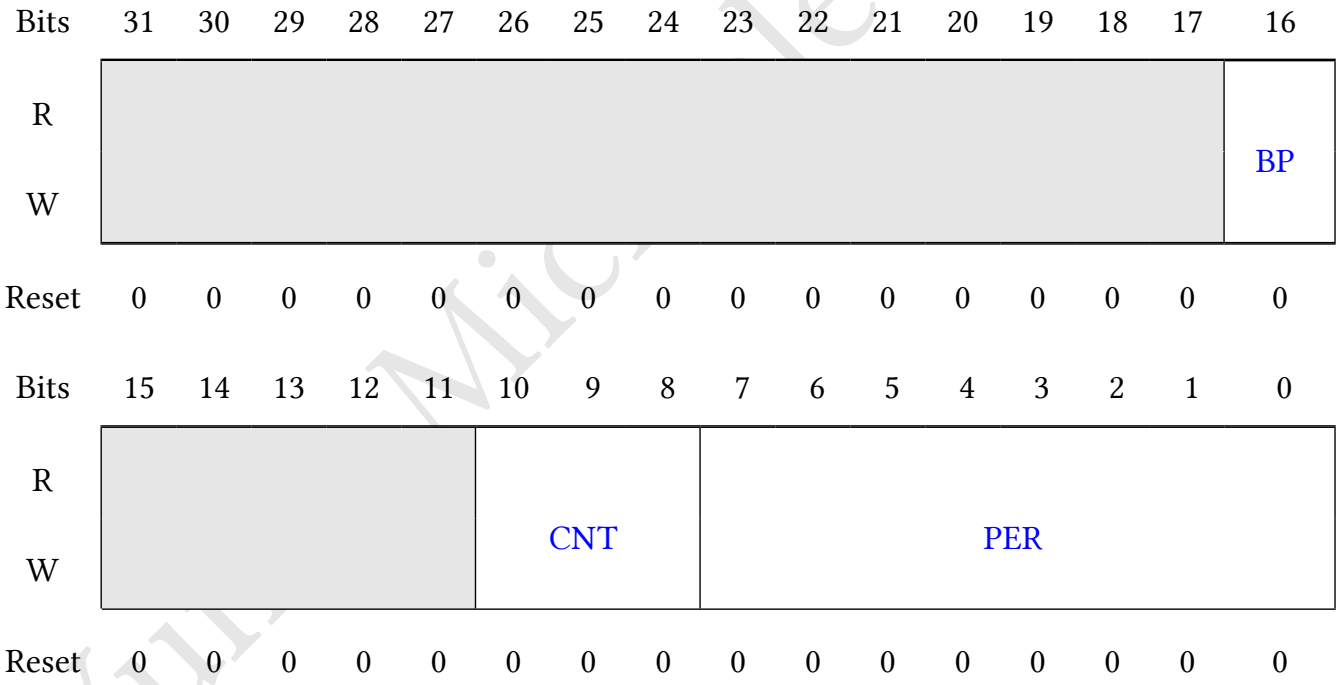


Table 26.11: ACMP FILT Register Description

Field	Function
16 BP	**Filter Bypass** Filter bypass control 0b - Filter is not bypassed 1b - Filter is bypassed forcibly
10 - 8	**Digital Filter Count**

Field	Function
CNT	Analog comparator output is regarded as a valid signal when the continuous sampling 0 or 1 reaches the configured value CNT, otherwise it is regarded as a glitch. Filter is bypassed when CNT=0
7 - 0 PER	**Digital Filter Period** Digital filter sampling period. Filter is bypassed when PER=0

26.3.1.6 ACMP CONT Register

Offset: 14h

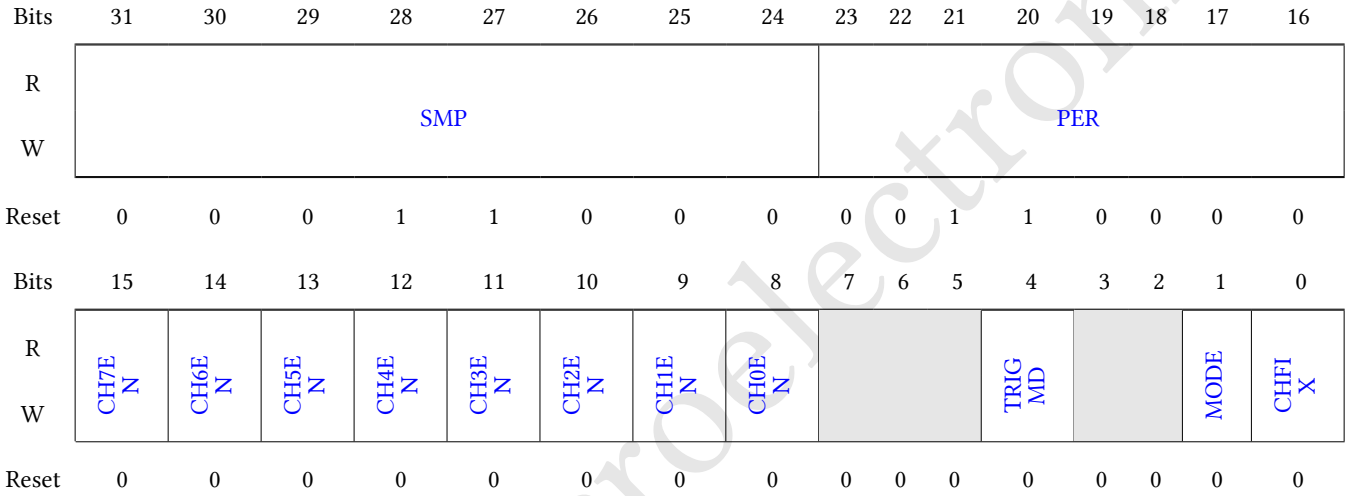


Table 26.12: ACMP CONT Register Description

Field	Function
31 - 24 SMP	**Channel Sample Position in Continuous Mode** After channel switching stabilization time, sample the comparator result
23 - 16 PER	**Continuous Mode Period** The sampled channel switching period in continuous mode
15 CH7EN	**Channel 7 Enable in Continuous Mode** If enabled, channel 7 participates in the continuous mode 0b - Channel 7 is disabled 1b - Channel 7 is enabled
14 CH6EN	**Channel 6 Enable in Continuous Mode** If enabled, channel 6 participates in the continuous mode 0b - Channel 6 is disabled 1b - Channel 6 is enabled
13	**Channel 5 Enable in Continuous Mode**

Yuntu Microelectronics

Field	Function
CH5EN	If enabled, channel 5 participates in the continuous mode 0b - Channel 5 is disabled 1b - Channel 5 is enabled
12 CH4EN	**Channel 4 Enable in Continuous Mode** If enabled, channel 4 participates in the continuous mode 0b - Channel 4 is disabled 1b - Channel 4 is enabled
11 CH3EN	**Channel 3 Enable in Continuous Mode** If enabled, channel 3 participates in the continuous mode 0b - Channel 3 is disabled 1b - Channel 3 is enabled
10 CH2EN	**Channel 2 Enable in Continuous Mode** If enabled, channel 2 participates in the continuous mode 0b - Channel 2 is disabled 1b - Channel 2 is enabled
9 CH1EN	**Channel 1 Enable in Continuous Mode** If enabled, channel 1 participates in the continuous mode 0b - Channel 1 is disabled 1b - Channel 1 is enabled
8 CH0EN	**Channel 0 Enable in Continuous Mode** If enabled, channel 0 participates in the continuous mode 0b - Channel 0 is disabled 1b - Channel 0 is enabled
4 TRIGMD	**Trigger Mode** Trigger mode control 0b - Trigger mode is disabled 1b - Trigger mode is enabled
1 MODE	**Continuous Mode Selection** Continuous mode selection 0b - Loop mode 1b - One-shot mode
0 CHFIX	**Channel Fix** Fix channel in continuous mode 0b - Fix negative channel, polling comparisons are made on the positive channels 1b - Fix positive channel, polling comparisons are made on the negative channels

26.3.1.7 ACMP CTRL Register

Offset: 18h

Yuntu Microelectronics

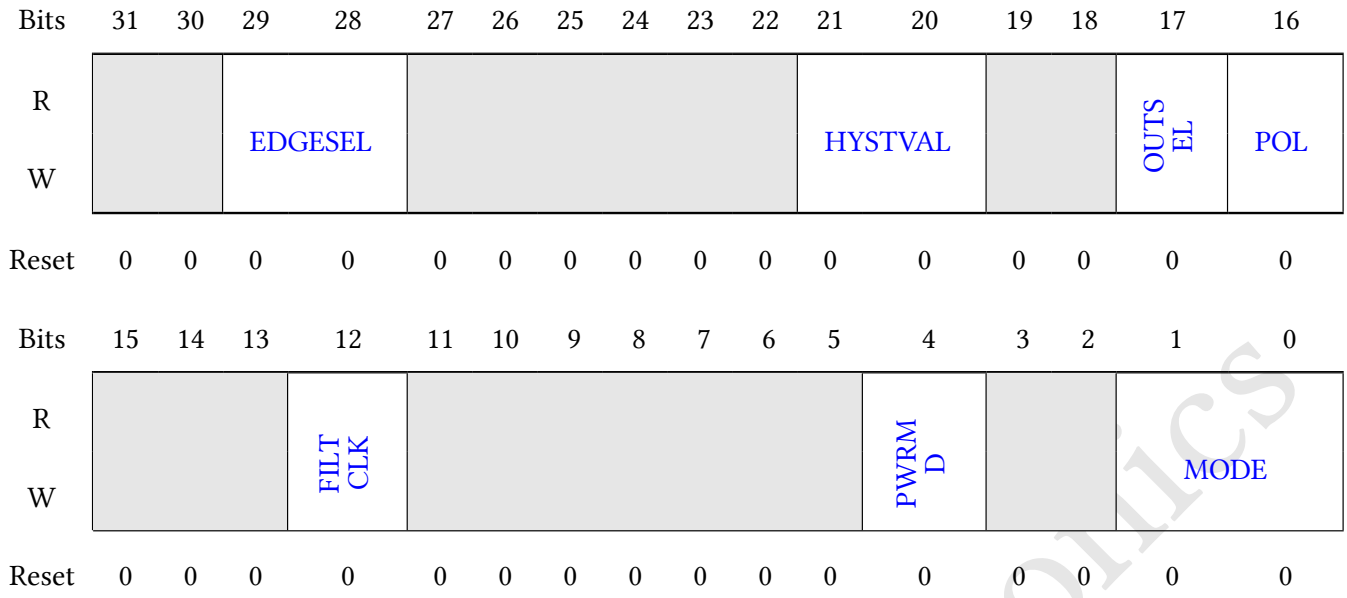


Table 26.13: ACMP CTRL Register Description

Field	Function
29 - 28 EDGESEL	**Edge Detection Selection** Edge detection condition, used for interrupt and DMA 00b - No edge detection 01b - Rising edge detection 10b - Falling edge detection 11b - Dual edges detection, only valid in interrupt, not for DMA
21 - 20 HYSTVAL	**Comparator Hysteresis Value** Hysteresis level selection bit 00b - Comparator hard block output has no hysteresis internally 01b - Comparator hard block output has 20mV hysteresis internally 10b - Comparator hard block output has 40mV hysteresis internally 11b - Reserved
17 OUTSEL	**Output Selection** Output selection bit 0b - Send ACMP output to pin, it's up to analog circuit and work mode 1b - Send comparator raw output to pin directly(Need to consider the configuration of CTRL[POL])
16 POL	**Output Polarity** Output polarity selection bit 0b - Not invert comparator output 1b - Invert comparator output
12	**Filter Clock Source**

