NuMicro® Family

Based on Arm® Cortex®-M23

M2354 Series
Technical Reference Manual

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

TABLE OF CONTENTS

1 GENERAL DESCRIPTION 31
2 FEATURES 33
3 PARTS INFORMATION 46
3.1 Package Type 46
3.2 M2354 Series Selection Guide 47
3.3 M2354 Series Selection Code 48
4 PIN CONFIGURATION 49
4.1 Pin Configuration 49
4.1.1 M2354 Pin Diagram 49
4.1.2 M2354 Multi-Function Pin Diagram 52
4.2 M2354 Pin Mapping 65
4.3 M2354 Pin Functional Description 69
5 BLOCK DIAGRAM 95
6 FUNCTIONAL DESCRIPTION 96
6.1 Arm® Cortex®-M23 Core 96
6.2 System Manager 98
6.2.1 Overview 98
6.2.2 Reset 98
6.2.3 Power Modes and Wake-up Sources 104
6.2.4 System Power Distribution 109
6.2.5 Bus Matrix 111
6.2.6 System Memory Map 111
6.2.7 Implementation Defined Attribution Unit (IDAU) 115
6.2.8 SRAM Memory Organization 117
6.2.9 Auto Trim 121
6.2.10 Register Lock Control 122
6.2.11 Register Map 125
6.2.12 Register Description 128
6.2.13 System Timer (SysTick) 194
6.2.14 Nested Vectored Interrupt Controller (NVIC) 198
6.2.15 System Control Register 239
6.2.16 Security Attribution Unit (SAU) 249
6.3 Clock Controller 255
6.3.1 Overview 255
6.3.2 Clock Generator 258
6.3.3 System Clock and SysTick Clock 260

6.3.4 Peripherals Clock 262
6.3.5 Power-down Mode Clock 263
6.3.6 Clock Output 263
6.3.7 Share Registers 264
6.3.8 Register Map 265
6.3.9 Register Description 267
6.4 Security Configuration Unit (SCU) 319
6.4.1 Overview 319
6.4.2 Features 319
6.4.3 Block Diagram 320
6.4.4 Functional Description 320
6.4.5 Register Map 336
6.4.6 Register Description 340
6.5 Arm® TrustZone® 420
6.5.1 Address Space Partition 421
6.5.2 Security Attribute Configuration 423
6.5.3 System Address Map and Access Scheme 424
6.6 Flash Memory Controller (FMC) 427
6.6.1 Overview 427
6.6.2 Features 427
6.6.3 Block Diagram 427
6.6.4 Functional Description 430
6.6.5 Register Map 474
6.6.6 Register Description 476
6.7 General Purpose I/O (GPIO) 515
6.7.1 Overview 515
6.7.2 Features 515
6.7.3 Block Diagram 516
6.7.4 Basic Configuration 517
6.7.5 Functional Description 518
6.7.6 Register Map 522
6.7.7 Register Description 527
6.8 PDMA Controller (PDMA) 544
6.8.1 Overview 544
6.8.2 Features 544
6.8.3 Block Diagram 544
6.8.4 Basic Configuration 545
6.8.5 Functional Description 545
6.8.6 Register Map 552
6.8.7 Register Description 554

6.9 Timer Controller (TMR) 587
6.9.1 Overview 587
6.9.2 Features 587
6.9.3 Block Diagram 589
6.9.4 Basic Configuration 595
6.9.5 Timer Functional Description 597
6.9.6 PWM Functional Description 603
6.9.7 Register Map 625
6.9.8 Register Description 632
6.10 Watchdog Timer (WDT) 679
6.10.1 Overview 679
6.10.2 Features 679
6.10.3 Block Diagram 679
6.10.4 Basic Configuration 680
6.10.5 Functional Description 680
6.10.6 Register Map 683
6.10.7 Register Description 684
6.11 Extra Watchdog Timer (EWDT) 688
6.11.1 Overview 688
6.11.2 Features 688
6.11.3 Block Diagram 688
6.11.4 Basic Configuration 689
6.11.5 Functional Description 689
6.11.6 Register Map 692
6.11.7 Register Description 693
6.12 Window Watchdog Timer (WWDT) 697
6.12.1 Overview 697
6.12.2 Features 697
6.12.3 Block Diagram 697
6.12.4 Basic Configuration 697
6.12.5 Functional Description 698
6.12.6 Register Map 702
6.12.7 Register Description 703
6.13 Extra Window Watchdog Timer (EWWDT) 708
6.13.1 Overview 708
6.13.2 Features 708
6.13.3 Block Diagram 708
6.13.4 Basic Configuration 708
6.13.5 Functional Description 709
6.13.6 Register Map 713

6.13.7 Register Description 714
6.14 Real Time Clock (RTC) 719
6.14.1 Overview 719
6.14.2 Features 719
6.14.3 Block Diagram 719
6.14.4 Basic Configuration 720
6.14.5 Functional Description 721
6.14.6 Register Map 728
6.14.7 Register Description 730
6.15 EPWM Generator and Capture Timer (EPWM) 770
6.15.1 Overview 770
6.15.2 Features 770
6.15.3 Block Diagram 772
6.15.4 Basic Configuration 776
6.15.5 Functional Description 778
6.15.6 Register Map 817
6.15.7 Register Description 824
6.16 Basic PWM Generator and Capture Timer (BPWM) 901
6.16.1 Overview 901
6.16.2 Features 901
6.16.3 Block Diagram 902
6.16.4 Basic Configuration 904
6.16.5 Functional Description 906
6.16.6 Register Map 921
6.16.7 Register Description 924
6.17 Quadrature Encoder Interface (QEI) 957
6.17.1 Overview 957
6.17.2 Features 957
6.17.3 Block Diagram 957
6.17.4 Basic Configuration 958
6.17.5 Functional Description 959
6.17.6 Register Map 967
6.17.7 Register Description 968
6.18 Enhanced Input Capture Timer (ECAP) 978
6.18.1 Overview 978
6.18.2 Features 978
6.18.3 Block Diagram 978
6.18.4 Basic Configuration 979
6.18.5 Functional Description 980
6.18.6 Register Map 985

6.18.7 Register Description 986
6.19 UART Interface Controller (UART) 997
6.19.1 Overview 997
6.19.2 Features 997
6.19.3 Block Diagram 998
6.19.4 Basic Configuration 1001
6.19.5 Functional Description 1006
6.19.6 Register Map 1031
6.19.7 Register Description 1033
6.20 Smart Card Host Interface (SC) 1069
6.20.1 Overview 1069
6.20.2 Features 1069
6.20.3 Block Diagram 1069
6.20.4 Basic Configuration 1071
6.20.5 Functional Description 1073
6.20.6 Register Map 1086
6.20.7 Register Description 1087
6.21 I²S Controller (I²S) 1112
6.21.1 Overview 1112
6.21.2 Features 1112
6.21.3 Block Diagram 1112
6.21.4 Basic Configuration 1112
6.21.5 Functional Description 1113
6.21.6 Register Map 1125
6.21.7 Register Description 1126
6.22 Serial Peripheral Interface (SPI) 1144
6.22.1 Overview 1144
6.22.2 Features 1144
6.22.3 Block Diagram 1144
6.22.4 Basic Configuration 1145
6.22.5 Functional Description 1149
6.22.6 Timing Diagram 1168
6.22.7 Programming Examples 1169
6.22.8 Register Map 1172
6.22.9 Register Description 1173
6.23 Quad Serial Peripheral Interface (QSPI) 1195
6.23.1 Overview 1195
6.23.2 Features 1195
6.23.3 Block Diagram 1195
6.23.4 Basic Configuration 1196

6.23.5 Functional Description 1197
6.23.6 Timing Diagram 1216
6.23.7 Programming Examples 1217
6.23.8 Register Map 1220
6.23.9 Register Description 1221
6.24 USCI - Universal Serial Control Interface Controller (USCI) 1236
6.24.1 Overview 1236
6.24.2 Features 1236
6.24.3 Block Diagram 1236
6.24.4 Functional Description 1236
6.25 I²C Serial Interface Controller (I²C) 1247
6.25.1 Overview 1247
6.25.2 Features 1247
6.25.3 Block Diagram 1248
6.25.4 Basic Configuration 1248
6.25.5 Functional Description 1250
6.25.6 Register Map 1273
6.25.7 Register Description 1274
6.26 USCI – UART Mode 1296
6.26.1 Overview 1296
6.26.2 Features 1296
6.26.3 Block Diagram 1296
6.26.4 Basic Configuration 1297
6.26.5 Functional Description 1298
6.26.6 Register Map 1307
6.26.7 Register Description 1308
6.27 USCI - SPI Mode 1331
6.27.1 Overview 1331
6.27.2 Features 1331
6.27.3 Block Diagram 1332
6.27.4 Basic Configuration 1332
6.27.5 Functional Description 1334
6.27.6 Register Map 1347
6.27.7 Register Description 1348
6.28 USCI - I²C Mode 1371
6.28.1 Overview 1371
6.28.2 Features 1371
6.28.3 Block Diagram 1372
6.28.4 Basic Configuration 1372
6.28.5 Functional Description 1374

6.28.6 Register Map 1393
6.28.7 Register Description 1394
6.29 Controller Area Network (CAN) 1413
6.29.1 Overview 1413
6.29.2 Features 1413
6.29.3 Block Diagram 1413
6.29.4 Basic Configuration 1414
6.29.5 Functional Description 1415
6.29.6 Test Mode 1416
6.29.7 CAN Communications 1418
6.29.8 Register Map 1436
6.29.9 Register Description 1441
6.30 Secure Digital Host Controller (SDH) 1476
6.30.1 Overview 1476
6.30.2 Features 1476
6.30.3 Block Diagram 1477
6.30.4 Basic Configuration 1477
6.30.5 Functional Description 1478
6.30.6 Register Map 1480
6.30.7 Register Description 1481
6.31 External Bus Interface (EBI) 1500
6.31.1 Overview 1500
6.31.2 Features 1500
6.31.3 Block Diagram 1501
6.31.4 Basic Configuration 1501
6.31.5 Functional Description 1503
6.31.6 Register Map 1513
6.31.7 Register Description 1514
6.32 USB 1.1 Device Controller (USBD) 1518
6.32.1 Overview 1518
6.32.2 Features 1518
6.32.3 Block Diagram 1519
6.32.4 Basic Configuration 1519
6.32.5 Functional Description 1519
6.32.6 Register Map 1525
6.32.7 Register Description 1528
6.33 USB 1.1 Host Controller (USBH) 1553
6.33.1 Overview 1553
6.33.2 Features 1553
6.33.3 Block Diagram 1554

6.33.4 Basic Configuration 1555
6.33.5 Functional Description 1555
6.33.6 Register Map 1557
6.33.7 Register Description 1558
6.34 USB On-The-Go (OTG) 1591
6.34.1 Overview 1591
6.34.2 Features 1591
6.34.3 Block Diagram 1591
6.34.4 Basic Configuration 1592
6.34.5 Functional Description 1592
6.34.6 Register Map 1596
6.34.7 Register Description 1597
6.35 CRC Controller (CRC) 1606
6.35.1 Overview 1606
6.35.2 Features 1606
6.35.3 Block Diagram 1606
6.35.4 Basic Configuration 1607
6.35.5 Functional Description 1607
6.35.6 Register Map 1609
6.35.7 Register Description 1610
6.36 Cryptographic Accelerator (CRYPTO) 1615
6.36.1 Overview 1615
6.36.2 Features 1615
6.36.3 Block Diagram 1616
6.36.4 Basic Configuration 1617
6.36.5 Functional Description 1617
6.36.6 Register Map 1656
6.36.7 Register Description 1667
6.37 Enhanced 12-bit Analog-to-Digital Converter (EADC) 1760
6.37.1 Overview 1760
6.37.2 Features 1760
6.37.3 Block Diagram 1761
6.37.4 Basic Configuration 1761
6.37.5 Functional Description 1762
6.37.6 Register Map 1777
6.37.7 Register Description 1780
6.38 True Random Number Generator (TRNG) 1812
6.38.1 Overview 1812
6.38.2 Features 1812
6.38.3 Block Diagram 1812

6.38.4 Basic Configuration 1812
6.38.5 Functional Description 1812
6.38.6 Register Map 1814
6.38.7 Register Description 1815
6.39 Key Store (KS) 1819
6.39.1 Overview 1819
6.39.2 Features 1819
6.39.3 Block Diagram 1820
6.39.4 Basic Configuration 1820
6.39.5 Functional Description 1820
6.39.6 Register Map 1828
6.39.7 Register Description 1829
6.40 LCD Controller 1841
6.40.1 Overview 1841
6.40.2 Features 1841
6.40.3 Block Diagram 1842
6.40.4 Basic Configuration 1842
6.40.5 Functional Description 1844
6.40.6 Register Map 1860
6.40.7 Register Description 1861
6.41 Tamper Controller (TC) 1874
6.41.1 Overview 1874
6.41.2 Features 1874
6.41.3 Block Diagram 1874
6.41.4 Basic Configuration 1875
6.41.5 Functional Description 1875
6.41.6 Register Map 1881
6.41.7 Register Description 1882
6.42 Digital to Analog Converter (DAC) 1910
6.42.1 Overview 1910
6.42.2 Features 1910
6.42.3 Block Diagram 1910
6.42.4 Basic Configuration 1911
6.42.5 Functional Description 1912
6.42.6 Register Map 1917
6.42.7 Register Description 1918
6.43 Analog Comparator Controller (ACMP) 1932
6.43.1 Overview 1932
6.43.2 Features 1932
6.43.3 Block Diagram 1933

6.43.4 Basic Configuration 1933
6.43.5 Functional Description 1934
6.43.6 Register Map 1939
6.43.7 Register Description 1940
6.44 Peripherals Interconnection 1947
6.44.1 Overview 1947
6.44.2 Peripherals Interconnect Matrix Table 1947
6.44.3 Functional Description 1948
7 APPLICATION CIRCUIT 1951
7.1 Power supply scheme with External V_{REF} 1951
7.2 Power supply scheme with internal V_{REF} 1952
7.3 Peripheral Application scheme 1953
8 ELECTRICAL CHARACTERISTIC 1954
9 PACKAGE DIMENSIONS 1955
9.1 LQFP 48 (7x7x1.4 mm^3 Footprint 2.0 mm) 1955
9.2 LQFP 64 (7x7x1.4 mm^3 Footprint 2.0 mm) 1956
9.3 LQFP 128 (14x14x1.4 mm^3 Footprint 2.0 mm) 1957
10 ABBREVIATIONS 1958
10.1 Abbreviations 1958
11 REVISION HISTORY 1960

LIST OF FIGURES

Figure 4.1-1 M2354 LQFP 48-pin Diagram 49
Figure 4.1-2 M2354 LQFP 64-pin Diagram 50
Figure 4.1-3 M2354 LQFP 128-pin Diagram 51
Figure 4.1-4 M2354LJFAE Multi-function Pin Diagram 52
Figure 4.1-5 M2354SJFAE Multi-function Pin Diagram 55
Figure 4.1-6 M2354KJFAE Multi-function Pin Diagram 59
Figure 5-1 M2354 Block Diagram 95
Figure 6.1-1 Cortex®-M23 Block Diagram 96
Figure 6.2-1 System Reset Sources 99
Figure 6.2-2 nRESET Reset Waveform 101
Figure 6.2-3 Power-on Reset (POR) Waveform 102
Figure 6.2-4 Low Voltage Reset (LVR) Waveform 102
Figure 6.2-5 Brown-out Detector (BOD) Waveform 103
Figure 6.2-6 Power Mode State Machine 106
Figure 6.2-7 Power Distribution Diagram 110
Figure 6.2-8 IDAU Memory Map 116
Figure 6.2-9 IDAU Block Diagram 117
Figure 6.2-10 SRAM Block Diagram 118
Figure 6.2-11 SRAM Memory Organization 118
Figure 6.2-12 SRAM Marco Organization 120
Figure 6.3-1 Clock Generator Global View Diagram (1/3) 256
Figure 6.3-2 Clock Generator Global View Diagram (2/3) 257
Figure 6.3-3 Clock Generator Global View Diagram (3/3) 258
Figure 6.3-4 Clock Generator Block Diagram 259
Figure 6.3-5 System Clock Block Diagram 261
Figure 6.3-6 HXT Stop Protect Procedure 262
Figure 6.3-7 SysTick Clock Control Block Diagram 262
Figure 6.3-8 Clock Output Block Diagram 264
Figure 6.4-1 SCU Block Diagram 320
Figure 6.4-2 Security and Privileged Violation Signal Block Diagram 325
Figure 6.4-3 Firmware Version Counter Block Diagram 327
Figure 6.4-4 FVC State Flow 328
Figure 6.4-5 Block Diagram of Debug Protection Mechanism Unit 329
Figure 6.4-6 DPM Main State 330
Figure 6.4-7 Debug Authority and Debug State Flow 333

Figure 6.4-8 Product Life-cycle Manager Block Diagram 335
Figure 6.5-1 Secure World View and Non-secure World View on a Chip 420
Figure 6.5-2 The 4 GB Memory Map Divided Into Secure and Non-secure Regions by IDAU 422
Figure 6.5-3 Typical Setting of SAU 423
Figure 6.5-4 Example of SRAM Divided Into Secure Block and Non-secure Block 425
Figure 6.5-5 Checking Point of Accesses 426
Figure 6.6-1 Flash Memory Controller Block Diagram 428
Figure 6.6-2 APROM Examples (512/1024 Kbytes) 431
Figure 6.6-3 Address Operation Model 432
Figure 6.6-4 APROM REMAP Examples 433
Figure 6.6-5 OTP Memory Map 457
Figure 6.6-6 Flash Memory Map 458
Figure 6.6-7 System Memory Map with IAP Mode 459
Figure 6.6-8 LDROM with IAP Mode 460
Figure 6.6-9 APROM with IAP Mode 460
Figure 6.6-10 Boot Loader with IAP Mode 461
Figure 6.6-11 ISP Procedure Example 464
Figure 6.6-12 ISP 32-bit Programming Procedure 467
Figure 6.6-13 ISP 64-bit Programming Procedure 467
Figure 6.6-14 Multi-word Programming Time 468
Figure 6.6-15 Firmware in SRAM for Multi-word Programming 468
Figure 6.6-16 Firmware in Boot Loader for Multi-word Programming 469
Figure 6.6-17 Multi-word Programming Flow 470
Figure 6.6-18 Fast Flash Programming Verification Flow 471
Figure 6.6-19 Verification Flow 472
Figure 6.6-20 Flash CRC32 Checksum Calculation 472
Figure 6.7-1 GPIO Controller Block Diagram 516
Figure 6.8-1 PDMA Controller Block Diagram 544
Figure 6.8-2 Descriptor Table Entry Structure 545
Figure 6.8-3 Basic Mode Finite State Machine 547
Figure 6.8-4 Descriptor Table Link List Structure 547
Figure 6.8-5 Scatter-gather Mode Finite State Machine 548
Figure 6.8-6 Example of Single Transfer Type and Burst Transfer Type in Basic Mode 549
Figure 6.8-7 Example of PDMA Channel 0 Time-out Counter Operation 550
Figure 6.8-8 Stride Function Block Transfer 550
Figure 6.8-9 Block Transfer with interval =3 and repeat count =2 551

Figure 6.9-1 Timer Controller Block Diagram 589
Figure 6.9-2 Clock Source of Timer0 ~ Timer 3 Controller 590
Figure 6.9-3 Clock Source of Timer4 and Timer5 Controller 591
Figure 6.9-4 Timer0 ~ Timer3 PWM Generator Overview Block Diagram 592
Figure 6.9-5 Timer0 ~ Timer3 PWM System Clock Source Control 592
Figure 6.9-6 Timer0 ~ Timer3 PWM Counter Clock Source Control 593
Figure 6.9-7 Timer0 ~ Timer3 PWM Independent Mode Architecture Diagram 593
Figure 6.9-8 Timer0 ~ Timer3 PWM Complementary Mode Architecture Diagram 594
Figure 6.9-9 Timer4 and Timer5 PWM Generator Overview Block Diagram 595
Figure 6.9-10 Timer4 and Timer5 PWM Architecture Diagram 595
Figure 6.9-11 Continuous Counting Mode 599
Figure 6.9-12 Capture Mode 600
Figure 6.9-13 Reset Counter Mode 600
Figure 6.9-14 Internal Timer Trigger 602
Figure 6.9-15 Inter-Timer Trigger Capture Timing 603
Figure 6.9-16 PWM Prescale Waveform in Up Count Type 604
Figure 6.9-17 Timer0 ~ Timer3 PWM Up Count Type 604
Figure 6.9-18 Timer4 and Timer5 PWM Up Count Type 604
Figure 6.9-19 Timer0 ~ Timer3 PWM Down Count Type 605
Figure 6.9-20 Timer0 ~ Timer3 PWM Up-Down Count Type 606
Figure 6.9-21 Timer0 ~ Timer3 PWM Comparator Events in Up-Down Count Type 607
Figure 6.9-22 Timer4 and Timer5 PWM Comparator Events in Up Count Type 607
Figure 6.9-23 Timer0 ~ Timer5 PWM Period Loading Mode with Up Count Type 608
Figure 6.9-24 Timer0 ~ Timer3 PWM Immediately Loading Mode with Up Count Type 609
Figure 6.9-25 Timer0 ~ Timer3 PWM Pulse Generation in Up-Down Count Type 609
Figure 6.9-26 Timer0 ~ Timer3 PWM Pulse Generation in Up Count Type 610
Figure 6.9-27 Timer0 ~ Timer3 PWM Pulse Generation in Down Count Type 610
Figure 6.9-28 Timer0 ~ Timer3 PWM 0% to 100% Duty Cycle in Up Count and Up-Down Count Type 611
Figure 6.9-29 Timer4 and Timer5 PWM Pulse Generation in Up Count Type 612
Figure 6.9-30 Timer4 and Timer5 PWM 0% to 100% Duty Cycle in Up Count Type 612
Figure 6.9-31 PWM Independent Mode Output Waveform 613
Figure 6.9-32 Timer0 ~ Timer3 PWM Complementary Mode Output Waveform 613
Figure 6.9-33 Timer0 ~ Timer3 PWMx_CH0 Output Control in Independent Mode 613
Figure 6.9-34 Timer0 ~ Timer3 PWMx_CH0 and PWMx_CH1 Output Control in Complementary Mode 614
Figure 6.9-35 Timer4 and Timer5 PWM Output PWMx_CH0 Control 614

Figure 6.9-36 Timer0 ~ Timer3 PWM Dead-Time Insertion 615
Figure 6.9-37 Timer0 ~ Timer3 PWM Output Mask Control Waveform 615
Figure 6.9-38 Timer0 ~ Timer3 PWM Brake Pin Noise Filter Block Diagram 616
Figure 6.9-39 Timer0 ~ Timer3 PWM Brake Event Block Diagram 617
Figure 6.9-40 Timer0 ~ Timer3 PWM Edge Detector Brake Waveform 618
Figure 6.9-41 Timer0 ~ Timer3 PWM Level Detector Brake Waveform 619
Figure 6.9-42 Timer0 ~ Timer3 PWM Brake Source Block Diagram 620
Figure 6.9-43 Timer0 ~ Timer3 System Fail Brake Block Diagram 621
Figure 6.9-44 Timer0 ~ Timer3 PWM Output Polarity Control with Dead-Time Insertion 621
Figure 6.9-45 Timer4 and Timer5 PWMx_CH0 Polarity Control 622
Figure 6.9-46 Timer0 ~ Timer3 PWM Interrupt Architecture Diagram 622
Figure 6.9-47 Timer4 and Timer5 PWM Interrupt Architecture Diagram 623
Figure 6.9-48 Timer4 and Timer5 PWM Wake-up Architecture Diagram 623
Figure 6.9-49 Timer0 ~ Timer3 PWM Trigger EADC Block Diagram 623
Figure 6.9-50 Timer4 and Timer5 PWM Trigger EADC, PDMA Block Diagram 624
Figure 6.10-1 Watchdog Timer Time-out Interval and Reset Period Timing 681
Figure 6.11-1 Extra Watchdog Timer Time-out Interval and Reset Period Timing 690
Figure 6.12-1 WWDT Reset and Reload Behavior 699
Figure 6.12-2 WWDT Reload Counter When CNTDAT > CMPDAT 699
Figure 6.12-3 WWDT Reload Counter When CNTDAT < CMPDAT 700
Figure 6.12-4 WWDT Interrupt and Reset Signals 700
Figure 6.13-1 Extra WWDT Reset and Reload Behavior 710
Figure 6.13-2 WWDT Reload Counter When CNTDAT > CMPDAT 710
Figure 6.13-3 Extra WWDT Reload Counter When CNTDAT < CMPDAT 711
Figure 6.13-4 Extra WWDT Interrupt and Reset Signals 711
Figure 6.14-1 RTC Block Diagram 720
Figure 6.14-2 Dynamic Rate Definition 725
Figure 6.14-3 Backup I/O Control Diagram 727
Figure 6.15-1 EPWM Generator Overview Block Diagram 772
Figure 6.15-2 EPWM Clock Source Control 773
Figure 6.15-3 EPWM Clock Source Control 774
Figure 6.15-4 EPWM Independent Mode Architecture Diagram 775
Figure 6.15-5 EPWM Complementary Mode Architecture Diagram 776
Figure 6.15-6 EPWM_CH0 Prescaler Waveform in Up Counter Type 778
Figure 6.15-7 EPWM Counter Waveform when Setting Clear Counter 779
Figure 6.15-8 EPWM Up Counter Type 779

Figure 6.15-9 EPWM Down Counter Type 780
Figure 6.15-10 EPWM Up-Down Counter Type 781
Figure 6.15-11 EPWM Compared point Events in Up-Down Counter Type 782
Figure 6.15-12 EPWM Double Buffering Illustration 783
Figure 6.15-13 Period Loading in Up-Count Mode 784
Figure 6.15-14 Immediately Loading in Up-Count Mode 785
Figure 6.15-15 Window Loading in Up-Count Mode 786
Figure 6.15-16 Center Loading in Up-Down-Count Mode 787
Figure 6.15-17 EPWM One-shot Mode Output Waveform 788
Figure 6.15-18 EPWM Pulse Generation 789
Figure 6.15-19 EPWM 0% to 100% Pulse Generation 789
Figure 6.15-20 EPWM Independent Mode Waveform 791
Figure 6.15-21 EPWM Complementary Mode Waveform 791
Figure 6.15-22 EPWM Group Function Waveform 792
Figure 6.15-23 EPWM SYNC_IN Noise Filter Block Diagram 793
Figure 6.15-24 EPWM Counter Synchronous Function Block Diagram 794
Figure 6.15-25 EPWM Synchronous Function with Synchronize source from SYNC_IN Signal 795
Figure 6.15-26 EPWMx_CH0 Output Control in Independent Mode 795
Figure 6.15-27 EPWMx_CH0 and EPWMx_CH1 Output Control in Complementary Mode 796
Figure 6.15-28 Dead-Time Insertion 797
Figure 6.15-29 Illustration of Mask Control Waveform 797
Figure 6.15-30 Brake Noise Filter Block Diagram 798
Figure 6.15-31 Brake Block Diagram for EPWMx_CH0 and EPWMx_CH1 Pair 799
Figure 6.15-32 Edge Detector Waveform for EPWMx_CH0 and EPWMx_CH1 Pair 800
Figure 6.15-33 Level Detector Waveform for EPWMx_CH0 and EPWMx_CH1 Pair 800
Figure 6.15-34 Brake Source Block Diagram 801
Figure 6.15-35 Brake System Fail Block Diagram 802
Figure 6.15-36 EPWM LEB Function Waveform 802
Figure 6.15-37 Initial State and Polarity Control with Rising Edge Dead-Time Insertion 803
Figure 6.15-38 EPWMx_CH0 Accumulate Interrupt Waveform 804
Figure 6.15-39 EPWMx_CH0 and EPWMx_CH1 Pair Interrupt Architecture Diagram 805
Figure 6.15-40 Fault Detect Function Interrupt Architecture Diagram 806
Figure 6.15-41 EPWMx_CH0 and EPWMx_CH1 Pair Trigger EADC Events 807
Figure 6.15-42 EPWMx_CH0 Trigger EADC Block Diagram 807
Figure 6.15-43 EPWM Trigger EADC in Up-Down Counter Type Timing Waveform 808
Figure 6.15-44 EPWM_CH0 and EPWM_CH1 Pair Trigger DAC Block Diagram 808

Figure 6.15-45 EPWM_CH0 Capture Block Diagram 809
Figure 6.15-46 Capture Operation Waveform 810
Figure 6.15-47 Capture PDMA Operation Waveform of Channel 0 812
Figure 6.15-48 Accumulator PDMA Function Architecture 813
Figure 6.15-49 EPWM Accumulator Stop Mode Waveform 814
Figure 6.15-50 Fault Detect Function Architecture 815
Figure 6.15-51 EPWM Mask Function Waveform 816
Figure 6.15-52 EPWM Deglitch Function Waveform 816
Figure 6.16-1 BPWM Generator Overview Block Diagram 902
Figure 6.16-2 BPWM Clock Source Control 903
Figure 6.16-3 BPWM Clock Source Control 903
Figure 6.16-4 BPWM Independent Mode Architecture Diagram 904
Figure 6.16-5 BPWM_CH0 CLKPSC Waveform 906
Figure 6.16-6 BPWM Counter Clear Waveform 907
Figure 6.16-7 BPWM Up Counter Type 907
Figure 6.16-8 BPWM Down Counter Type 908
Figure 6.16-9 BPWM Up-Down Counter Type 908
Figure 6.16-10 BPWM CMPDAT Events in Up-Down Counter Type 909
Figure 6.16-11 Period Loading Mode with Up-Counter Type 910
Figure 6.16-12 Immediately Loading Mode with Up-Counter Type 911
Figure 6.16-13 Center Loading Mode with Up-Down-Counter Type 912
Figure 6.16-14 BPWM Pulse Generation (Left: Asymmetric Pulse, Right: Variety Pulse) 913
Figure 6.16-15 BPWM 0% to 100% Pulse Generation (Left: Up Counter Type, Right: Up-down Counter Type) 913
Figure 6.16-16 BPWM_CH0 Output Control 3 Steps 914
Figure 6.16-17 Mask Control Waveform Illustration 915
Figure 6.16-18 Initial State and Polarity Control 915
Figure 6.16-19 BPWM_CH0 and BPWM_CH1 Pair Interrupt Architecture Diagram 916
Figure 6.16-20 BPWM_CH0 and BPWM_CH1 Pair Trigger EADC Source Block Diagram 917
Figure 6.16-21 BPWM CH0~ CH5 Trigger EADC Block Diagram 917
Figure 6.16-22 BPWM Trigger EADC in Up-Down Counter Type Timing Waveform 918
Figure 6.16-23 BPWM_CH0 Capture Block Diagram 919
Figure 6.16-24 Capture Operation Waveform 920
Figure 6.17-1 QEI Block Diagram 957
Figure 6.17-2 QEI Clock Source Control 958
Figure 6.17-3 Noise Filter 959
Figure 6.17-4 Noise Filter Sampling Clock Selection 960

Figure 6.17-5 QEA/QEB/IDX Timing Requirement through Noise Filter 960
Figure 6.17-6 X4 Counting Mode 961
Figure 6.17-7 X2 Counting Mode 962
Figure 6.17-8 Compare Operation 963
Figure 6.17-9 QEI_CNT Reload/Reset Control 964
Figure 6.17-10 Trigger Control of Capturing QEI Counter 964
Figure 6.17-11 Capture and Latch QEI Counter 965
Figure 6.17-12 Quadrature Encoder Interface Interrupt Architecture Diagram 966
Figure 6.18-1 Input Capture Timer/Counter Architecture 978
Figure 6.18-2 Input Capture Timer/Counter Clock Source Control 980
Figure 6.18-3 Noise Filter Sampling Clock Selection 981
Figure 6.18-4 Input Capture Timer/Counter Function Block 982
Figure 6.18-5 Input Capture Timer/Counter Interrupt Architecture Diagram 984
Figure 6.19-1 UART Clock Control Diagram 1000
Figure 6.19-2 UART Block Diagram 1000
Figure 6.19-3 Auto-Baud Rate Measurement 1009
Figure 6.19-4 Transmit Delay Time Operation 1009
Figure 6.19-5 UART nCTS Wake-up Case1 1010
Figure 6.19-6 UART nCTS Wake-up Case2 1010
Figure 6.19-7 UART Data Wake-up 1011
Figure 6.19-8 UART Received Data FIFO reached threshold wake-up 1011
Figure 6.19-9 UART RS-485 AAD Mode Address Match Wake-up 1012
Figure 6.19-10 UART Received Data FIFO threshold time-out wake-up 1012
Figure 6.19-11 Auto-Flow Control Block Diagram 1016
Figure 6.19-12 UART nCTS Auto-Flow Control Enabled 1017
Figure 6.19-13 UART nRTS Auto-Flow Control Enabled 1017
Figure 6.19-14 UART nRTS Auto-Flow with Software Control 1018
Figure 6.19-15 IrDA Control Block Diagram 1018
Figure 6.19-16 IrDA TX/RX Timing Diagram 1019
Figure 6.19-17 Structure of LIN Frame 1020
Figure 6.19-18 Structure of LIN Byte 1020
Figure 6.19-19 Break Detection in LIN Mode 1022
Figure 6.19-20 LIN Frame ID and Parity Format 1022
Figure 6.19-21 LIN Sync Field Measurement 1024
Figure 6.19-22 UART_BAUD Update Sequence in AR mode if SLVDUEN is 1 1025
Figure 6.19-23 UART_BAUD Update Sequence in AR mode if SLVDUEN is 0 1026