Yuntu Microelectronics

Field	Function
FILTCLK	Filter clock source 0b - Select bus clock for filter 1b - Select function clock for filter
4 PWRMD	**Power Mode** Power mode 0b - High speed mode 1b - Low power mode
1 - 0 MODE	**ACMP Function Mode** ACMP function mode selection 00b - Common mode 01b - Sample mode 10b - Window mode 11b - Continuous mode

26.3.1.8 ACMP EN Register

Offset: 1ch

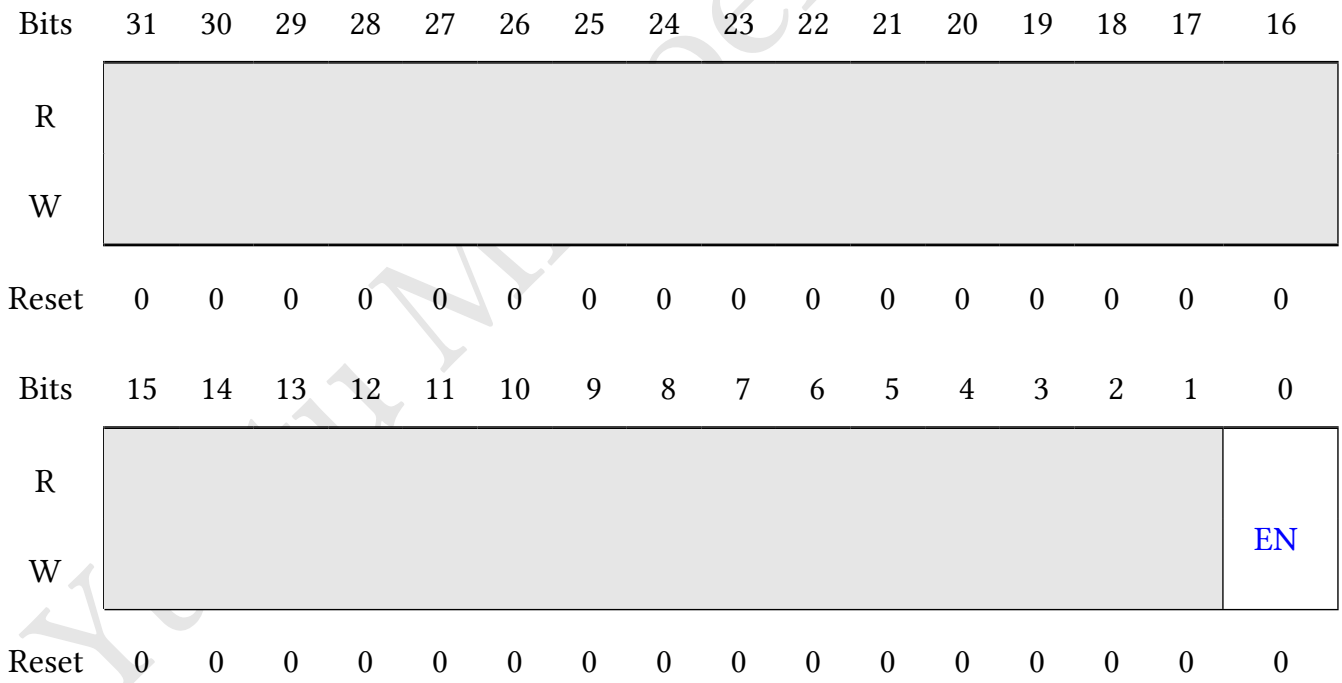


Table 26.14: ACMP EN Register Description

Field	Function
0 EN	**Module Enable** Module enable bit 0b - ACMP module is disabled 1b - ACMP module is enabled

Yuntu Microelectronics

26.3.1.9 ACMP DMACR Register

Offset: 20h

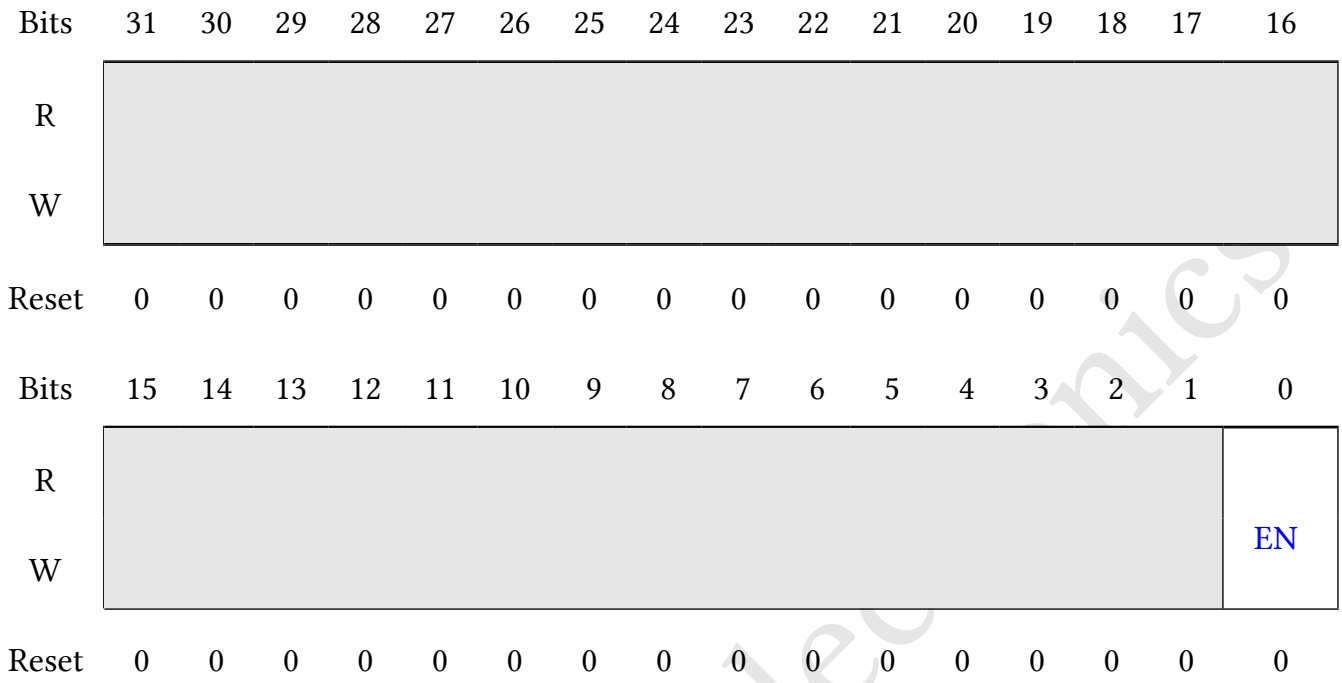


Table 26.15: ACMP DMACR Register Description

Field	Function
0 EN	**Direct Memory Access Enable** Combine EDGESEL and OUTNF/OUTPF to generate DMA request 0b - DMA request is disabled 1b - DMA request is enabled

26.4 Functional Description

The ACMP module has two 8-to-1 analog MUXs with eight analog input channels. Digital filter of comparator output is supported. ACMP module has four function modes: common mode, sample mode, window mode and continuous mode. The continuous mode includes two sub-modes: one-shot mode and loop mode. ACMP module supports to work in low power mode, and interrupt can wake up the chip from low power mode. In addition, the ACMP supports DMA.

26.4.1 Input Control

26.4.1.1 Input Selection

The input source of the ACMP can be selected by the MUX register, there are two sources to choose from the external signal from the pad or the internal signal from the DAC. The positive input source is selected by MUX[INPSEL], and the negative input source is selected by MUX[INNSEL]. ACMP module has two 8-to-1 analog MUXs with eight analog input channels which for external signal. The positive

and negative input of the comparator can both select one of eight external signals by MUX[CHPSEL] and MUX[CHNSEL].

NOTE: It is forbidden to select the same signal for both positive and negative input.

26.4.1.2 DAC (Digital to Analog Converter)

The ACMP has an internal 8-bit DAC module. The DAC reference voltage is from external reference voltage VREFH. The output voltage configured by DAC[VAL], the voltage value is $(\text{DAC[VAL]}/255) * \text{VREFH}$. Setting DAC[EN]=1 to enable the DAC module.

26.4.2 Output Control

26.4.2.1 Polarity

The comparator output can be inverted or keep by configuring CTRL[POL] bit.

26.4.2.2 Filter

Comparator output result exists glitches. The digital filter is used to filter out glitches to avoid mistakes, it has a sampling counter with configurable sampling period. Filter is bypassed when FILT[BP]=1 or FILT[CNT]=0 or FILT[PER]=0.

As long as the comparator output changes, the sampling counter reset. If the comparator output remains unchanged, the sampling counter will accumulate to FILT[CNT] value according to the sampling period and hold. Analog comparator output is regarded as a valid signal only when the sampling counter reaches the configured value (FILT[CNT]).

26.4.3 Function Mode

ACMP module has four function modes: common mode, sample mode, window mode and continuous mode.

26.4.3.1 Common Mode

ACMP module is in common mode when CTRL[MODE]=0. In common mode, the comparator output (STS[OUT]) reflects the comparison result of the selected two signals in real time. The interrupt and DMA request is generated according to STS[OUTNF]/STS[OUTPF].

26.4.3.2 Sample Mode

ACMP module is in sample mode when CTRL[MODE]=1. In sample mode, the comparator output (STS[OUT]) updates with the comparison result of the selected two signals when get sample signal from the TMU module (ACMP_SAMPLE). The comparison result is sampled by hardware. The interrupt and DMA request is generated according to STS[OUTNF]/STS[OUTPF].

26.4.3.3 Window Mode

ACMP module is in window mode when CTRL[MODE]=2. In window mode, the comparator output (STS[OUT]) updates with the comparison result of the selected two signals when getting a valid window signal from the TMU module (ACMP_SAMPLE). The comparison result works under window control. The interrupt and DMA request is generated according to STS[OUTNF]/STS[OUTPF].

Yuntu Microelectronics

26.4.3.4 Continuous Mode

ACMP module is in continuous mode when CTRL[MODE]=3. In continuous mode, the comparator output (STS[OUT]) reflects the comparison result of the current two signals in real time. Continuous mode includes two ways: one-shot mode and loop mode.

26.4.3.4.1 Channel Fix

In continuous mode, one of the two inputs is fixed and polling comparisons are made on the positive channels. The fixed input is chosen by CONT[CHFIX]. When CONT[CHFIX]=0, the negative input channel fix, and the positive input switches from the selected channels; When CONT[CHFIX]=1, the positive input channel fix, and the negative input port switches from the selected channels relatively. The input to be scanned is required to come from external analog signals, and the fixed input source could select an external analog signal or the internal temp sensor output.

26.4.3.4.2 Channel Select

The scanned channels is enabled by configuring CONT[CHnEN] (n=0,1,...,7), the scan sequence is from channel 0 to channel 7. For example, when configure CONT[CHFIX]=0 and CONT[CHnEN]=1 (n=0,1,2,3), the negative port of the comparator fix and positive input channels (ACMP_IN0/ACMP_IN1/ACMP_IN2/ACMP_IN3) participate in automatic switching continuous one by one. If both two input sources select the external analog signal (MUX[INPSEL]=1 and MUX[INNSEL]=1), the fixed channel will not be scanned for the other input even if enabled by CONT[CHnEN].

26.4.3.4.3 Timing

The channel switching period is configurable. After channel switching stabilization time, the internal counter starts counting until CONT[PER] is reached, then the comparator switches to the next selected channel. When the internal counter reaches CONT[SMP], the comparator output result is sampled. Show as bellow "Continuous Mode of ACMP".

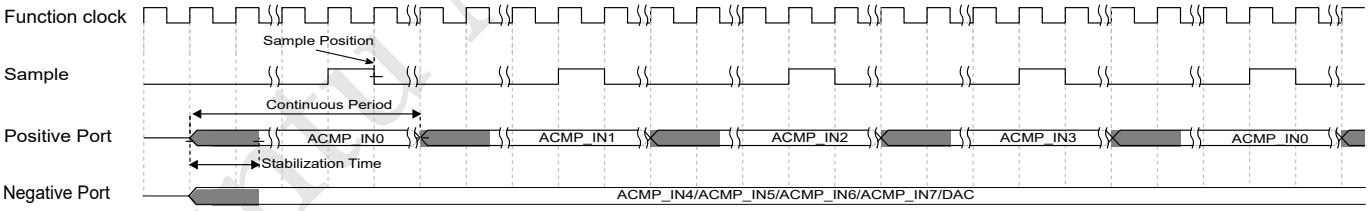


Figure 26.2: Continuous Mode of ACMP

NOTE: The stabilization time is up to the hardware, it's around (todo) ns.

26.4.3.4.4 Scan Result

The scan result in continuous mode is reflected by STS[CHnF] (n=0,1,...,7). Before starting a continuous comparison, the initial value requests to be written into STS[CHnOUT] (n=0,1,...,7). During the comparison, the initial value is regarded as the expected result and compared with the comparator output result. The corresponding flag set (STS[CHnF]=1) when the comparison result of the current channel is not equal to the expected result. The interrupt and DMA request is generated according to STS[CHnF].

Yuntu Microelectronics

26.4.3.4.5 Loop Mode

In continuous mode, loop mode is enabled by setting CONT[CONTMD]=0. In loop mode, the comparator restarts scanning the selected channels after finishing the comparison of the last channel. Configuring EN[EN]=0 to end the continuous comparison.

26.4.3.4.6 One-shot Mode

In continuous mode, one-shot mode is enabled by setting CONT[CONTMD]=1. In one-shot mode, the comparator stops the continuous comparison after finishing the comparison of the last channel. In one-shot mode, the ACMP can switch the selected channel by hardware trigger, this function is enabled by setting CONT[TRIGMD]=1.

The hardware trigger is only from the TMU module, and users can config the CIM register to select the trigger type. There are three types of triggers to be chosen:

- Raw signal which from the TMU module (CIM->CTRL[ACMPTSS]=0)
- Signal from the TMU module and synchronized by bus clock (CIM->CTRL[ACMPTSS]=1)
- Signal from the TMU module and synchronized by function clock (CIM->CTRL[ACMPTSS]=2)

NOTE: Users should select the prepare trigger type according to the different applications.

26.4.4 Modes of Operation

ACMP module can work in Active mode, Sleep mode, Deepsleep mode and Standby mode. As show ”ACMP Power Mode Table”.

26.4.5 Interrupts

ACMP module support asynchronous interrupt generation regardless of the existence of the bus clock. In continuous mode, the interrupt is generated when INTE[CONTIE]=1 and any channel flag set (STS[CHnF]=1). In other modes, select the expected interrupt edge by configuring CTRL[EDGESEL] first, then enable the interrupt by setting INTE[IE]=1, interrupt will be generated by STS[OUTNF]/STS[OUTPF] due to the comparator output changes. To clear the interrupt, write 1 to corresponding flag by software.

NOTE: Users should avoid configuring INTE[IE]=1 and INTE[CONTIE]=1 at the same time.

26.4.6 DMA Usage

The ACMP module supports DMA, this function is enabled by setting DMA[DMAEN]=1. The DMA request is generated when a valid edge defined in CTRL[EDGESEL] occurs. The flag STS[OUTNF]/STS[OUTPF] is set when a valid edge defined in CTRL[EDGESEL] occurs. The DMA request is cleared by clearing the flag STS[OUTNF]/STS[OUTPF]. The DMA priority is higher than the interrupt. As show ”ACMP DMA Table”.

26.4.7 Reset

During a reset, the ACMP is in the default mode and disabled.

Yuntu Microelectronics

Chapter 27

Temperature Sensor (TEMPSENSOR)

27.1 Introduction

The temperature sensor provides a circuit to measure the temperature of the chip. It converts the temperature to the corresponding voltage, and ADC module can convert it to digital value through TEMP_SENSOR channel. According to the result of ADC conversion and the following formula, we can obtain the temperature of the chip.

$$\mathrm{T} = \mathrm{C} - \mathrm{K} * \mathrm{V}$$

K= 616.9

V is the voltage calculated from the ADC conversion result, formula is $V = (D/4096) * Vrefh$, Vrefh is the reference voltage of the ADC, D is the ADC conversion digital value.

C is an coefficient calibrated at 25 °C

For YTM32B1LE0x device, users can obtain C through the following formula.

$$C = \frac{(C_{pre} - 25)}{530} * 616.9 + 25$$

The value of C_{pre} can be read from 0x10000f34.

Yuntu Microelectronics

Part IX

Security, Integrity and Safety

Yuntu Microelectronics

Chapter 28

Cyclic Redundancy Check (CRC)

28.1 Introduction

The CRC (Cyclic Redundancy Check) module is used to compute the 32/16/4-bit CRC value for error detection. The CRC value is generated by fixed CRC polynomials. Bit-swap is optional for CRC input and output. Bit-inversion is only optional for CRC output.

28.1.1 Features

- The following CRC polynomials are implemented:
- CRC4 (CRC-ITU):
  $X^4 + X + 1$
- CRC16 (CRC-CCITT):
  $X^{16} + X^{12} + X^5 + 1$
- CRC32 (CRC-ethernet):
  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Programmable initial seed
- Optional bit-swap in one byte is available for input data
- Optional bit-swap in one word is available for output data
- Optional bit-inversion is available for output data
- 8/16/32-bit access for CRC input data

28.1.2 Block Diagram

The following figure shows the CRC block diagram.

Yuntu Microelectronics

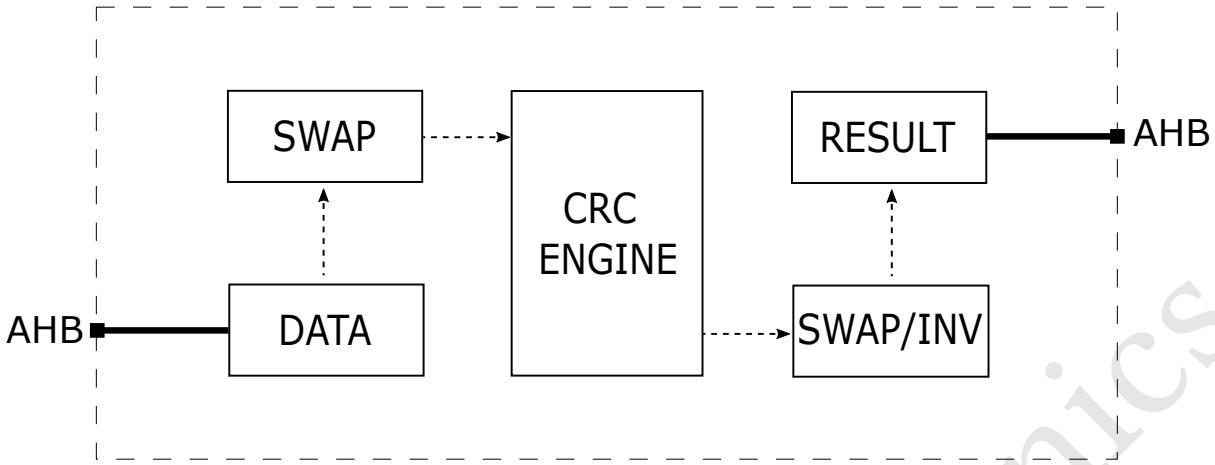


Figure 28.1: CRC Block Diagram

28.2 Register Definition

28.2.1 CRC Memory Map

CRC base address 40032000h

Table 28.1: CRC Memory Map

Offset	Register	Width	Access	Reset Value
0h	Control Register (CTRL)	32	RW	00000000h
4h	Initial Seed Register (INIT)	32	RW	00000000h
10h	Data in 32 bits (DATA_IN32)	32	WO	00000000h
10h	Data in 16 bits (DATA_IN16)	16	WO	0000h
10h	Data in 8 bits (DATA_IN8)	8	WO	00h
ch	Result Register (RESULT)	32	RO	00000000h

28.2.1.1 CRC CTRL Register

Offset: 0h

Yuntu Microelectronics

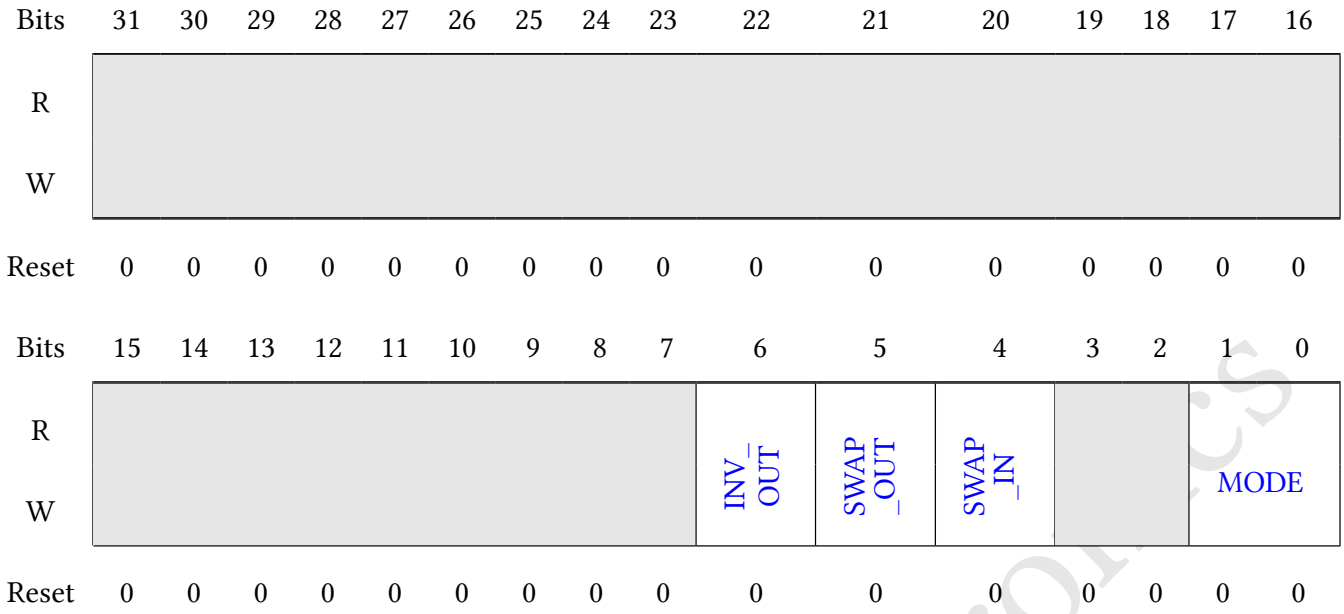


Table 28.2: CRC CTRL Register Description

Field	Function
6 INV_OUT	**Invert Output** Bit-invertion for all CRC output bits. 0b - Disable the bit-inversion 1b - Enable the bit-inversion
5 SWAP_OUT	**Swap Output** Swap all the valid bits (4/16/32-bit) for CRC output data. 0b - Disable the output bit-swap 1b - Enable the output bit-swap
4 SWAP_IN	**Swap Input** Enable bit-swap in each byte for CRC input data. 0b - Disable the input bit-swap 1b - Enable the input bit-swap
1 - 0 MODE	**CRC Mode** Select CRC calculation mode. 00b - Select CRC-4 mode 01b - Select CRC-16 mode 10b - Select CRC-32 mode

28.2.1.2 CRC INIT Register

Offset: 4h

Yuntu Microelectronics

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SEED															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SEED															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28.3: CRC INIT Register Description

Field	Function
31 - 0 SEED	**Initial Seed** This register contains the CRC initial seed value.