Figure 6.19-24 RS-485 nRTS Driving Level in Auto Direction Mode 1028
Figure 6.19-25 RS-485 nRTS Driving Level with Software Control 1028
Figure 6.19-26 Structure of RS-485 Frame 1029
Figure 6.20-1 SC Clock Control Diagram (8-bit Pre-scale Counter in Clock Controller) 1070
Figure 6.20-2 SC Controller Block Diagram 1070
Figure 6.20-3 SC Data Character 1074
Figure 6.20-4 SC Activation Sequence 1075
Figure 6.20-5 SC Warm Reset Sequence 1077
Figure 6.20-6 SC Deactivation Sequence 1078
Figure 6.20-7 Basic Operation Flow 1079
Figure 6.20-8 Initial Character TS 1080
Figure 6.20-9 SC Error Signal 1081
Figure 6.20-10 Transmit Direction Block Guard Time Operation 1084
Figure 6.20-11 Receive Direction Block Guard Time Operation 1084
Figure 6.20-12 Extra Guard Time Operation 1084
Figure 6.21-1 I^{2}S Controller Block Diagram 1112
Figure 6.21-2 I^{2}S Clock Control Diagram 1114
Figure 6.21-3 Master Mode Interface Block Diagram 1114
Figure 6.21-4 Slave Mode Interface Block Diagram 1115
Figure 6.21-5 I^{2}S Channel Width and Data Width (CHWIDTH≦DATWIDTH) 1115
Figure 6.21-6 I^{2}S Channel Width and Data Width (CHWIDTH > DATWIDTH) 1115
Figure 6.21-7 I^{2}S Data Format Timing Diagram (FORMAT = 0x0 ; CHWIDTH≦DATWIDTH) 1116
Figure 6.21-8 I^{2}S with MSB Justified Data Format (FORMAT = 0x1 ; CHWIDTH > DATWIDTH) 1116
Figure 6.21-9 I^{2}S with LSB Justified Data Format (FORMAT = 0x2 ; CHWIDTH > DATWIDTH) 1116
Figure 6.21-10 PCM Data Format Timing Diagram (FORMAT = 0x4 ; CHWIDTH≦DATWIDTH) 1117
Figure 6.21-11 PCM with MSB Justified Data Format (FORMAT = 0x5 ; CHWIDTH > DATWIDTH) 1117
Figure 6.21-12 PCM with LSB Justified Data Format (FORMAT = 0x6 ; CHWIDTH > DATWIDTH) 1117
Figure 6.21-13 TDM 6-channel Audio Format with 24-bit Data in 32-bit Channel Block (PCM Standard Data Format; FORMAT=0x4) 1118
Figure 6.21-14 TDM 6-channel Audio Format with 24-bit Data in 32-bit Channel Block (PCM with MSB Justified; FORMAT=0x5) 1118
Figure 6.21-15 TDM 6-channel Audio Format with 24-bit Data in 32-bit Channel Block (PCM with LSB Justified; FORMAT=0x6) 1118
Figure 6.21-16 I^{2}S Interrupts 1120

Figure 6.21-17 FIFO Contents for Various 2-channel Audio Modes 1121
Figure 6.21-18 FIFO Contents for Various 4-channel Audio Modes 1122
Figure 6.21-19 FIFO Contents for Various 6-channel Audio Modes (Part-1) 1123
Figure 6.21-20 FIFO Contents for Various 6-channel Audio Modes (Part-2) 1124
Figure 6.22-1 SPI Block Diagram 1145
Figure 6.22-2 SPI Peripheral Clock 1149
Figure 6.22-3 SPI Full-Duplex Master Mode Application Block Diagram 1150
Figure 6.22-4 SPI Full-Duplex Slave Mode Application Block Diagram 1150
Figure 6.22-5 32-bit in One Transaction 1151
Figure 6.22-6 Automatic Slave Selection (SSACTPOL = 0, SUSPITV > 0x2) 1152
Figure 6.22-7 Automatic Slave Selection (SSACTPOL = 0, SUSPITV < 0x3) 1152
Figure 6.22-8 Byte Reorder Function 1153
Figure 6.22-9 Timing Waveform for Byte Suspend 1153
Figure 6.22-10 SPI Half-Duplex Master Mode Application Block Diagram 1154
Figure 6.22-11 SPI Half-Duplex Slave Mode Application Block Diagram 1154
Figure 6.22-12 FIFO Threshold Comparator 1156
Figure 6.22-13 Transmit FIFO Buffer Example for 8~16 bits of data length 1157
Figure 6.22-14 Transmit FIFO Buffer Example for 17~32 bits of data length 1157
Figure 6.22-15 Receive FIFO Buffer Example for 16 bits of Data Length 1158
Figure 6.22-16 Receive FIFO Buffer Example for 32 bits of Data Length 1159
Figure 6.22-17 TX Underflow Event and Slave Under Run Event 1159
Figure 6.22-18 TX Underflow Event (SPI Slave 3-Wire Mode Enabled) 1160
Figure 6.22-19 Slave Mode Bit Count Error and Effective Bit Number of Uncompleted RX Data 1161
Figure 6.22-20 I^{2}S Data Format Timing Diagram 1163
Figure 6.22-21 MSB Justified Data Format Timing Diagram 1163
Figure 6.22-22 PCM Mode A Timing Diagram 1163
Figure 6.22-23 PCM Mode B Timing Diagram 1164
Figure 6.22-24 FIFO Contents for Various I^{2}S Modes 1165
Figure 6.22-25 SPI Timing in Master Mode 1168
Figure 6.22-26 SPI Timing in Master Mode (Alternate Phase of SPIx_CLK) 1168
Figure 6.22-27 SPI Timing in Slave Mode 1169
Figure 6.22-28 SPI Timing in Slave Mode (Alternate Phase of SPIx_CLK) 1169
Figure 6.23-1 QSPI Block Diagram 1196
Figure 6.23-2 QSPI Peripheral Clock 1198
Figure 6.23-3 QSPI Full-Duplex Master Mode Application Block Diagram 1198
Figure 6.23-4 QSPI Full-Duplex Slave Mode Application Block Diagram 1198

Figure 6.23-5 32-bit in One Transaction 1199
Figure 6.23-6 Automatic Slave Selection (SSACTPOL = 0, SUSPITV > 0x2) 1200
Figure 6.23-7 Automatic Slave Selection (SSACTPOL = 0, SUSPITV < 0x3) 1200
Figure 6.23-8 Byte Reorder Function 1201
Figure 6.23-9 Timing Waveform for Byte Suspend 1201
Figure 6.23-10 QSPI Half-Duplex Master Mode Application Block Diagram 1202
Figure 6.23-11 QSPI Half-Duplex Slave Mode Application Block Diagram 1202
Figure 6.23-12 Two-bit Transfer Mode System Architecture 1204
Figure 6.23-13 Two-bit Transfer Mode Timing (Master Mode) 1204
Figure 6.23-14 Bit Sequence of Dual Output Mode 1205
Figure 6.23-15 Bit Sequence of Dual Input Mode 1205
Figure 6.23-16 Bit Sequence of Quad Output Mode 1206
Figure 6.23-17 Bit Sequence of Quad Input Mode 1207
Figure 6.23-18 Timing Diagram of QSPI Output Data for Transmit Double Transfer Rate Mode 1208
Figure 6.23-19 FIFO Threshold Comparator 1209
Figure 6.23-20 Transmit FIFO Buffer Example 1210
Figure 6.23-21 Receive FIFO Buffer Example 1211
Figure 6.23-22 TX Underflow Event and Slave Under Run Event 1211
Figure 6.23-23 Two-bit Transfer Mode FIFO Buffer Example 1212
Figure 6.23-24 TX Underflow Event (QSPI0 Slave 3-Wire Mode Enabled) 1212
Figure 6.23-25 Slave Mode Bit Count Error and Effective Bit Number of Uncompleted RX Data 1213
Figure 6.23-26 Slave Time-out Event 1214
Figure 6.23-27 QSPI Timing in Master Mode 1216
Figure 6.23-28 QSPI Timing in Master Mode (Alternate Phase of QSPIx_CLK) 1216
Figure 6.23-29 QSPI Timing in Slave Mode 1217
Figure 6.23-30 QSPI Timing in Slave Mode (Alternate Phase of QSPIx_CLK) 1217
Figure 6.24-1 USCI Block Diagram 1236
Figure 6.24-2 Input Conditioning for USCIx_DAT[1:0] and USCIx_CTL[1:0] 1237
Figure 6.24-3 Input Conditioning for USCIx_CLK 1238
Figure 6.24-4 Block Diagram of Data Buffering 1239
Figure 6.24-5 Data Access Structure 1240
Figure 6.24-6 Transmit Data Path 1240
Figure 6.24-7 Receive Data Path 1241
Figure 6.24-8 Protocol-Relative Clock Generator 1242
Figure 6.24-9 Basic Clock Divider Counter 1243

Figure 6.24-10 Block of Timing Measurement Counter 1243
Figure 6.24-11 Sample Time Counter 1244
Figure 6.24-12 Event and Interrupt Structure 1245
Figure 6.25-1 I²C Controller Block Diagram 1248
Figure 6.25-2 I²C Bus Timing 1250
Figure 6.25-3 I²C Protocol 1251
Figure 6.25-4 START and STOP Conditions 1251
Figure 6.25-5 Bit Transfer on the I²C Bus 1252
Figure 6.25-6 Acknowledge on the I²C Bus 1252
Figure 6.25-7 Master Transmits Data to Slave by 7-bit 1253
Figure 6.25-8 Master Reads Data from Slave by 7-bit 1253
Figure 6.25-9 Master Transmits Data to Slave by 10-bit 1253
Figure 6.25-10 Master Reads Data from Slave by 10-bit 1254
Figure 6.25-11 Control I²C Bus according to the Current I²C Status 1254
Figure 6.25-12 Master Transmitter Mode Control Flow 1255
Figure 6.25-13 Master Receiver Mode Control Flow 1256
Figure 6.25-14 Slave Mode Control Flow 1257
Figure 6.25-15 GC Mode 1258
Figure 6.25-16 Arbitration Lost 1259
Figure 6.25-17 Bus Management Packet Protocol Diagram Element Key 1261
Figure 6.25-187-bit Addressable Device to Host Communication 1262
Figure 6.25-197-bit Addressable Device Responds to an ARA 1262
Figure 6.25-20 Bus Management ALERT function 1263
Figure 6.25-21 Bus Management Time Out Timing 1264
Figure 6.25-22 Bus Clock Low Time Out Timing 1264
Figure 6.25-23 Setup Time Wrong Adjustment 1266
Figure 6.25-24 Hold Time Wrong Adjustment 1266
Figure 6.25-25 I²C Data Shifting Direction 1267
Figure 6.25-26 I²C Time-out Count Block Diagram 1269
Figure 6.25-27 I²C Wake-Up Related Signals Waveform 1270
Figure 6.25-28 EEPROM Random Read 1271
Figure 6.25-29 Protocol of EEPROM Random Read 1272
Figure 6.26-1 USCI-UART Mode Block Diagram 1296
Figure 6.26-2 UART Signal Connection for Full-Duplex Communication 1298
Figure 6.26-3 UART Standard Frame Format 1299
Figure 6.26-4 UART Bit Timing (Data Sample Time) 1301

Figure 6.26-5 UART Auto Baud Rate Control 1303
Figure 6.26-6 Incoming Data Wake-Up 1304
Figure 6.26-7 nCTS Wake-Up Case 1 1304
Figure 6.26-8 nCTS Wake-Up Case 2 1304
Figure 6.27-1 SPI Master Mode Application Block Diagram 1331
Figure 6.27-2 SPI Slave Mode Application Block Diagram 1331
Figure 6.27-3 USCI SPI Mode Block Diagram 1332
Figure 6.27-4 4-Wire Full-Duplex SPI Communication Signals (Master Mode) 1334
Figure 6.27-5 4-Wire Full-Duplex SPI Communication Signals (Slave Mode) 1335
Figure 6.27-6 SPI Communication with Different SPI Clock Configuration (SCLKMODE=0x0) 1336
Figure 6.27-7 SPI Communication with Different SPI Clock Configuration (SCLKMODE=0x1) 1336
Figure 6.27-8 SPI Communication with Different SPI Clock Configuration (SCLKMODE=0x2) 1337
Figure 6.27-9 SPI Communication with Different SPI Clock Configuration (SCLKMODE=0x3) 1337
Figure 6.27-10 16-bit Data Length in One Word Transaction with MSB First Format 1338
Figure 6.27-11 Word Suspend Interval between Two Transaction Words 1339
Figure 6.27-12 Auto Slave Select (SUSPITV ≧ 0x3) 1339
Figure 6.27-13 Auto Slave Select (SUSPITV < 0x3) 1340
Figure 6.27-14 One Output Data Channel Half-duplex (SPI Master Mode) 1341
Figure 6.27-15 One Input Data Channel Half-duplex (SPI Master Mode) 1341
Figure 6.27-16 SPI Timing in Master Mode 1343
Figure 6.27-17 SPI Timing in Master Mode (Alternate Phase of Serial Bus Clock) 1343
Figure 6.27-18 SPI Timing in Slave Mode 1344
Figure 6.27-19 SPI Timing in Slave Mode (Alternate Phase of Serial Bus Clock) 1344
Figure 6.28-1 I^2C Bus Timing 1371
Figure 6.28-2 USCI I²C Mode Block Diagram 1372
Figure 6.28-3 I^2C Protocol 1374
Figure 6.28-4 START and STOP Conditions 1374
Figure 6.28-5 Bit Transfer on the I^2C Bus 1375
Figure 6.28-6 Acknowledge on the I^2C Bus 1376
Figure 6.28-7 Arbitration Lost 1377
Figure 6.28-8 Control I^2C Bus according to Current I^2C Status 1379
Figure 6.28-9 Master Transmits Data to Slave with a 7-bit Address 1380
Figure 6.28-10 Master Reads Data from Slave with a 7-bit Address 1380
Figure 6.28-11 Master Transmits Data to Slave by 10-bit Address 1380
Figure 6.28-12 Master Reads Data from Slave by 10-bit Address 1381
Figure 6.28-13 Master Transmitter Mode Control Flow with 7-bit Address 1381

Figure 6.28-14 Master Receiver Mode Control Flow with 7-bit Address 1382
Figure 6.28-15 Master Transmitter Mode Control Flow with 10-bit Address 1383
Figure 6.28-16 Master Receiver Mode Control Flow with 10-bit Address 1384
Figure 6.28-17 Save Mode Control Flow with 7-bit Address 1385
Figure 6.28-18 Save Mode Control Flow with 10-bit Address 1386
Figure 6.28-19 GC Mode with 7-bit Address 1387
Figure 6.28-20 Setup Time Wrong Adjustment 1389
Figure 6.28-21 Hold Time Wrong Adjustment 1389
Figure 6.28-22 I^2C Time-out Count Block Diagram 1390
Figure 6.28-23 EEPROM Random Read 1391
Figure 6.28-24 Protocol of EEPROM Random Read 1391
Figure 6.29-1 CAN Peripheral Block Diagram 1414
Figure 6.29-2 CAN Core in Silent Mode 1416
Figure 6.29-3 CAN Core in Loop Back Mode 1417
Figure 6.29-4 CAN Core in Loop Back Mode Combined with Silent Mode 1417
Figure 6.29-5 Data transfer between IFn Registers and Message 1420
Figure 6.29-6 Application Software Handling of a FIFO Buffer 1425
Figure 6.29-7 Bit Timing 1427
Figure 6.29-8 Propagation Time Segment 1428
Figure 6.29-9 Synchronization on “late” and “early” Edges 1430
Figure 6.29-10 Filtering of Short Dominant Spikes 1431
Figure 6.29-11 Structure of the CAN Core’s CAN Protocol Controller 1432
Figure 6.30-1 SD Host Controller Block Diagram 1477
Figure 6.30-2 PAD (Physical Address Descriptor) Table Format 1479
Figure 6.31-1 EBI Block Diagram 1501
Figure 6.31-2 Connection of 16-bit EBI Data Width with 16-bit Device 1504
Figure 6.31-3 Connection of 8-bit EBI Data Width with 8-bit Device 1504
Figure 6.31-4 Connection of 16-bit EBI Data Width with 16-bit Device in Separate mode 1505
Figure 6.31-5 Connection of 8-bit EBI Data Width with 8-bit Device in Separate mode 1505
Figure 6.31-6 Timing Control Waveform for 16-bit Data Width 1507
Figure 6.31-7 Timing Control Waveform for 8-bit Data Width 1508
Figure 6.31-8 Timing Control Waveform for Byte Write in 16-bit Data Mode 1509
Figure 6.31-9 Timing Control Waveform for Insert Idle Cycle 1510
Figure 6.31-10 Timing Control Waveform for 16-bit Data Width for Separate Mode 1511
Figure 6.31-11 Timing Control Waveform for Continuous Data Access Mode 1512
Figure 6.32-1 USB Block Diagram 1519

Figure 6.32-2 NEVWK Interrupt Operation Flow 1521
Figure 6.32-3 Endpoint SRAM Structure 1521
Figure 6.32-4 Setup Transaction Followed by Data IN Transaction 1522
Figure 6.32-5 Data Out Transfer 1522
Figure 6.32-6 LPM State Transition Diagram 1523
Figure 6.33-1 USB 1.1 Host Controller Block Diagram 1554
Figure 6.34-1 USB OTG Block Diagram 1592
Figure 6.34-2 USB Device Mode 1593
Figure 6.34-3 USB Host Mode 1593
Figure 6.35-1 CRC Generator Block Diagram 1606
Figure 6.35-2 CHECKSUM Bit Order Reverse Functional Block 1607
Figure 6.35-3 Write Data Bit Order Reverse Functional Block 1608
Figure 6.36-1 Cryptographic Accelerator Block Diagram 1617
Figure 6.36-2 PRNG Function Diagram 1620
Figure 6.36-3 Electronic Codebook Mode 1623
Figure 6.36-4 Cipher Block Chaining Mode 1624
Figure 6.36-5 Cipher Feedback Mode 1625
Figure 6.36-6 Output Feedback Mode 1626
Figure 6.36-7 Counter Mode 1627
Figure 6.36-8 CBC-CS1 Encryption 1627
Figure 6.36-9 CBC-CS1 Decryption 1628
Figure 6.36-10 CCM Encryption and Decryption Flow 1629
Figure 6.36-11 GCM encryption and decryption flow 1631
Figure 6.36-12 Main Hierarchy Chart of ECC 1636
Figure 6.37-1 ADC Converter Block Diagram 1761
Figure 6.37-2 Sample Module 0~3 Block Diagram 1763
Figure 6.37-3 Sample Module 4~15 Block Diagram 1764
Figure 6.37-4 Sample Module 16~18 Block Diagram 1764
Figure 6.37-5 EADC Clock Control 1765
Figure 6.37-6 Example ADC Conversion Timing Diagram, n=0~18 1766
Figure 6.37-7 Sample Module Conversion Priority Arbitrator Diagram 1767
Figure 6.37-8 Specific Sample Module ADC EOC Signal for ADINT0~3 Interrupt 1769
Figure 6.37-9 EADC0_ST De-bounce Timing Diagram 1769
Figure 6.37-10 EPWM-triggered ADC Start Conversion 1770
Figure 6.37-11 External Triggered ADC Start Conversion 1770
Figure 6.37-12 Conversion Start Delay Timing Diagram 1771

Figure 6.37-13 ADC Extend Sampling Timing Diagram 1772
Figure 6.37-14 ADC Conversion Result Monitor Logics Diagram 1773
Figure 6.37-15 ADC Controller Interrupts 1774
Figure 6.37-16 ADC Start up Sequence with Calibration 1775
Figure 6.38-11.1 True Random Number Generator Block Diagram 1812
Figure 6.39-1 Key Store Block Diagram 1820
Figure 6.40-1 LCD Controller Block Diagram 1842
Figure 6.40-2 LCD Output Waveform of Type A, Duty 1/4, Bias 1/2 1847
Figure 6.40-3 LCD Output Waveform of Type A, Duty 1/4 Bias 1/3 1848
Figure 6.40-4 LCD Output Waveform of Type A, Duty 1/4, Bias 1/4 1849
Figure 6.40-5 LCD Output Waveform of Type B, Duty 1/4, Bias 1/2 1850
Figure 6.40-6 LCD Output Waveform of Type B, Duty 1/4, Bias 1/3 1851
Figure 6.40-7 LCD Output Waveform of Type B, Duty 1/4, Bias 1/4 1852
Figure 6.40-8 Waveform Inverse 1853
Figure 6.40-9 Frame Counting 1854
Figure 6.40-10 Blinking 1855
Figure 6.40-11 Resistive Network and Voltage Buffers 1856
Figure 6.40-12 Null Frame 1857
Figure 6.40-13 Power Saving Mode 1858
Figure 6.41-1 Tamper Block Diagram (Normal Mode) 1875
Figure 6.42-1 Digital-to-Analog Converter Block Diagram 1911
Figure 6.42-2 Data Holding Register Format 1913
Figure 6.42-3 DAC Conversion Started by Software Write Trigger 1913
Figure 6.42-4 DAC Conversion Started by Hardware Trigger Event 1914
Figure 6.42-5 DAC0 and DAC1 Group and Ungroup Update Example 1914
Figure 6.42-6 DAC PDMA Under-Run Condition Example 1915
Figure 6.42-7 DAC Continuous Conversion with Software PDMA Mode 1916
Figure 6.42-8 DAC Interrupt Source 1916
Figure 6.43-1 Analog Comparator Block Diagram 1933
Figure 6.43-2 Comparator Hysteresis Function of ACMP0 1934
Figure 6.43-3 Window Latch Mode 1935
Figure 6.43-4 Example of Filter Function 1935
Figure 6.43-5 Comparator Controller Interrupt 1936
Figure 6.43-6 Comparator Reference Voltage Block Diagram 1936
Figure 6.43-7 Example of Window Compare Mode 1937
Figure 6.43-8 Example of Window Compare Mode 1938

List of Tables

Table 3-1 M2354 Series Selection Guide 47
Table 3-2 M2354 Series Selection Code 48
Table 4.1-1 M2354LJFAE Multi-function Pin Table 54
Table 4.1-2 M2354SJFAE Multi-function Pin Table 58
Table 6.2-1 Reset Value of Registers 101
Table 6.2-2 Power Mode Table 105
Table 6.2-3 Power Mode Entry Setting Table 105
Table 6.2-4 Power Mode Difference Table 105
Table 6.2-5 Clocks in Power Modes 107
Table 6.2-6 Condition of Entering Power-down Mode Again 109
Table 6.2-7 Address Space Assignments for On-Chip Controllers 113
Table 6.2-8 SRAM Power Mode Behavior 121
Table 6.2-9 List of Registers with Write Protection 124
Table 6.2-10 Exception Model 199
Table 6.2-11 Interrupt Number Table 202
Table 6.2-12 Priority Grouping 229
Table 6.3-1 Each Clock Source Enable Bit and Corresponding Stable Flag Table 260
Table 6.3-2 Clock Controller Share Register list 264
Table 6.3-3 Symbol Definition of PLL Output Frequency Formula 293
Table 6.4-1 Security Memory Access Policy (Master is Core Processor) 322
Table 6.4-2 Security Memory Access Policy (Master is Not Core Processor) 322
Table 6.4-3 Privileged Memory Access Policy (Master is Core Processor) 323
Table 6.4-4 Privileged Memory Access Policy (Master is Not Core Processor) 323
Table 6.4-5 Shared Register Memory Access Policy (Master Is Core Processor) 324
Table 6.4-6 Shared Register Memory Access Policy (Master Is not Core Processor) 324
Table 6.4-7 Shared Registers List for Non-secure Access 324
Table 6.4-8 GPIO Memory Access Policy 324
Table 6.4-9 Master ID Number List 326
Table 6.4-10 Debug State and Debug Authority 332
Table 6.4-11 Function of Stage of PLM 336
Table 6.5-1 Peripherals and Regions that are Always Secure 424
Table 6.6-1 Dual-Bank Block Address Range 430
Table 6.6-2 ISP Command List 463
Table 6.6-3 NS-ISP Command List 466
Table 6.6-4 FMC Control Registers for Flash Programming 466

Table 6.8-1 Channel Priority Table 546
Table 6.9-1 Timer0 and Timer1 Pin Configuration 596
Table 6.9-2 Timer2 and Timer3 Pin Configuration 596
Table 6.9-3 Timer4 and Timer5 Pin Configuration 597
Table 6.9-4 Timer0 ~ Timer3 PWM Pulse Generation Event Priority in Up Count Type 610
Table 6.9-5 Timer0 ~ Timer3 PWM Pulse Generation Event Priority in Down Count Type 611
Table 6.9-6 Timer0 ~ Timer3 PWM Pulse Generation Event Priority in Up-Down Count Type 611
Table 6.9-7 Timer4 and Timer5 PWM Output Level 612
Table 6.10-1 Watchdog Timer Time-out Interval Period Selection 681
Table 6.11-1 Extra Watchdog Timer Time-out Interval Period Selection 690
Table 6.12-1 WWDT Prescaler Value Selection 698
Table 6.12-2 CMPDAT Setting Limitation 701
Table 6.13-1 Extra WWDT Prescaler Value Selection 709
Table 6.13-2 CMPDAT Setting Limitation 712
Table 6.14-1 RTC Read/Write Enable 722
Table 6.14-212/24 Hour Time Scale Selection 723
Table 6.14-3 Registers Value after Powered On 724
Table 6.14-4 Dynamic Pattern Source Selection 725
Table 6.14-5 Tamper Control Bit Effect for Pair 0 725
Table 6.14-6 Tamper Control Bit Effect for Pair 1 and 2 726
Table 6.15-1 EPWM Clock Source Control Registers Setting Table 773
Table 6.15-2 EPWM Pulse Generation Event Priority for Up-Counter 790
Table 6.15-3 EPWM Pulse Generation Event Priority for Down-Counter 790
Table 6.15-4 EPWM Pulse Generation Event Priority for Up-Down-Counter 790
Table 6.16-1 BPWM Pulse Generation Event Priority for Up-Counter 913
Table 6.16-2 BPWM Pulse Generation Event Priority for Down-Counter 914
Table 6.16-3 BPWM Pulse Generation Event Priority for Up-Down-Counter 914
Table 6.17-1 Direction of Counting 963
Table 6.18-1 Typical Case of Noise Filter Settings 980
Table 6.19-1 NuMicro® M2354 Series UART Features 998
Table 6.19-2 UART Interrupt 1001
Table 6.19-3 UART Interface Controller Pin 1006
Table 6.19-4 UART controller Baud Rate Equation Table 1006
Table 6.19-5 UART controller Baud Rate Parameter Setting Example Table 1007
Table 6.19-6 UART controller Baud Rate Register Setting Example Table 1007
Table 6.19-7 Baud Rate Compensation Example Table 1 1008

Table 6.19-8 Baud Rate Compensation Example Table 2 1008
Table 6.19-9 UART controller Interrupt Source and Flag List 1015
Table 6.19-10 UART Line Control of Word and Stop Length Setting 1015
Table 6.19-11 UART Line Control of Parity Bit Setting 1016
Table 6.19-12 LIN Header Selection in Master Mode 1021
Table 6.20-1 SC Host Controller Pin Description 1071
Table 6.20-2 UART Mode Pin Description 1071
Table 6.20-3 Timer0/Timer1/Timer2 Operation Mode 1084
Table 6.21-1 Pin Configuration of I^{2}S Controller 1113
Table 6.22-1 SPI/I2S Interface Controller Pin Description (SPI0~SPI3) 1149
Table 6.22-2 Dummy Data Number for I^{2}S / PCM Master Mode and Monaural Mode 1166
Table 6.22-3 Dummy Data Number for I^{2}S / PCM Master Mode and Stereo Mode 1166
Table 6.22-4 Dummy Data Number for I^{2}S Slave Mode and Monaural Mode 1166
Table 6.22-5 Dummy Data Number for PCM Slave Mode and Monaural Mode 1167
Table 6.22-6 Dummy Data Number for I^{2}S Slave Mode and Stereo Mode 1167
Table 6.22-7 Dummy Data Number for PCM Slave Mode and Stereo Mode 1167
Table 6.24-1 Input Signals for Different Protocols 1237
Table 6.24-2 Output Signals for Different Protocols 1238
Table 6.24-3 Data Transfer Events and Interrupt Handling 1245
Table 6.24-4 Protocol-specific Events and Interrupt Handling 1246
Table 6.25-1 Reserved SMBus Address 1260
Table 6.25-2 Relationship between I^{2}C Baud Rate and PCLK 1265
Table 6.25-3 I^{2}C Status Code Description 1268
Table 6.26-1 Input Signals for UART Protocol 1299
Table 6.26-2 Output Signals for UART Protocol 1299
Table 6.26-3 Baud Rate Relationship 1302
Table 6.27-1 Signals for SPI communication 1334
Table 6.27-2 Serial Bus Clock Configuration 1335
Table 6.28-1 Relationship between I^{2}C Baud Rate and PCLK 1389
Table 6.29-1 Initialization of a Transmit Object 1422
Table 6.29-2 Initialization of a Receive Object 1423
Table 6.29-3 CAN Bit Time Parameters 1427
Table 6.29-4 CAN Register Map for Each Bit Function 1440
Table 6.29-5 Last Error Code 1445
Table 6.29-6 Source of Interrupts 1448
Table 6.29-7 IF1 and IF2 Message Interface Register 1451

Table 6.29-8 Structure of a Message Object in the Message Memory 1465
Table 6.30-1 SD0 Pin Configuration 1478
Table 6.31-1 EBI Address Mapping 1503
Table 6.31-2 Timing Control Parameter 1506
Table 6.32-1 USB Link Power Management (Lx) States 1523
Table 6.36-1 Each Engine Error Conditions and Error Flag 1618
Table 6.36-2 DMA Enable Bit Table 1619
Table 6.36-3 DMA Cascade Bit Table 1619
Table 6.36-4 ECC Parameters and Corresponding Registers Table 1636
Table 6.36-5 Required Input Data of Various Operations 1637
Table 6.36-6 Low-Weight Binary Irreducible Polynomials 1642
Table 6.36-7 RSA Accelerator Support Situation 1653
Table 6.37-1 Relation between Resolution and Conversion Cycles 1767
Table 6.37-2 EADC Differential Model Channel Selection 1773
Table 6.37-3 EADC Power Saving Mode 1775
Table 6.37-4 EADC Start up with Calibration 1775
Table 6.39-1 Detailed Information for SRAM, Flash and OTP 1823
Table 6.39-2 Security Policy for Accessing Key 1825
Table 6.39-3 Privilege Policy for Accessing Key 1825
Table 6.39-4 CPU and Crypto Engine Access Key for All Situations 1826
Table 6.40-1 COM/SEG Output Configuration 1844
Table 6.41-1 Tamper Power Condition in Each Mode 1879
Table 6.41-2 TLVD Detect Level 1880
Table 6.41-3 TOVD Detect Level 1880
Table 6.43-1 Truth Table of Window Compare Logic 1937
Table 6.22-1 Peripherals Interconnect Matrix Table 1948
Table 10.1-1 List of Abbreviations 1959

1 GENERAL DESCRIPTION

The NuMicro® M2354 Series is a TrustZone® for Armv8-M architecture empowered microcontroller series focusing on IoT Security based on Arm® Cortex®-M23 CPU core technology. It runs up to 96 MHz with 1024 Kbytes embedded Flash memory and 256 Kbytes SRAM, supporting Flash in dual-bank mode, secure firmware OTA (Over-The-Air) update, ultra-low power consumption in normal run with 89.3 uA/MHz in LDO mode, 39.6 uA/MHz in DC-DC mode and an 8x40 COM/SEG LCD driver inside. Besides the fundamental microcontroller security features, it further enhances the chip-level security in covering side-channel attacks mitigation to crypto hardware engine, fault injection mitigation for operating voltage and clock as well as active shield to cryptographic key storage. The series supports power supply voltage from 1.7V ~ 3.6V in operating temperature range from -40°C to +105°C, and is equipped with both LDO and DC-DC power supply functionalities. The M2354 Series is quite competitive for those devices that need more secure, fast computing and low power in the IoT market.

The one of major challenges for IoT devices that are connected to cloud services or other devices by network communication is security, so the IoT devices must meet some security requirements to protect firmware, software and secure assets from being stolen or modified by an attacker. "**Execution**", "**Storage**", and "**Connectivity**" are the three important security targets for IoT devices.

The TrustZone® technology based on Armv8-M architecture is a System-on-Chip (SoC) and CPU system-wide approach to microcontroller security. The whole system isolates secure and normal worlds to avoid the trusted assets being accessed by a non-secure process. In addition to the firmware-level security, the M2354 series is also equipped with rich functions to improve system security. The Secure Bootloader supports trusted system-boot feature which can protect certificated firmware from being replaced with malware possibly in the upgrade processing and taking control of system resource finally. The hardware crypto accelerators, including AES, ECC and RSA, support encryption and decryption operations to offload the main processor's computing power and ensure data transmission in secure.

The M2354 series also enhances firmware update security requirement with monotonic version counter. The firmware cannot be rollback to older one which has lower security protection. Furthermore, there is a secure crypto keys storage protected by the chip-level active shield function to physical intrusion. The series addresses the physical attack protection and system security certification for Arm® PSA Certified™ Level 2 even for PSA Certified™ Level 3.

Other than security, low power is also vital for IoT applications. The M2354 series supports 4 core power levels with both LDO and DC-DC power supply mechanism. Except normal run mode, the series also provides idle run mode with power consumption 31.5 μA/MHz in LDO mode and 14.3 μA/MHz in DC-DC mode. The current consumption of Deep Power-Down mode without V_{BAT} is less than 0.1 μA.

The M2354 series is equipped with plenty of peripherals such as Timers, Watchdog Timers, RTC, PDMA, UART, Universal Serial Control Interface (USCI), SPI/ I²S, I^2C, GPIOs, makes it highly suitable for connecting comprehensive external modules. The M2354 integrates high performance analog front-end circuit blocks, such as 16 channels of 12-bit 6 MSPS ADC, temperature sensor, low voltage reset (LVR) and brown-out detector (BOD) to enhance product performance, reduce external components and form factor simultaneously. Moreover, it supports up to 8x40 COM/SEG for segment LCD display needed such as metering devices.

The M2354 series provides LQFP48 (7mm x 7mm), LQFP64 (7mm x 7mm) and LQFP128 (14mm x 14mm).