28.2.1.3 CRC DATA_IN32 Register

Offset: 10h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28.4: CRC DATA_IN32 Register Description

Field	Function
31 - 0 DATA	**CRC Data Input** CRC data input in 32 bits

Yuntu Microelectronics

28.2.1.4 CRC DATA_IN16 Register

Offset: 10h

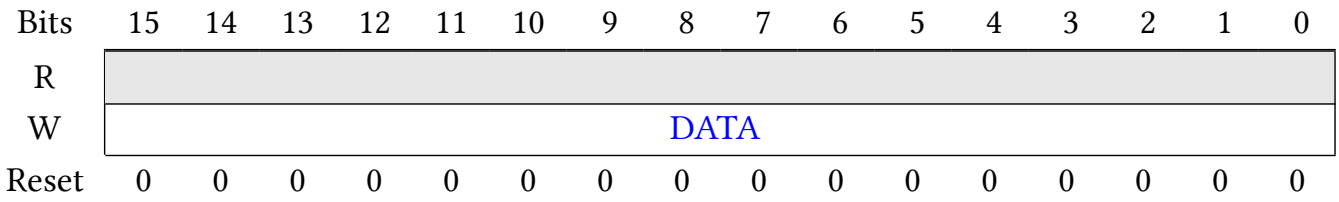


Table 28.5: CRC DATA_IN16 Register Description

Field	Function
15 - 0 DATA	**CRC Data Input** CRC data input in 16 bits

28.2.1.5 CRC DATA_IN8 Register

Offset: 10h

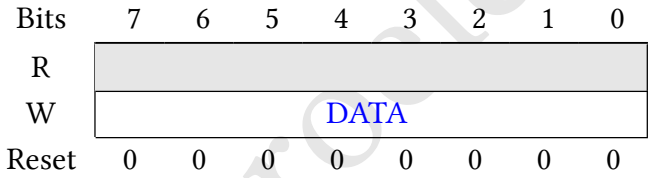


Table 28.6: CRC DATA_IN8 Register Description

Field	Function
7 - 0 DATA	**CRC Data Input** CRC data input in 8 bits

28.2.1.6 CRC RESULT Register

Offset: ch

Yuntu Microelectronics

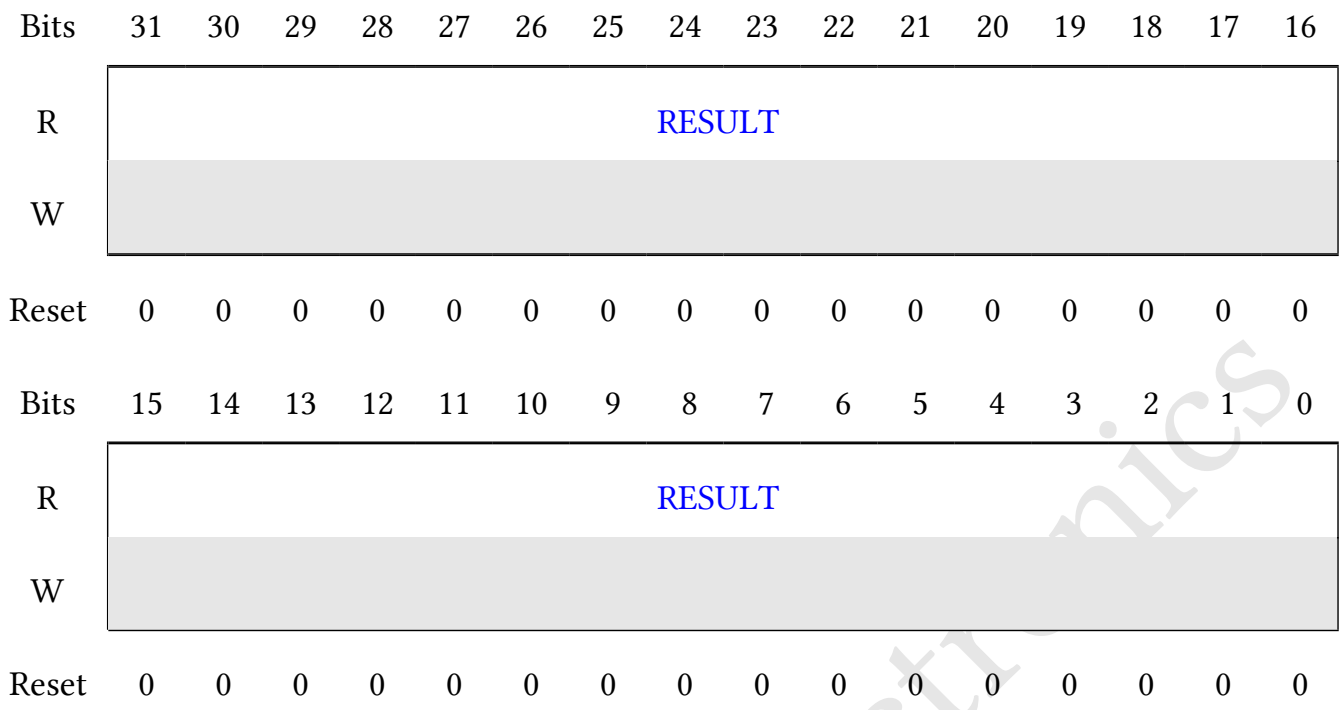


Table 28.7: CRC RESULT Register Description

Field	Function
31 - 0 RESULT	**CRC Result** This register contains the result of CRC calculation. In CRC-16 mode, only the low 16 bits are valid. In CRC-4 mode, only the low 4 bits are valid.

28.3 Functional Description

28.3.1 CRC Initialization

Before the user starts a CRC calculation, it is necessary to set several configurations appropriately.

- Configure bit swap and bit inversion if necessary.
- Select appropriate CRC mode
- Program initial CRC seed
- Program CRC input data to DATA register
- Get CRC result from RESULT register

28.3.2 CRC Calculation

The following fixed polynomials are implemented for 32/16/4-bit CRC calculation:

- CRC-04: $X^4 + X + 1$
- CRC-16: $X^{16} + X^{12} + X^5 + 1$
- CRC-32: $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$

For 32/16/4-bit CRC modes, 8/16/32-bit access of CRC input data with all contiguous bytes is available.

Yuntu Microelectronics

- If CRC input data is programmed by 8/16-bit, the calculation will be done in one clock cycle.
- If CRC input data is programmed by 32-bit, the calculation will be done in two clock cycles.

Please note that for 32-bit write access, the CRC engine will calculate from bit31 to bit0 which means for 8-bit write access, in order to get same result of 32-bit write access, user should send data[31:24] first and data[7:0] last.

For example, for 32-bit write access like 0x12345678, for 16-bit write access should be 0x1234 first, then 0x5678, for 8-bit write access should be 0x12 first, then 0x34, 0x56, 0x78, so that user could get same crc result.

28.3.3 Bit Swap and Inversion

The CRC module provides the swap function for both writing the input data and reading the CRC result.

The following types of swap function are available:

- If CTRL[SWAP_IN] is 1, the DATA[31:0] will become {DATA[24:31], DATA[16:23], DATA[8:15], DATA[0:7]}.
- If CTRL[SWAP_OUT] is 1, the RESULT[31:0] will become RESULT[0:31]. This module also provides the invert function for reading the CRC result.
- If CTRL[INV_OUT] is 1, each bit of RESULT will be inverted.

Yuntu Microelectronics

Chapter 29

Watch Dog (WDG)

29.1 Introduction

The Watchdog Timer(WDG) modules is an independent timer for safety monitor during software execution. It is a 32-bit timer that enables system to recover from unexpected situations like software stuck at a loop or executing unintended code. User can configure WDG to generate interrupt or reset for the first timeout, and it always generates reset request for the second consecutive timeout.

29.1.1 Features

- 32-bit countdown timer
- Functional clock can be selected from SIRC and SXOSC
- Support regular or window servicing mode
- Support reset request or interrupt for the first timeout
- Hard and soft configuration lock bits

29.2 Register Definition

This subsection provides a description of the memory map and registers.