The NuMicro® M2354 is suitable for a wide range of applications such as:

- IoT Devices with Secure Connection
- Collaborative Secure Software Development Business Model

- Secure Fingerprint Lock
- Smart Home Appliance
- Smart City Facilities
- Wireless Sensor Node Device (WSND)
- Secure Wireless Connectivity Module (SWCM)
- Auto Meter Reading (AMR)
- Digital Currency Authentication
- Trusted Execution Environment (TEE) with Trusted Applications (TAs)

2 FEATURES

Core And System	
Arm® Cortex®-M23	• Arm® Cortex®-M23 processor, running up to 96 MHz • 96 MHz at 1.8V-3.63V; 84 MHz at 1.7V • Supports Arm® TrustZone® Technology • Built-in PMSAv8 Memory Protection Unit (MPU) • Built-in Security Attribution Unit (SAU) • Built-in Nested Vectored Interrupt Controller (NVIC) • Built-in Embedded Trace Macrocell (ETM) • 32-bit Single-cycle hardware multiplier and 32-bit 17-cycle hardware divider • 24-bit system tick timer • Supports Programmable and maskable interrupt • Supports Low Power Sleep mode by WFI and WFE instructions • Supports single cycle I/O access
Secure Configuration Unit (SCU)	• Configure SRAM's security and privilege attribution block by block • Configure GPIOs' security and privilege attribution port by port • Configure peripherals' security and privilege attribution • Generates secure and privilege violation interrupt • Equipped with a 24-bit timer as a non-secure state monitor • Monotonic firmware version counter • Debug protection mechanism • Product life-cycle management
Brown-out Detector (BOD)	• Eight-level BOD with brown-out interrupt and reset option (3.0V/2.8V/2.6V/2.4V/2.2V/2.0V/1.8V/1.6V)
Low Voltage Reset (LVR)	• LVR with 1.5V threshold voltage level
Power Manager	• Dual voltage regulator is available for DC-DC converter or LDO • Supports 1.26V, 1.2V, 1.1V and 0.9V core voltage while operating • Supports Power-down mode • Supports Standby Power -down mode • Supports Low Leakage Power-down mode • Supports Ultra-low Leakage Power-down mode • Supports Fast Wake-up Power-down mode • Supports Deep Power-down mode

Security	• 128-bit Unique ID (UID) • 128-bit Unique Customer ID (UCID) • One built-in temperature sensor with 1˚C resolution
Memories	
Boot Loader	• Factory pre-loaded 16 KB mask ROM for secure boot procedure • Uses SHA-256 and ECC-256 to validate data in APROM, LDROM and external SPI Flash • Nuvoton ISP (In-System-Programming) tool for firmware upgrade via UART and high speed USB device • ISP/IAP libraries
Flash	• Dual bank 512/1024 KB on-chip Application ROM (APROM) for Over-The-Air (OTA) upgrade • 16 KB on-chip Flash for user-defined loader (LDROM) • Excute Only Memory (XOM) for intelectual property protection • All on-chip Flash supporting 2 KB page erase • Fast Flash programming verification with CRC • On-chip Flash programming with In-Chip Programming (ICP), In-System Programming (ISP) and In-Application Programming (IAP) capabilities • Always boot from boot loader • 2-wired ICP Flash updating through SWD interface • 32-bit/64-bit and multi-word Flash programming function
SRAM	• Up to 256 KB on-chip SRAM includes: • 32 KB SRAM located in bank 0 that supports hardware parity check; Exception (NMI) generated upon a parity check error • 128/128 KB SRAM located in bank 1 and bank2 • Byte-, half-word- and word-access • PDMA operation
Cyclic Redundancy Calculation (CRC)	• Supports CRC-CCITT, CRC-8, CRC-16 and CRC-32 polynomials • Programmable initial value and seed value • Programmable order reverse setting and one's complement setting for input data and CRC checksum • 8-bit, 16-bit, and 32-bit data width • 8-bit write mode with 1-AHB clock cycle operation • 16-bit write mode with 2-AHB clock cycle operation • 32-bit write mode with 4-AHB clock cycle operation • Uses DMA to write data with performing CRC operation

Peripheral DMA (PDMA)	• 16 independent and configurable channels for automatic data transfer between memories and peripherals • 8 channels of PDMA1 can be configured as secure or non-secure channels • Supports time-out function when transfer time-out • Basic and Scatter-Gather transfer modes • Each channel supports circular buffer management using Scatter-Gather Transfer mode • Stride function for rectangle image data movement • Fixed-priority and Round-robin priorities modes • Single and burst transfer types • Byte-, half-word- and word tranfer unit with count up to 65536 • Incremental or fixed source and destination address
Clocks	
External Clock Source	• 4~24 MHz High-speed external crystal oscillator (HXT) for precise timing operation • 32.768 kHz Low-speed external crystal oscillator (extLXT) for RTC function and low-power system operation • Supports clock failure detection for external crystal oscillators and exception generation (NMI)
Internal Clock Source	• 12 MHz High-speed Internal RC oscillator (HIRC) trimmed to 0.25% accuracy that can optionally be used as a system clock • 48 MHz High-speed Internal RC oscillator (HIRC48) trimmed to 0.25% accuracy that can optionally be used as a system clock • 32 kHz Low-speed Internal RC oscillator (LIRC32) for RTC function • Up to 200 MHz on-chip PLL, sourced from HIRC or HXT, allows CPU operation up to the maximum CPU frequency without the need for a high-frequency crystal
Real-Time Clock (RTC)	• Real-Time Clock with a separate power domain • The RTC clock source includes Low-speed external crystal oscillator (extLXT) and 32 kHz Low-speed Internal RC oscillator (LIRC32) • The RTC block includes 80 bytes of battery-powered backup registers, which can be cleared by tamper pins • Supports 6 static and dynamic tamper pins • Able to wake up CPU from any reduced power mode • Supports Alarm registers (second, minute, hour, day, month, year) • Supports RTC Time Tick and Alarm Match interrupt • Automatic leap year recognition

		• Supports 1 Hz clock output for calibration • Frequency of RTC clock source compensated by RTC_FRWQADJ register
Timers		
32-bit Timer	**TIMER**	• Four sets of 32-bit timers with 24-bit up counter and one 8-bit pre-scale counter from independent clock source • One-shot, Periodic, Toggle and Continuous Counting operation modes • Supports event counting function to count the event from external pins • Supports external capture pin for interval measurement and resetting 24-bit up counter • Supports chip wake-up function, if a timer interrupt signal is generated
	PWM	• Eight 16-bit PWM counters with 12-bit clock prescale with up to 64 MHz • Supports 12-bit deadband (dead time) • Up, down or up-down PWM counter type • Supports brake function • Supports mask function and tri-state output for each PWM channel
Enhanced PWM (EPWM)		• Twelve 16-bit counters with 12-bit clock prescale for twelve 36 MHz PWM output channels • Up to 12 independent input capture channels with 16-bit resolution counter • Supports dead time with maximum divided 12-bit prescale • Up, down or up-down PWM counter type • Supports complementary mode for 3 complementary paired PWM output channels • Synchronous function for phase control • Counter synchronous start function • Brake function with auto recovery mechanism • Mask function and tri-state output for each PWM channel • Able to trigger EADC or DAC to start conversion
Basic PWM (BPWM)		• Two 16-bit counters with 12-bit clock prescale for twelve 36 MHz PWM output channels • Up to 6 independent input capture channels with 16-bit resolution counter • Up, down or up-down PWM counter type • Counter synchronous start function • Complementary mode for 3 complementary paired PWM

	output channels • Mask function and tri-state output for each PWM channel • Able to trigger EADC to start conversion
Watchdog	• 18-bit free running up counter for WDT time-out interval • Supports multiple clock sources from LIRC (default selection), HCLK/2048 and LXT with 8 selectable time-out period • Able to wake up system from Power-down or Idle mode • Time-out event to trigger interrupt or reset system • Supports four WDT reset delay periods, including 1026, 130, 18 or 3 WDT_CLK reset delay period • Configured to force WDT enabled on chip power-on or reset
Window Watchdog	• Clock sourced from HCLK/2048 or LIRC; the window set by 6-bit down counter with 11-bit prescale • Suspended in Idle/Power-down mode
Analog Interfaces	
Enhanced Analog-to-Digital Converter (EADC)	• One 12-bit, 19-ch SAR EADC with up to 16 single-ended input channels or 8 differential input pairs; 10-bit accuracy is guaranteed • Three internal channels for V_{BAT}, band-gap VBG input and Temperature sensor input • Supports external V_{REF} pin or internal reference voltage V_{REF}: 1.6V, 2.0V, 2.5V, and 3.0V • Two power saving modes: Power-down mode and Standby mode. • Supports calibration capability • Analog-to-Digital conversion can be triggered by software enable, external pin, Timer 0~3 overflow pulse trigger or EPWM trigger • Configurable EADC sampling time • Up to 19 sample modules • Double data buffers for sample module 0~3 • PDMA operation
Digital-to-Analog Converter (DAC)	• Two 12-bit, 1 MSPS voltage type DAC with 8-bit mode and 8μs rail-to-rail settle time • Maximum output voltage AV_{DD} -0.2V at buffer mode • Digital-to-Analog conversion triggered by Timer0~3, EPWM0, EPWM1, external trigger pin to start DAC conversion or software • Supports group mode for synchronized data update of two DACs • PDMA operation

Analog Comparator (ACMP)	• Two rail-to-rail Analog Comparators • Supports four multiplexed I/O pins at positive input • Supports I/O pins, band-gap, DAC output, and 16-level Voltage divider from AVDD or VREF at negative input • Supports four programmable propagation speeds for power saving. • Supports wake up from Power-down by interrput • Supports triggers for brake events and cycle-by-cycle control for PWM • Supports window compare mode and window latch mode • Supports programmable hysteresis window: 0mV, 10mV, 20mV and 30mV
LCD	• Supports the following COM/SEG configurations: – 320 dots (8-COM x 40-SEG) – 252 dots (6-COM x 42-SEG) – 176 dots (4-COM x 44-SEG) – 104 dots (8-COM x 13-SEG) for M2354SIFAE • Supports maximum 8 COM driving pins, multiplexed with GPIO pins • Supports maximum 44 SEG driving pins, multiplexed with GPIO pins • Supports 3 bias voltage levels 1/2, 1/3, and 1/4 • Supports 8 duty ratios 1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, and 1/8 • Supports clock frequency divider from 2, 4, 6… 2048 to configure the LCD operating frequency • Configurable frame counting event interrupt period • Supports LCD blinking display controlled by frame counting event. • Supports LCD frame end interrupt • LCD keeps display or blinking even if in Power-down mode when LCD clock source is selected as LIRC or LXT • Supports both type A and type B driving waveforms
Communication Interfaces	
Low-power UART	• Auto-Baud Rate measurement and baud rate compensation function • Supports low power UART (LPUART): baud rate clock from LXT(32.768 kHz) with 9600bps in Power-down mode even system clock is stopped • 16-byte FIFOs with programmable level trigger • Auto flow control (nCTS and nRTS) • Supports IrDA (SIR) function • Supports LIN function on UART0 and UART1

		• Supports RS-485 9-bit mode and direction control • Supports nCTS, incoming data, Received Data FIFO reached threshold and RS-485 Address Match (AAD mode) wake-up function in idle mode • Supports hardware or software enables to program nRTS pin to control RS-485 transmission direction • Supports wake-up function • 8-bit receiver FIFO time-out detection function • Supports break error, frame error, parity error and receive/transmit FIFO overflow detection function • PDMA operation
Smart Card Interface		• Three sets of ISO-7816-3 which are compliant with ISO-7816-3 T=0, T=1 • Supports full duplex UART function • 4-byte FIFOs with programmable level trigger • Programmable guard time selection (11 ETU ~ 266 ETU) • One 24-bit and two 8 bit time-out counters for Answer to Request (ATR) and waiting times processing • Auto inverse convention function • Stop clock level and clock stop (clock keep) function • Transmitter and receiver error retry function • Supports hardware activation, deactivation and warm reset sequence process • Supports hardware auto deactivation sequence after card removal
I^2C		• Three sets of I^2C devices with Master/Slave mode • Supports Standard mode (100 kbps), Fast mode (400 kbps) and Fast mode plus (1 Mbps) • Supports 10 bits mode • Programmable clocks allowing for versatile rate control • Supports multiple address recognition (four slave address with mask option) • Supports SMBus and PMBus • Supports multi-address power-down wake-up function • PDMA operation
SPI/I^2S	**SPI**	• Up to four sets of SPI/I^2S controllers with Master/Slave mode • SPI/I^2S provides separate 4-level of 32-bit (or 8-level of 16-bit) transmit and receive FIFO buffers • Configurable bit length of a transfer word from 8 to 32-bit • MSB first or LSB first transfer sequence • Byte reorder function

		• Supports Byte or Word Suspend mode • Supports one data channel half-duplex transfer • Supports receive-only mode • PDMA operation
	I²S	• Supports mono and stereo audio data with 8-, 16-, 24- and 32-bit audio data sizes • Supports PCM mode A, PCM mode B, I²S and MSB justified data format • PDMA operation
QSPI		• One set of SPI Quad controller with Master/Slave mode • 2-bit Transfer mode • Dual and Quad I/O Transfer mode • QSPI provides separate 8-level of 32-bit transmit and receive FIFO buffers • Configurable bit length of a transfer word from 8 to 32-bit • MSB first or LSB first transfer sequence • Byte reorder function • Supports Byte or Word Suspend mode • 3-wired, no slave select signal, bi-direction interface • Supports one data channel half-duplex transfer • Supports receive-only mode • PDMA operation
I²S		• One set of I²S interface with Master/Slave mode • Supports mono and stereo audio data with 8-, 16-, 24- and 32-bit word sizes • Two 16-level FIFO data buffers, one for transmitting and the other for receiving • Supports I²S protocols: Philips standard, MSB-justified, and LSB-justified data format • Supports PCM protocols: PCM standard, MSB-justified, and LSB-justified data format • PCM protocol supports TDM multi-channel transmission in one audio sample; the number of data channel can be set as 2, 4, 6 or 8 • PDMA operation
Universal Serial Control Interface (USCI)	**UART**	• Two sets of USCI, configured as UART, SPI or I²C function • Supports single byte TX and RX buffer mode • Supports one transmit buffer and two receive buffers for data payload • Supports hardware auto flow control function and

	programmable flow control trigger level • 9-bit Data Transfer • Baud rate detection by built-in capture event of baud rate generator • Supports wake-up function • PDMA operation
	SPI • Supports Master or Slave mode operation • Supports one transmit buffer and two receive buffer for data payload • Configurable bit length of a transfer word from 4 to 16-bit • Supports MSB first or LSB first transfer sequence • Supports Word Suspend function • Supports 3-wire, no slave select signal, bi-direction interface • Supports wake-up function by slave select signal in slave mode • Supports one data channel half-duplex transfer • PDMA operation
	I^2C • Supports master and slave device capability • Supports one transmit buffer and two receive buffer for data payload • Communication in standard mode (100 kbps), fast mode (up to 400 kbps), and Fast mode plus (1 Mbps) • Supports 10-bit mode • Supports 10-bit bus time out capability • Supports bus monitor mode • Supports power-down wake-up by data toggle or address match • Supports multiple address recognition • Supports device address flag • Programmable setup/hold time
Controller Area Network (CAN)	• One set of CAN 2.0B controller • Each supports 32 Message Objects; each Message Object has its own identifier mask • Programmable FIFO mode (concatenation of Message Object) • Disabled Automatic Re-transmission mode for Time Triggered CAN applications • Supports power-down wake-up function
Secure Digital Host Controller (SDHC)	• One set of Secure Digital Host Controller, compliant with SD Memory Card Specification Version 2.0

	• Supports 36 MHz to achieve 192 Mbps at 3.3V operation • Supports dedicated DMA master with Scatter-Gather function to accelerate the data transfer between system memory and SD/SDHC/SDIO card
External Bus Interface (EBI)	• Supports up to three memory banks with individual adjustment of timing parameter • Each bank supports dedicated external chip select pin with polarity control and up to 1 MB addressing space • 8-/16-bit data width • Supports byte write in 16-bit data width mode • Supports variable external bus base clock (MCLK) which based on HCLK • Configurable idle cycle for different access condition: Idle of Write command finish (W2X) and Idle of Read-to-Read (R2R) • Supports Address/Data multiplexed mode • Supports address bus and data bus separate mode • Supports LCD interface i80 mode • PDMA operation
GPIO	• Supports four I/O modes: Quasi bi-direction, Push-Pull output, Open-Drain output and Input only with high impendence mode • Selectable TTL/Schmitt trigger input • Configured as interrupt source with edge/level trigger setting • Supports independent pull-up/pull-down control • Supports high driver and high sink current I/O • Supports software selectable slew rate control • Supports 5V-tolerance function except analog I/O • Improve access efficiency by using single cycle I/O bus
Control Interfaces	
Quadrature Encoder Interface (QEI)	• Two QEI phase inputs (QEI_A, QEI_B) and one Index input (QEI_INDEX) • Supports 2/4 times free-counting mode and 2/4 compare-counting mode • Supports encoder pulse width measurement mode with ECAP
Enhanced Capture (ECAP)	• Input Capture Timer/Counter • Supports three input channels with independent capture counter hold register • 24-bit Input Capture up-counting timer/counter supports captured events reset and/or reload capture counter • Supports rising edge, falling edge and both edge detector options with noise filter in front of input ports

	• Supports compare-match function
Advanced Connectivity	
USB 2.0 Full Speed with on-chip transceiver	**USB 2.0 Full Speed OTG (On-The-Go)** • On-chip USB 2.0 full speed OTG transceiver • Compliant with USB OTG Supplement 2.0 • Configurable as host-only, device-only, ID-dependent or OTG device **USB 1.1 Host Controller** • Compliant with USB Revision 1.1 Specification • Compatible with OHCI (Open Host Controller Interface) Revision 1.0 • Supports full-speed (12Mbps) and low-speed (1.5Mbps) USB devices • Supports Control, Bulk, Interrupt, Isochronous and Split transfers • Integrated a port routing logic to route full/low speed device to OHCI controller • Supports an integrated Root Hub • Supports port power control and port overcurrent detection • Built-in DMA **USB 2.0 Full Speed Device Controller** • Compliant with USB Revision 2.0 Specification • Supports suspend function when no bus activity existing for 3 ms • 12 configurable endpoints for configurable Isochronous, Bulk, Interrupt and Control transfer types • 1024 bytes configurable RAM for endpoint buffer • Remote wake-up capability
Cryptography Accelerator	
Elliptic Curve Cryptography (ECC)	• Hardware ECC accelerator • Supports both prime field GF(p) and binary field GF(2m) • Supports NIST P-192, P-224, P-256, P-384 and P-521 curve sizes • Supports NIST B-163, B-233, B-283, B-409 and B-571 curve sizes • Supports NIST K-163, K-233, K-283, K-409 and K-571 curve sizes • Supports Curve25519 • Supports point multiplication, addition and doubling operations in GF(p) and GF(2m) • Supports modulus division, multiplication, addition and subtraction operations in GF(p) • Supports three techniques to improve side-channel attack

	protection ability • Supports Public Key Cryptographic Algorithm SM2 Based on Elliptic Curves
Advanced Encryption Standard (AES)	• Hardware AES accelerator • Supports 128-bit, 192-bit and 256-bit key length and key expander, and is compliant with FIPS 197 • Supports ECB, CBC, CFB, OFB, CTR, CBC-CS1, CBC-CS2 and CBC-CS3 block cipher modes • Compliant with NIST SP800-38A and addendum • Supports SM4 cipher block alogrithm
Secure Hash Algorithm (SHA)	• Hardware SHA accelerator • Supports SHA-160, SHA-224, SHA-256, SHA-384, and SHA-512 • Compliant with FIPS 180/180-2 • Supprots SM3 Cryptographic Hash Alogrithm
Rivest, Shamir and Adleman Cryptography (RSA)	• Hardware RSA accelerator • Supports both encryption and decryption with 1024, 2048, 3072 and 4096 bits • Supports Chinese Remainder Theorem (CRT) decryption with 2048, 3072 and 4096 bits • Supports three techniques to improve side-channel attack protection ability
Pseudo Random Number Generator (PRNG)	• Supports 128, 163, 192, 224, 233, 255, 256, 283, 384, 409, 512, 521 and 571 bits random number generation (283~571 bits only generated for Key Store) • Able to take the true random number seed from TRNG
True Randon Number Generator (TRNG)	• Up to 800 random bits per second • Provides the true random number seed for PRNG
Key Store	
Key Store	• Supports programming interface for key management • Supports multiple key size from 128 bits to 4096 bits • Supports 4 Kbytes SRAM, 2 Kbytes Flash and 544bytes OTP for key storage • Supports 32 keys for SRAM, 32 keys for Flash and 8 keys for OTP at most • Supports crypto engine access or store key in key store directly • Supports ECDH operation with ECC and PRNG engine • Supports to store middle data for RSA CRT and SCAP mode • Supports revoke operation for each key • Supports erase key in SRAM/Flash and revoke key in OTP

	while tamper detected • Supports integrity checking • Supports data scrambling at SRAM, Flash and OTP • Supports data remanence prevention at SRAM • Supports silent access for side-channel protection at SRAM, Flash and OTP
Attack Detection	
Attack Detection (TAMPER)	• Includes voltage, clock and I/O tamper detectors: – Voltage: ◆ HV detector detects if V_{DD} › 4.0V ◆ LV detector detects if LDO_CAP 〉 ± 20% – Clock detector: ◆ detects if external clock (LXT) is failed or stopped – I/O tamper detector: ◆ detects GPF6~11 pins • Provides event response after an attack detected: – Clear key or data content in SRAM and Flash of Key Store, and revoke the OTP in Key Store – Clear RTC spare register – Reset Crypto – Chip reset – Interrupt – Wake up the system • Not supported in DPD mode.

3 PARTS INFORMATION

3.1 Package Type

Part No.	LQFP48	LQFP64	LQFP128
M2354	M2354LJFAE	M2354SJFAE	M2354KJFAE

3.2 M2354 Series Selection Guide

PART NUMBER			M2354					
			LJFAE		SJFAE		KJFAE	
Flash (KB)			1024		1024		1024	
SRAM (KB)			256		256		256	
ISP Loader ROM (KB)			16					
I/O			40		50		106	
32-bit Timer			4					
Tamper I/O			1		1		6	
RTC			√					
Connectivity	LPUART		6					
	ISO-7816		3					
	Quad SPI	SPI/I²S	1	3	1	4	1	4
	I²S		1					
	I²C		3					
	USCI (UART/I²C/SPI)		2					
	CAN		1					
	LIN		2					
	SDHC		1					
Crypto	TRNG		√					
	AES		√					
	ECC		√					
	SHA/HMAC		√					
	RSA		√					
Enhanced Security	FVC		√					
	DPM		√					
	PLM		√					
	Key Store		√					
Power Glitch Detector			√					
LCD (COMXSEG)			-		8 X 13		8 X 40	
16-bit Enhanced PWM			12					
16-bit Basic PWM			12					
QEI			2					
ECAP			1					
USB 2.0 FS OTG			√					
12-bit ADC			11		16		16	
12-bit DAC			2					
Analog Comparator			2					
External Bus Interface			√		√		√	
Package			LQFP48		LQFP 64		LQFP 128	

Table 3-1 M2354 Series Selection Guide

3.3 M2354 Series Selection Code

M23	54	K	J	F	A	E
Secure Core	Line	Package	Flash	SRAM	Rev.	Temperature
Cortex®-M23	54: Ultra Line (Segment LCD)	L: LQFP48 (7x7 mm) S: LQFP64 (7x7 mm) K: LQFP128 (14x14 mm)	J: 1024 KB	F: 256 kB		E: -40°C~105°C

Table 3-2 M2354 Series Selection Code

4 PIN CONFIGURATION

Users can find pin configuration information in the M2354 Multi-function Pin diagram sections or by using NuTool - PinConfigure. The NuTool - PinConfigure contains all NuMicro® Family chip series with all part number, and helps users configure GPIO multi-function correctly and handily.

4.1 Pin Configuration

4.1.1 M2354 Pin Diagram

4.1.1.1 M2354 LQFP 48-Pin Diagram

Corresponding Part Number: M2354LJFAE

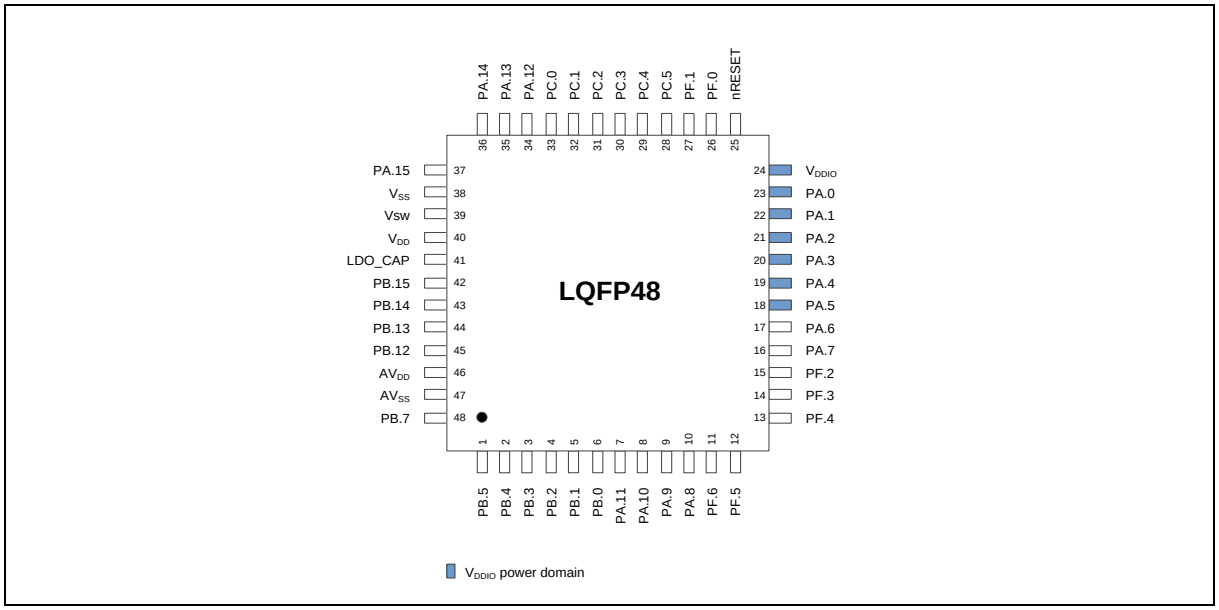


Figure 4.1-1 M2354 LQFP 48-pin Diagram

4.1.1.2 M2354 LQFP 64-Pin Diagram

Corresponding Part Number: M2354SJFAE

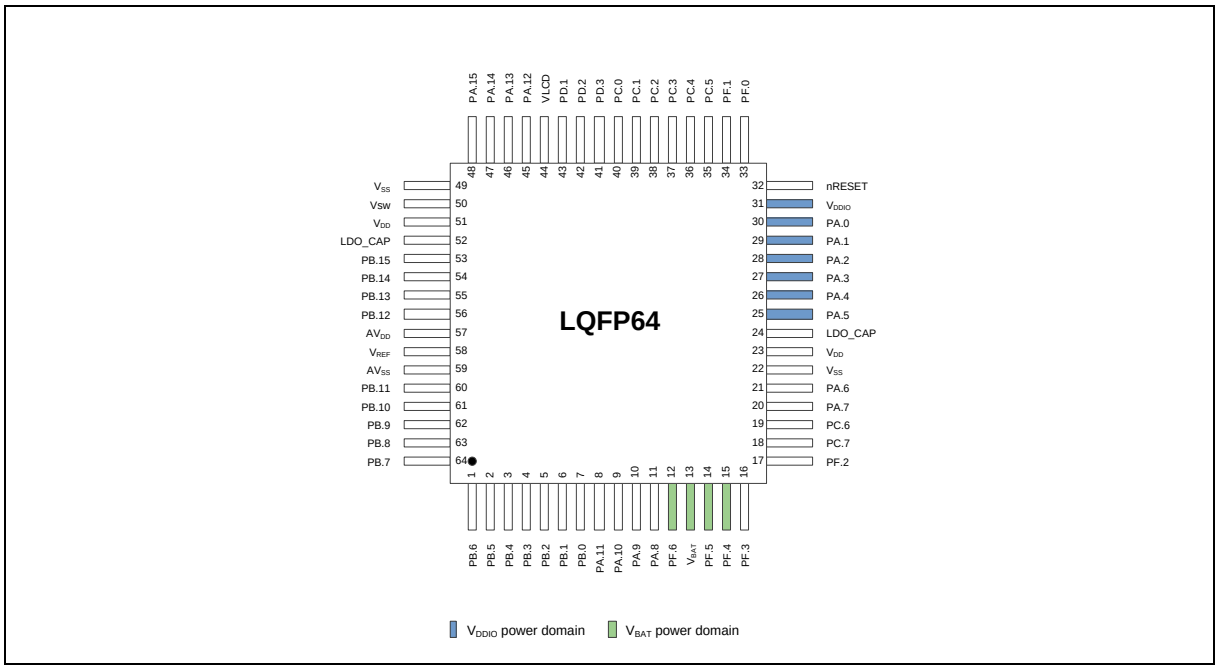


Figure 4.1-2 M2354 LQFP 64-pin Diagram

4.1.1.3 M2354 LQFP 128-Pin Diagram

Corresponding Part Number: M2354KJFAE

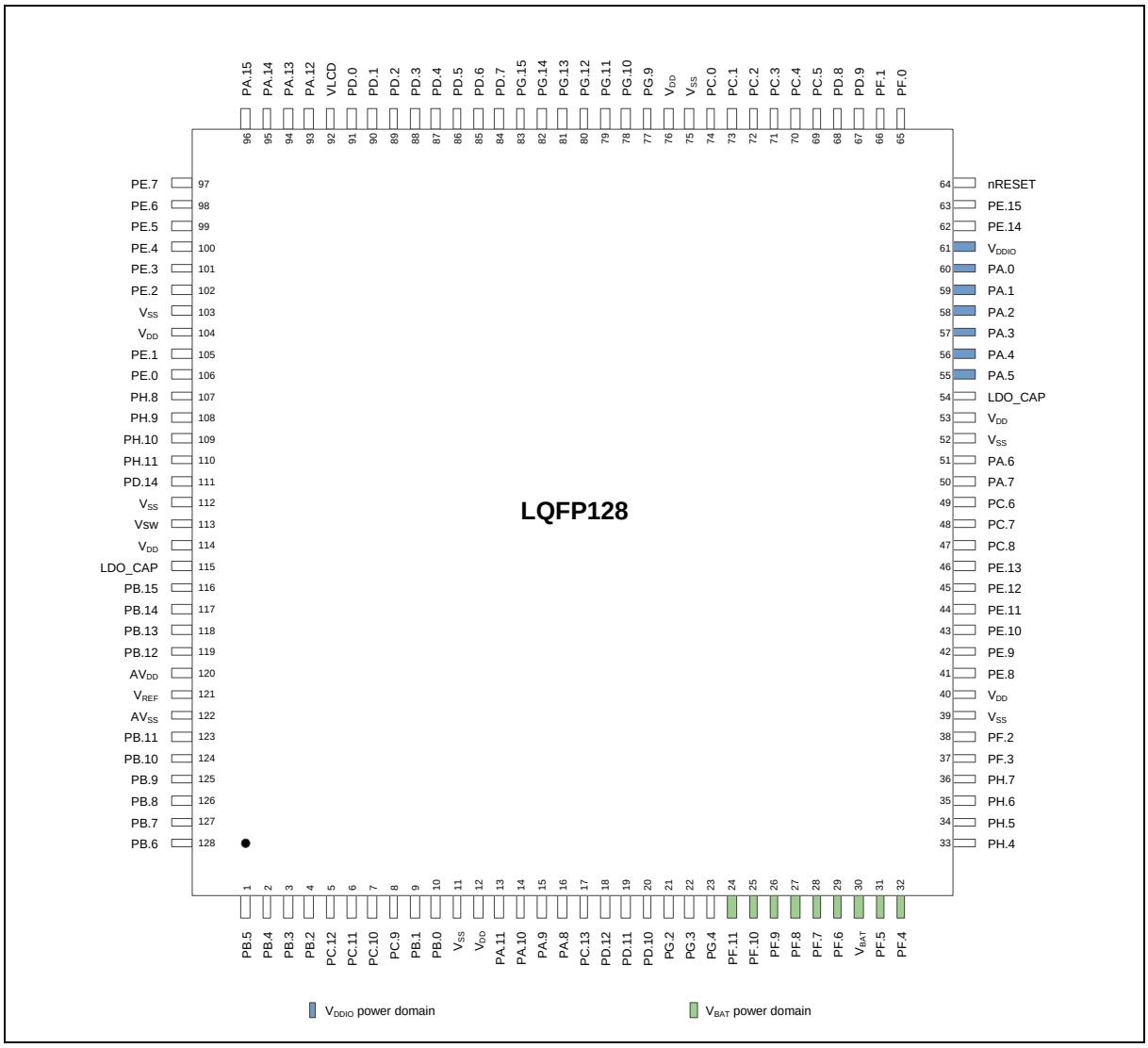


Figure 4.1-3 M2354 LQFP 128-pin Diagram

4.1.2 M2354 Multi-Function Pin Diagram

4.1.2.1 M2354 LQFP 48-Pin Multi-function Pin Diagram

Corresponding Part Number: M2354LJFAE

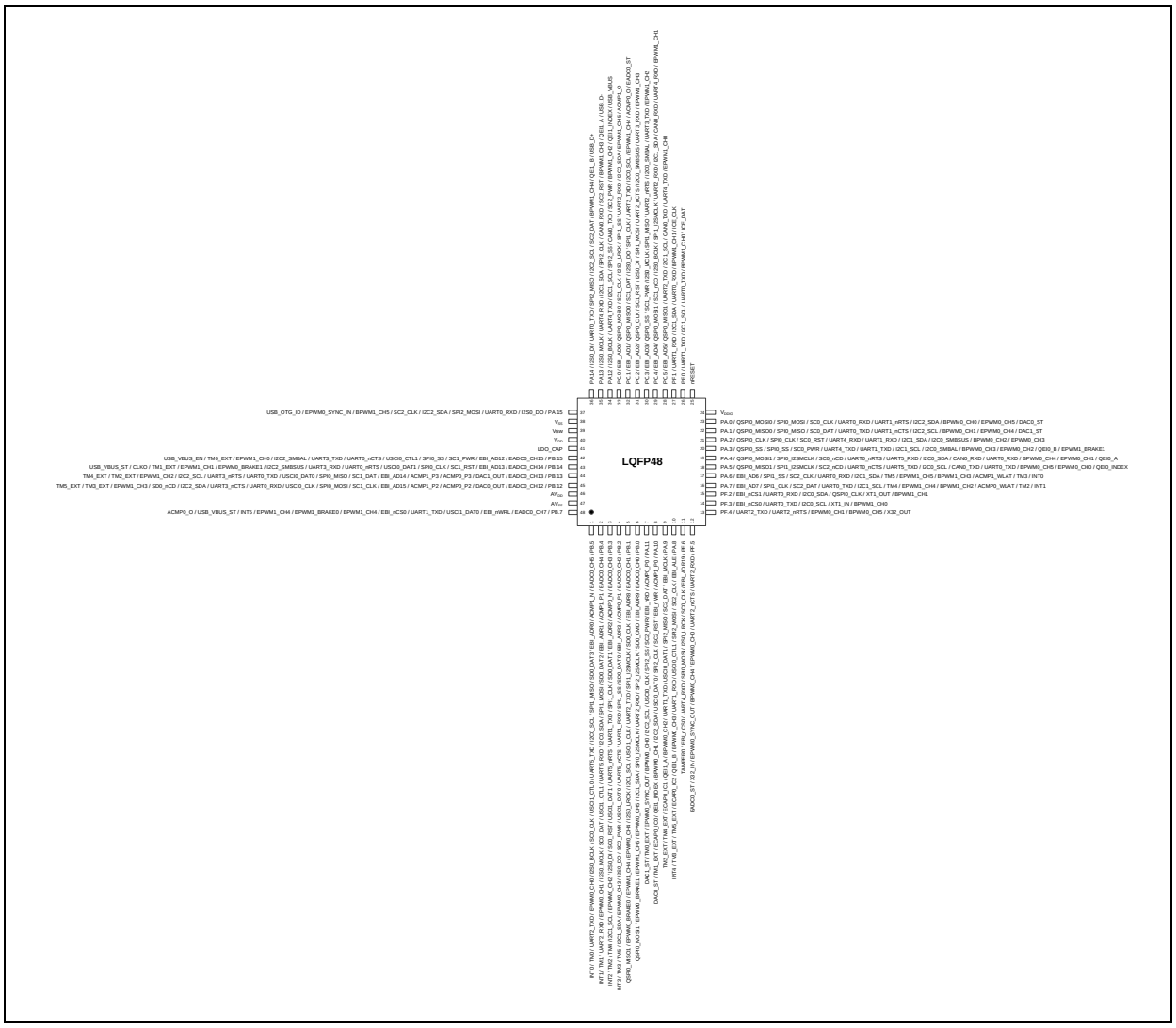


Figure 4.1-4 M2354LJFAE Multi-function Pin Diagram

Pin	M2354LJFAE Pin Function
1	PB.5 / EADC0_CH5 / ACMP1_N / EBI_ADR0 / SD0_DAT3 / SPI1_MISO / I2C0_SCL / UART5_TXD / USCI1_CTL0 / SC0_CLK / I2S0_BCLK / EPWM0_CH0 / UART2_TXD / TM0 / INT0
2	PB.4 / EADC0_CH4 / ACMP1_P1 / EBI_ADR1 / SD0_DAT2 / SPI1_MOSI / I2C0_SDA / UART5_RXD / USCI1_CTL1 / SC0_DAT / I2S0_MCLK / EPWM0_CH1 / UART2_RXD / TM1 / INT1
3	PB.3 / EADC0_CH3 / ACMP0_N / EBI_ADR2 / SD0_DAT1 / SPI1_CLK / UART1_TXD / UART5_nRTS / USCI1_DAT1 / SC0_RST / I2S0_DI / EPWM0_CH2 / I2C1_SCL / TM4 / TM2 / INT2
4	PB.2 / EADC0_CH2 / ACMP0_P1 / EBI_ADR3 / SD0_DAT0 / SPI1_SS / UART1_RXD / UART5_nCTS / USCI1_DAT0 / SC0_PWR / I2S0_DO / EPWM0_CH3 / I2C1_SDA / TM5 / TM3 / INT3