29.2.1 WDG Memory Map

WDG0 base address 40052000h

Table 29.1: WDG Memory Map

Offset	Register	Width	Access	Reset Value
0h	Service Code Register (SVCR)	32	RW	00000000h
4h	Control Register (CR)	32	RW	00000042h
8h	Lock Register (LR)	32	RW	00000001h
ch	Timer Overflow Value (TOVR)	32	RW	00000300h
10h	Window Value Register (WVR)	32	RW	00000000h
14h	Counter Current Value Register (CNTCVR)	32	RW	00000000h
18h	Interrupt Flag Register (INTF)	32	RW	00000000h

Yuntu Microelectronics

29.2.1.1 WDG SVCR Register

Watchdog Service Code Register

Offset: 0h

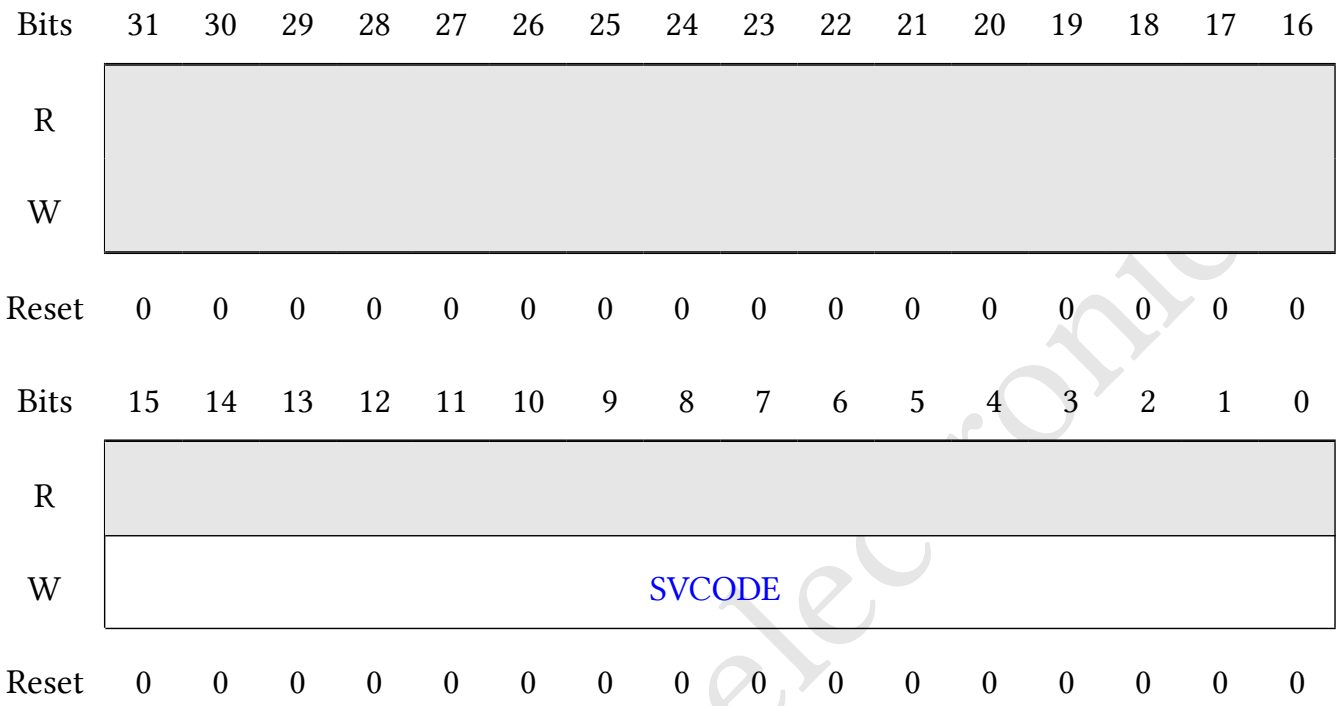


Table 29.2: WDG SVCR Register Description

Field	Function
15 - 0 SVCODE	**Watchdog Service Code** It can be used to service the watchdog or to clear the soft lock bit. Writing 0xB631 followed by 0xC278 will clear the soft lock bit, writing 0xA518 followed by 0xD826 to service the watchdog.

Yuntu Microelectronics

29.2.1.2 WDG CR Register

Watchdog Control Register

Offset: 4h

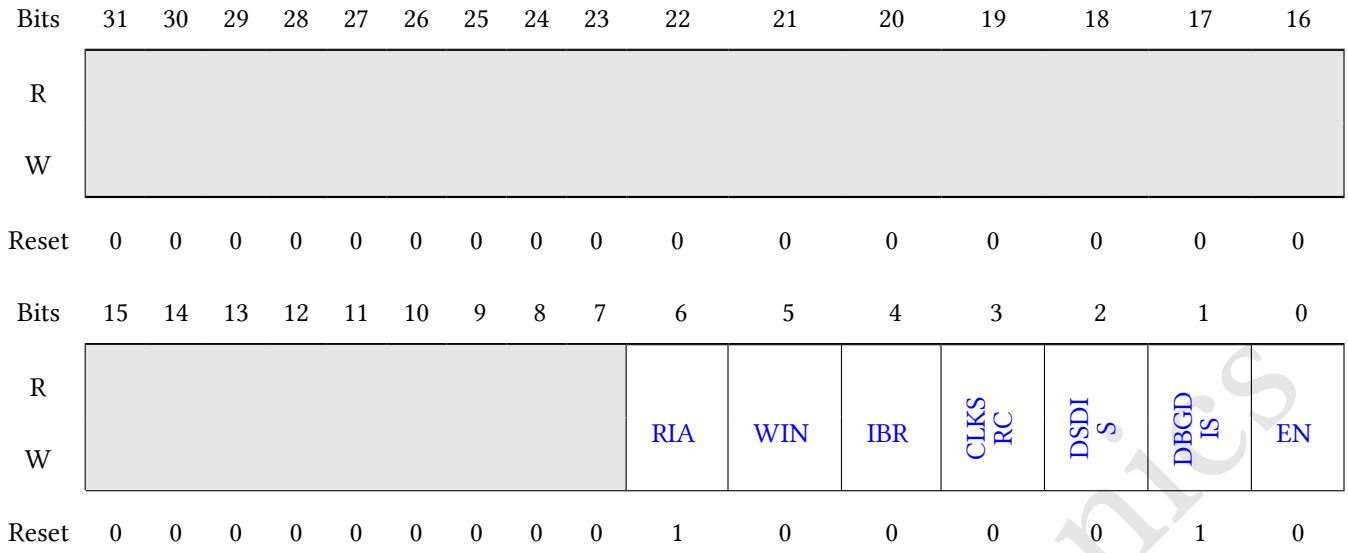


Table 29.3: WDG CR Register Description

Field	Function
6 RIA	**Reset Invalid Apb Enable** Enable reset for invalid APB access. 0b - Disable to generate reset by invalid APB access 1b - Enable to generate reset by invalid APB access
5 WIN	**Window Mode** Enable window mode. 0b - Disable window mode, watchdog service can be done at any time. 1b - Enable window mode, watchdog service is can only be done when timer value is less than the value of WIN register.
4 IBR	**Interrupt Before Reset** Enable interrupt before reset function 0b - Disable interrupt before reset. 1b - Enable interrupt before reset, when watchdog time-out, generate an interrupt first and when next time-out happens, generate reset.
3 CLKSRC	**Clock Source** 0b - Clock from SXOSC 1b - Clock from SIRC
2 DSDIS	**Enable In Deepsleep Mode** Enable or disable watchdog in deepsleep mode 0b - Enable watchdog in deepsleep mode. 1b - Disable watchdog in deepsleep mode.
1	**Disable In Debug Mode**

Yuntu Microelectronics

Field	Function
DBGDIS	Enable or disable watchdog in debug mode. 0b - Enable watchdog in debug mode. 1b - Disable watchdog in debug mode.
0 EN	**Module Enable** Enable watchdog module. 0b - Disable watchdog module 1b - Enable watchdog module

29.2.1.3 WDG LR Register

Watchdog Lock Register

Offset: 8h

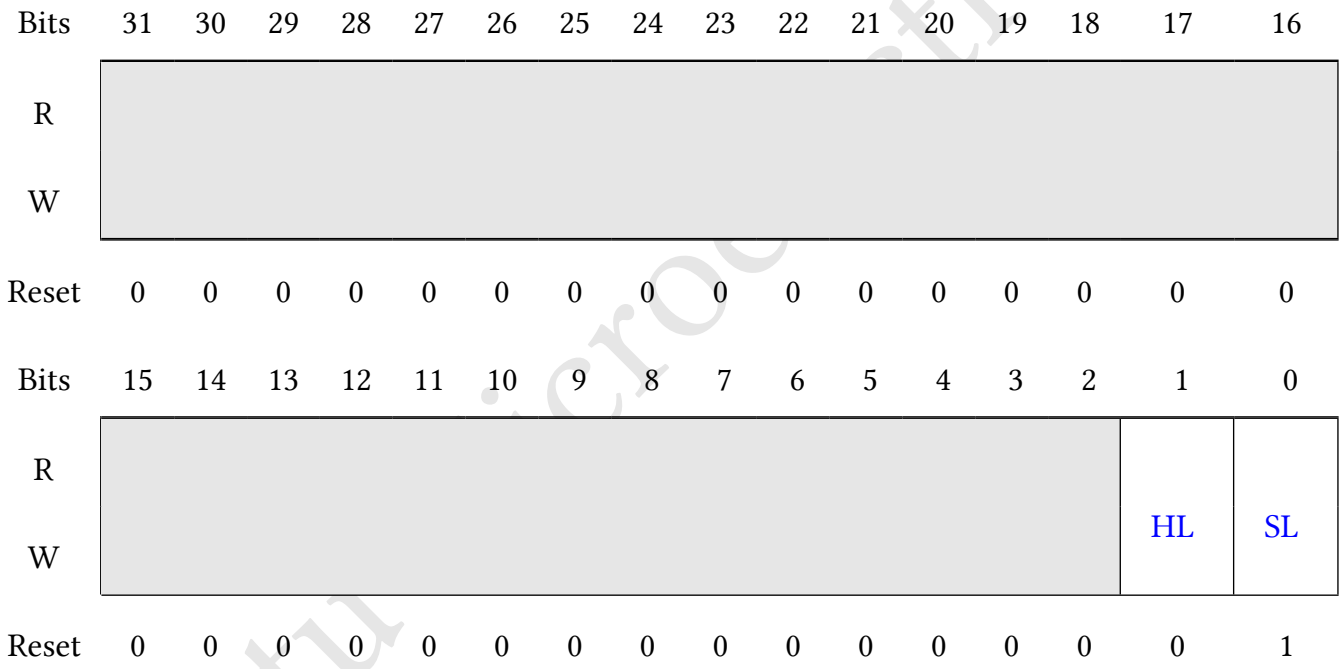


Table 29.4: WDG LR Register Description

Field	Function
1 HL	**Hard Lock** Hard lock, when this bit set, the CR, WIN, TO are read-only. once set, the bit can only be clear by reset.
0 SL	**Soft Lock** Soft lock, when this bit set, the CR, WIN, TO are read-only. this bit can be clear by writing a valid sequence to SVR register.