Pin	M2354LJFAE Pin Function
5	PB.1 / EADC0_CH1 / EBI_ADR8 / SD0_CLK / SPI1_I2SMCLK / UART2_TXD / USCI1_CLK / I2C1_SCL / I2S0_LRCK / EPWM0_CH4 / EPWM1_CH4 / EPWM0_BRAKE0 / QSPI0_MISO1
6	PB.0 / EADC0_CH0 / EBI_ADR9 / SD0_CMD / SPI2_I2SMCLK / UART2_RXD / SPI0_I2SMCLK / I2C1_SDA / EPWM0_CH5 / EPWM1_CH5 / EPWM0_BRAKE1 / QSPI0_MOSI1
7	PA.11 / ACMP0_P0 / EBI_nRD / SC2_PWR / SPI2_SS / USCI0_CLK / I2C2_SCL / BPWM0_CH0 / EPWM0_SYNC_OUT / TM0_EXT / DAC1_ST
8	PA.10 / ACMP1_P0 / EBI_nWR / SC2_RST / SPI2_CLK / USCI0_DAT0 / I2C2_SDA / BPWM0_CH1 / QEI1_INDEX / ECAP0_IC0 / TM1_EXT / DAC0_ST
9	PA.9 / EBI_MCLK / SC2_DAT / SPI2_MISO / USCI0_DAT1 / UART1_TXD / BPWM0_CH2 / QEI1_A / ECAP0_IC1 / TM4_EXT / TM2_EXT
10	PA.8 / EBI_ALE / SC2_CLK / SPI2_MOSI / USCI0_CTL1 / UART1_RXD / BPWM0_CH3 / QEI1_B / ECAP0_IC2 / TM5_EXT / TM3_EXT / INT4
11	PF.6 / EBI_ADR19 / SC0_CLK / I2S0_LRCK / SPI0_MOSI / UART4_RXD / EBI_nCS0 / TAMPER0
12	PF.5 / UART2_RXD / UART2_nCTS / EPWM0_CH0 / BPWM0_CH4 / EPWM0_SYNC_OUT / X32_IN / EADC0_ST
13	PF.4 / UART2_TXD / UART2_nRTS / EPWM0_CH1 / BPWM0_CH5 / X32_OUT
14	PF.3 / EBI_nCS0 / UART0_TXD / I2C0_SCL / XT1_IN / BPWM1_CH0
15	PF.2 / EBI_nCS1 / UART0_RXD / I2C0_SDA / QSPI0_CLK / XT1_OUT / BPWM1_CH1
16	PA.7 / EBI_AD7 / SPI1_CLK / SC2_DAT / UART0_TXD / I2C1_SCL / TM4 / EPWM1_CH4 / BPWM1_CH2 / ACMP0_WLAT / TM2 / INT1
17	PA.6 / EBI_AD6 / SPI1_SS / SC2_CLK / UART0_RXD / I2C1_SDA / TM5 / EPWM1_CH5 / BPWM1_CH3 / ACMP1_WLAT / TM3 / INT0
18	PA.5 / QSPI0_MISO1 / SPI1_I2SMCLK / SC2_nCD / UART0_nCTS / UART5_TXD / I2C0_SCL / CAN0_TXD / UART0_TXD / BPWM0_CH5 / EPWM0_CH0 / QEI0_INDEX
19	PA.4 / QSPI0_MOSI1 / SPI0_I2SMCLK / SC0_nCD / UART0_nRTS / UART5_RXD / I2C0_SDA / CAN0_RXD / UART0_RXD / BPWM0_CH4 / EPWM0_CH1 / QEI0_A
20	PA.3 / QSPI0_SS / SPI0_SS / SC0_PWR / UART4_TXD / UART1_TXD / I2C1_SCL / I2C0_SMBAL / BPWM0_CH3 / EPWM0_CH2 / QEI0_B / EPWM1_BRAKE1
21	PA.2 / QSPI0_CLK / SPI0_CLK / SC0_RST / UART4_RXD / UART1_RXD / I2C1_SDA / I2C0_SMBSUS / BPWM0_CH2 / EPWM0_CH3
22	PA.1 / QSPI0_MISO0 / SPI0_MISO / SC0_DAT / UART0_TXD / UART1_nCTS / I2C2_SCL / BPWM0_CH1 / EPWM0_CH4 / DAC1_ST
23	PA.0 / QSPI0_MOSI0 / SPI0_MOSI / SC0_CLK / UART0_RXD / UART1_nRTS / I2C2_SDA / BPWM0_CH0 / EPWM0_CH5 / DAC0_ST
24	vref
25	nRESET
26	PF.0 / UART1_TXD / I2C1_SCL / UART0_TXD / BPWM1_CH0 / ICE_DAT
27	PF.1 / UART1_RXD / I2C1_SDA / UART0_RXD / BPWM1_CH1 / ICE_CLK
28	PC.5 / EBI_AD5 / QSPI0_MISO1 / UART2_TXD / I2C1_SCL / CAN0_TXD / UART4_TXD / EPWM1_CH0
29	PC.4 / EBI_AD4 / QSPI0_MOSI1 / SC1_nCD / I2S0_BCLK / SPI1_I2SMCLK / UART2_RXD / I2C1_SDA / CAN0_RXD / UART4_RXD / EPWM1_CH1
30	PC.3 / EBI_AD3 / QSPI0_SS / SC1_PWR / I2S0_MCLK / SPI1_MISO / UART2_nRTS / I2C0_SMBAL / UART3_TXD /

Pin	M2354LJFAE Pin Function
	EPWM1_CH2
31	PC.2 / EBI_AD2 / QSPI0_CLK / SC1_RST / I2S0_DI / SPI1_MOSI / UART2_nCTS / I2C0_SMBSUS / UART3_RXD / EPWM1_CH3
32	PC.1 / EBI_AD1 / QSPI0_MISO0 / SC1_DAT / I2S0_DO / SPI1_CLK / UART2_TXD / I2C0_SCL / EPWM1_CH4 / ACMP0_O / EADC0_ST
33	PC.0 / EBI_AD0 / QSPI0_MOSI0 / SC1_CLK / I2S0_LRCK / SPI1_SS / UART2_RXD / I2C0_SDA / EPWM1_CH5 / ACMP1_O
34	PA.12 / I2S0_BCLK / UART4_TXD / I2C1_SCL / SPI2_SS / CAN0_TXD / SC2_PWR / BPWM1_CH2 / QEI1_INDEX / USB_VBUS
35	PA.13 / I2S0_MCLK / UART4_RXD / I2C1_SDA / SPI2_CLK / CAN0_RXD / SC2_RST / BPWM1_CH3 / QEI1_A / USB_D-
36	PA.14 / I2S0_DI / UART0_TXD / SPI2_MISO / I2C2_SCL / SC2_DAT / BPWM1_CH4 / QEI1_B / USB_D+
37	PA.15 / I2S0_DO / UART0_RXD / SPI2_MOSI / I2C2_SDA / SC2_CLK / BPWM1_CH5 / EPWM0_SYNC_IN / USB_OTG_ID
38	V_{SS}
39	Vsw
40	V_{DD}
41	LDO_CAP
42	PB.15 / EADC0_CH15 / EBI_AD12 / SC1_PWR / SPI0_SS / USCI0_CTL1 / UART0_nCTS / UART3_TXD / I2C2_SMBAL / EPWM1_CH0 / TM0_EXT / USB_VBUS_EN
43	PB.14 / EADC0_CH14 / EBI_AD13 / SC1_RST / SPI0_CLK / USCI0_DAT1 / UART0_nRTS / UART3_RXD / I2C2_SMBSUS / EPWM0_BRAKE1 / EPWM1_CH1 / TM1_EXT / CLKO / USB_VBUS_ST
44	PB.13 / EADC0_CH13 / DAC1_OUT / ACMP0_P3 / ACMP1_P3 / EBI_AD14 / SC1_DAT / SPI0_MISO / USCI0_DAT0 / UART0_TXD / UART3_nRTS / I2C2_SCL / EPWM1_CH2 / TM2_EXT / TM4_EXT
45	PB.12 / EADC0_CH12 / DAC0_OUT / ACMP0_P2 / ACMP1_P2 / EBI_AD15 / SC1_CLK / SPI0_MOSI / USCI0_CLK / UART0_RXD / UART3_nCTS / I2C2_SDA / SD0_nCD / EPWM1_CH3 / TM3_EXT / TM5_EXT
46	AV_{DD}
47	AV_{SS}
48	PB.7 / EADC0_CH7 / EBI_nWRL / USCI1_DAT0 / UART1_TXD / EBI_nCS0 / BPWM1_CH4 / EPWM1_BRAKE0 / EPWM1_CH4 / INT5 / USB_VBUS_ST / ACMP0_O

Table 4.1-1 M2354LJFAE Multi-function Pin Table

4.1.2.2 M2354 LQFP 64-Pin Multi-function Pin Diagram

Corresponding Part Number: M2354SJFAE

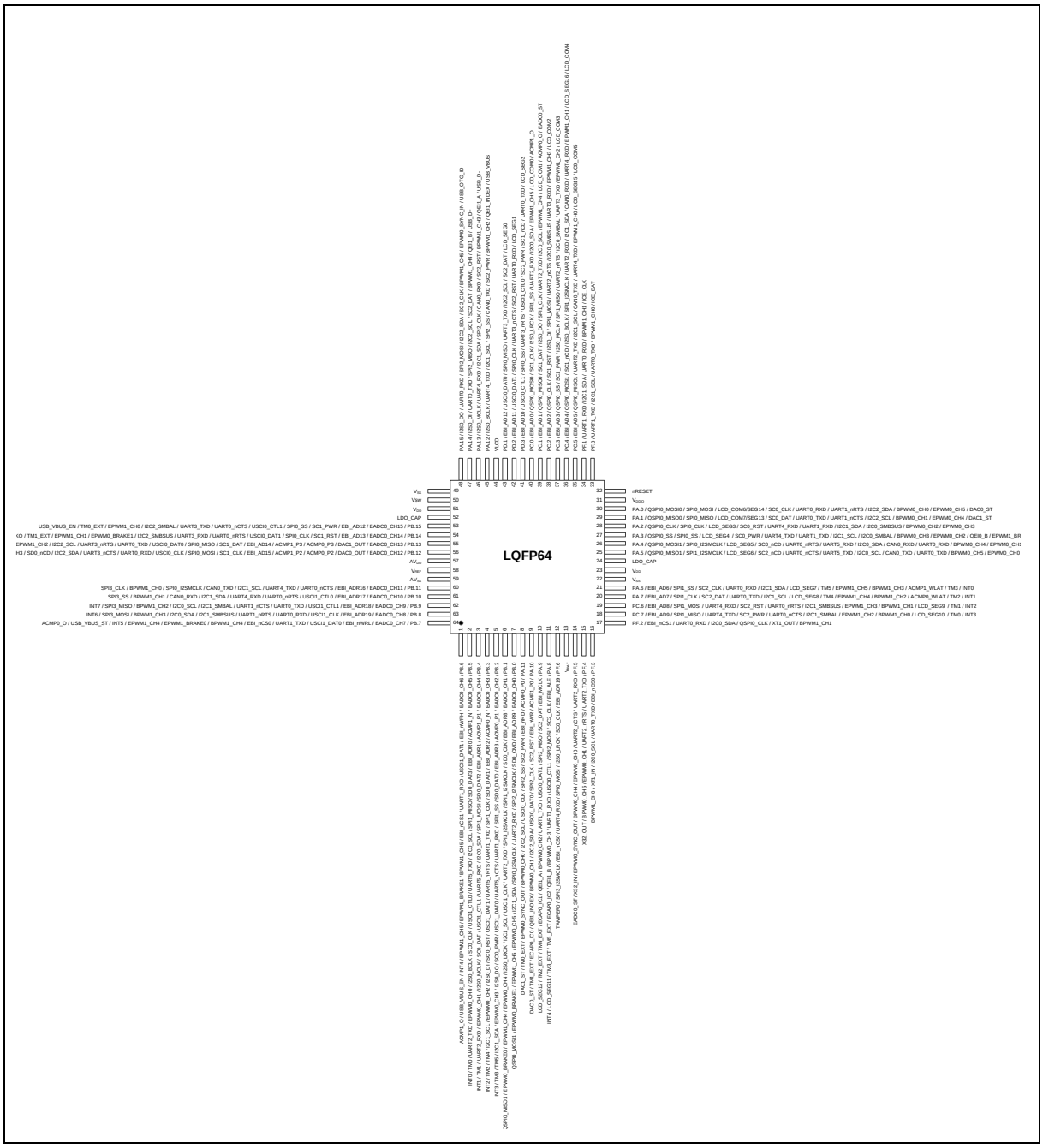


Figure 4.1-5 M2354SJFAE Multi-function Pin Diagram

Pin	M2354SJFAE Pin Function
1	PB.6 / EADC0_CH6 / EBI_nWRH / USCI1_DAT1 / UART1_RXD / EBI_nCS1 / BPWM1_CH5 / EPWM1_BRAKE1 / EPWM1_CH5 / INT4 / USB_VBUS_EN / ACMP1_O

Pin	M2354SJFAE Pin Function
2	PB.5 / EADC0_CH5 / ACMP1_N / EBI_ADR0 / SD0_DAT3 / SPI1_MISO / I2C0_SCL / UART5_TXD / USCI1_CTL0 / SC0_CLK / I2S0_BCLK / EPWM0_CH0 / UART2_TXD / TM0 / INT0
3	PB.4 / EADC0_CH4 / ACMP1_P1 / EBI_ADR1 / SD0_DAT2 / SPI1_MOSI / I2C0_SDA / UART5_RXD / USCI1_CTL1 / SC0_DAT / I2S0_MCLK / EPWM0_CH1 / UART2_RXD / TM1 / INT1
4	PB.3 / EADC0_CH3 / ACMP0_N / EBI_ADR2 / SD0_DAT1 / SPI1_CLK / UART1_TXD / UART5_nRTS / USCI1_DAT1 / SC0_RST / I2S0_DI / EPWM0_CH2 / I2C1_SCL / TM4 / TM2 / INT2
5	PB.2 / EADC0_CH2 / ACMP0_P1 / EBI_ADR3 / SD0_DAT0 / SPI1_SS / UART1_RXD / UART5_nCTS / USCI1_DAT0 / SC0_PWR / I2S0_DO / EPWM0_CH3 / I2C1_SDA / TM5 / TM3 / INT3
6	PB.1 / EADC0_CH1 / EBI_ADR8 / SD0_CLK / SPI1_I2SMCLK / SPI3_I2SMCLK / UART2_TXD / USCI1_CLK / I2C1_SCL / I2S0_LRCK / EPWM0_CH4 / EPWM1_CH4 / EPWM0_BRAKE0 / QSPI0_MISO1
7	PB.0 / EADC0_CH0 / EBI_ADR9 / SD0_CMD / SPI2_I2SMCLK / UART2_RXD / SPI0_I2SMCLK / I2C1_SDA / EPWM0_CH5 / EPWM1_CH5 / EPWM0_BRAKE1 / QSPI0_MOSI1
8	PA.11 / ACMP0_P0 / EBI_nRD / SC2_PWR / SPI2_SS / USCI0_CLK / I2C2_SCL / BPWM0_CH0 / EPWM0_SYNC_OUT / TM0_EXT / DAC1_ST
9	PA.10 / ACMP1_P0 / EBI_nWR / SC2_RST / SPI2_CLK / USCI0_DAT0 / I2C2_SDA / BPWM0_CH1 / QEI1_INDEX / ECAP0_IC0 / TM1_EXT / DAC0_ST
10	PA.9 / EBI_MCLK / SC2_DAT / SPI2_MISO / USCI0_DAT1 / UART1_TXD / BPWM0_CH2 / QEI1_A / ECAP0_IC1 / TM4_EXT / TM2_EXT / LCD_SEG12
11	PA.8 / EBI_ALE / SC2_CLK / SPI2_MOSI / USCI0_CTL1 / UART1_RXD / BPWM0_CH3 / QEI1_B / ECAP0_IC2 / TM5_EXT / TM3_EXT / LCD_SEG11 / INT4
12	PF.6 / EBI_ADR19 / SC0_CLK / I2S0_LRCK / SPI0_MOSI / UART4_RXD / EBI_nCS0 / SPI3_I2SMCLK / TAMPER0
13	V_{BAT}
14	PF.5 / UART2_RXD / UART2_nCTS / EPWM0_CH0 / BPWM0_CH4 / EPWM0_SYNC_OUT / X32_IN / EADC0_ST
15	PF.4 / UART2_TXD / UART2_nRTS / EPWM0_CH1 / BPWM0_CH5 / X32_OUT
16	PF.3 / EBI_nCS0 / UART0_TXD / I2C0_SCL / XT1_IN / BPWM1_CH0
17	PF.2 / EBI_nCS1 / UART0_RXD / I2C0_SDA / QSPI0_CLK / XT1_OUT / BPWM1_CH1
18	PC.7 / EBI_AD9 / SPI1_MISO / UART4_TXD / SC2_PWR / UART0_nCTS / I2C1_SMBAL / EPWM1_CH2 / BPWM1_CH0 / LCD_SEG10 / TM0 / INT3
19	PC.6 / EBI_AD8 / SPI1_MOSI / UART4_RXD / SC2_RST / UART0_nRTS / I2C1_SMBSUS / EPWM1_CH3 / BPWM1_CH1 / LCD_SEG9 / TM1 / INT2
20	PA.7 / EBI_AD7 / SPI1_CLK / SC2_DAT / UART0_TXD / I2C1_SCL / LCD_SEG8 / TM4 / EPWM1_CH4 / BPWM1_CH2 / ACMP0_WLAT / TM2 / INT1
21	PA.6 / EBI_AD6 / SPI1_SS / SC2_CLK / UART0_RXD / I2C1_SDA / LCD_SEG7 / TM5 / EPWM1_CH5 / BPWM1_CH3 / ACMP1_WLAT / TM3 / INT0
22	V_{SS}
23	V_{DD}
24	LDO_CAP
25	PA.5 / QSPI0_MISO1 / SPI1_I2SMCLK / LCD_SEG6 / SC2_nCD / UART0_nCTS / UART5_TXD / I2C0_SCL / CAN0_TXD / UART0_TXD / BPWM0_CH5 / EPWM0_CH0 / QEI0_INDEX
26	PA.4 / QSPI0_MOSI1 / SPI0_I2SMCLK / LCD_SEG5 / SC0_nCD / UART0_nRTS / UART5_RXD / I2C0_SDA / CAN0_RXD / UART0_RXD / BPWM0_CH4 / EPWM0_CH1 / QEI0_A

Pin	M2354SJFAE Pin Function
27	PA.3 / QSPI0_SS / SPI0_SS / LCD_SEG4 / SC0_PWR / UART4_TXD / UART1_TXD / I2C1_SCL / I2C0_SMBAL / BPWM0_CH3 / EPWM0_CH2 / QEI0_B / EPWM1_BRAKE1
28	PA.2 / QSPI0_CLK / SPI0_CLK / LCD_SEG3 / SC0_RST / UART4_RXD / UART1_RXD / I2C1_SDA / I2C0_SMBSUS / BPWM0_CH2 / EPWM0_CH3
29	PA.1 / QSPI0_MISO0 / SPI0_MISO / LCD_COM7/SEG13 / SC0_DAT / UART0_TXD / UART1_nCTS / I2C2_SCL / BPWM0_CH1 / EPWM0_CH4 / DAC1_ST
30	PA.0 / QSPI0_MOSI0 / SPI0_MOSI / LCD_COM6/SEG14 / SC0_CLK / UART0_RXD / UART1_nRTS / I2C2_SDA / BPWM0_CH0 / EPWM0_CH5 / DAC0_ST
31	V_{DDIO}
32	nRESET
33	PF.0 / UART1_TXD / I2C1_SCL / UART0_TXD / BPWM1_CH0 / ICE_DAT
34	PF.1 / UART1_RXD / I2C1_SDA / UART0_RXD / BPWM1_CH1 / ICE_CLK
35	PC.5 / EBI_AD5 / QSPI0_MISO1 / UART2_TXD / I2C1_SCL / CAN0_TXD / UART4_TXD / EPWM1_CH0 / LCD_SEG15 / LCD_COM5
36	PC.4 / EBI_AD4 / QSPI0_MOSI1 / SC1_nCD / I2S0_BCLK / SPI1_I2SMCLK / UART2_RXD / I2C1_SDA / CAN0_RXD / UART4_RXD / EPWM1_CH1 / LCD_SEG16 / LCD_COM4
37	PC.3 / EBI_AD3 / QSPI0_SS / SC1_PWR / I2S0_MCLK / SPI1_MISO / UART2_nRTS / I2C0_SMBAL / UART3_TXD / EPWM1_CH2 / LCD_COM3
38	PC.2 / EBI_AD2 / QSPI0_CLK / SC1_RST / I2S0_DI / SPI1_MOSI / UART2_nCTS / I2C0_SMBSUS / UART3_RXD / EPWM1_CH3 / LCD_COM2
39	PC.1 / EBI_AD1 / QSPI0_MISO0 / SC1_DAT / I2S0_DO / SPI1_CLK / UART2_TXD / I2C0_SCL / EPWM1_CH4 / LCD_COM1 / ACMP0_O / EADC0_ST
40	PC.0 / EBI_AD0 / QSPI0_MOSI0 / SC1_CLK / I2S0_LRCK / SPI1_SS / UART2_RXD / I2C0_SDA / EPWM1_CH5 / LCD_COM0 / ACMP1_O
41	PD.3 / EBI_AD10 / USCI0_CTL1 / SPI0_SS / UART3_nRTS / USCI1_CTL0 / SC2_PWR / SC1_nCD / UART0_TXD / LCD_SEG2
42	PD.2 / EBI_AD11 / USCI0_DAT1 / SPI0_CLK / UART3_nCTS / SC2_RST / UART0_RXD / LCD_SEG1
43	PD.1 / EBI_AD12 / USCI0_DAT0 / SPI0_MISO / UART3_TXD / I2C2_SCL / SC2_DAT / LCD_SEG0
44	V_{LCD}
45	PA.12 / I2S0_BCLK / UART4_TXD / I2C1_SCL / SPI2_SS / CAN0_TXD / SC2_PWR / BPWM1_CH2 / QEI1_INDEX / USB_VBUS
46	PA.13 / I2S0_MCLK / UART4_RXD / I2C1_SDA / SPI2_CLK / CAN0_RXD / SC2_RST / BPWM1_CH3 / QEI1_A / USB_D-
47	PA.14 / I2S0_DI / UART0_TXD / SPI2_MISO / I2C2_SCL / SC2_DAT / BPWM1_CH4 / QEI1_B / USB_D+
48	PA.15 / I2S0_DO / UART0_RXD / SPI2_MOSI / I2C2_SDA / SC2_CLK / BPWM1_CH5 / EPWM0_SYNC_IN / USB_OTG_ID
49	V_{SS}
50	Vsw
51	V_{DD}
52	LDO_CAP

Pin	M2354SJFAE Pin Function
53	PB.15 / EADC0_CH15 / EBI_AD12 / SC1_PWR / SPI0_SS / USCI0_CTL1 / UART0_nCTS / UART3_TXD / I2C2_SMBAL / EPWM1_CH0 / TM0_EXT / USB_VBUS_EN
54	PB.14 / EADC0_CH14 / EBI_AD13 / SC1_RST / SPI0_CLK / USCI0_DAT1 / UART0_nRTS / UART3_RXD / I2C2_SMBSUS / EPWM0_BRAKE1 / EPWM1_CH1 / TM1_EXT / CLKO / USB_VBUS_ST
55	PB.13 / EADC0_CH13 / DAC1_OUT / ACMP0_P3 / ACMP1_P3 / EBI_AD14 / SC1_DAT / SPI0_MISO / USCI0_DAT0 / UART0_TXD / UART3_nRTS / I2C2_SCL / EPWM1_CH2 / TM2_EXT / TM4_EXT
56	PB.12 / EADC0_CH12 / DAC0_OUT / ACMP0_P2 / ACMP1_P2 / EBI_AD15 / SC1_CLK / SPI0_MOSI / USCI0_CLK / UART0_RXD / UART3_nCTS / I2C2_SDA / SD0_nCD / EPWM1_CH3 / TM3_EXT / TM5_EXT
57	AV_{DD}
58	V_{REF}
59	AV_{SS}
60	PB.11 / EADC0_CH11 / EBI_ADR16 / UART0_nCTS / UART4_TXD / I2C1_SCL / CAN0_TXD / SPI0_I2SMCLK / BPWM1_CH0 / SPI3_CLK
61	PB.10 / EADC0_CH10 / EBI_ADR17 / USCI1_CTL0 / UART0_nRTS / UART4_RXD / I2C1_SDA / CAN0_RXD / BPWM1_CH1 / SPI3_SS
62	PB.9 / EADC0_CH9 / EBI_ADR18 / USCI1_CTL1 / UART0_TXD / UART1_nCTS / I2C1_SMBAL / I2C0_SCL / BPWM1_CH2 / SPI3_MISO / INT7
63	PB.8 / EADC0_CH8 / EBI_ADR19 / USCI1_CLK / UART0_RXD / UART1_nRTS / I2C1_SMBSUS / I2C0_SDA / BPWM1_CH3 / SPI3_MOSI / INT6
64	PB.7 / EADC0_CH7 / EBI_nWRL / USCI1_DAT0 / UART1_TXD / EBI_nCS0 / BPWM1_CH4 / EPWM1_BRAKE0 / EPWM1_CH4 / INT5 / USB_VBUS_ST / ACMP0_O

Table 4.1-2 M2354SJFAE Multi-function Pin Table

4.1.2.3 M2354 LQFP 128-Pin Multi-function Pin Diagram

Corresponding Part Number: M2354KJFAE

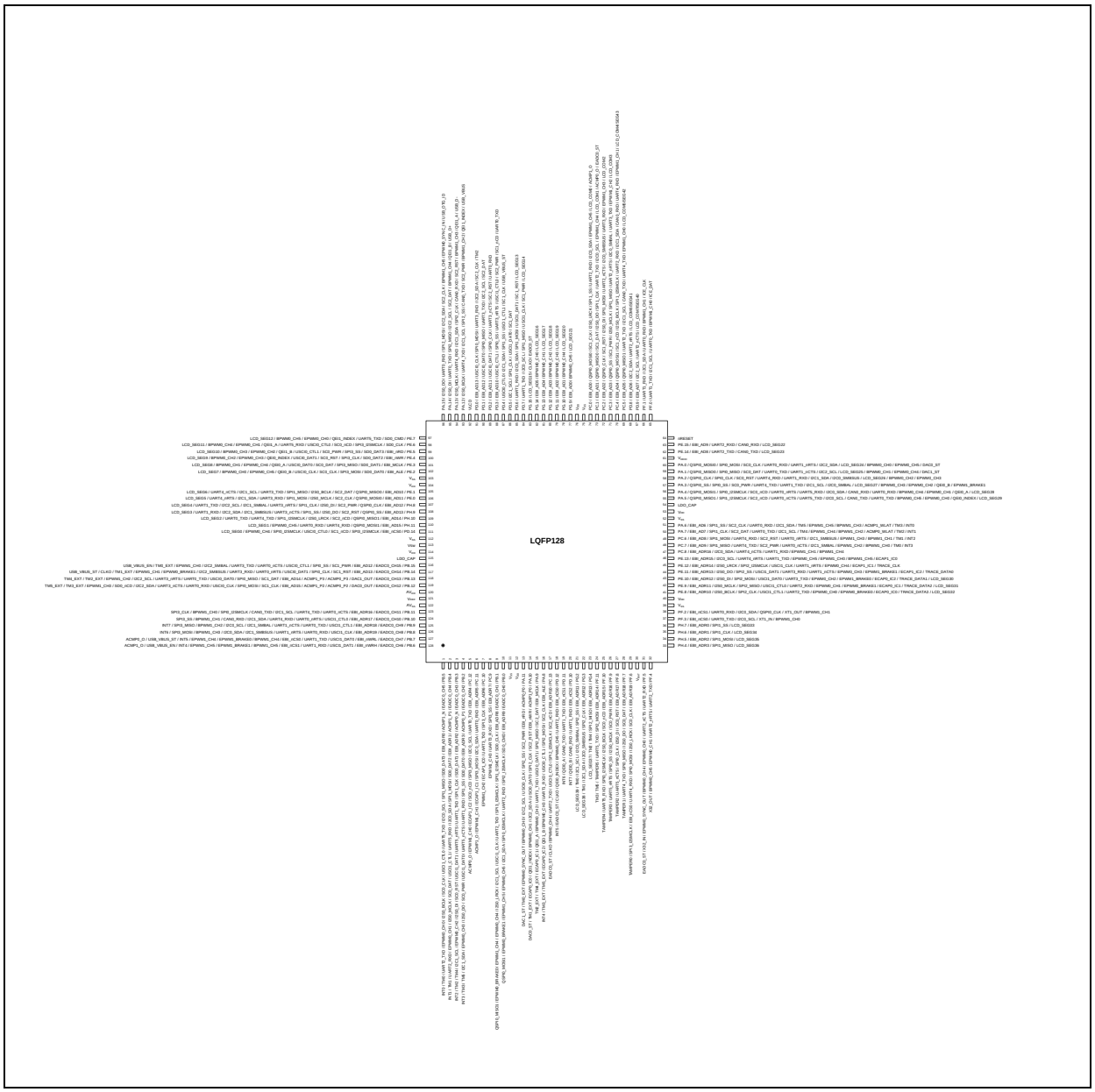


Figure 4.1-6 M2354KJFAE Multi-function Pin Diagram

Pin	M2354KJFAE Pin Function
1	PB.5 / EADC0_CH5 / ACMP1_N / EBI_ADR0 / SD0_DAT3 / SPI1_MISO / I2C0_SCL / UART5_TXD / USCI1_CTL0 / SC0_CLK / I2S0_BCLK / EPWM0_CH0 / UART2_TXD / TM0 / INT0
2	PB.4 / EADC0_CH4 / ACMP1_P1 / EBI_ADR1 / SD0_DAT2 / SPI1_MOSI / I2C0_SDA / UART5_RXD / USCI1_CTL1 / SC0_DAT / I2S0_MCLK / EPWM0_CH1 / UART2_RXD / TM1 / INT1
3	PB.3 / EADC0_CH3 / ACMP0_N / EBI_ADR2 / SD0_DAT1 / SPI1_CLK / UART1_TXD / UART5_nRTS / USCI1_DAT1

Pin	M2354KJFAE Pin Function
	/ SC0_RST / I2S0_DI / EPWM0_CH2 / I2C1_SCL / TM4 / TM2 / INT2
4	PB.2 / EADC0_CH2 / ACMP0_P1 / EBI_ADR3 / SD0_DAT0 / SPI1_SS / UART1_RXD / UART5_nCTS / USCI1_DAT0 / SC0_PWR / I2S0_DO / EPWM0_CH3 / I2C1_SDA / TM5 / TM3 / INT3
5	PC.12 / EBI_ADR4 / UART0_TXD / I2C0_SCL / SPI3_MISO / SC0_nCD / ECAP1_IC2 / EPWM1_CH0 / ACMP0_O
6	PC.11 / EBI_ADR5 / UART0_RXD / I2C0_SDA / SPI3_MOSI / ECAP1_IC1 / EPWM1_CH1 / ACMP1_O
7	PC.10 / EBI_ADR6 / SPI3_CLK / UART3_TXD / ECAP1_IC0 / EPWM1_CH2
8	PC.9 / EBI_ADR7 / SPI3_SS / UART3_RXD / EPWM1_CH3
9	PB.1 / EADC0_CH1 / EBI_ADR8 / SD0_CLK / SPI1_I2SMCLK / SPI3_I2SMCLK / UART2_TXD / USCI1_CLK / I2C1_SCL / I2S0_LRCK / EPWM0_CH4 / EPWM1_CH4 / EPWM0_BRAKE0 / QSPI0_MISO1
10	PB.0 / EADC0_CH0 / EBI_ADR9 / SD0_CMD / SPI2_I2SMCLK / UART2_RXD / SPI0_I2SMCLK / I2C1_SDA / EPWM0_CH5 / EPWM1_CH5 / EPWM0_BRAKE1 / QSPI0_MOSI1
11	V_{SS}
12	V_{DD}
13	PA.11 / ACMP0_P0 / EBI_nRD / SC2_PWR / SPI2_SS / USCI0_CLK / I2C2_SCL / BPWM0_CH0 / EPWM0_SYNC_OUT / TM0_EXT / DAC1_ST
14	PA.10 / ACMP1_P0 / EBI_nWR / SC2_RST / SPI2_CLK / USCI0_DAT0 / I2C2_SDA / BPWM0_CH1 / QEI1_INDEX / ECAP0_IC0 / TM1_EXT / DAC0_ST
15	PA.9 / EBI_MCLK / SC2_DAT / SPI2_MISO / USCI0_DAT1 / UART1_TXD / BPWM0_CH2 / QEI1_A / ECAP0_IC1 / TM4_EXT / TM2_EXT
16	PA.8 / EBI_ALE / SC2_CLK / SPI2_MOSI / USCI0_CTL1 / UART1_RXD / BPWM0_CH3 / QEI1_B / ECAP0_IC2 / TM5_EXT / TM3_EXT / INT4
17	PC.13 / EBI_ADR10 / SC2_nCD / SPI2_I2SMCLK / USCI0_CTL0 / UART2_TXD / BPWM0_CH4 / CLKO / EADC0_ST
18	PD.12 / EBI_nCS0 / UART2_RXD / BPWM0_CH5 / QEI0_INDEX / CLKO / EADC0_ST / INT5
19	PD.11 / EBI_nCS1 / UART1_TXD / CAN0_TXD / QEI0_A / INT6
20	PD.10 / EBI_nCS2 / UART1_RXD / CAN0_RXD / QEI0_B / INT7
21	PG.2 / EBI_ADR11 / SPI2_SS / I2C0_SMBAL / I2C1_SCL / TM0 / LCD_SEG39
22	PG.3 / EBI_ADR12 / SPI2_CLK / I2C0_SMBSUS / I2C1_SDA / TM1 / LCD_SEG38
23	PG.4 / EBI_ADR13 / SPI2_MISO / TM4 / TM2 / LCD_SEG37
24	PF.11 / EBI_ADR14 / SPI2_MOSI / UART5_TXD / TAMPER5 / TM5 / TM3
25	PF.10 / EBI_ADR15 / SC0_nCD / I2S0_BCLK / SPI0_I2SMCLK / UART5_RXD / TAMPER4
26	PF.9 / EBI_ADR16 / SC0_PWR / I2S0_MCLK / SPI0_SS / UART5_nRTS / TAMPER3
27	PF.8 / EBI_ADR17 / SC0_RST / I2S0_DI / SPI0_CLK / UART5_nCTS / TAMPER2
28	PF.7 / EBI_ADR18 / SC0_DAT / I2S0_DO / SPI0_MISO / UART4_TXD / TAMPER1
29	PF.6 / EBI_ADR19 / SC0_CLK / I2S0_LRCK / SPI0_MOSI / UART4_RXD / EBI_nCS0 / SPI3_I2SMCLK / TAMPER0
30	V_{BAT}
31	PF.5 / UART2_RXD / UART2_nCTS / EPWM0_CH0 / BPWM0_CH4 / EPWM0_SYNC_OUT / X32_IN / EADC0_ST
32	PF.4 / UART2_TXD / UART2_nRTS / EPWM0_CH1 / BPWM0_CH5 / X32_OUT

Pin	M2354KJFAE Pin Function
33	PH.4 / EBI_ADR3 / SPI1_MISO / LCD_SEG36
34	PH.5 / EBI_ADR2 / SPI1_MOSI / LCD_SEG35
35	PH.6 / EBI_ADR1 / SPI1_CLK / LCD_SEG34
36	PH.7 / EBI_ADR0 / SPI1_SS / LCD_SEG33
37	PF.3 / EBI_nCS0 / UART0_TXD / I2C0_SCL / XT1_IN / BPWM1_CH0
38	PF.2 / EBI_nCS1 / UART0_RXD / I2C0_SDA / QSPI0_CLK / XT1_OUT / BPWM1_CH1
39	V_{SS}
40	V_{DD}
41	PE.8 / EBI_ADR10 / I2S0_BCLK / SPI2_CLK / USCI1_CTL1 / UART2_TXD / EPWM0_CH0 / EPWM0_BRAKE0 / ECAP0_IC0 / TRACE_DATA3 / LCD_SEG32
42	PE.9 / EBI_ADR11 / I2S0_MCLK / SPI2_MISO / USCI1_CTL0 / UART2_RXD / EPWM0_CH1 / EPWM0_BRAKE1 / ECAP0_IC1 / TRACE_DATA2 / LCD_SEG31
43	PE.10 / EBI_ADR12 / I2S0_DI / SPI2_MOSI / USCI1_DAT0 / UART3_TXD / EPWM0_CH2 / EPWM1_BRAKE0 / ECAP0_IC2 / TRACE_DATA1 / LCD_SEG30
44	PE.11 / EBI_ADR13 / I2S0_DO / SPI2_SS / USCI1_DAT1 / UART3_RXD / UART1_nCTS / EPWM0_CH3 / EPWM1_BRAKE1 / ECAP1_IC2 / TRACE_DATA0
45	PE.12 / EBI_ADR14 / I2S0_LRCK / SPI2_I2SMCLK / USCI1_CLK / UART1_nRTS / EPWM0_CH4 / ECAP1_IC1 / TRACE_CLK
46	PE.13 / EBI_ADR15 / I2C0_SCL / UART4_nRTS / UART1_TXD / EPWM0_CH5 / EPWM1_CH0 / BPWM1_CH5 / ECAP1_IC0
47	PC.8 / EBI_ADR16 / I2C0_SDA / UART4_nCTS / UART1_RXD / EPWM1_CH1 / BPWM1_CH4
48	PC.7 / EBI_AD9 / SPI1_MISO / UART4_TXD / SC2_PWR / UART0_nCTS / I2C1_SMBAL / EPWM1_CH2 / BPWM1_CH0 / TM0 / INT3
49	PC.6 / EBI_AD8 / SPI1_MOSI / UART4_RXD / SC2_RST / UART0_nRTS / I2C1_SMBSUS / EPWM1_CH3 / BPWM1_CH1 / TM1 / INT2
50	PA.7 / EBI_AD7 / SPI1_CLK / SC2_DAT / UART0_TXD / I2C1_SCL / TM4 / EPWM1_CH4 / BPWM1_CH2 / ACMP0_WLAT / TM2 / INT1
51	PA.6 / EBI_AD6 / SPI1_SS / SC2_CLK / UART0_RXD / I2C1_SDA / TM5 / EPWM1_CH5 / BPWM1_CH3 / ACMP1_WLAT / TM3 / INT0
52	V_{SS}
53	V_{DD}
54	LDO_CAP
55	PA.5 / QSPI0_MISO1 / SPI1_I2SMCLK / SC2_nCD / UART0_nCTS / UART5_TXD / I2C0_SCL / CAN0_TXD / UART0_TXD / BPWM0_CH5 / EPWM0_CH0 / QEI0_INDEX / LCD_SEG29
56	PA.4 / QSPI0_MOSI1 / SPI0_I2SMCLK / SC0_nCD / UART0_nRTS / UART5_RXD / I2C0_SDA / CAN0_RXD / UART0_RXD / BPWM0_CH4 / EPWM0_CH1 / QEI0_A / LCD_SEG28
57	PA.3 / QSPI0_SS / SPI0_SS / SC0_PWR / UART4_TXD / UART1_TXD / I2C1_SCL / I2C0_SMBAL / LCD_SEG27 / BPWM0_CH3 / EPWM0_CH2 / QEI0_B / EPWM1_BRAKE1
58	PA.2 / QSPI0_CLK / SPI0_CLK / SC0_RST / UART4_RXD / UART1_RXD / I2C1_SDA / I2C0_SMBSUS / LCD_SEG26 / BPWM0_CH2 / EPWM0_CH3

Pin	M2354KJFAE Pin Function
59	PA.1 / QSPI0_MISO0 / SPI0_MISO / SC0_DAT / UART0_TXD / UART1_nCTS / I2C2_SCL / LCD_SEG25 / BPWM0_CH1 / EPWM0_CH4 / DAC1_ST
60	PA.0 / QSPI0_MOSI0 / SPI0_MOSI / SC0_CLK / UART0_RXD / UART1_nRTS / I2C2_SDA / LCD_SEG24 / BPWM0_CH0 / EPWM0_CH5 / DAC0_ST
61	V_{DDIO}
62	PE.14 / EBI_AD8 / UART2_TXD / CAN0_TXD / LCD_SEG23
63	PE.15 / EBI_AD9 / UART2_RXD / CAN0_RXD / LCD_SEG22
64	nRESET
65	PF.0 / UART1_TXD / I2C1_SCL / UART0_TXD / BPWM1_CH0 / ICE_DAT
66	PF.1 / UART1_RXD / I2C1_SDA / UART0_RXD / BPWM1_CH1 / ICE_CLK
67	PD.9 / EBI_AD7 / I2C2_SCL / UART2_nCTS / LCD_COM7/SEG40
68	PD.8 / EBI_AD6 / I2C2_SDA / UART2_nRTS / LCD_COM6/SEG41
69	PC.5 / EBI_AD5 / QSPI0_MISO1 / UART2_TXD / I2C1_SCL / CAN0_TXD / UART4_TXD / EPWM1_CH0 / LCD_COM5/SEG42
70	PC.4 / EBI_AD4 / QSPI0_MOSI1 / SC1_nCD / I2S0_BCLK / SPI1_I2SMCLK / UART2_RXD / I2C1_SDA / CAN0_RXD / UART4_RXD / EPWM1_CH1 / LCD_COM4/SEG43
71	PC.3 / EBI_AD3 / QSPI0_SS / SC1_PWR / I2S0_MCLK / SPI1_MISO / UART2_nRTS / I2C0_SMBAL / UART3_TXD / EPWM1_CH2 / LCD_COM3
72	PC.2 / EBI_AD2 / QSPI0_CLK / SC1_RST / I2S0_DI / SPI1_MOSI / UART2_nCTS / I2C0_SMBSUS / UART3_RXD / EPWM1_CH3 / LCD_COM2
73	PC.1 / EBI_AD1 / QSPI0_MISO0 / SC1_DAT / I2S0_DO / SPI1_CLK / UART2_TXD / I2C0_SCL / EPWM1_CH4 / LCD_COM1 / ACMP0_O / EADC0_ST
74	PC.0 / EBI_AD0 / QSPI0_MOSI0 / SC1_CLK / I2S0_LRCK / SPI1_SS / UART2_RXD / I2C0_SDA / EPWM1_CH5 / LCD_COM0 / ACMP1_O
75	V_{SS}
76	V_{DD}
77	PG.9 / EBI_AD0 / BPWM0_CH5 / LCD_SEG21
78	PG.10 / EBI_AD1 / BPWM0_CH4 / LCD_SEG20
79	PG.11 / EBI_AD2 / BPWM0_CH3 / LCD_SEG19
80	PG.12 / EBI_AD3 / BPWM0_CH2 / LCD_SEG18
81	PG.13 / EBI_AD4 / BPWM0_CH1 / LCD_SEG17
82	PG.14 / EBI_AD5 / BPWM0_CH0 / LCD_SEG16
83	PG.15 / LCD_SEG15 / CLKO / EADC0_ST
84	PD.7 / UART1_TXD / I2C0_SCL / SPI1_MISO / USCI1_CLK / SC1_PWR / LCD_SEG14
85	PD.6 / UART1_RXD / I2C0_SDA / SPI1_MOSI / USCI1_DAT1 / SC1_RST / LCD_SEG13
86	PD.5 / I2C1_SCL / SPI1_CLK / USCI1_DAT0 / SC1_DAT
87	PD.4 / USCI0_CTL0 / I2C1_SDA / SPI1_SS / USCI1_CTL1 / SC1_CLK / USB_VBUS_ST

Pin	M2354KJFAE Pin Function
88	PD.3 / EBI_AD10 / USCI0_CTL1 / SPI0_SS / UART3_nRTS / USCI1_CTL0 / SC2_PWR / SC1_nCD / UART0_TXD
89	PD.2 / EBI_AD11 / USCI0_DAT1 / SPI0_CLK / UART3_nCTS / SC2_RST / UART0_RXD
90	PD.1 / EBI_AD12 / USCI0_DAT0 / SPI0_MISO / UART3_TXD / I2C2_SCL / SC2_DAT
91	PD.0 / EBI_AD13 / USCI0_CLK / SPI0_MOSI / UART3_RXD / I2C2_SDA / SC2_CLK / TM2
92	V_{LCD}
93	PA.12 / I2S0_BCLK / UART4_TXD / I2C1_SCL / SPI2_SS / CAN0_TXD / SC2_PWR / BPWM1_CH2 / QEI1_INDEX / USB_VBUS
94	PA.13 / I2S0_MCLK / UART4_RXD / I2C1_SDA / SPI2_CLK / CAN0_RXD / SC2_RST / BPWM1_CH3 / QEI1_A / USB_D-
95	PA.14 / I2S0_DI / UART0_TXD / SPI2_MISO / I2C2_SCL / SC2_DAT / BPWM1_CH4 / QEI1_B / USB_D+
96	PA.15 / I2S0_DO / UART0_RXD / SPI2_MOSI / I2C2_SDA / SC2_CLK / BPWM1_CH5 / EPWM0_SYNC_IN / USB_OTG_ID
97	PE.7 / SD0_CMD / UART5_TXD / QEI1_INDEX / EPWM0_CH0 / BPWM0_CH5 / LCD_SEG12
98	PE.6 / SD0_CLK / SPI3_I2SMCLK / SC0_nCD / USCI0_CTL0 / UART5_RXD / QEI1_A / EPWM0_CH1 / BPWM0_CH4 / LCD_SEG11
99	PE.5 / EBI_nRD / SD0_DAT3 / SPI3_SS / SC0_PWR / USCI0_CTL1 / QEI1_B / EPWM0_CH2 / BPWM0_CH3 / LCD_SEG10
100	PE.4 / EBI_nWR / SD0_DAT2 / SPI3_CLK / SC0_RST / USCI0_DAT1 / QEI0_INDEX / EPWM0_CH3 / BPWM0_CH2 / LCD_SEG9
101	PE.3 / EBI_MCLK / SD0_DAT1 / SPI3_MISO / SC0_DAT / USCI0_DAT0 / QEI0_A / EPWM0_CH4 / BPWM0_CH1 / LCD_SEG8
102	PE.2 / EBI_ALE / SD0_DAT0 / SPI3_MOSI / SC0_CLK / USCI0_CLK / QEI0_B / EPWM0_CH5 / BPWM0_CH0 / LCD_SEG7
103	V_{SS}
104	V_{DD}
105	PE.1 / EBI_AD10 / QSPI0_MISO0 / SC2_DAT / I2S0_BCLK / SPI1_MISO / UART3_TXD / I2C1_SCL / UART4_nCTS / LCD_SEG6
106	PE.0 / EBI_AD11 / QSPI0_MOSI0 / SC2_CLK / I2S0_MCLK / SPI1_MOSI / UART3_RXD / I2C1_SDA / UART4_nRTS / LCD_SEG5
107	PH.8 / EBI_AD12 / QSPI0_CLK / SC2_PWR / I2S0_DI / SPI1_CLK / UART3_nRTS / I2C1_SMBAL / I2C2_SCL / UART1_TXD / LCD_SEG4
108	PH.9 / EBI_AD13 / QSPI0_SS / SC2_RST / I2S0_DO / SPI1_SS / UART3_nCTS / I2C1_SMBSUS / I2C2_SDA / UART1_RXD / LCD_SEG3
109	PH.10 / EBI_AD14 / QSPI0_MISO1 / SC2_nCD / I2S0_LRCK / SPI1_I2SMCLK / UART4_TXD / UART0_TXD / LCD_SEG2
110	PH.11 / EBI_AD15 / QSPI0_MOSI1 / UART4_RXD / UART0_RXD / EPWM0_CH5 / LCD_SEG1
111	PD.14 / EBI_nCS0 / SPI3_I2SMCLK / SC1_nCD / USCI0_CTL0 / SPI0_I2SMCLK / EPWM0_CH4 / LCD_SEG0
112	V_{SS}
113	Vsw
114	V_{DD}

Pin	M2354KJFAE Pin Function
115	LDO_CAP
116	PB.15 / EADC0_CH15 / EBI_AD12 / SC1_PWR / SPI0_SS / USCI0_CTL1 / UART0_nCTS / UART3_TXD / I2C2_SMBAL / EPWM1_CH0 / TM0_EXT / USB_VBUS_EN
117	PB.14 / EADC0_CH14 / EBI_AD13 / SC1_RST / SPI0_CLK / USCI0_DAT1 / UART0_nRTS / UART3_RXD / I2C2_SMBSUS / EPWM0_BRAKE1 / EPWM1_CH1 / TM1_EXT / CLKO / USB_VBUS_ST
118	PB.13 / EADC0_CH13 / DAC1_OUT / ACMP0_P3 / ACMP1_P3 / EBI_AD14 / SC1_DAT / SPI0_MISO / USCI0_DAT0 / UART0_TXD / UART3_nRTS / I2C2_SCL / EPWM1_CH2 / TM2_EXT / TM4_EXT
119	PB.12 / EADC0_CH12 / DAC0_OUT / ACMP0_P2 / ACMP1_P2 / EBI_AD15 / SC1_CLK / SPI0_MOSI / USCI0_CLK / UART0_RXD / UART3_nCTS / I2C2_SDA / SD0_nCD / EPWM1_CH3 / TM3_EXT / TM5_EXT
120	AV_{DD}
121	V_{REF}
122	AV_{SS}
123	PB.11 / EADC0_CH11 / EBI_ADR16 / UART0_nCTS / UART4_TXD / I2C1_SCL / CAN0_TXD / SPI0_I2SMCLK / BPWM1_CH0 / SPI3_CLK
124	PB.10 / EADC0_CH10 / EBI_ADR17 / USCI1_CTL0 / UART0_nRTS / UART4_RXD / I2C1_SDA / CAN0_RXD / BPWM1_CH1 / SPI3_SS
125	PB.9 / EADC0_CH9 / EBI_ADR18 / USCI1_CTL1 / UART0_TXD / UART1_nCTS / I2C1_SMBAL / I2C0_SCL / BPWM1_CH2 / SPI3_MISO / INT7
126	PB.8 / EADC0_CH8 / EBI_ADR19 / USCI1_CLK / UART0_RXD / UART1_nRTS / I2C1_SMBSUS / I2C0_SDA / BPWM1_CH3 / SPI3_MOSI / INT6
127	PB.7 / EADC0_CH7 / EBI_nWRL / USCI1_DAT0 / UART1_TXD / EBI_nCS0 / BPWM1_CH4 / EPWM1_BRAKE0 / EPWM1_CH4 / INT5 / USB_VBUS_ST / ACMP0_O
128	PB.6 / EADC0_CH6 / EBI_nWRH / USCI1_DAT1 / UART1_RXD / EBI_nCS1 / BPWM1_CH5 / EPWM1_BRAKE1 / EPWM1_CH5 / INT4 / USB_VBUS_EN / ACMP1_O

4.2 M2354 Pin Mapping

Different part number with same package might has different function. Please refer to the selection guide in section 3.2, Pin Configuration in section 4.1 or NuTool - PinConfigure.

Corresponding Part Number: M2354

	M2354		
Pin Name	48 Pin	64 Pin	128 Pin
PB.6		1	128
PB.5	1	2	1
PB.4	2	3	2
PB.3	3	4	3
PB.2	4	5	4
PC.12			5
PC.11			6
PC.10			7
PC.9			8
PB.1	5	6	9
PB.0	6	7	10
V_{SS}			11
V_{DD}			12
PA.11	7	8	13
PA.10	8	9	14
PA.9	9	10	15
PA.8	10	11	16
PC.13			17
PD.12			18
PD.11			19
PD.10			20
PG.2			21
PG.3			22
PG.4			23
PF.11			24
PF.10			25
PF.9			26
PF.8			27
PF.7			28

PF.6	11	12	29
V_{BAT}		13	30
PF.5	12	14	31
PF.4	13	15	32
PH.4			33
PH.5			34
PH.6			35
PH.7			36
PF.3	14	16	37
PF.2	15	17	38
V_{SS}			39
V_{DD}			40
PE.8			41
PE.9			42
PE.10			43
PE.11			44
PE.12			45
PE.13			46
PC.8			47
PC.7		18	48
PC.6		19	49
PA.7	16	20	50
PA.6	17	21	51
V_{SS}		22	52
V_{DD}		23	53
LDO_CAP		24	54
PA.5	18	25	55
PA.4	19	26	56
PA.3	20	27	57
PA.2	21	28	58
PA.1	22	29	59
PA.0	23	30	60
V_{DDIO}	24	31	61
PE.14			62

PE.15			63
nRESET	25	32	64
PF.0	26	33	65
PF.1	27	34	66
PD.9			67
PD.8			68
PC.5	28	35	69
PC.4	29	36	70
PC.3	30	37	71
PC.2	31	38	72
PC.1	32	39	73
PC.0	33	40	74
V_{SS}			75
V_{DD}			76
PG.9			77
PG.10			78
PG.11			79
PG.12			80
PG.13			81
PG.14			82
PG.15			83
PD.7			84
PD.6			85
PD.5			86
PD.4			87
PD.3		41	88
PD.2		42	89
PD.1		43	90
PD.0			91
V_{LCD}		44	92
PA.12	34	45	93
PA.13	35	46	94
PA.14	36	47	95
PA.15	37	48	96

PE.7			97
PE.6			98
PE.5			99
PE.4			100
PE.3			101
PE.2			102
V_{SS}	38	49	103
V_{DD}			104
PE.1			105
PE.0			106
PH.8			107
PH.9			108
PH.10			109
PH.11			110
PD.14			111
V_{SS}			112
V_{sw}	39	50	113
V_{DD}	40	51	114
LDO_CAP	41	52	115
PB.15	42	53	116
PB.14	43	54	117
PB.13	44	55	118
PB.12	45	56	119
AV_{DD}	46	57	120
V_{REF}		58	121
AV_{SS}	47	59	122
PB.11		60	123
PB.10		61	124
PB.9		62	125
PB.8		63	126
PB.7	48	64	127

4.3 M2354 Pin Functional Description

Group	Pin Name	GPIO	MFP	Type	Description
ACMP0	ACMP0_N	PB.3	MFP1	A	Analog comparator 0 negative input pin.
	ACMP0_O	PC.12	MFP14	O	Analog comparator 0 output pin.
		PC.1	MFP14	O	
		PB.7	MFP15	O	
	ACMP0_P0	PA.11	MFP1	A	Analog comparator 0 positive input 0 pin.
	ACMP0_P1	PB.2	MFP1	A	Analog comparator 0 positive input 1 pin.
	ACMP0_P2	PB.12	MFP1	A	Analog comparator 0 positive input 2 pin.
	ACMP0_P3	PB.13	MFP1	A	Analog comparator 0 positive input 3 pin.
	ACMP0_WLAT	PA.7	MFP13	I	Analog comparator 0 window latch input pin
ACMP1	ACMP1_N	PB.5	MFP1	A	Analog comparator 1 negative input pin.
	ACMP1_O	PB.6	MFP15	O	Analog comparator 1 output pin.
		PC.11	MFP14	O	
		PC.0	MFP14	O	
	ACMP1_P0	PA.10	MFP1	A	Analog comparator 1 positive input 0 pin.
	ACMP1_P1	PB.4	MFP1	A	Analog comparator 1 positive input 1 pin.
	ACMP1_P2	PB.12	MFP1	A	Analog comparator 1 positive input 2 pin.
	ACMP1_P3	PB.13	MFP1	A	Analog comparator 1 positive input 3 pin.
	ACMP1_WLAT	PA.6	MFP13	I	Analog comparator 1 window latch input pin
BPWM0	BPWM0_CH0	PA.11	MFP9	I/O	BPWM0 channel 0 output/capture input.
		PA.0	MFP12	I/O	
		PG.14	MFP12	I/O	
		PE.2	MFP13	I/O	
	BPWM0_CH1	PA.10	MFP9	I/O	BPWM0 channel 1 output/capture input.
		PA.1	MFP12	I/O	
		PG.13	MFP12	I/O	
		PE.3	MFP13	I/O	
	BPWM0_CH2	PA.9	MFP9	I/O	BPWM0 channel 2 output/capture input.
		PA.2	MFP12	I/O	
		PG.12	MFP12	I/O	
		PE.4	MFP13	I/O	
	BPWM0_CH3	PA.8	MFP9	I/O	BPWM0 channel 3 output/capture input.
		PA.3	MFP12	I/O	

Group	Pin Name	GPIO	MFP	Type	Description
		PG.11	MFP12	I/O	
		PE.5	MFP13	I/O	
	BPWM0_CH4	PC.13	MFP9	I/O	BPWM0 channel 4 output/capture input.
		PF.5	MFP8	I/O	
		PA.4	MFP12	I/O	
		PG.10	MFP12	I/O	
		PE.6	MFP13	I/O	
	BPWM0_CH5	PD.12	MFP9	I/O	BPWM0 channel 5 output/capture input.
		PF.4	MFP8	I/O	
		PA.5	MFP12	I/O	
		PG.9	MFP12	I/O	
		PE.7	MFP13	I/O	
BPWM1	BPWM1_CH0	PF.3	MFP11	I/O	BPWM1 channel 0 output/capture input.
		PC.7	MFP12	I/O	
		PF.0	MFP12	I/O	
		PB.11	MFP10	I/O	
	BPWM1_CH1	PF.2	MFP11	I/O	BPWM1 channel 1 output/capture input.
		PC.6	MFP12	I/O	
		PF.1	MFP12	I/O	
		PB.10	MFP10	I/O	
	BPWM1_CH2	PA.7	MFP12	I/O	BPWM1 channel 2 output/capture input.
		PA.12	MFP11	I/O	
		PB.9	MFP10	I/O	
	BPWM1_CH3	PA.6	MFP12	I/O	BPWM1 channel 3 output/capture input.
		PA.13	MFP11	I/O	
		PB.8	MFP10	I/O	
	BPWM1_CH4	PC.8	MFP12	I/O	BPWM1 channel 4 output/capture input.
		PA.14	MFP11	I/O	
		PB.7	MFP10	I/O	
	BPWM1_CH5	PB.6	MFP10	I/O	BPWM1 channel 5 output/capture input.
		PE.13	MFP12	I/O	
		PA.15	MFP11	I/O	
CAN0	CAN0_RXD	PD.10	MFP4	I	CAN0 bus receiver input.

Group	Pin Name	GPIO	MFP	Type	Description
		PA.4	MFP10	I	
		PE.15	MFP4	I	
		PC.4	MFP10	I	
		PA.13	MFP6	I	
		PB.10	MFP8	I	
	CAN0_TXD	PD.11	MFP4	O	CAN0 bus transmitter output.
		PA.5	MFP10	O	
		PE.14	MFP4	O	
		PC.5	MFP10	O	
		PA.12	MFP6	O	
		PB.11	MFP8	O	
CLKO	CLKO	PC.13	MFP13	O	Clock Out
		PD.12	MFP13	O	
		PG.15	MFP14	O	
		PB.14	MFP14	O	
DAC0	DAC0_OUT	PB.12	MFP1	A	DAC0 channel analog output.
		PB.12	MFP1	A	
	DAC0_ST	PA.10	MFP14	I	DAC0 external trigger input.
		PA.0	MFP15	I	
DAC1	DAC1_OUT	PB.13	MFP1	A	DAC1 channel analog output.
		PB.13	MFP1	A	
	DAC1_ST	PA.11	MFP14	I	DAC1 external trigger input.
		PA.1	MFP15	I	
EADC0	EADC0_CH0	PB.0	MFP1	A	EADC0 channel 0 analog input.
	EADC0_CH1	PB.1	MFP1	A	EADC0 channel 1 analog input.
	EADC0_CH2	PB.2	MFP1	A	EADC0 channel 2 analog input.
	EADC0_CH3	PB.3	MFP1	A	EADC0 channel 3 analog input.
	EADC0_CH4	PB.4	MFP1	A	EADC0 channel 4 analog input.
	EADC0_CH5	PB.5	MFP1	A	EADC0 channel 5 analog input.
	EADC0_CH6	PB.6	MFP1	A	EADC0 channel 6 analog input.
	EADC0_CH7	PB.7	MFP1	A	EADC0 channel 7 analog input.
	EADC0_CH8	PB.8	MFP1	A	EADC0 channel 8 analog input.
	EADC0_CH9	PB.9	MFP1	A	EADC0 channel 9 analog input.

Group	Pin Name	GPIO	MFP	Type	Description
	EADC0_CH10	PB.10	MFP1	A	EADC0 channel 10 analog input.
	EADC0_CH11	PB.11	MFP1	A	EADC0 channel 11 analog input.
	EADC0_CH12	PB.12	MFP1	A	EADC0 channel 12 analog input.
	EADC0_CH13	PB.13	MFP1	A	EADC0 channel 13 analog input.
	EADC0_CH14	PB.14	MFP1	A	EADC0 channel 14 analog input.
	EADC0_CH15	PB.15	MFP1	A	EADC0 channel 15 analog input.
	EADC0_ST	PC.13	MFP14	I	EADC0 external trigger input.
		PD.12	MFP14	I	
		PF.5	MFP11	I	
		PC.1	MFP15	I	
		PG.15	MFP15	I	
EBI	EBI_AD0	PC.0	MFP2	I/O	EBI address/data bus bit 0.
		PG.9	MFP2	I/O	
	EBI_AD1	PC.1	MFP2	I/O	EBI address/data bus bit 1.
		PG.10	MFP2	I/O	
	EBI_AD2	PC.2	MFP2	I/O	EBI address/data bus bit 2.
		PG.11	MFP2	I/O	
	EBI_AD3	PC.3	MFP2	I/O	EBI address/data bus bit 3.
		PG.12	MFP2	I/O	
	EBI_AD4	PC.4	MFP2	I/O	EBI address/data bus bit 4.
		PG.13	MFP2	I/O	
	EBI_AD5	PC.5	MFP2	I/O	EBI address/data bus bit 5.
		PG.14	MFP2	I/O	
	EBI_AD6	PA.6	MFP2	I/O	EBI address/data bus bit 6.
		PD.8	MFP2	I/O	
	EBI_AD7	PA.7	MFP2	I/O	EBI address/data bus bit 7.
		PD.9	MFP2	I/O	
	EBI_AD8	PC.6	MFP2	I/O	EBI address/data bus bit 8.
		PE.14	MFP2	I/O	
	EBI_AD9	PC.7	MFP2	I/O	EBI address/data bus bit 9.
		PE.15	MFP2	I/O	
	EBI_AD10	PD.3	MFP2	I/O	EBI address/data bus bit 10.
		PE.1	MFP2	I/O	

Group	Pin Name	GPIO	MFP	Type	Description
	EBI_AD11	PD.2	MFP2	I/O	EBI address/data bus bit 11.
		PE.0	MFP2	I/O	
	EBI_AD12	PD.1	MFP2	I/O	EBI address/data bus bit 12.
		PH.8	MFP2	I/O	
		PB.15	MFP2	I/O	
	EBI_AD13	PD.0	MFP2	I/O	EBI address/data bus bit 13.
		PH.9	MFP2	I/O	
		PB.14	MFP2	I/O	
	EBI_AD14	PH.10	MFP2	I/O	EBI address/data bus bit 14.
		PB.13	MFP2	I/O	
	EBI_AD15	PH.11	MFP2	I/O	EBI address/data bus bit 15.
		PB.12	MFP2	I/O	
	EBI_ADR0	PB.5	MFP2	O	EBI address bus bit 0.
		PH.7	MFP2	O	
	EBI_ADR1	PB.4	MFP2	O	EBI address bus bit 1.
		PH.6	MFP2	O	
	EBI_ADR2	PB.3	MFP2	O	EBI address bus bit 2.
		PH.5	MFP2	O	
	EBI_ADR3	PB.2	MFP2	O	EBI address bus bit 3.
		PH.4	MFP2	O	
	EBI_ADR4	PC.12	MFP2	O	EBI address bus bit 4.
	EBI_ADR5	PC.11	MFP2	O	EBI address bus bit 5.
	EBI_ADR6	PC.10	MFP2	O	EBI address bus bit 6.
	EBI_ADR7	PC.9	MFP2	O	EBI address bus bit 7.
	EBI_ADR8	PB.1	MFP2	O	EBI address bus bit 8.
	EBI_ADR9	PB.0	MFP2	O	EBI address bus bit 9.
	EBI_ADR10	PC.13	MFP2	O	EBI address bus bit 10.
		PE.8	MFP2	O	
	EBI_ADR11	PG.2	MFP2	O	EBI address bus bit 11.
		PE.9	MFP2	O	
	EBI_ADR12	PG.3	MFP2	O	EBI address bus bit 12.
		PE.10	MFP2	O	
	EBI_ADR13	PG.4	MFP2	O	EBI address bus bit 13.