Yuntu Microelectronics

29.2.1.4 WDG TOVR Register

Watchdog Timer Overflow Value

Offset: ch

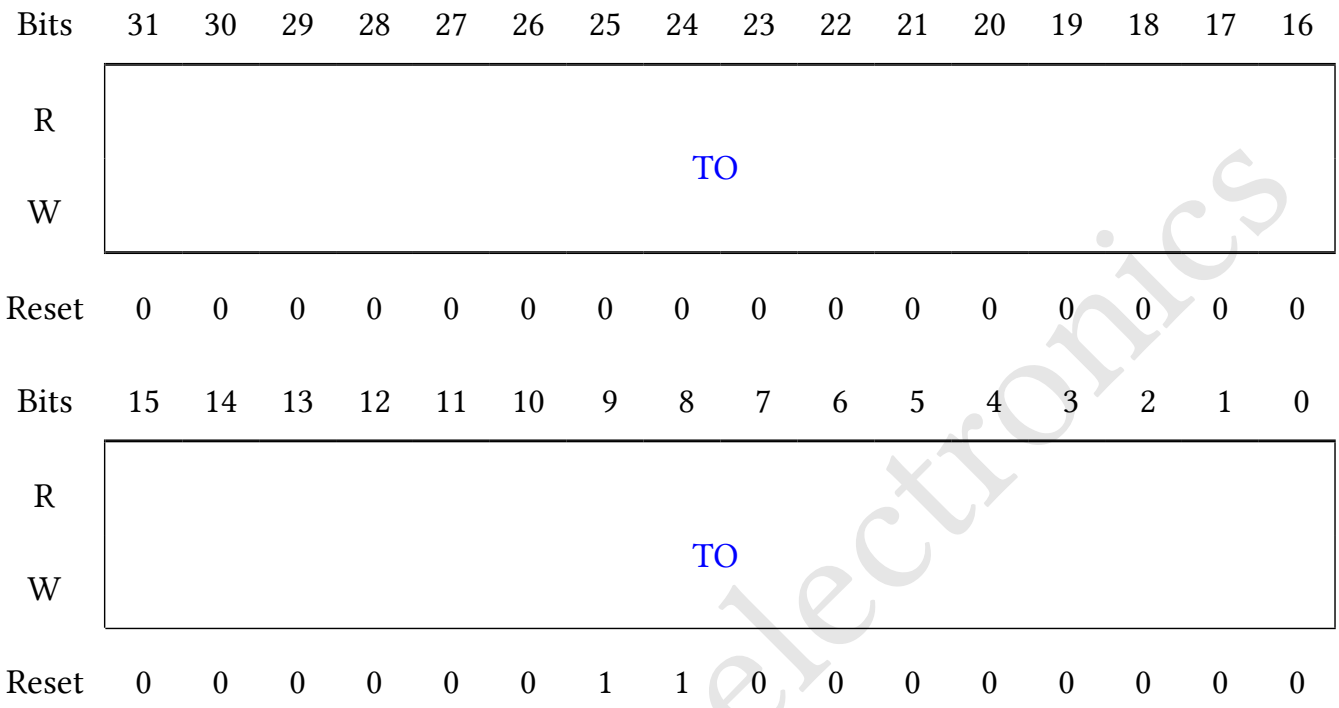


Table 29.5: WDG TOVR Register Description

Field	Function
31 - 0 TO	**Timer Overflow Value** A 32bit down counter uses this value to start count down. **NOTE**. This register can be configured to any value, it is crucial to verify the alignment between the actual WDG reset time and the TOVR[TO] register value prior to its configuration. For instance, when WDG clock source is chosen from a clock with the period of 500ns, and this register bit is set to a relatively low value such as 0x10, the WDG reset interval will be adjust to 16*0.5us = 8us upon enabling WDG. This implies that WDG must be serviced every 8us to prevent a WDG reset. Obviously if the core is excuting a complex application, this service interval may be impractically short. The method for calculating WDG reset time remains consistent when alternative clock sources are selected. **In general, it is advisable to avoid configuring this register with a value less than 0x100!**

29.2.1.5 WDG WVR Register

Offset: 10h

Yuntu Microelectronics

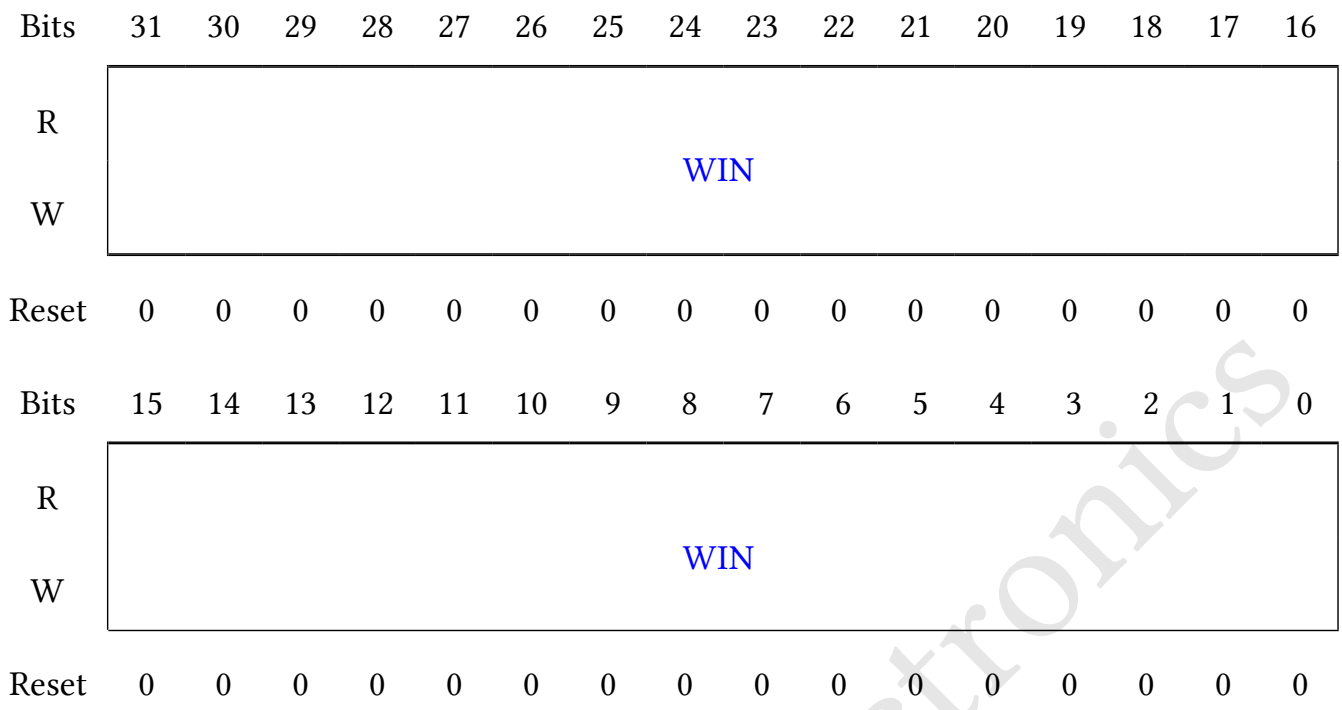


Table 29.6: WDG WVR Register Description

Field	Function
31 - 0 WIN	**Watchdog Window Value** When CR[WIN] is set, the window mode is enabled, the service sequence can only be written when the timer value is less than the value of this register.

Yuntu Microelectronics

29.2.1.6 WDG CNTCVR Register

Watchdog Counter Current Value Register

Offset: 14h

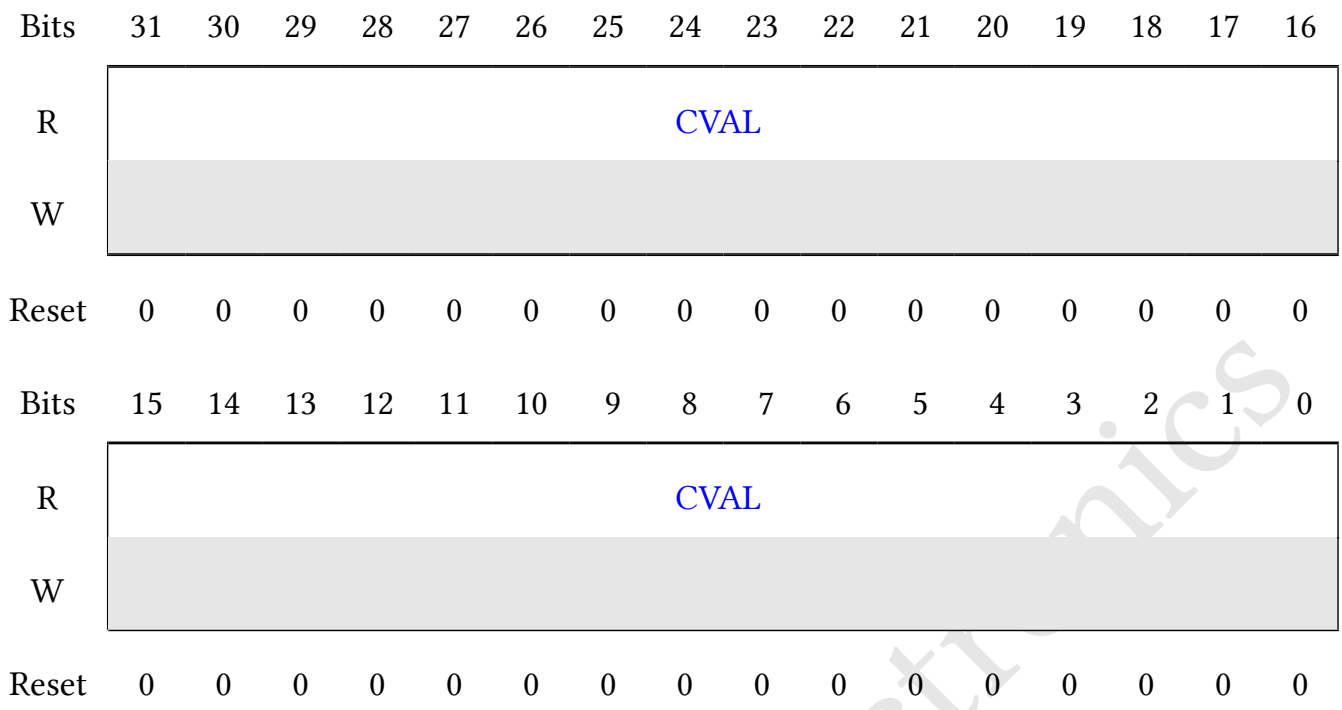


Table 29.7: WDG CNTCVR Register Description

Field	Function
31 - 0 CVAL	**Counter Current Value** Watchdog counter current value.

29.2.1.7 WDG INTF Register

Watchdog Interrupt Flag register

Offset: 18h

Yuntu Microelectronics

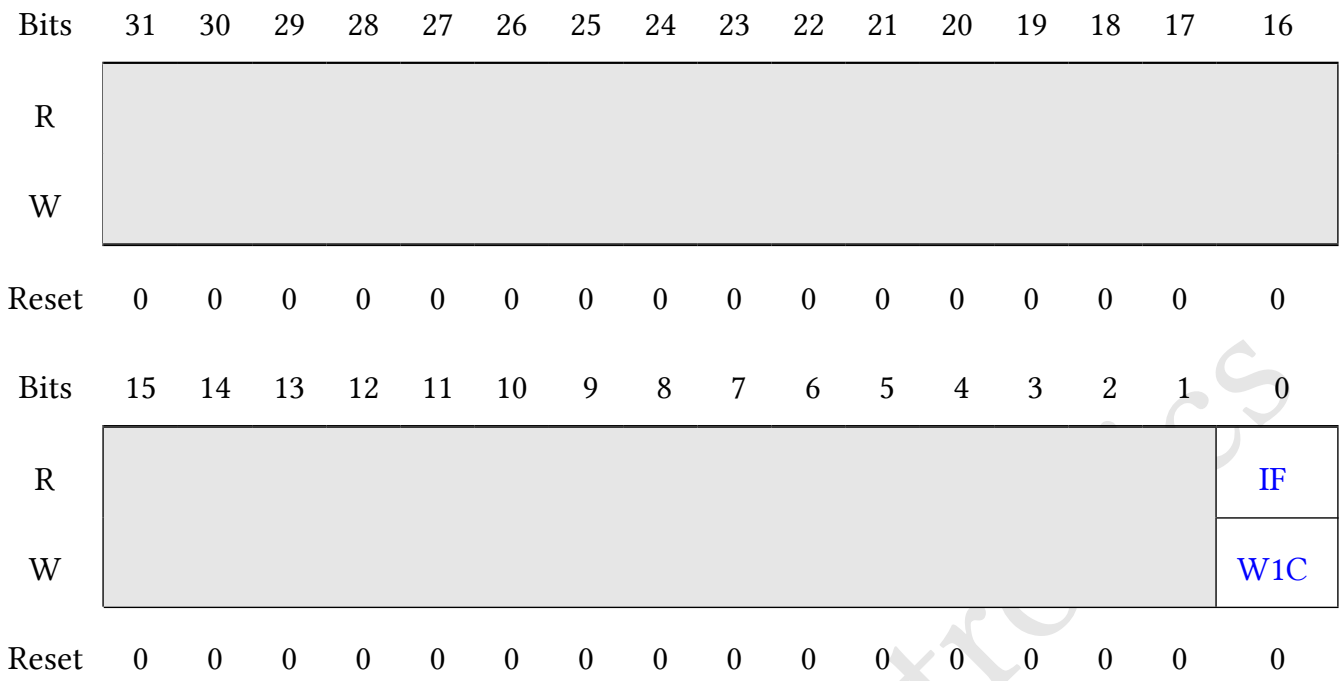


Table 29.8: WDG INTF Register Description

Field	Function
0 IF	**Interrupt Flag** Watchdog interrupt flag. Write 1 to this field to clear the flag and interrupt. 0b - No watchdog interrupt is pending 1b - Watchdog timer time-out interrupt is pending

29.3 Functional Description

29.3.1 General

This section gives detailed information on the internal operation of the module. The WDG is a 32-bit window watchdog that enables the system to recover from situations such as:

- Software trapped in a loop
- A bus transaction failing to terminate

In regular operation, WDG requires periodic execution of a watchdog servicing operation. The servicing operation resets the timer to a specified timeout period. If this servicing action does not occur before the timer expires, WDG generates an interrupt or a hardware reset request.