Group	Pin Name	GPIO	MFP	Type	Description
		PE.11	MFP2	O	
	EBI_ADR14	PF.11	MFP2	O	EBI address bus bit 14.
		PE.12	MFP2	O	
	EBI_ADR15	PF.10	MFP2	O	EBI address bus bit 15.
		PE.13	MFP2	O	
	EBI_ADR16	PF.9	MFP2	O	EBI address bus bit 16.
		PC.8	MFP2	O	
		PB.11	MFP2	O	
	EBI_ADR17	PF.8	MFP2	O	EBI address bus bit 17.
		PB.10	MFP2	O	
	EBI_ADR18	PF.7	MFP2	O	EBI address bus bit 18.
		PB.9	MFP2	O	
	EBI_ADR19	PF.6	MFP2	O	EBI address bus bit 19.
		PB.8	MFP2	O	
	EBI_ALE	PA.8	MFP2	O	EBI address latch enable output pin.
		PE.2	MFP2	O	
	EBI_MCLK	PA.9	MFP2	O	EBI external clock output pin.
		PE.3	MFP2	O	
	EBI_nCS0	PD.12	MFP2	O	EBI chip select 0 output pin.
		PF.6	MFP7	O	
		PF.3	MFP2	O	
		PD.14	MFP2	O	
		PB.7	MFP8	O	
	EBI_nCS1	PB.6	MFP8	O	EBI chip select 1 output pin.
		PD.11	MFP2	O	
		PF.2	MFP2	O	
	EBI_nCS2	PD.10	MFP2	O	EBI chip select 2 output pin.
	EBI_nRD	PA.11	MFP2	O	EBI read enable output pin.
		PE.5	MFP2	O	
	EBI_nWR	PA.10	MFP2	O	EBI write enable output pin.
		PE.4	MFP2	O	
	EBI_nWRH	PB.6	MFP2	O	EBI high byte write enable output pin
	EBI_nWRL	PB.7	MFP2	O	EBI low byte write enable output pin.

Group	Pin Name	GPIO	MFP	Type	Description
ECAP0	ECAP0_IC0	PA.10	MFP11	I	Enhanced capture unit 0 input 0 pin.
		PE.8	MFP12	I	
	ECAP0_IC1	PA.9	MFP11	I	Enhanced capture unit 0 input 1 pin.
		PE.9	MFP12	I	
	ECAP0_IC2	PA.8	MFP11	I	Enhanced capture unit 0 input 2 pin.
		PE.10	MFP12	I	
ECAP1	ECAP1_IC0	PC.10	MFP11	I	Enhanced capture unit 1 input 0 pin.
		PE.13	MFP13	I	
	ECAP1_IC1	PC.11	MFP11	I	Enhanced capture unit 1 input 1 pin.
		PE.12	MFP13	I	
	ECAP1_IC2	PC.12	MFP11	I	Enhanced capture unit 1 input 2 pin.
		PE.11	MFP13	I	
EPWM0	EPWM0_BRAKE0	PB.1	MFP13	I	EPWM0 Brake 0 input pin.
		PE.8	MFP11	I	
	EPWM0_BRAKE1	PB.0	MFP13	I	EPWM0 Brake 1 input pin.
		PE.9	MFP11	I	
		PB.14	MFP10	I	
	EPWM0_CH0	PB.5	MFP11	I/O	EPWM0 channel 0 output/capture input.
		PF.5	MFP7	I/O	
		PE.8	MFP10	I/O	
		PA.5	MFP13	I/O	
		PE.7	MFP12	I/O	
	EPWM0_CH1	PB.4	MFP11	I/O	EPWM0 channel 1 output/capture input.
		PF.4	MFP7	I/O	
		PE.9	MFP10	I/O	
		PA.4	MFP13	I/O	
		PE.6	MFP12	I/O	
	EPWM0_CH2	PB.3	MFP11	I/O	EPWM0 channel 2 output/capture input.
		PE.10	MFP10	I/O	
		PA.3	MFP13	I/O	
		PE.5	MFP12	I/O	
	EPWM0_CH3	PB.2	MFP11	I/O	EPWM0 channel 3 output/capture input.
		PE.11	MFP10	I/O	

Group	Pin Name	GPIO	MFP	Type	Description
		PA.2	MFP13	I/O	
		PE.4	MFP12	I/O	
	EPWM0_CH4	PB.1	MFP11	I/O	EPWM0 channel 4 output/capture input.
		PE.12	MFP10	I/O	
		PA.1	MFP13	I/O	
		PE.3	MFP12	I/O	
		PD.14	MFP11	I/O	
	EPWM0_CH5	PB.0	MFP11	I/O	EPWM0 channel 5 output/capture input.
		PE.13	MFP10	I/O	
		PA.0	MFP13	I/O	
		PE.2	MFP12	I/O	
		PH.11	MFP11	I/O	
	EPWM0_SYNC_IN	PA.15	MFP12	I	EPWM0 counter synchronous trigger input pin.
	EPWM0_SYNC_OUT	PA.11	MFP10	O	EPWM0 counter synchronous trigger output pin.
		PF.5	MFP9	O	
EPWM1	EPWM1_BRAKE0	PE.10	MFP11	I	EPWM1 Brake 0 input pin.
		PB.7	MFP11	I	
	EPWM1_BRAKE1	PB.6	MFP11	I	EPWM1 Brake 1 input pin.
		PE.11	MFP11	I	
		PA.3	MFP15	I	
	EPWM1_CH0	PC.12	MFP12	I/O	EPWM1 channel 0 output/capture input.
		PE.13	MFP11	I/O	
		PC.5	MFP12	I/O	
		PB.15	MFP11	I/O	
	EPWM1_CH1	PC.11	MFP12	I/O	EPWM1 channel 1 output/capture input.
		PC.8	MFP11	I/O	
		PC.4	MFP12	I/O	
		PB.14	MFP11	I/O	
	EPWM1_CH2	PC.10	MFP12	I/O	EPWM1 channel 2 output/capture input.
		PC.7	MFP11	I/O	
		PC.3	MFP12	I/O	
		PB.13	MFP11	I/O	
	EPWM1_CH3	PC.9	MFP12	I/O	EPWM1 channel 3 output/capture input.

Group	Pin Name	GPIO	MFP	Type	Description
		PC.6	MFP11	I/O	
		PC.2	MFP12	I/O	
		PB.12	MFP11	I/O	
	EPWM1_CH4	PB.1	MFP12	I/O	EPWM1 channel 4 output/capture input.
		PA.7	MFP11	I/O	
		PC.1	MFP12	I/O	
		PB.7	MFP12	I/O	
	EPWM1_CH5	PB.6	MFP12	I/O	EPWM1 channel 5 output/capture input.
		PB.0	MFP12	I/O	
		PA.6	MFP11	I/O	
		PC.0	MFP12	I/O	
I2C0	I2C0_SCL	PB.5	MFP6	I/O	I2C0 clock pin.
		PC.12	MFP4	I/O	
		PF.3	MFP4	I/O	
		PE.13	MFP4	I/O	
		PA.5	MFP9	I/O	
		PC.1	MFP9	I/O	
		PD.7	MFP4	I/O	
		PB.9	MFP9	I/O	
	I2C0_SDA	PB.4	MFP6	I/O	I2C0 data input/output pin.
		PC.11	MFP4	I/O	
		PF.2	MFP4	I/O	
		PC.8	MFP4	I/O	
		PA.4	MFP9	I/O	
		PC.0	MFP9	I/O	
		PD.6	MFP4	I/O	
		PB.8	MFP9	I/O	
	I2C0_SMBAL	PG.2	MFP4	O	I2C0 SMBus SMBALTER pin
		PA.3	MFP10	O	
		PC.3	MFP9	O	
	I2C0_SMBSUS	PG.3	MFP4	O	I2C0 SMBus SMBSUS pin (PMBus CONTROL pin)
		PA.2	MFP10	O	
		PC.2	MFP9	O	

Group	Pin Name	GPIO	MFP	Type	Description
I2C1	I2C1_SCL	PB.3	MFP12	I/O	I2C1 clock pin.
		PB.1	MFP9	I/O	
		PG.2	MFP5	I/O	
		PA.7	MFP8	I/O	
		PA.3	MFP9	I/O	
		PF.0	MFP3	I/O	
		PC.5	MFP9	I/O	
		PD.5	MFP4	I/O	
		PA.12	MFP4	I/O	
		PE.1	MFP8	I/O	
		PB.11	MFP7	I/O	
	I2C1_SDA	PB.2	MFP12	I/O	I2C1 data input/output pin.
		PB.0	MFP9	I/O	
		PG.3	MFP5	I/O	
		PA.6	MFP8	I/O	
		PA.2	MFP9	I/O	
		PF.1	MFP3	I/O	
		PC.4	MFP9	I/O	
		PD.4	MFP4	I/O	
		PA.13	MFP4	I/O	
		PE.0	MFP8	I/O	
		PB.10	MFP7	I/O	
	I2C1_SMBAL	PC.7	MFP8	O	I2C1 SMBus SMBALTER pin
		PH.8	MFP8	O	
		PB.9	MFP7	O	
	I2C1_SMBSUS	PC.6	MFP8	O	I2C1 SMBus SMBSUS pin (PMBus CONTROL pin)
		PH.9	MFP8	O	
		PB.8	MFP7	O	
I2C2	I2C2_SCL	PA.11	MFP7	I/O	I2C2 clock pin.
		PA.1	MFP9	I/O	
		PD.9	MFP3	I/O	
		PD.1	MFP6	I/O	
		PA.14	MFP6	I/O	

Group	Pin Name	GPIO	MFP	Type	Description
		PH.8	MFP9	I/O	
		PB.13	MFP8	I/O	
	I2C2_SDA	PA.10	MFP7	I/O	I2C2 data input/output pin.
		PA.0	MFP9	I/O	
		PD.8	MFP3	I/O	
		PD.0	MFP6	I/O	
		PA.15	MFP6	I/O	
		PH.9	MFP9	I/O	
		PB.12	MFP8	I/O	
	I2C2_SMBAL	PB.15	MFP8	O	I2C2 SMBus SMBALTER pin
	I2C2_SMBSUS	PB.14	MFP8	O	I2C2 SMBus SMBSUS pin (PMBus CONTROL pin)
I2S0	I2S0_BCLK	PB.5	MFP10	O	I2S0 bit clock output pin.
		PF.10	MFP4	O	
		PE.8	MFP4	O	
		PC.4	MFP6	O	
		PA.12	MFP2	O	
		PE.1	MFP5	O	
	I2S0_DI	PB.3	MFP10	I	I2S0 data input pin.
		PF.8	MFP4	I	
		PE.10	MFP4	I	
		PC.2	MFP6	I	
		PA.14	MFP2	I	
		PH.8	MFP5	I	
	I2S0_DO	PB.2	MFP10	O	I2S0 data output pin.
		PF.7	MFP4	O	
		PE.11	MFP4	O	
		PC.1	MFP6	O	
		PA.15	MFP2	O	
		PH.9	MFP5	O	
	I2S0_LRCK	PB.1	MFP10	O	I2S0 left right channel clock output pin.
		PF.6	MFP4	O	
		PE.12	MFP4	O	
		PC.0	MFP6	O	

Group	Pin Name	GPIO	MFP	Type	Description
		PH.10	MFP5	O	
	I2S0_MCLK	PB.4	MFP10	O	I2S0 master clock output pin.
		PF.9	MFP4	O	
		PE.9	MFP4	O	
		PC.3	MFP6	O	
		PA.13	MFP2	O	
		PE.0	MFP5	O	
ICE	ICE_CLK	PF.1	MFP14	I	Serial wired debugger clock pin. **Note:** It is recommended to use 100 kΩ pull-up resistor on ICE_CLK pin.
	ICE_DAT	PF.0	MFP14	I/O	Serial wired debugger data pin. **Note:** It is recommended to use 100 kΩ pull-up resistor on ICE_DAT pin.
INT0	INT0	PB.5	MFP15	I	External interrupt 0 input pin.
		PA.6	MFP15	I	
INT1	INT1	PB.4	MFP15	I	External interrupt 1 input pin.
		PA.7	MFP15	I	
INT2	INT2	PB.3	MFP15	I	External interrupt 2 input pin.
		PC.6	MFP15	I	
INT3	INT3	PB.2	MFP15	I	External interrupt 3 input pin.
		PC.7	MFP15	I	
INT4	INT4	PB.6	MFP13	I	External interrupt 4 input pin.
		PA.8	MFP15	I	
INT5	INT5	PD.12	MFP15	I	External interrupt 5 input pin.
		PB.7	MFP13	I	
INT6	INT6	PD.11	MFP15	I	External interrupt 6 input pin.
		PB.8	MFP13	I	
INT7	INT7	PD.10	MFP15	I	External interrupt 7 input pin.
		PB.9	MFP13	I	
LCD	LCD_COM0	PC.0	MFP13	A	LCD common 0 output pin
	LCD_COM1	PC.1	MFP13	A	LCD common 1 output pin
	LCD_COM2	PC.2	MFP15	A	LCD common 2 output pin
	LCD_COM3	PC.3	MFP15	A	LCD common 3 output pin
	LCD_COM4/SEG43	PC.4	MFP15	A	LCD common 4 output pin
	LCD_COM5/SEG42	PC.5	MFP15	A	LCD common 5 output pin

Group	Pin Name	GPIO	MFP	Type	Description
	LCD_COM6/SEG41	PD.8	MFP15	A	LCD common 6 output pin
	LCD_COM7/SEG40	PD.9	MFP15	A	LCD common 7 output pin
	LCD_SEG0	PD.14	MFP15	A	LCD segment 0 output pin
	LCD_SEG1	PH.11	MFP15	A	LCD segment 1 output pin
	LCD_SEG2	PH.10	MFP15	A	LCD segment 2 output pin
	LCD_SEG3	PH.9	MFP15	A	LCD segment 3 output pin
	LCD_SEG4	PH.8	MFP15	A	LCD segment 4 output pin
	LCD_SEG5	PE.0	MFP15	A	LCD segment 5 output pin
	LCD_SEG6	PE.1	MFP15	A	LCD segment 6 output pin
	LCD_SEG7	PE.2	MFP15	A	LCD segment 7 output pin
	LCD_SEG8	PE.3	MFP15	A	LCD segment 8 output pin
	LCD_SEG9	PE.4	MFP15	A	LCD segment 9 output pin
	LCD_SEG10	PE.5	MFP15	A	LCD segment 10 output pin
	LCD_SEG11	PE.6	MFP15	A	LCD segment 11 output pin
	LCD_SEG12	PE.7	MFP15	A	LCD segment 12 output pin
	LCD_SEG13	PD.6	MFP15	A	LCD segment 13 output pin
	LCD_SEG14	PD.7	MFP15	A	LCD segment 14 output pin
	LCD_SEG15	PG.15	MFP13	A	LCD segment 15 output pin
	LCD_SEG16	PG.14	MFP15	A	LCD segment 16 output pin
	LCD_SEG17	PG.13	MFP15	A	LCD segment 17 output pin
	LCD_SEG18	PG.12	MFP15	A	LCD segment 18 output pin
	LCD_SEG19	PG.11	MFP15	A	LCD segment 19 output pin
	LCD_SEG20	PG.10	MFP15	A	LCD segment 20 output pin
	LCD_SEG21	PG.9	MFP15	A	LCD segment 21 output pin
	LCD_SEG22	PE.15	MFP15	A	LCD segment 22 output pin
	LCD_SEG23	PE.14	MFP15	A	LCD segment 23 output pin
	LCD_SEG24	PA.0	MFP11	A	LCD segment 24 output pin
	LCD_SEG25	PA.1	MFP11	A	LCD segment 25 output pin
	LCD_SEG26	PA.2	MFP11	A	LCD segment 26 output pin
	LCD_SEG27	PA.3	MFP11	A	LCD segment 27 output pin
	LCD_SEG28	PA.4	MFP15	A	LCD segment 28 output pin
	LCD_SEG29	PA.5	MFP15	A	LCD segment 29 output pin
	LCD_SEG30	PE.10	MFP15	A	LCD segment 30 output pin

Group	Pin Name	GPIO	MFP	Type	Description
	LCD_SEG31	PE.9	MFP15	A	LCD segment 31 output pin
	LCD_SEG32	PE.8	MFP15	A	LCD segment 32 output pin
	LCD_SEG33	PH.7	MFP15	A	LCD segment 33 output pin
	LCD_SEG34	PH.6	MFP15	A	LCD segment 34 output pin
	LCD_SEG35	PH.5	MFP15	A	LCD segment 35 output pin
	LCD_SEG36	PH.4	MFP15	A	LCD segment 36 output pin
	LCD_SEG37	PG.4	MFP15	A	LCD segment 37 output pin
	LCD_SEG38	PG.3	MFP15	A	LCD segment 38 output pin
	LCD_SEG39	PG.2	MFP15	A	LCD segment 39 output pin
QEI0	QEI0_A	PD.11	MFP10	I	Quadrature encoder 0 phase A input
		PA.4	MFP14	I	
		PE.3	MFP11	I	
	QEI0_B	PD.10	MFP10	I	Quadrature encoder 0 phase B input
		PA.3	MFP14	I	
		PE.2	MFP11	I	
	QEI0_INDEX	PD.12	MFP10	I	Quadrature encoder 0 index input
		PA.5	MFP14	I	
		PE.4	MFP11	I	
QEI1	QEI1_A	PA.9	MFP10	I	Quadrature encoder 1 phase A input
		PA.13	MFP12	I	
		PE.6	MFP11	I	
	QEI1_B	PA.8	MFP10	I	Quadrature encoder 1 phase B input
		PA.14	MFP12	I	
		PE.5	MFP11	I	
	QEI1_INDEX	PA.10	MFP10	I	Quadrature encoder 1 index input
		PA.12	MFP12	I	
		PE.7	MFP11	I	
QSPI0	QSPI0_CLK	PF.2	MFP5	I/O	Quad SPI0 serial clock pin.
		PA.2	MFP3	I/O	
		PC.2	MFP4	I/O	
		PH.8	MFP3	I/O	
	QSPI0_MISO0	PA.1	MFP3	I/O	Quad SPI0 MISO0 (Master In, Slave Out) pin.
		PC.1	MFP4	I/O	

Group	Pin Name	GPIO	MFP	Type	Description
		PE.1	MFP3	I/O	
	QSPI0_MISO1	PB.1	MFP15	I/O	Quad SPI0 MISO1 (Master In, Slave Out) pin.
		PA.5	MFP3	I/O	
		PC.5	MFP4	I/O	
		PH.10	MFP3	I/O	
	QSPI0_MOSI0	PA.0	MFP3	I/O	Quad SPI0 MOSI0 (Master Out, Slave In) pin.
		PC.0	MFP4	I/O	
		PE.0	MFP3	I/O	
	QSPI0_MOSI1	PB.0	MFP15	I/O	Quad SPI0 MOSI1 (Master Out, Slave In) pin.
		PA.4	MFP3	I/O	
		PC.4	MFP4	I/O	
		PH.11	MFP3	I/O	
	QSPI0_SS	PA.3	MFP3	I/O	Quad SPI0 slave select pin.
		PC.3	MFP4	I/O	
		PH.9	MFP3	I/O	
SC0	SC0_CLK	PB.5	MFP9	O	Smart Card 0 clock pin.
		PF.6	MFP3	O	
		PA.0	MFP6	O	
		PE.2	MFP6	O	
	SC0_DAT	PB.4	MFP9	I/O	Smart Card 0 data pin.
		PF.7	MFP3	I/O	
		PA.1	MFP6	I/O	
		PE.3	MFP6	I/O	
	SC0_PWR	PB.2	MFP9	O	Smart Card 0 power pin.
		PF.9	MFP3	O	
		PA.3	MFP6	O	
		PE.5	MFP6	O	
	SC0_RST	PB.3	MFP9	O	Smart Card 0 reset pin.
		PF.8	MFP3	O	
		PA.2	MFP6	O	
		PE.4	MFP6	O	
	SC0_nCD	PC.12	MFP9	I	Smart Card 0 card detect pin.
		PF.10	MFP3	I	

Group	Pin Name	GPIO	MFP	Type	Description
		PA.4	MFP6	I	
		PE.6	MFP6	I	
SC1	SC1_CLK	PC.0	MFP5	O	Smart Card 1 clock pin.
		PD.4	MFP8	O	
		PB.12	MFP3	O	
	SC1_DAT	PC.1	MFP5	I/O	Smart Card 1 data pin.
		PD.5	MFP8	I/O	
		PB.13	MFP3	I/O	
	SC1_PWR	PC.3	MFP5	O	Smart Card 1 power pin.
		PD.7	MFP8	O	
		PB.15	MFP3	O	
	SC1_RST	PC.2	MFP5	O	Smart Card 1 reset pin.
		PD.6	MFP8	O	
		PB.14	MFP3	O	
	SC1_nCD	PC.4	MFP5	I	Smart Card 1 card detect pin.
		PD.3	MFP8	I	
		PD.14	MFP4	I	
SC2	SC2_CLK	PA.8	MFP3	O	Smart Card 2 clock pin.
		PA.6	MFP6	O	
		PD.0	MFP7	O	
		PA.15	MFP7	O	
		PE.0	MFP4	O	
	SC2_DAT	PA.9	MFP3	I/O	Smart Card 2 data pin.
		PA.7	MFP6	I/O	
		PD.1	MFP7	I/O	
		PA.14	MFP7	I/O	
		PE.1	MFP4	I/O	
	SC2_PWR	PA.11	MFP3	O	Smart Card 2 power pin.
		PC.7	MFP6	O	
		PD.3	MFP7	O	
		PA.12	MFP7	O	
		PH.8	MFP4	O	
	SC2_RST	PA.10	MFP3	O	Smart Card 2 reset pin.

Group	Pin Name	GPIO	MFP	Type	Description
		PC.6	MFP6	O	
		PD.2	MFP7	O	
		PA.13	MFP7	O	
		PH.9	MFP4	O	
	SC2_nCD	PC.13	MFP3	I	Smart Card 2 card detect pin.
		PA.5	MFP6	I	
		PH.10	MFP4	I	
SD0	SD0_CLK	PB.1	MFP3	O	SD/SDIO0 clock output pin
		PE.6	MFP3	O	
	SD0_CMD	PB.0	MFP3	I/O	SD/SDIO0 command/response pin
		PE.7	MFP3	I/O	
	SD0_DAT0	PB.2	MFP3	I/O	SD/SDIO0 data line bit 0.
		PE.2	MFP3	I/O	
	SD0_DAT1	PB.3	MFP3	I/O	SD/SDIO0 data line bit 1.
		PE.3	MFP3	I/O	
	SD0_DAT2	PB.4	MFP3	I/O	SD/SDIO0 data line bit 2.
		PE.4	MFP3	I/O	
	SD0_DAT3	PB.5	MFP3	I/O	SD/SDIO0 data line bit 3.
		PE.5	MFP3	I/O	
	SD0_nCD	PB.12	MFP9	I	SD/SDIO0 card detect input pin
SPI0	SPI0_CLK	PF.8	MFP5	I/O	SPI0 serial clock pin.
		PA.2	MFP4	I/O	
		PD.2	MFP4	I/O	
		PB.14	MFP4	I/O	
	SPI0_I2SMCLK	PB.0	MFP8	I/O	SPI0 I^2S master clock output pin
		PF.10	MFP5	I/O	
		PA.4	MFP4	I/O	
		PD.14	MFP6	I/O	
		PB.11	MFP9	I/O	
	SPI0_MISO	PF.7	MFP5	I/O	SPI0 MISO (Master In, Slave Out) pin.
		PA.1	MFP4	I/O	
		PD.1	MFP4	I/O	
		PB.13	MFP4	I/O	

Group	Pin Name	GPIO	MFP	Type	Description
	SPI0_MOSI	PF.6	MFP5	I/O	SPI0 MOSI (Master Out, Slave In) pin.
		PA.0	MFP4	I/O	
		PD.0	MFP4	I/O	
		PB.12	MFP4	I/O	
	SPI0_SS	PF.9	MFP5	I/O	SPI0 slave select pin.
		PA.3	MFP4	I/O	
		PD.3	MFP4	I/O	
		PB.15	MFP4	I/O	
SPI1	SPI1_CLK	PB.3	MFP5	I/O	SPI1 serial clock pin.
		PH.6	MFP3	I/O	
		PA.7	MFP4	I/O	
		PC.1	MFP7	I/O	
		PD.5	MFP5	I/O	
		PH.8	MFP6	I/O	
	SPI1_I2SMCLK	PB.1	MFP5	I/O	SPI1 I^2S master clock output pin
		PA.5	MFP4	I/O	
		PC.4	MFP7	I/O	
		PH.10	MFP6	I/O	
	SPI1_MISO	PB.5	MFP5	I/O	SPI1 MISO (Master In, Slave Out) pin.
		PH.4	MFP3	I/O	
		PC.7	MFP4	I/O	
		PC.3	MFP7	I/O	
		PD.7	MFP5	I/O	
		PE.1	MFP6	I/O	
	SPI1_MOSI	PB.4	MFP5	I/O	SPI1 MOSI (Master Out, Slave In) pin.
		PH.5	MFP3	I/O	
		PC.6	MFP4	I/O	
		PC.2	MFP7	I/O	
		PD.6	MFP5	I/O	
		PE.0	MFP6	I/O	
	SPI1_SS	PB.2	MFP5	I/O	SPI1 slave select pin.
		PH.7	MFP3	I/O	
		PA.6	MFP4	I/O	

Group	Pin Name	GPIO	MFP	Type	Description
		PC.0	MFP7	I/O	
		PD.4	MFP5	I/O	
		PH.9	MFP6	I/O	
SPI2	SPI2_CLK	PA.10	MFP4	I/O	SPI2 serial clock pin.
		PG.3	MFP3	I/O	
		PE.8	MFP5	I/O	
		PA.13	MFP5	I/O	
	SPI2_I2SMCLK	PB.0	MFP4	I/O	SPI2 I^2S master clock output pin
		PC.13	MFP4	I/O	
		PE.12	MFP5	I/O	
	SPI2_MISO	PA.9	MFP4	I/O	SPI2 MISO (Master In, Slave Out) pin.
		PG.4	MFP3	I/O	
		PE.9	MFP5	I/O	
		PA.14	MFP5	I/O	
	SPI2_MOSI	PA.8	MFP4	I/O	SPI2 MOSI (Master Out, Slave In) pin.
		PF.11	MFP3	I/O	
		PE.10	MFP5	I/O	
		PA.15	MFP5	I/O	
	SPI2_SS	PA.11	MFP4	I/O	SPI2 slave select pin.
		PG.2	MFP3	I/O	
		PE.11	MFP5	I/O	
		PA.12	MFP5	I/O	
SPI3	SPI3_CLK	PC.10	MFP6	I/O	SPI3 serial clock pin.
		PE.4	MFP5	I/O	
		PB.11	MFP11	I/O	
	SPI3_I2SMCLK	PB.1	MFP6	I/O	SPI3 I^2S master clock output pin
		PF.6	MFP9	I/O	
		PE.6	MFP5	I/O	
		PD.14	MFP3	I/O	
	SPI3_MISO	PC.12	MFP6	I/O	SPI3 MISO (Master In, Slave Out) pin.
		PE.3	MFP5	I/O	
		PB.9	MFP11	I/O	
	SPI3_MOSI	PC.11	MFP6	I/O	SPI3 MOSI (Master Out, Slave In) pin.

Group	Pin Name	GPIO	MFP	Type	Description
		PE.2	MFP5	I/O	
		PB.8	MFP11	I/O	
	SPI3_SS	PC.9	MFP6	I/O	SPI3 slave select pin.
		PE.5	MFP5	I/O	
		PB.10	MFP11	I/O	
TAMPER0	TAMPER0	PF.6	MFP10	I/O	TAMPER detector loop pin 0.
TAMPER1	TAMPER1	PF.7	MFP10	I/O	TAMPER detector loop pin 1.
TAMPER2	TAMPER2	PF.8	MFP10	I/O	TAMPER detector loop pin 2.
TAMPER3	TAMPER3	PF.9	MFP10	I/O	TAMPER detector loop pin 3.
TAMPER4	TAMPER4	PF.10	MFP10	I/O	TAMPER detector loop pin 4.
TAMPER5	TAMPER5	PF.11	MFP10	I/O	TAMPER detector loop pin 5.
TM0	TM0	PB.5	MFP14	I/O	Timer0 event counter input/toggle output pin.
		PG.2	MFP13	I/O	
		PC.7	MFP14	I/O	
	TM0_EXT	PA.11	MFP13	I/O	Timer0 external capture input/toggle output pin.
		PB.15	MFP13	I/O	
TM1	TM1	PB.4	MFP14	I/O	Timer1 event counter input/toggle output pin.
		PG.3	MFP13	I/O	
		PC.6	MFP14	I/O	
	TM1_EXT	PA.10	MFP13	I/O	Timer1 external capture input/toggle output pin.
		PB.14	MFP13	I/O	
TM2	TM2	PB.3	MFP14	I/O	Timer2 event counter input/toggle output pin.
		PG.4	MFP13	I/O	
		PA.7	MFP14	I/O	
		PD.0	MFP14	I/O	
	TM2_EXT	PA.9	MFP13	I/O	Timer2 external capture input/toggle output pin.
		PB.13	MFP13	I/O	
TM3	TM3	PB.2	MFP14	I/O	Timer3 event counter input/toggle output pin.
		PF.11	MFP13	I/O	
		PA.6	MFP14	I/O	
	TM3_EXT	PA.8	MFP13	I/O	Timer3 external capture input/toggle output pin.
		PB.12	MFP13	I/O	
TM4	TM4	PB.3	MFP13	I/O	Timer4 event counter input/toggle output pin.

Group	Pin Name	GPIO	MFP	Type	Description
		PG.4	MFP12	I/O	
		PA.7	MFP10	I/O	
	TM4_EXT	PA.9	MFP12	I/O	Timer4 external capture input/toggle output pin.
		PB.13	MFP14	I/O	
TM5	TM5	PB.2	MFP13	I/O	Timer5 event counter input/toggle output pin.
		PF.11	MFP12	I/O	
		PA.6	MFP10	I/O	
	TM5_EXT	PA.8	MFP12	I/O	Timer5 external capture input/toggle output pin.
		PB.12	MFP14	I/O	
TRACE	TRACE_CLK	PE.12	MFP14	O	ETM Trace Clock output pin
	TRACE_DATA0	PE.11	MFP14	O	ETM Trace Data 0 output pin
	TRACE_DATA1	PE.10	MFP14	O	ETM Trace Data 1 output pin
	TRACE_DATA2	PE.9	MFP14	O	ETM Trace Data 2 output pin
	TRACE_DATA3	PE.8	MFP14	O	ETM Trace Data 3 output pin
UART0	UART0_RXD	PC.11	MFP3	I	UART0 data receiver input pin.
		PF.2	MFP3	I	
		PA.6	MFP7	I	
		PA.4	MFP11	I	
		PA.0	MFP7	I	
		PF.1	MFP4	I	
		PD.2	MFP9	I	
		PA.15	MFP3	I	
		PH.11	MFP8	I	
		PB.12	MFP6	I	
		PB.8	MFP5	I	
	UART0_TXD	PC.12	MFP3	O	UART0 data transmitter output pin.
		PF.3	MFP3	O	
		PA.7	MFP7	O	
		PA.5	MFP11	O	
		PA.1	MFP7	O	
		PF.0	MFP4	O	
		PD.3	MFP9	O	
		PA.14	MFP3	O	

Group	Pin Name	GPIO	MFP	Type	Description
		PH.10	MFP8	O	
		PB.13	MFP6	O	
		PB.9	MFP5	O	
	UART0_nCTS	PC.7	MFP7	I	UART0 clear to Send input pin.
		PA.5	MFP7	I	
		PB.15	MFP6	I	
		PB.11	MFP5	I	
	UART0_nRTS	PC.6	MFP7	O	UART0 request to Send output pin.
		PA.4	MFP7	O	
		PB.14	MFP6	O	
		PB.10	MFP5	O	
UART1	UART1_RXD	PB.6	MFP6	I	UART1 data receiver input pin.
		PB.2	MFP6	I	
		PA.8	MFP7	I	
		PD.10	MFP3	I	
		PC.8	MFP8	I	
		PA.2	MFP8	I	
		PF.1	MFP2	I	
		PD.6	MFP3	I	
		PH.9	MFP10	I	
	UART1_TXD	PB.3	MFP6	O	UART1 data transmitter output pin.
		PA.9	MFP7	O	
		PD.11	MFP3	O	
		PE.13	MFP8	O	
		PA.3	MFP8	O	
		PF.0	MFP2	O	
		PD.7	MFP3	O	
		PH.8	MFP10	O	
		PB.7	MFP6	O	
	UART1_nCTS	PE.11	MFP8	I	UART1 clear to Send input pin.
		PA.1	MFP8	I	
		PB.9	MFP6	I	
	UART1_nRTS	PE.12	MFP8	O	UART1 request to Send output pin.

Group	Pin Name	GPIO	MFP	Type	Description
		PA.0	MFP8	O	
		PB.8	MFP6	O	
UART2	UART2_RXD	PB.4	MFP12	I	UART2 data receiver input pin.
		PB.0	MFP7	I	
		PD.12	MFP7	I	
		PF.5	MFP2	I	
		PE.9	MFP7	I	
		PE.15	MFP3	I	
		PC.4	MFP8	I	
		PC.0	MFP8	I	
	UART2_TXD	PB.5	MFP12	O	UART2 data transmitter output pin.
		PB.1	MFP7	O	
		PC.13	MFP7	O	
		PF.4	MFP2	O	
		PE.8	MFP7	O	
		PE.14	MFP3	O	
		PC.5	MFP8	O	
		PC.1	MFP8	O	
	UART2_nCTS	PF.5	MFP4	I	UART2 clear to Send input pin.
		PD.9	MFP4	I	
		PC.2	MFP8	I	
	UART2_nRTS	PF.4	MFP4	O	UART2 request to Send output pin.
		PD.8	MFP4	O	
		PC.3	MFP8	O	
UART3	UART3_RXD	PC.9	MFP7	I	UART3 data receiver input pin.
		PE.11	MFP7	I	
		PC.2	MFP11	I	
		PD.0	MFP5	I	
		PE.0	MFP7	I	
		PB.14	MFP7	I	
	UART3_TXD	PC.10	MFP7	O	UART3 data transmitter output pin.
		PE.10	MFP7	O	
		PC.3	MFP11	O	

Group	Pin Name	GPIO	MFP	Type	Description
		PD.1	MFP5	O	
		PE.1	MFP7	O	
		PB.15	MFP7	O	
	UART3_nCTS	PD.2	MFP5	I	UART3 clear to Send input pin.
		PH.9	MFP7	I	
		PB.12	MFP7	I	
	UART3_nRTS	PD.3	MFP5	O	UART3 request to Send output pin.
		PH.8	MFP7	O	
		PB.13	MFP7	O	
UART4	UART4_RXD	PF.6	MFP6	I	UART4 data receiver input pin.
		PC.6	MFP5	I	
		PA.2	MFP7	I	
		PC.4	MFP11	I	
		PA.13	MFP3	I	
		PH.11	MFP7	I	
		PB.10	MFP6	I	
	UART4_TXD	PF.7	MFP6	O	UART4 data transmitter output pin.
		PC.7	MFP5	O	
		PA.3	MFP7	O	
		PC.5	MFP11	O	
		PA.12	MFP3	O	
		PH.10	MFP7	O	
		PB.11	MFP6	O	
	UART4_nCTS	PC.8	MFP5	I	UART4 clear to Send input pin.
		PE.1	MFP9	I	
	UART4_nRTS	PE.13	MFP5	O	UART4 request to Send output pin.
		PE.0	MFP9	O	
UART5	UART5_RXD	PB.4	MFP7	I	UART5 data receiver input pin.
		PF.10	MFP6	I	
		PA.4	MFP8	I	
		PE.6	MFP8	I	
	UART5_TXD	PB.5	MFP7	O	UART5 data transmitter output pin.
		PF.11	MFP6	O	

Group	Pin Name	GPIO	MFP	Type	Description
		PA.5	MFP8	O	
		PE.7	MFP8	O	
	UART5_nCTS	PB.2	MFP7	I	UART5 clear to Send input pin.
		PF.8	MFP6	I	
	UART5_nRTS	PB.3	MFP7	O	UART5 request to Send output pin.
		PF.9	MFP6	O	
USB	USB_D+	PA.14	MFP14	A	USB differential signal D+.
	USB_D-	PA.13	MFP14	A	USB differential signal D-.
	USB_OTG_ID	PA.15	MFP14	I	USB_ identification.
	USB_VBUS	PA.12	MFP14	P	Power supply from USB host or HUB.
	USB_VBUS_EN	PB.6	MFP14	O	USB external VBUS regulator enable pin.
		PB.15	MFP14	O	
	USB_VBUS_ST	PD.4	MFP14	I	USB external VBUS regulator status pin.
		PB.14	MFP15	I	
		PB.7	MFP14	I	
USCI0	USCI0_CLK	PA.11	MFP6	I/O	USCI0 clock pin.
		PD.0	MFP3	I/O	
		PE.2	MFP7	I/O	
		PB.12	MFP5	I/O	
	USCI0_CTL0	PC.13	MFP6	I/O	USCI0 control 0 pin.
		PD.4	MFP3	I/O	
		PE.6	MFP7	I/O	
		PD.14	MFP5	I/O	
	USCI0_CTL1	PA.8	MFP6	I/O	USCI0 control 1 pin.
		PD.3	MFP3	I/O	
		PE.5	MFP7	I/O	
		PB.15	MFP5	I/O	
	USCI0_DAT0	PA.10	MFP6	I/O	USCI0 data 0 pin.
		PD.1	MFP3	I/O	
		PE.3	MFP7	I/O	
		PB.13	MFP5	I/O	
	USCI0_DAT1	PA.9	MFP6	I/O	USCI0 data 1 pin.
		PD.2	MFP3	I/O	

Group	Pin Name	GPIO	MFP	Type	Description
		PE.4	MFP7	I/O	
		PB.14	MFP5	I/O	
USCI1	USCI1_CLK	PB.1	MFP8	I/O	USCI1 clock pin.
		PE.12	MFP6	I/O	
		PD.7	MFP6	I/O	
		PB.8	MFP4	I/O	
	USCI1_CTL0	PB.5	MFP8	I/O	USCI1 control 0 pin.
		PE.9	MFP6	I/O	
		PD.3	MFP6	I/O	
		PB.10	MFP4	I/O	
	USCI1_CTL1	PB.4	MFP8	I/O	USCI1 control 1 pin.
		PE.8	MFP6	I/O	
		PD.4	MFP6	I/O	
		PB.9	MFP4	I/O	
	USCI1_DAT0	PB.2	MFP8	I/O	USCI1 data 0 pin.
		PE.10	MFP6	I/O	
		PD.5	MFP6	I/O	
		PB.7	MFP4	I/O	
	USCI1_DAT1	PB.6	MFP4	I/O	USCI1 data 1 pin.
		PB.3	MFP8	I/O	
		PE.11	MFP6	I/O	
		PD.6	MFP6	I/O	
X32	X32_IN	PF.5	MFP10	I	External 32.768 kHz crystal input pin.
	X32_OUT	PF.4	MFP10	O	External 32.768 kHz crystal output pin.
XT1	XT1_IN	PF.3	MFP10	I	External 4~24 MHz (high speed) crystal input pin.
	XT1_OUT	PF.2	MFP10	O	External 4~24 MHz (high speed) crystal output pin.

5 BLOCK DIAGRAM

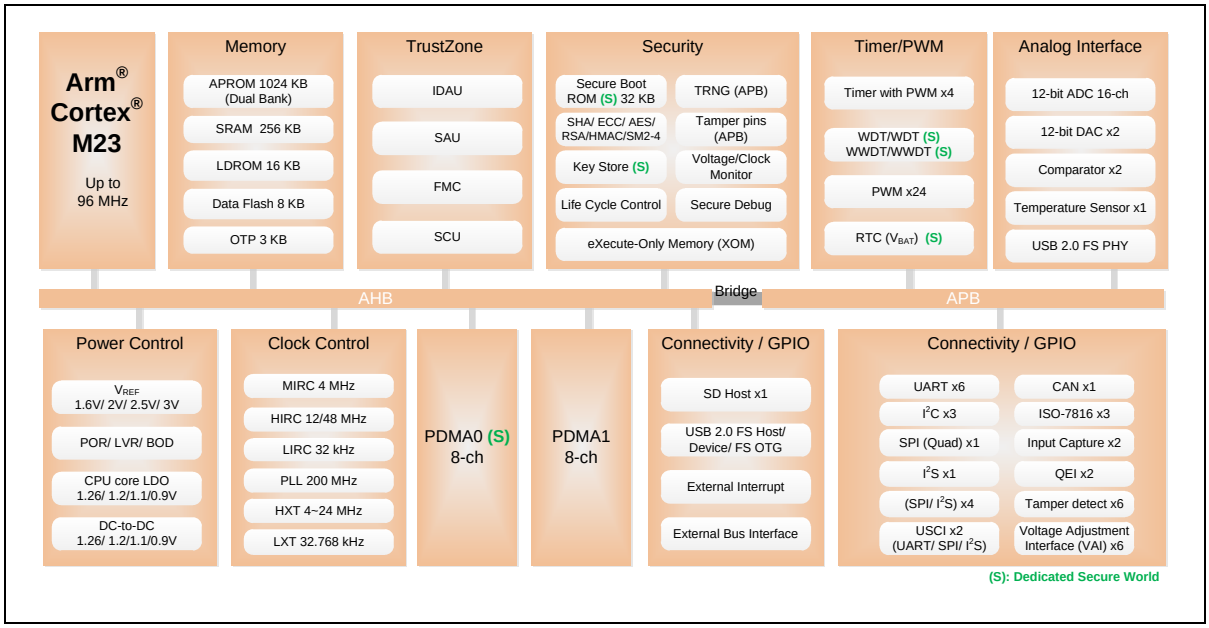


Figure 5-1 M2354 Block Diagram

6 FUNCTIONAL DESCRIPTION

6.1 Arm® Cortex®-M23 Core

The NuMicro® M2354 series is embedded with the Cortex®-M23 processor. The Cortex®-M23 processor is a low gate count, two-stage, and highly energy efficient 32-bit RISC processor, which has an AMBA AHB5 interface supporting Arm® TrustZone® technology, a debug access port supporting serial wire debug and single-cycle I/O ports. It has an NVIC component and MPU for memory-protection functionality. The processor also supports Security Extension.Figure 6.1-1 shows the functional controller of the processor.

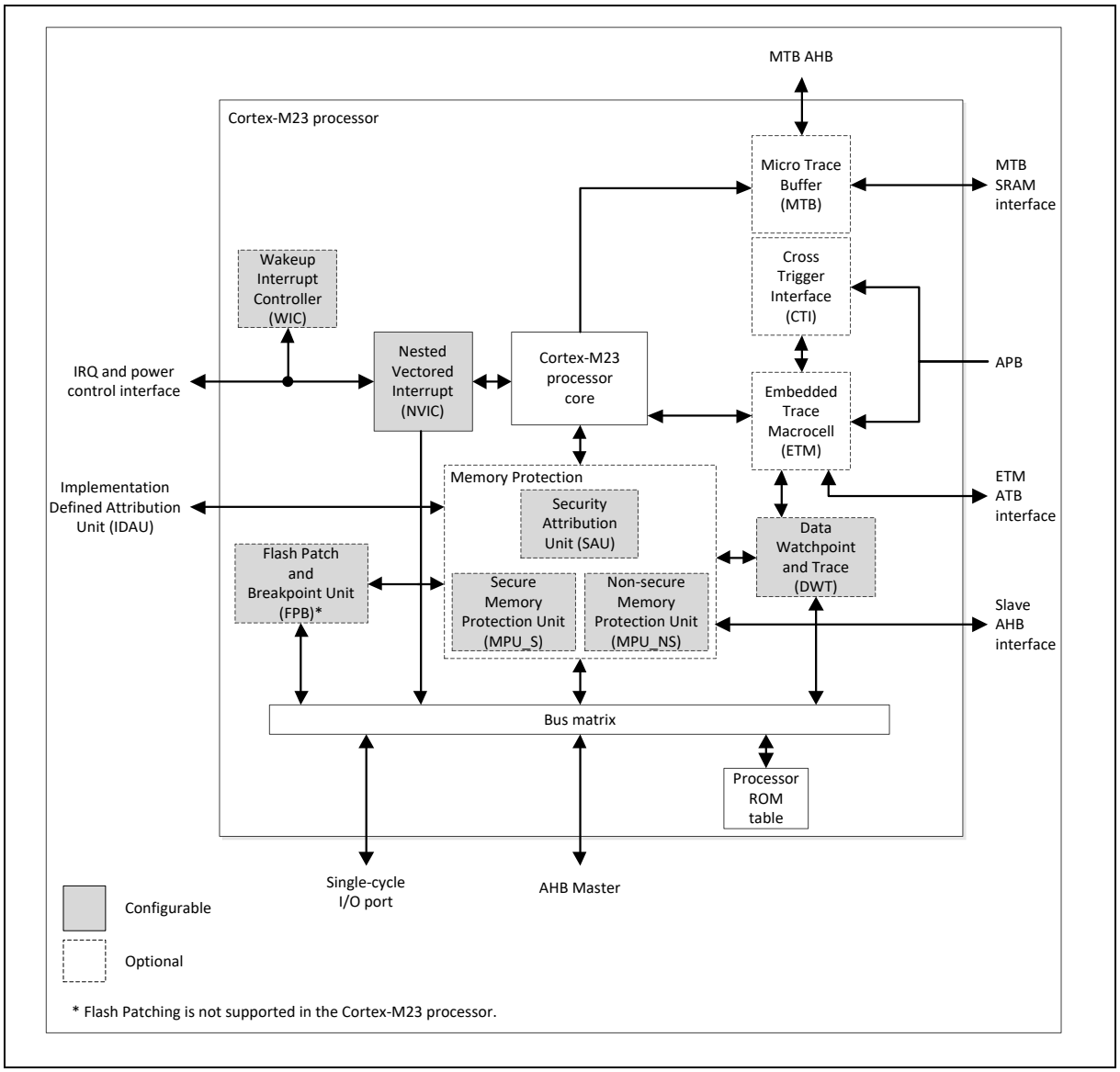


Figure 6.1-1 Cortex®-M23 Block Diagram

Cortex®-M23 processor features:

- Arm®v8-M Baseline architecture.
- Arm®v8-M Baseline Thumb®-2 instruction set that combines high code density with 32-bit performance.
- Support for single-cycle I/O access.
- Power control optimization of system components.
- Integrated sleep modes for low power consumption.
- Optimized code fetching for reduced Flash and ROM power consumption.
- A 32-bit Single cycle Hardware multiplier.
- A 32-bit Hardware divider.
- Deterministic, high-performance interrupt handling for time-critical applications.
- Deterministic instruction cycle timing.
- Support for system level debug authentication.
- Support for Arm® Debug Interface Architecture ADIv5.1 Serial Wire Debug (SWD).
- ETM for instruction trace.
- Separated privileged and unprivileged modes.
- Security Extension supporting a Secure and a Non-secure state.
- Protected Memory System Architecture (PMSAv8) Memory Protection Units (MPUs) for both Secure and Non-secure states.
- Security Attribution Unit (SAU).
- SysTick timers for both Secure and Non-secure states.
- A Nested Vectored Interrupt Controller (NVIC) closely integrated with the processor with up to 240 interrupts.

6.2 System Manager

6.2.1 Overview

System management includes the following sections:

- System Reset
- System Power Distribution
- SRAM Memory Organization
- System Timer (SysTick)
- Nested Vectored Interrupt Controller (NVIC)
- System Control register

6.2.2 Reset

The system reset can be issued by one of the events listed below. These reset event flags can be read from SYS_RSTSTS register to determine the reset source. Hardware reset source are from peripheral signals. Software reset can trigger reset through setting control registers.

- Hardware Reset Sources
  - Power-on Reset (POR)
  - Low level on the nRESET pin
  - Watchdog Time-out Reset and Window Watchdog Reset (WDT/WWDT Reset)
  - Low Voltage Reset (LVR)
  - Brown-out Detector Reset (BOD Reset)
  - CPU Lockup Reset
- Software Reset Sources
  - CHIP Reset will reset whole chip by writing 1 to CHIPRST (SYS_IPRST0[0])
  - System Reset to reboot but keeping the booting setting from APROM or LDROM by writing 1 to SYSRESETREQ (AIRCR[2])
  - CPU Reset for Cortex®-M23 core only by writing 1 to CPURST (SYS_IPRST0[1])

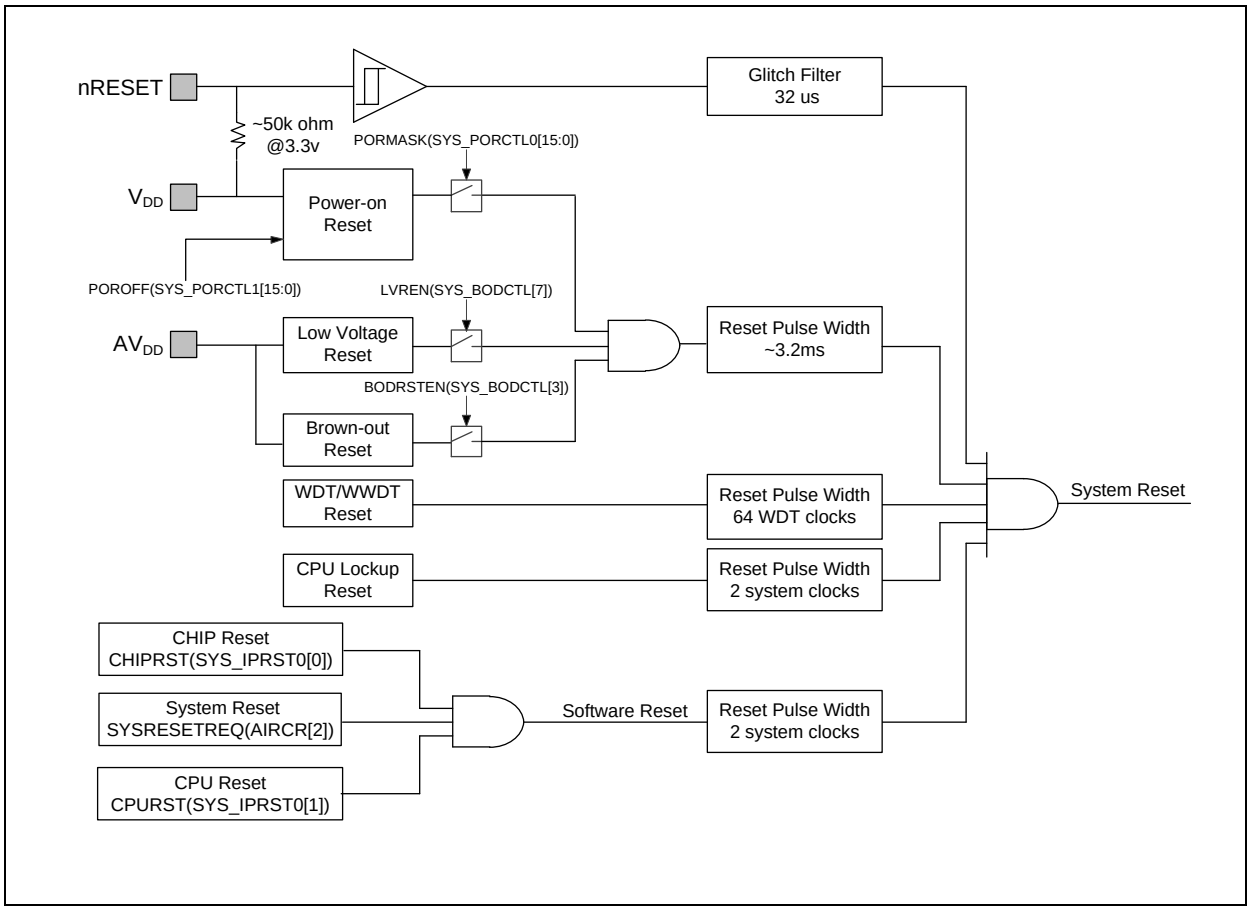


Figure 6.2-1 System Reset Sources

There are a total of 9 reset sources in the NuMicro® family. In general, CPU reset is used to reset Cortex®-M23 only; the other reset sources will reset Cortex®-M23 and all peripherals. However, there are small differences between each reset source and they are listed in Table 6.2-1.

Reset Sources Register	POR	NRESET	WDT	LVR	BOD	Lockup	CHIP	SYSTEM	CPU
SYS_RSTSTS	Bit 0 = 1	Bit 1 = 1	Bit 2 = 1	Bit 3 = 1	Bit 4 = 1	Bit 8 = 1	Bit 0 = 1	Bit 5 = 1	Bit 7 = 1
CHIPRST (SYS_IPRST0[0])	0x0	-	-	-	-	-	-	-	-
HCLKSEL (CLK_CLKSEL0[2:0])	0x5 HIRC48	0x5 HIRC48	0x5 HIRC48	0x5 HIRC48	0x5 HIRC48	0x6 MIRC	0x5 HIRC48	0x6 MIRC	-
HCLKDIV (CLK_CLKDIV0[3:0])	0x0	0x0	0x0	0x0	0x0	0x3	0x0	0x3	-
PLSTATUS (SYS_PLSTS[9:8])	0x2 PL2	0x2 PL2	0x2 PL2	0x2 PL2	0x2 PL2	-	0x2 PL2	-	-
CURMVR (SYS_PLSTS[12])	0x0 LDO	-	-	-	-	-	-	-	-

Reset Sources Register	POR	NRESET	WDT	LVR	BOD	Lockup	CHIP	SYSTEM	CPU
BODEN (SYS_BODCTL[0])	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	-	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	-
BODVL (SYS_BODCTL[18:16])									
BODRSTEN (SYS_BODCTL[3])									
SYS_SRAMPC0	0x0	-	-	-	-	-	-	-	-
KS (SYS_SRAMPC1[29:28])	0x0	0x0	0x0	0x0	0x0	-	0x0	-	-
RSA (SYS_SRAMPC1[27:26])	0x2	0x2	0x2	0x2	0x2	-	0x2	-	-
SYS_SRAMPC1 expect KS(bit[29:28]) and RSA(bit[27:26])	0x0800_0000	-	-	-	-	-	-	-	-
LXTEN (CLK_PWRCTL[1])	0x0	-	-	-	-	-	-	-	-
WDTCKEN (CLK_APBCLK0[0])	0x1	-	0x1	-	-	-	0x1	-	-
WDTSEL (CLK_CLKSEL1[1:0])	0x3	0x3	-	-	-	-	-	-	-
HXTSTB (CLK_STATUS[0])	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	-
LXTSTB (CLK_STATUS[1])	0x0	-	-	-	-	-	-	-	-
PLLSTB (CLK_STATUS[2])	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	-
HIRCSTB (CLK_STATUS[4])	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	-
CLKSFAIL (CLK_STATUS[7])	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	-
CLK_PLLCTL	0x0009_440A	0x0009_440A	0x0009_440A	0x0009_440A	0x0009_440A	0x0009_440A	0x0009_440A	0x0009_440A	-
PDMSEL (CLK_PMUCTL [2:0])	0x0	-	-	-	-	-	-	-	-
RSTEN (WDT_CTL[1])	Reload from	Reload from	Reload from	Reload from	Reload from	-	Reload from	-	-

Reset Sources Register	POR	NRESET	WDT	LVR	BOD	Lockup	CHIP	SYSTEM	CPU
WDTEN (WDT_CTL[7])	CONFIG0	CONFIG0	CONFIG0	CONFIG0	CONFIG0		CONFIG0		
WDT_CTL except bit 1 and bit 7.	0x0800	0x0800	0x0800	0x0800	0x0800	-	0x0800	-	-
WDT_ALTCTL	0x0000	0x0000	0x0000	0x0000	0x0000	-	0x0000	-	-
WWDT_RLDCNT	0x0000	0x0000	0x0000	0x0000	0x0000	-	0x0000	-	-
WWDT_CTL	0x3F0800	0x3F0800	0x3F0800	0x3F0800	0x3F0800	-	0x3F0800	-	-
WWDT_STATUS	0x0000	0x0000	0x0000	0x0000	0x0000	-	0x0000	-	-
WWDT_CNT	0x3F	0x3F	0x3F	0x3F	0x3F	-	0x3F	-	-
Other Peripheral Registers	Reset Value								-
Note: '-' means that the value of register keeps original setting.									

Table 6.2-1 Reset Value of Registers

6.2.2.1 nRESET Reset

The nRESET reset means to generate a reset signal by pulling low nRESET pin, which is an asynchronous reset input pin and can be used to reset system at any time. When the nRESET voltage is lower than 0.2 V_{DD} and the state keeps longer than 32 us (glitch filter), chip will be reset. The nRESET reset will control the chip in reset state until the nRESET voltage rises above 0.7 V_{DD} and the state keeps longer than 32 us (glitch filter). The PINRF(SYS_RSTSTS[1]) will be set to 1 if the previous reset source is nRESET reset. Figure 6.2-2 shows the nRESET reset waveform.

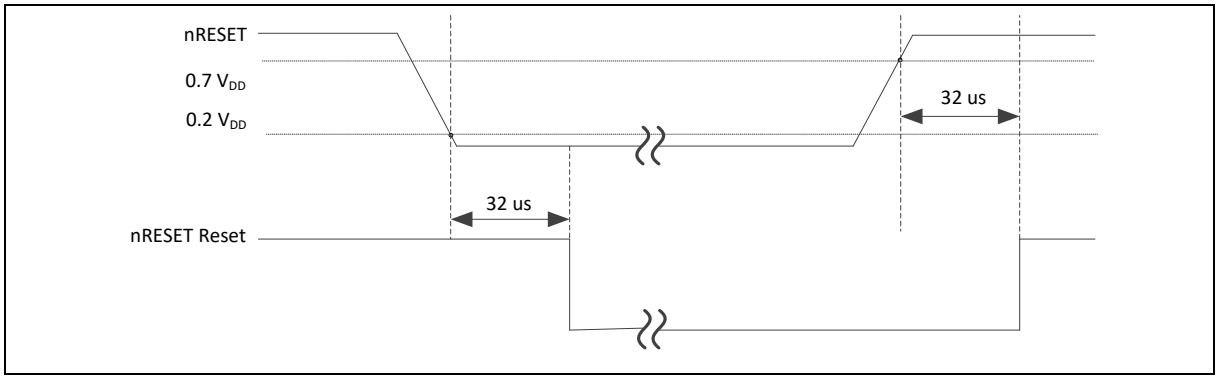


Figure 6.2-2 nRESET Reset Waveform

6.2.2.2 Power-on Reset (POR)

The Power-on reset (POR) is used to generate a stable system reset signal and forces the system to be reset when power-on to avoid unexpected behavior of MCU. When applying the power to MCU, the POR module will detect the rising voltage and generate reset signal to system until the voltage is ready for MCU operation. At POR reset, the PORF(SYS_RSTSTS[0]) will be set to 1 to indicate there is a

POR reset event. The PORF(SYS_RSTSTS[0]) bit can be cleared by writing 1 to it. Figure 6.2-3 shows the power-on reset waveform.

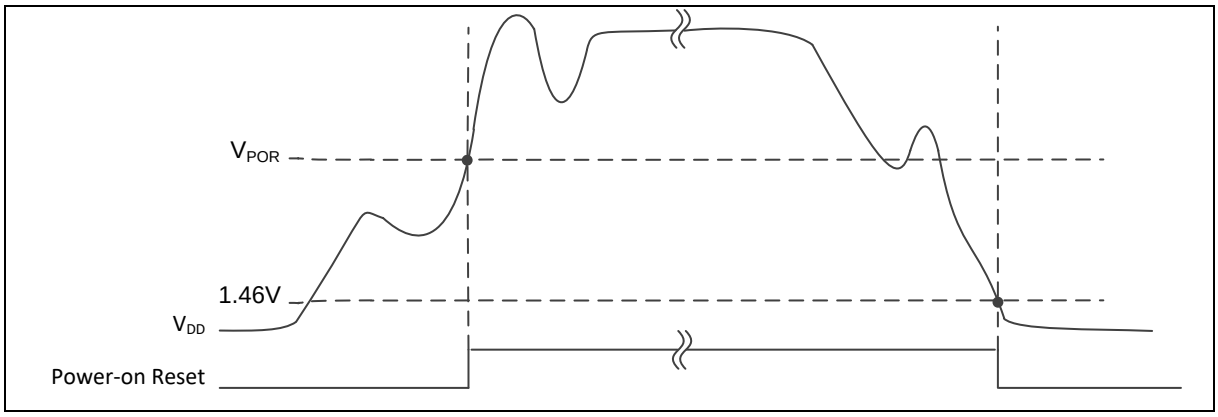


Figure 6.2-3 Power-on Reset (POR) Waveform

6.2.2.3 Low Voltage Reset (LVR)

If the Low Voltage Reset function is enabled by setting the Low Voltage Reset Enable Bit LVREN (SYS_BODCTL[7]) to 1, and wait LVR detection circuit stable flag (SYS_BODCTL[23]) to 1, LVR detection circuit will be stable and the LVR function will be active. Then LVR function will detect AV_{DD} during system operation. When the AV_{DD} voltage is lower than V_{LVR} and the state keeps longer than De-glitch time set by LVRDGSEL (SYS_BODCTL[14:12]), chip will be reset. The LVR reset will control the chip in reset state until the AV_{DD} voltage rises above V_{LVR} and the state keeps longer than De-glitch time set by LVRDGSEL (SYS_BODCTL[14:12]). The default setting of Low Voltage Reset is enabled without De-glitch function. Figure 6.2-4 shows the Low Voltage Reset waveform.

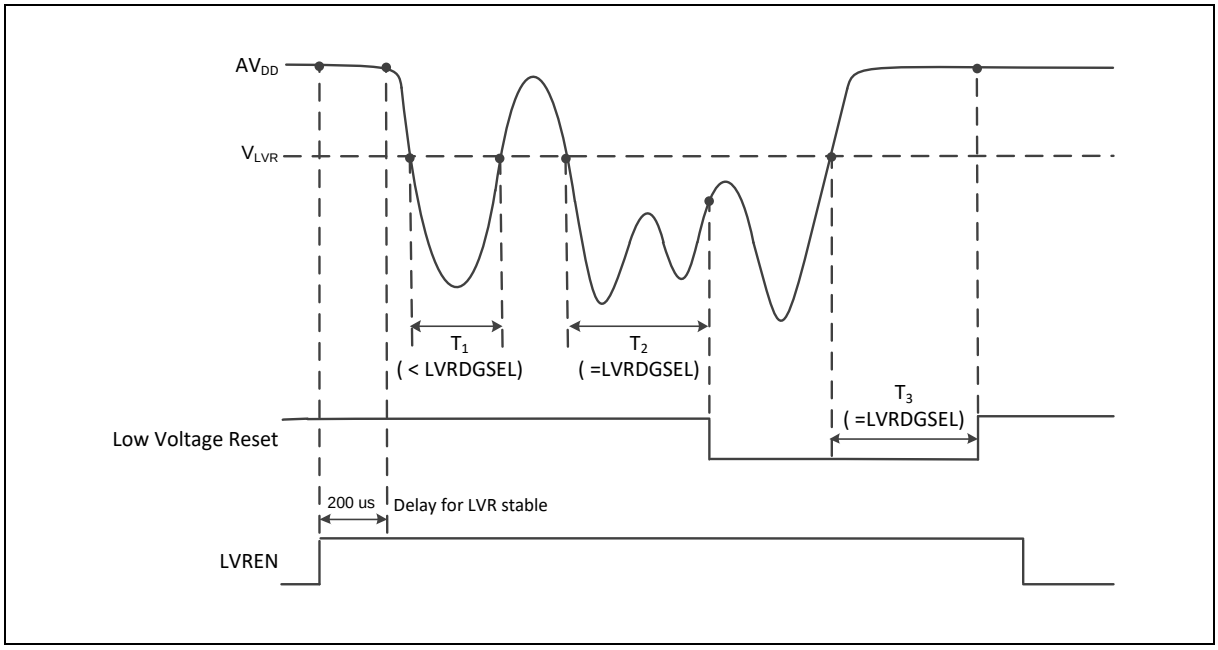


Figure 6.2-4 Low Voltage Reset (LVR) Waveform

6.2.2.4 Brown-out Detector Reset (BOD Reset)

If the Brown-out Detector (BOD) function is enabled by setting the Brown-out Detector Enable Bit BODEN (SYS_BODCTL[0]), and wait BOD detection circuit stable flag STB(SYS_BODCTL[23]) to 1,

BOD detection circuit will be stable and the BOD function will be active. Brown-out Detector function will detect AV_{DD} during system operation. When the AV_{DD} voltage is lower than V_{BOD} that is decided by BODEN and BODVL (SYS_BODCTL[18:16]) and the state keeps longer than De-glitch time set by BODDGSEL (SYS_BODCTL[10:8]), chip will be reset. The BOD reset will control the chip in reset state until the AV_{DD} voltage rises above V_{BOD} and the state keeps longer than De-glitch time set by BODDGSEL. The default value of BODEN, BODVL and BODRSTEN (SYS_BODCTL[3]) is set by Flash controller user configuration register CBODEN (CONFIG0 [19]), CBOV (CONFIG0 [23:21]) and CBORST(CONFIG0[20]) respectively. User can determine the initial BOD setting by setting the CONFIG0 register. Figure 6.2-5 shows the Brown-out Detector waveform.