You can configure WDG to generate a reset request or an interrupt on the initial timeout. WDG generates a reset request if a second consecutive timeout occurs before writing the service sequence for the first timeout.

29.3.1.1 Operation Modes

WDG supports the chip modes of operation as follows:

Yuntu Microelectronics

- Normal(Active): When WDG is enabled(CR[EN]=1), the WDG timer counts down continuously.
- Debug: If CR[DBGDIS] = 1, WDG stops the timer; otherwise, the timer continues to run.
- Sleep: Timer continues to run according to the clock selection between SIRC and SXOSC.
- DeepSleep: Timer continues to run according to the clock selection between SIRC and SXOSC.
- Standby: Timer continues to run according to the clock selection between SIRC and SXOSC.

As for How to enable the SIRC or SXOSC in corresponding power mode, please see Power modes table for more information.

29.3.2 Interrupts

WDG module just supports one interrupt. Please refer to the table of interrupt vectors. WDG interrupt can be enabled by setting the CR[IBR] bit to 1. The WDG interrupt flag(INTF[IF]) is set to 1 while timeout occurs, and the interrupt flag can be cleared by setting INTF[IF] to 1.

Yuntu Microelectronics

Chapter 30

ECC Management Unit (EMU)

30.1 Chip-specific EMU Information

ECC Management Unit(EMU) have two main functions: Error Injection and Error Report.

There is just one channel that could be injected and/or reported, called SRAM. This channel have 32-bits read data and 7-bits ecc check-bit.

30.2 Introduction

30.2.1 Error Injection

Error injection is mainly used for diagnostic purposes. It provides a method for diagnostic coverage of SRAM.

Error injection uses two-stage enable mechanism: EIGCR[GEN] which enables error injection globally and EICHCR[CH0EN] which enables error injection functionality for a particular channel. This two-stage enable mechanism provides protection against accidental enabling and reconfiguration of the error injection function.

Error injection provides support for inducing single-bit or multi-bit inversions on read data when accessing SRAM.

NOTE: more than two bits inversions may result in undefined behavior.

30.2.2 Error Report

Error report provides information and optionally interrupt notification on SRAM error events associated with ECC (Error Correction Code) and parity.

Error report provides count registers which count all correctable error so far.

30.3 Register Definition

30.3.1 Programming Model Access

The EMU provides a programming model mapped to an on-platform peripheral slot.

- All system bus masters can access the programming model:

Yuntu Microelectronics

  - Only in supervisor mode
  - Using only 32-bit (word) accesses
- Any of the following attempted references to the programming model generates an bus error termination:
  - In user mode
  - Using non-32-bit access sizes
  - To undefined (reserved) addresses

30.3.2 Error Injection Registers

- Error injection have two-stage enable mechanism: EIGCR[GEN] and EICHCR[CH0EN].
- Error injection have two array registers for SRAM.
- Specifies an addr that defines error will be injected to all read access or an read access(EICHD0_ADDR).
- Specifies a checkbit mask that defines which bits of the checkbit bus from target SRAM are inverted on a read access.
- Specifies a data mask that defines which bits of the data bus from target SRAM are inverted on a read access.

30.3.3 Error Report Registers

- Specifies a register for error status and a register for error interrupt enable.
- Error report have one array registers for SRAM.
- Error report will keep the last recorded ECC event(ADDR, SYND).
- Error report will count the correctable error so far.

30.3.4 EMU Memory Map

EMU0 base address 40018000h

Table 30.1: EMU Memory Map

Offset	Register	Width	Access	Reset Value
0h	Error Injection Global Control Register (EIGCR)	32	RW	00000000h
4h	Error Injection Channel Control Register (EICHCR)	32	RW	00000000h
20h	Channel 0 Error Injection Address Register (EICHD0_ADDR)	32	RW	00000000h
24h	Error Injection CheckBit Mask Register (EICHD0_CHKBIT)	32	RW	00000000h
28h	Error Injection DATA Mask Register (EICHD0_DATA)	32	RW	00000000h
100h	Error Report Interrupt Enable Register (ERIE)	32	RW	00000000h
104h	Error Report Interrupt Flag Register (ERIF)	32	RW	00000000h
220h	Error Report Address Register (ERINFO0_ADDR)	32	RO	00000000h
224h	Error Report Syndrome Register (ERINFO0_SYND)	32	RO	00000000h
228h	Correctable Error Count Register (ERINFO0_CNT)	32	RW	00000000h

Yuntu Microelectronics

30.3.4.1 EMU EIGCR Register

Offset: 0h

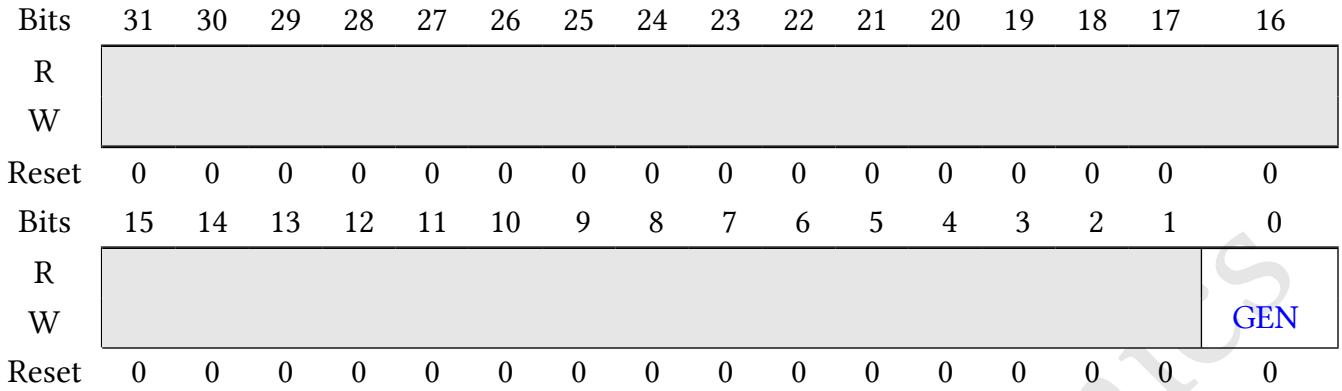
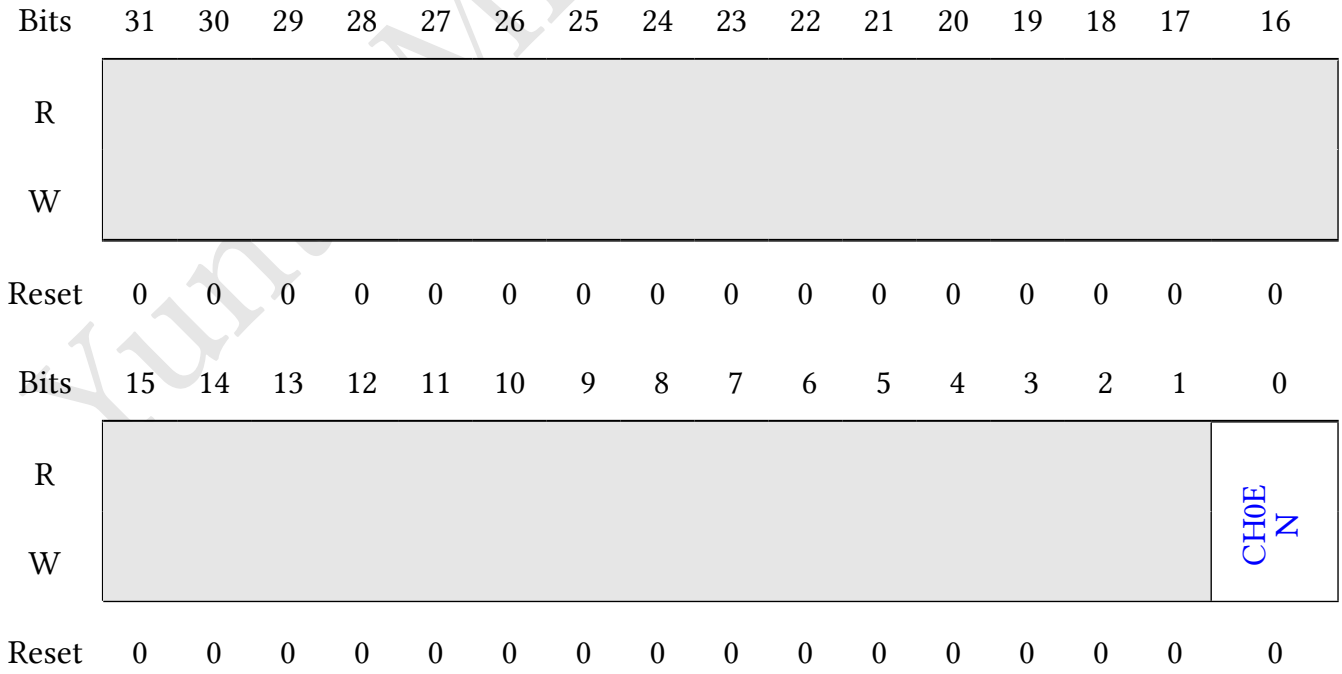


Table 30.2: EMU EIGCR Register Description

Field	Function
0 GEN	**Global Error Injection Enable** This bit will enable or disable all error injection functions. 0b - Global error injection is disabled 1b - Global error injection is enabled

30.3.4.2 EMU EICHCR Register

Offset: 4h



Yuntu Microelectronics

Table 30.3: EMU EICHCR Register Description

Field	Function
0 CH0EN	**Channel 0 Error Injection Enable** This field enable channel 0 error injection function. Enable channel 0 error injection function should enable this field and EIGCR[GEN]. Any write to EICHD0_ADDR, EICHD0_CHKBIT, EICHD0_DATA0 register will clear this field. 0b - Channel 0 error injection is disabled 1b - Channel 0 error injection is enabled

30.3.4.3 EMU EICHD_ADDR Register

Offset: 20h + n * 10h

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

Table 30.4: EMU EICHD_ADDR Register Description

Field	Function
31 - 0 ADDR	**Channel 0 Error Injection Address** This field defines which addr will be injected error when read access. when the ADDR is not 0, error will be injected according to this field's value. if ADDR is 0, Error will be injected at all read access.

30.3.4.4 EMU EICHD_CHKBIT Register

Offset: 24h + n * 10h

Yuntu Microelectronics

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

Table 30.5: EMU EICHD_CHKBIT Register Description

Field	Function
6 - 0 CHKBIT	**Error Injection CheckBit Mask** This field defines a bit-mapped mask that specifies whether the corresponding bit of the checkbit bus from the target SRAM should be inverted or remain unmodified. 0000000b (0x00) - The corresponding bit of the checkbit bus remains unmodified. 0000001b (0x01) - The corresponding bit of the checkbit bus is inverted.

30.3.4.5 EMU EICHD_DATA Register

Offset: 28h + n * 10h

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

Table 30.6: EMU EICHD_DATA Register Description

Field	Function
31 - 0 DATA	**Data Mask** This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target SRAM should be inverted or remain unmodified. 0b - The corresponding bit of the checkbit bus remains unmodified. 1b - The corresponding bit of the checkbit bus is inverted.

Yuntu Microelectronics

30.3.4.6 EMU ERIE Register

Offset: 100h

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

Table 30.7: EMU ERIE Register Description

Field	Function
16 NCIE	**Non-Correctable Interrupt Enable** SRAM Non-Correctable error interrupt enable. 0b - SRAM Non-Correctable Error interrupt is disabled 1b - SRAM Non-Correctable Error interrupt is enabled
0 SCIE	**Single Correctable Interrupt Enable** SRAM Single correctable error interrupt enable. 0b - SRAM Single Correctable Error interrupt is disabled 1b - SRAM Single Correctable Error interrupt is enabled

30.3.4.7 EMU ERIF Register

Offset: 104h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																NCIF
W																W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																SCIF
W																W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Yuntu Microelectronics

Table 30.8: EMU ERIF Register Description

Field	Function
16 NCIF	**Non-Correctable Interrupt Flag** SRAM Non-Correctable error interrupt flag. 0b - SRAM Non-Correctable Error interrupt not detected 1b - SRAM Non-Correctable Error interrupt detected
0 SCIF	**Single Correctable Interrupt Flag** SRAM Single Correctable error interrupt flag. 0b - SRAM Single Correctable Error interrupt not detected 1b - SRAM Single Correctable Error interrupt detected

30.3.4.8 EMU ERINFO_ADDR Register

Offset: 220h + n * 10h

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

Table 30.9: EMU ERINFO_ADDR Register Description

Field	Function
31 - 0 ADDR	**Error Report Address 0** This field contains the faulting system address of the last recorded ECC event on SRAM.

30.3.4.9 EMU ERINFO_SYND Register

Offset: 224h + n * 10h

Yuntu Microelectronics

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R										SYNDROME						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 30.10: EMU ERINFO_SYND Register Description

Field	Function
6 - 0 SYNDROME	**Error Report Syndrome 0** This field contains the syndrome of the last recorded ECC event on SRAM.

30.3.4.10 EMU ERINFO_CNT Register

Offset: 228h + n * 10h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W									CNT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 30.11: EMU ERINFO_CNT Register Description

Field	Function
7 - 0 CNT	**SRAM Correctable Error Count** For each correctable error event, the EMU increments this field's error count value until the counter reaches its maximum value 0xFF. CNT will not change once it reaches 0xFF and will not wrap even if error occurs. Read this field to determine the correctable error count value so far. Write zero to this field to reset the counter. Writing non-zero (or partial zero) values has no effect.

Yuntu Microelectronics

30.4 Functional Description

30.4.1 General

This section gives detailed information on the internal operation of the module.

30.4.2 Error Injection

- Error data could be injected to read access. (EICH0_ADDR/CHKBIT/DATA registers for SRAM).
- Error data could be injected to overall read access or a particular read access. when EICH0_ADDR is cleared 0, error data will be injected to all read access, otherwise, the error data will be injected to a particular read access.
- Injection method: when read access and addr match, the bus read data will be the XOR of SRAM read data(data and its ecc check-bit) and EICH0_CHKBIT/DATA. EMU support signle_bit error and multi_bit error, an attempt to invert more than 2 bits in one operation might result in undefined behavior.
- Use two-stage enable mechanism providing protection. After config channel's EICH0_ADDR, EICH0_CHKBIT and EICH0_DATA, To properly enable the error injection mechanism, please set channel enable and then set global enable.

30.4.3 Error Report

- Error Report could repert error when read SRAM(singel-bit or multi-bit error). This channel has an array registers to state the error(ERADDR0/ERSYND0/ERCNT0).
- When detected an error(single-bit error or multi-bit error), the register will keep its address and syndrome. if the error is single-bit error, the ERCNT0 will increase(until the count reaches its max value).
- Error report provides an option to generate an interrupt, upon the report of an error and the corresponding interrupt enable bit set.
- To clear error flag, write 1 to the interrupt flag. To clear the correctable error count, write 0 to the count register.

Yuntu Microelectronics

Part X

Appendix

Yuntu Microelectronics

Appendix A

A.1 General Changes

- Editorial Changes
  - Each new release of this document includes editorial improvements such as:
    - Spelling and grammar
    - Punctuation
    - Capitalization
    - Formatting
    - Navigation
    - Acronyms
    - Voices
    - Tense
- Update details for various products of YTM32B1LE0x throughout the reference manual.

Yuntu Microelectronics

Appendix B

B.1 Revision History

Table B.1: Reference Manual Revision History

Rev.No.	Release Date
1.0	2022/10/18
1.1	2023/1/11
1.2	2023/2/24
1.3	2023/3/13
1.4	2023/5/24
1.5	2023/7/21
1.6	2023/12/8
1.7	2024/2/22
1.8	2025/8/29

Yuntu Microelectronics

B.2 Release Notes

B.2.1 Introduction

- In section “Overview” and “Feature Summary”, updated the Min of supply voltage from 2.7V to 2.97V.
- In section “Feature Summary”, added the information about ASIL/AEC-Q100/RoHs.

B.2.2 Memory Map

- No substantial content changes.

B.2.3 Embedded Flash Module (EFM)

- In subsection “Memory Map”, added the UID information to the table.
- In subsection “EFM Memory Map”, added the register of “CTRL_OVRD”.
- In subsubsection “EFM CTRL Register”, added the information of bit24.

B.2.4 Pinmux

- No substantial content changes.

B.2.5 General Purpose Input/Output (GPIO)

- In subsubsection “GPIO PCR Register”, modified the note in bit of “DFE”.

B.2.6 Port Control (PCTRL)

- No substantial content changes.

Yuntu Microelectronics

B.2.7 CM0+ Overview

- No substantial content changes.

B.2.8 Direct Memory Access (DMA)

- In subsubsection “DMA CTRL Register”, added the description of bit17.

B.2.9 Divide and Square Root (DIVSQRT)

- No substantial content changes.

B.2.10 Trigger Multiplexer Module (TMU)

- No substantial content changes.

B.2.11 Chip Integration Module (CIM)

- In section “Chip-specific CIM Information”, updated the UID address.
- In subsubsection “CIM CHIPINFO Register”, updated the description in bits of “SRAMS”.

B.2.12 System Clock Unit (SCU)

- In subsubsection “SCU FXOSC_CTRL Register”, updated the bit name and information of 8-13.
- In subsubsection “SCU CMUCMP_HIGH Register” and “SCU CMUCMP_LOW Register”, added the formula information of bits.

Yuntu Microelectronics

B.2.13 Power Control Unit (PCU)

- In subsubsection “PCU BORCR Register”, added two notes to the two bits function description.
- In subsubsection “PCU LVDCR Register”, removed 1.5V LVD information of bit8/9.
- In subsubsection “PCU CSR Register”, added one note to the “ESCFG” bit function description.
- In subsection “Brown-out”, added one note about BOR function.
- In subsection “Low Voltage Detect”, removed 1.5V LVD information.

B.2.14 IP Clock Control (IPC)

- No substantial content changes.

B.2.15 Reset Control Unit (RCU)

- No substantial content changes.

B.2.16 Power Mode

- Added the new chapter.

B.2.17 Flexible Controller Area Network (FlexCAN)

- In subsubsection “CAN WMB_WMBn_CS Register”, “CAN WMB_WMBn_ID Register”, “CAN WMB_WMBn_D03 Register”, and “CAN WMB_WMBn_D47 Register”, updated the value of offset.
- In subsubsection “CAN ERFCR Register”, corrected the function description in bit of “ERFEN”.

B.2.18 Universal Asynchronous Receiver Transmitter (UART)

- In subsubsection “UART BAUD Register”, updated the function description in bits of “OSRVAL” and “DIV”.

Yuntu Microelectronics

B.2.19 Serial Port Interface (SPI)

- No substantial content changes.

B.2.20 Inter-Integrated Circuit (I2C)

- No substantial content changes.

B.2.21 Periodic Timer (pTMR)

- No substantial content changes.

B.2.22 Low Power Timer (lpTMR)

- No substantial content changes.

B.2.23 Enhanced Timer (eTMR)

- No substantial content changes.

B.2.24 Real Time Clock Module (RTC)

- In subsection “Operation Mode”, removed the information about Powerdown mode.

B.2.25 Analog-to- Digital Converter (ADC)

- In subsubsection “ADC CFG1 Register”, updated the function description in bits of “PRS” and “STCNT”.

Yuntu Microelectronics

B.2.26 Analog Comparator (ACMP)

- No substantial content changes.

B.2.27 Temperature Sensor (TEMPSENSOR)

- In subsection “Introduction”, corrected the value of K in the formula.

B.2.28 Cyclic Redundancy Check (CRC)

- No substantial content changes.

B.2.29 Watch Dog (WDG)

- In subsection“WDG TOVR Register”, updated the description of bits.
- In subsection “Operation Modes”, removed the information about Powerdown mode.

B.2.30 ECC Management Unit (EMU)

- No substantial content changes.

Yuntu Microelectronics

Copyright and Contact

Information in this document is provided solely to enable system and software implementers to use Yuntu Microelectronics products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. Yuntu Microelectronics reserves the right to change the detail specifications as may be required to permit improvements in the design of its products.

Yuntu Microelectronics makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Yuntu Microelectronics assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Yuntu Microelectronics data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. Yuntu Microelectronics does not convey any license under its patent rights nor the rights of others. Yuntu Microelectronics sells products pursuant to standard terms and conditions of sale, which can be found at the following address: www.ytmicro.com

While Yuntu Microelectronics has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer’s applications and products, and Yuntu Microelectronics accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

Yuntu Microelectronics reserves the right to change the production detail specifications as may be required to permit improvements in the design of its products.

©2020 - 2025 Jiangsu Yuntu Microelectronics Co., LTD.

How to reach us:
Home Page: www.ytmicro.com

Yuntu Microelectronics