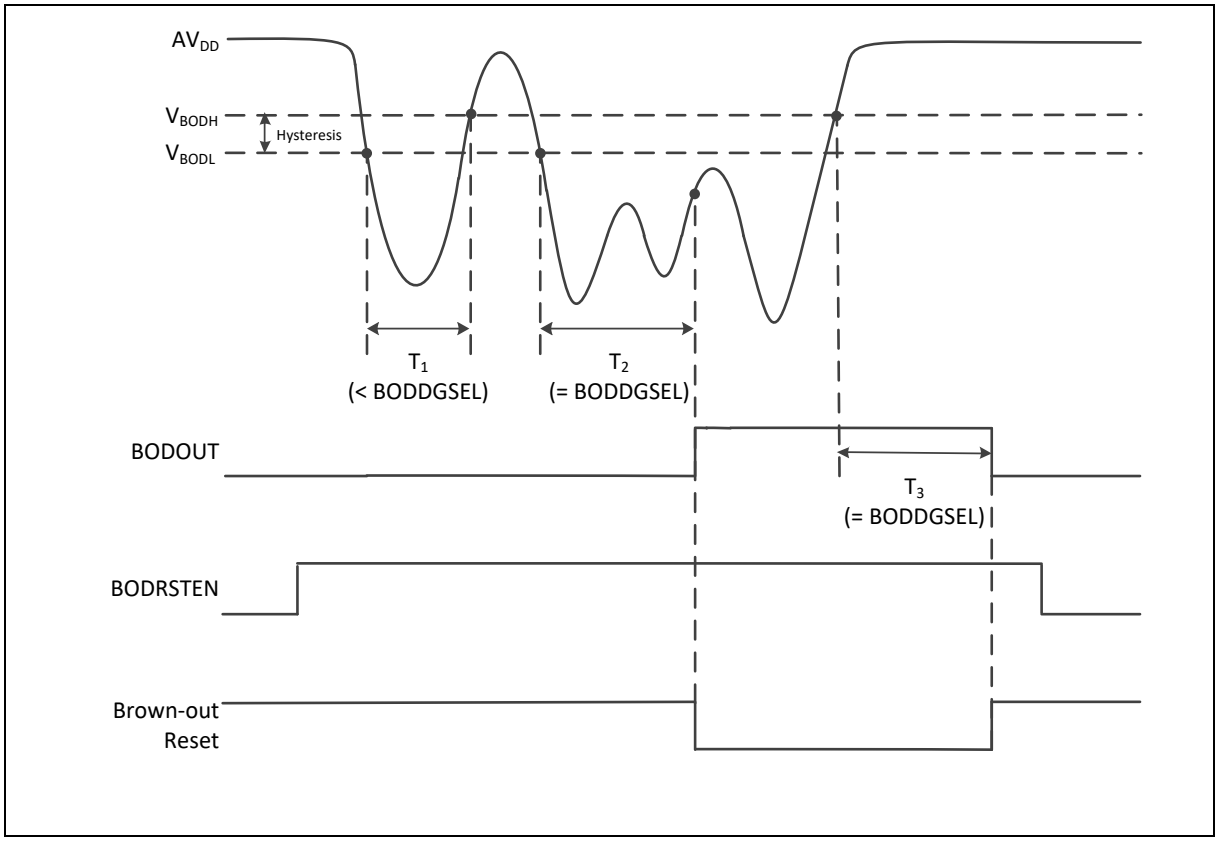


Figure 6.2-5 Brown-out Detector (BOD) Waveform

6.2.2.5 Watchdog Timer Reset (WDT)

In most industrial applications, system reliability is very important. To automatically recover the MCU from failure status is one way to improve system reliability. The watchdog timer(WDT) is widely used to check if the system works fine. If the MCU is crashed or out of control, it may cause the watchdog time-out. User may decide to enable system reset during watchdog time-out to recover the system and take action for the system crash/out-of-control after reset.

Software can check if the reset is caused by watchdog time-out to indicate the previous reset is a watchdog reset and handle the failure of MCU after watchdog time-out reset by checking WDTRF(SYS_RSTSTS[2]).

6.2.2.6 CPU Lockup Reset

CPU enters lockup status after CPU produces hardfault at hardfault handler and chip gives immediate indication of seriously errant kernel software. This is the result of the CPU being locked because of an

unrecoverable exception following the activation of the processor's built in system state protection hardware. When chip enters debug mode, the CPU lockup reset will be ignored.

6.2.2.7 CPU Reset, CHIP Reset and System Reset

The CPU Reset means only Cortex®-M23 core is reset and all other peripherals remain the same status after CPU reset. User can set the CPURST(SYS_IPRST0[1]) to 1 to assert the CPU Reset signal.

The CHIP Reset is same with Power-on Reset. The CPU and all peripherals are reset and BS(FMC_ISPCTL[1]) bit is automatically reloaded from CONFIG0 setting. User can set the CHIPRST(SYS_IPRST0[1]) to 1 to assert the CHIP Reset signal.

The System Reset is similar with CHIP Reset. The difference is that BS(FMC_ISPCTL[1]) will not be reloaded from CONFIG0 setting and keep its original software setting for booting from APROM or LDROM. User can set the SYSRESETREQ(AIRCR[2]) to 1 to assert the System Reset.

6.2.3 Power Modes and Wake-up Sources

The NuMicro® M2354 series has a power manager unit to support several operating modes for saving power. Table 6.2-2 lists all power modes in the NuMicro® M2354 series.

Mode	CPU Operating Maximum Speed (MHz)	LDO_CAP (V)	Clock Disable
Power level 0	96 MHz	1.26	All clocks are disabled by control register. CLK_AHBCLK, CLK_APBCLK0 and CLK_APBCLK1.
Power level 1	84 MHz	1.2	All clocks are disabled by control register. CLK_AHBCLK, CLK_APBCLK0 and CLK_APBCLK1.
Power level 2	48 MHz	1.1	All clocks are disabled by control register. CLK_AHBCLK, CLK_APBCLK0 and CLK_APBCLK1.
Power level 3	4 MHz	0.9	All clocks are disabled by control register. CLK_AHBCLK, CLK_APBCLK0 and CLK_APBCLK1.
Idle mode	CPU enter Sleep mode	keep	Only CPU clock is disabled.
Power-down mode (PD)	CPU enters Deep Sleep mode	keep	Most clocks are disabled except LIRC/LXT/MIRC, and only RTC/WDT/EWDT/Timer/UART/LCD peripheral clocks still enable if their clock sources are selected as LIRC/LXT/MIRC.
Fast Wake-up Power-down mode (FWPD)	CPU enters Deep Sleep mode	keep	Most clocks are disabled except LIRC/LXT/MIRC, and only RTC/WDT/EWDT/Timer/UART/LCD peripheral clocks still enable if their clock sources are selected as LIRC/LXT/MIRC.
Low leakage Power-down mode (LLPD)	CPU enters Deep Sleep mode	0.9	Most clocks are disabled except LIRC/LXT/MIRC, and only RTC/WDT/EWDT/Timer/UART/LCD peripheral clocks still enable if their clock sources are selected as LIRC/LXT/MIRC.
Ultra Low leakage Power-down mode	CPU enters Deep Sleep mode	0.8	Most clocks are disabled except LIRC/LXT, and only RTC/WDT/EWDT/Timer/UART/LCD

(ULLPD)			peripheral clocks still enable if their clock sources are selected as LIRC/LXT.
Standby Power-down mode (SPD)	Power off	0.9 or keep	Only LIRC/LXT still enable for RTC function and wake-up timer usage.
Deep Power-down mode (DPD)	Power off	Floating	Only LIRC/LXT still enable for RTC function and wake-up timer usage.

Table 6.2-2 Power Mode Table

Each power mode has different entry setting and leaving condition. Table 6.2-3 shows the entry setting for each power mode. When chip power-on, chip is running in normal mode. User can enter each mode by setting SLEEPDEEP (SCR[2]), PDEN (CLK_PWRCT[7]) and PDMSEL (CLK_PMUCTL[2:0]) and execute WFI instruction.

Register/Instruction Mode	SLEEPDEEP (SCR[2])	PDEN (CLK_PWRCTL[7])	PDMSEL (CLK_PMUCTL[2:0])	CPU Run WFI Instruction
Normal mode	0	0	0	NO
Idle mode	0	0	0	YES
Power-down mode	1	1	0	YES
Low leakage Power-down mode	1	1	1	YES
Ultra Low leakage Power-down mode	1	1	3	YES
Fast Wake-up Power-down mode	1	1	2	YES
Standby Power-down mode	1	1	4	YES
Deep Power-down mode	1	1	6	YES

Table 6.2-3 Power Mode Entry Setting Table

There are several wake-up sources in Idle mode and Power-down mode. Table 6.2-4 lists the available clocks for each power mode.

Power Mode	Normal Mode	Idle Mode	Power-Down Mode
Definition	CPU is in active state	CPU is in sleep state	CPU is in sleep state and all clocks stop except LXT and LIRC.
Entry Condition	Chip is in normal mode after system reset released	CPU executes WFI instruction.	CPU sets sleep mode enable and power down enable and executes WFI instruction.
Wake-up Sources	N/A	All interrupts	EINT, GPIO, UART, USBD, USBH, OTG, CAN, BOD, WDT, EWDT, SDH, Timer, I²C, USCI, RTC, ACMP, TAMPER and CLKD.
Available Clocks	All	All except CPU clock	LXT, LIRC and MIRC
After Wake-up	N/A	CPU back to normal mode	CPU back to normal mode

Table 6.2-4 Power Mode Difference Table

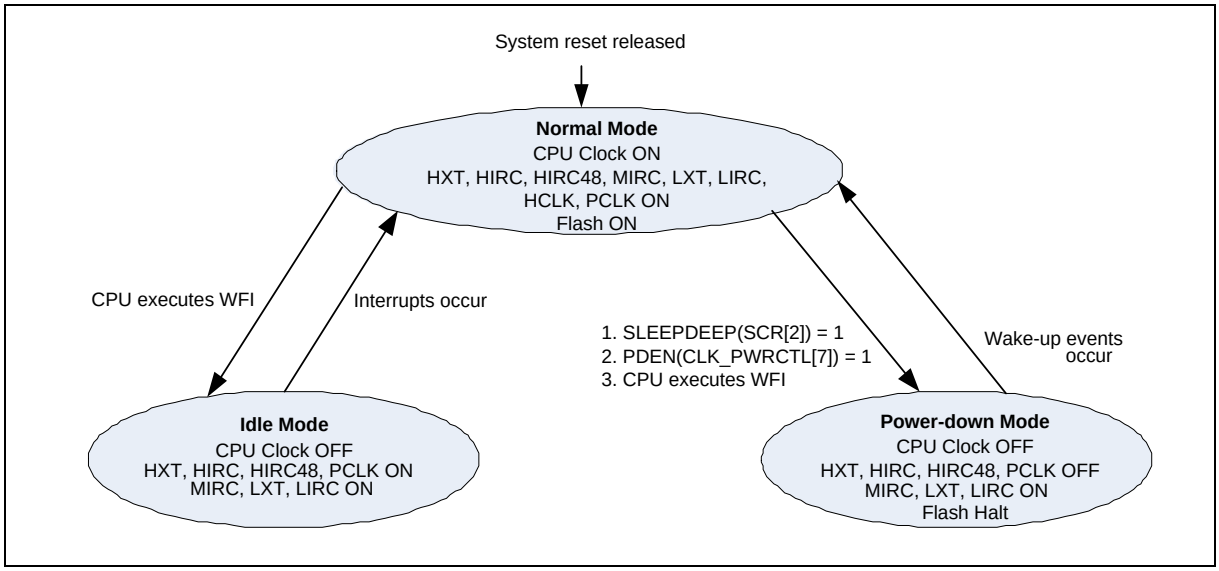


Figure 6.2-6 Power Mode State Machine

1. LXT ON or OFF depends on software setting in normal mode.
2. LIRC ON or OFF depends on software setting in normal mode.
3. MIRC ON or OFF depends on software setting in normal mode.
4. If TIMER clock source is selected as LIRC/LXT/MIRC and LIRC/LXT/MIRC is on.
5. If WDT clock source is selected as LIRC/LXT and LIRC/LXT is on.
6. If RTC clock source is selected as LIRC/LXT and LIRC/LXT is on.
7. If UART clock source is selected as LXT and LXT is on.
8. If LCD clock source is selected as LIRC/LXT and LIRC/LXT is on.
   If LCD charge pump clock source is selected as MIRC/MIRC1P2M and MIRC/MIRC1P2M is on.
9. If EWDT clock source is selected as LIRC/LXT and LIRC/LXT is on.

	Normal Mode	Idle Mode	Power-Down Mode (PD/FWPD/LLPD/ULLPD)	Power-Down Mode (SPD/DPD)
HXT	ON	ON	Halt	Halt
HIRC	ON	ON	Halt	Halt
HIRC48	ON	ON	Halt	Halt
MIRC	ON	ON	ON/OFF[3]	OFF
LXT	ON	ON	ON/OFF[1]	ON/OFF[1]
LIRC	ON	ON	ON/OFF[2]	ON/OFF[2]
PLL	ON	ON	Halt	Halt
CPU	ON	Halt	Halt	Halt
HCLK/PCLK	ON	ON	Halt	Halt

FLASH	ON	ON	Halt	Halt
TIMER	ON	ON	ON/OFF[4]	Halt
WDT	ON	ON	ON/OFF[5]	Halt
RTC	ON	ON	ON/OFF[6]	ON/OFF[6]
UART	ON	ON	ON/OFF[7]	Halt
LCD	ON	ON	ON/OFF[8]	Halt
EWDT	ON	ON	ON/OFF[9]	Halt
Others	ON	ON	Halt	Halt

Table 6.2-5 Clocks in Power Modes

Wake-up sources in Power-down mode:

EINT, GPIO, UART, USBD, USBH, OTG, CAN, BOD, ACMP, WDT, EWDT, SDH, Timer, I²C, USCI, RTC, TAMPER and CLKD.

After chip enters power down, the following wake-up sources can wake chip up to normal mode. Table 6.2-6 lists the condition about how to enter Power-down mode again for each peripheral.

*User needs to wait this condition before setting PDEN(CLK_PWRCTL[7]) and execute WFI to enter Power-down mode.

Wake-Up Source	Wake-Up Condition	Power-down mode			System Can Enter Power-Down Mode Again Condition*
		PD LLPD ULLPD FWPD	SPD	DPD	
BOD	Brown-out Detector Reset / Interrupt	V	-	-	After software writes 1 to clear BODIF (SYS_BODCTL[4]).
	Brown-out Detector Reset	-	V	-	After software writes 1 to clear BODWK (CLK_PMUSTS[13]) when SPD mode is entered.
LVR	LVR Reset	V	-	-	After software writes 1 to clear LVRF (SYS_RSTSTS[3])
		-	V	-	After software writes 1 to clear LVRWK (CLK_PMUSTS[12]) when SPD mode is entered.
POR	POR Reset	V	V	V	After software writes 1 to clear PORF (SYS_RSTSTS[0]).
EINT	External Interrupt	V	-	-	After software write 1 to clear the Px_INTSRC[n] bit.
GPIO	GPIO Interrupt	V	-	-	After software write 1 to clear the Px_INTSRC[n] bit.
GPIO(PA6-PA15,PB~PD) Wake-up pin	rising or falling edge event, 50-pin	-	V	-	GPxWK(CLK_PMUSTS[11:8]) is cleared when SPD mode is entered.
GPIO(PC.0)	rising or falling edge event, 1-pin	-	-	V	PINWK0(CLK_PMUSTS[0]) is cleared when

Wake-up pin					DPD mode is entered.
GPIO(PB.0) Wake-up pin	rising or falling edge event, 1-pin	-	-	V	PINWK1(CLK_PMUSTS[3]) is cleared when DPD mode is entered.
GPIO(PB.2) Wake-up pin	rising or falling edge event, 1-pin	-	-	V	PINWK2(CLK_PMUSTS[4]) is cleared when DPD mode is entered.
GPIO(PB.12) Wake-up pin	rising or falling edge event, 1-pin	-	-	V	PINWK3(CLK_PMUSTS[5]) is cleared when DPD mode is entered.
GPIO(PF.6) Wake-up pin	rising or falling edge event, 1-pin	-	-	V	PINWK4(CLK_PMUSTS[6]) is cleared when DPD mode is entered.
TIMER	Timer Interrupt	V	-	-	After software writes 1 to clear TWKF (TIMERx_INTSTS[1]) and TIF (TIMERx_INTSTS[0]).
Wakeup timer	Wakeup by wake-up timer time-out	-	V	V	After software writes 1 to clear TMRWK (CLK_PMUSTS[1]) when SPD or DPD mode is entered.
WDT	WDT Interrupt	V	-	-	After software writes 1 to clear WKF (WDT_CTL[5]) (Write Protect).
EWDT	EWDT Interrupt	V	-	-	After software writes 1 to clear WKF (EWDT_CTL[5]) (Write Protect).
RTC	Alarm Interrupt	V	-	-	After software writes 1 to clear ALMIF (RTC_INTSTS[0]).
	Time Tick Interrupt	V	-	-	After software writes 1 to clear TICKIF (RTC_INTSTS[1]).
	RTC Tamper Interrupt	V	-	-	After software writes 1 to clear TAMPxIF (RTC_INTSTS[8:13]).
	Wakeup by RTC alarm	-	V	V	RTCWK (CLK_PMUSTS[2]) is cleared when DPD or SPD mode is entered.
	Wakeup by RTC tick time	-	V	V	RTCWK (CLK_PMUSTS[2]) is cleared when DPD or SPD mode is entered.
	Wakeup by tamper event	-	V	V	RTCWK (CLK_PMUSTS[2]) is cleared when DPD or SPD mode is entered.
UART	nCTS wake-up	V	-	-	After software writes 1 to clear CTSWKF (UARTx_WKSTS[0]).
	RX Data wake-up	V	-	-	After software writes 1 to clear DATWKF (UARTx_WKSTS[1]).
	Received FIFO Threshold Wake-up	V	-	-	After software writes 1 to clear RFRTWKF (UARTx_WKSTS[2]).
	RS-485 AAD Mode Wake-up	V	-	-	After software writes 1 to clear RS485WKF (UARTx_WKSTS[3]).
	Received FIFO Threshold Time-out Wake-up	V	-	-	After software writes 1 to clear TOUTWKF (UARTx_WKSTS[4]).
USCI UART	CTS Toggle	V	-	-	After software writes 1 to clear WKF (UUART_WKSTS[0]).
	Data Toggle	V	-	-	After software writes 1 to clear WKF (UUART_WKSTS[0]).

USCI I²C	Data toggle	V	-	-	After software writes 1 to clear WKF (UI2C_WKSTS[0]).
	Address match	V	-	-	After software writes 1 to clear WKAKDONE (UI2C_PROTSTS[16], then writes 1 to clear WKF (UI2C_WKSTS[0]).
USCI SPI	SS Toggle	V	-	-	After software writes 1 to clear WKF (USPI_WKSTS[0]).
I²C	Address match wake-up	V	-	-	After software writes 1 to clear WKAKDONE (I2C_WKSTS[1]). Then software writes 1 to clear WKIF(I2C_WKSTS[0]).
USBD	1.Remote wake-up 2.Plug in wake-up	V	-	-	After software writes 1 to clear BUSIF (USBD_INTSTS[0]).
USBH	1.Connection detected 2.Disconnect detected 3.Remote-wakeup	V	-	-	1.After write 1 to clear RHSC (HcInterruptStatus[7]). 2.After write 1 to clear RHSC (HcInterruptStatus[7]). 3.After write 1 to clear RHSC (HcInterruptStatus[7]). and port suspended.
OTG	ID pin state be change	V	-	-	After software writes 1 to set WKEN(OTG_CTL[5]).
ACMP	Comparator Power-Down Wake-Up Interrupt	V	-	-	After software writes 1 to clear WKIF0 (ACMP_STATUS[8]) and WKIF1 (ACMP_STATUS[9]).
	ACMPO status change	-	V	-	ACMPWK (CLK_PMUSTS[14]) is cleared when SPD mode is entered.
CAN	Incoming Data Toggle	V	-	-	After software writes 0 to clear WAKUP_STS (CAN_WU_STATUS[0])
SDH	Card detection	V	-	-	Clear CDIF0 (SDH_INTSTS[8]) after SDH wake-up.
TAMPER	Event detection	V	-	-	Clear TAMP_EVSTS or disable Enable in TAMP_INTEVEN after TAMPER wake-up.
	Wakeup by Event detection	-	V	-	TAMPERWK (CLK_PMUSTS[15]) is cleared when SPD mode is entered.
CLKD	LXT clock fail interrupt	V	-	-	After software writes 1 to clear LXTFIF (CLK_CLKDSTS[1]).

Table 6.2-6 Condition of Entering Power-down Mode Again

6.2.4 System Power Distribution

In this chip, power distribution is divided into four segments:

- Analog power from AV_{DD} and AV_{SS} provides the power for analog components operation.
- Digital power from V_{DD} and V_{SS} supplies the power to the internal regulator which provides a fixed 0.9V, 1.1V, 1.2V or 1.26V power for digital operation and I/O pins.
- USB transceiver power from V_{DD} offers the power for operating the USB transceiver.
- RTC power from V_{BAT} provides the power for RTC and 80 bytes backup registers.

The outputs of internal voltage regulators, LDO and V_{DD}, require an external capacitor which should be located close to the corresponding pin. Analog power (AV_{DD}) should be the same voltage level of the

digital power (V_{DD}). If system enters SPD mode SW_SPD switch is turned off, and internal voltage regulator can be set to LDO mode or DC-DC converter mode. Figure 6.2-7 shows the power distribution.

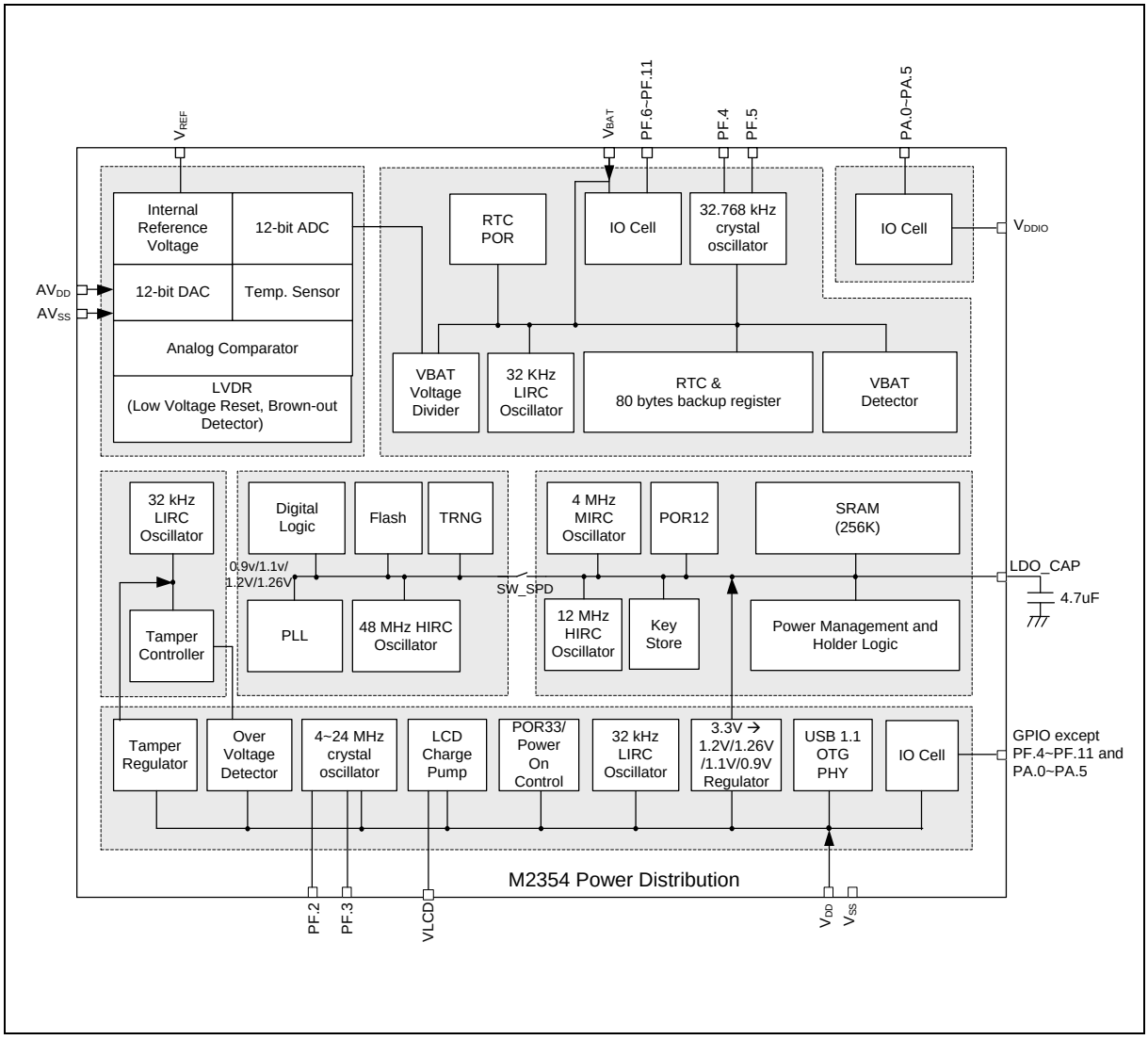


Figure 6.2-7 Power Distribution Diagram

6.2.5 Bus Matrix

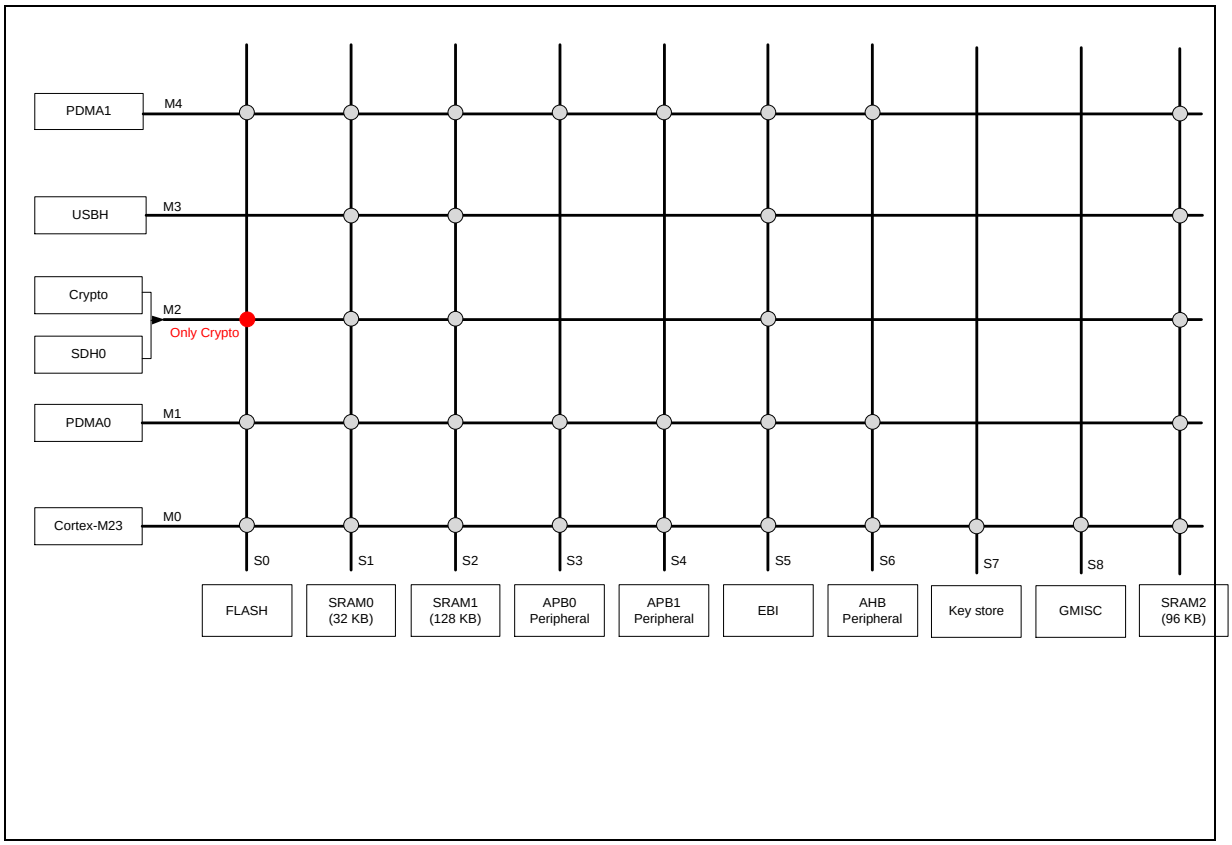


Figure 6.2-2 M2354 Bus Matrix Architecture Diagram

Refer to Figure 6.2-2. This chip uses Advanced Microcontroller Bus Architecture (AMBA) protocol to implement system bus. The system has five masters and nine slaves, in which a different master can communicate with a different slave at the same time through Bus Matrix. The Cortex®-M23 core processor acts as the master in Bus Matrix, located on M0 to communicate with any slaves through Bus Matrix. PDMA0 and PDMA1 are Peripheral Direct Memory Access and act as the master in Bus Matrix, respectively located on M1 and M4, which can communicate with any slaves through Bus Matrix. SDH0 and Crypto share the same master bandwidth located on M2. USBH acts as the master role in Bus Matrix and is located on M3. The slave AHB Peripheral is the Advanced High-performance Bus (AHB) controller, and any master can communicate with any AHB peripheral through Bus Matrix.

6.2.6 System Memory Map

This chip provides 4G-byte addressing space. The memory locations assigned to each on-chip controllers are shown in Table 6.2-7. The detailed register definition, memory space, and programming will be described in the following sections for each on-chip peripheral. This chip implement Arm® TrustZone Architecture as well as memory alias technique, secure code and non-secure code can run together on the chip well, while both have different memory view. Secure code view is shown in Table 6.2-1 and non-secure code view is shown in Table 6.2-2.

The NuMicro® M2354 series only supports little-endian data format.

Address Space	Token	Controllers
Flash and SRAM Memory Space		

0x0000_0000 – 0x0003_FFFF	FLASH_BA	FLASH Memory Space (256 KB)
0x0000_0000 – 0x0007_FFFF	FLASH_BA	FLASH Memory Space (512 KB)
0x0000_0000 – 0x000F_FFFF	FLASH_BA	FLASH Memory Space (1024 KB)
0x2000_0000 – 0x2000_7FFF	SRAM0_BA	SRAM Memory Space (32 KB)
0x2000_8000 – 0x2002_7FFF	SRAM1_BA	SRAM Memory Space (128 KB)
0x2002_8000 – 0x2003_FFFF	SRAM2_BA	SRAM Memory Space (96 KB)
0x6000_0000 – 0x6FFF_FFFF	EXTMEM_BA	External Memory Space (256 MB)
Secure Peripheral Controllers Space (0x4000_0000 – 0x400F_FFFF)		
0x4000_0000 – 0x4000_01FF	SYS_BA	System Control Registers (always secure)
0x4000_0200 – 0x4000_02FF	CLK_BA	Clock Control Registers (always secure)
0x4000_0300 – 0x4000_03FF	NMI_BA	NMI Control Registers (always secure)
0x4000_4000 – 0x4000_4FFF	GPIO_BA	GPIO Control Registers
0x4000_8000 – 0x4000_8FFF	PDMA0_BA	Peripheral DMA 0 Control Registers (always secure)
0x4000_9000 – 0x4000_9FFF	USBH_BA	USB Host Control Registers
0x4000_C000 – 0x4000_CFFF	FMC_BA	Flash Memory Control Registers (always secure)
0x4000_D000 – 0x4000_DFFF	SDH0_BA	SDHOST0 Control Registers
0x4001_0000 – 0x4001_0FFF	EBI_BA	External Bus Interface Control Registers
0x4001_8000 – 0x4000_8FFF	PDMA1_BA	Peripheral DMA 1 Control Registers (secure or non-secure)
0x4003_1000 – 0x4003_1FFF	CRC_BA	CRC Generator Registers
0x4003_2000 – 0x4003_4FFF	CRPT_BA	Cryptographic Accelerator Registers
0x4003_5000 – 0x4003_5FFF	KS_BA	Key Store Registers (always secure)
0x4002_F000 – 0x4002_FFFF	SCU_BA	Secure Configuration Unit Registers (always secure)
Secure APB Controllers Space (0x4004_0000 ~ 0x400F_FFFF)		
0x4004_0000 – 0x4004_0FFF	WDT_BA	Watchdog Timer Control Registers (always secure)
0x4004_1000 – 0x4004_1FFF	RTC_BA	Real Time Clock (RTC) Control Register (always secure)
0x4004_2000 – 0x4004_2FFF	EWDT_BA	Extra Watchdog Timer Control Registers
0x4004_3000 – 0x4004_3FFF	EADC_BA	Enhanced Analog-Digital-Converter (EADC) Control Registers
0x4004_5000 – 0x4004_5FFF	ACMP01_BA	Analog Comparator 0/1 Control Registers
0x4004_7000 – 0x4004_7FFF	DAC_BA	DAC Control Registers
0x4004_8000 – 0x4004_8FFF	I2S0_BA	I2S0 Interface Control Registers
0x4004_D000 – 0x4004_DFFF	OTG_BA	OTG Control Registers
0x4005_0000 – 0x4005_0FFF	TMR01_BA	Timer0/Timer1 Control Registers (always secure)
0x4005_1000 – 0x4005_1FFF	TMR23_BA	Timer2/Timer3 Control Registers
0x4005_2000 – 0x4005_2FFF	TMR45_BA	Timer4/Timer5 Control Registers

0x4005_8000 – 0x4005_8FFF	EPWM0_BA	EPWM0 Control Registers
0x4005_9000 – 0x4005_9FFF	EPWM1_BA	EPWM1 Control Registers
0x4005_A000 – 0x4005_AFFF	BPWM0_BA	BPWM0 Control Registers
0x4005_B000 – 0x4005_BFFF	BPWM1_BA	BPWM1 Control Registers
0x4006_0000 – 0x4006_0FFF	QSPI0_BA	QSPI0 Control Registers
0x4006_1000 – 0x4006_1FFF	SPI0_BA	SPI0 Control Registers
0x4006_2000 – 0x4006_2FFF	SPI1_BA	SPI1 Control Registers
0x4006_3000 – 0x4006_3FFF	SPI2_BA	SPI2 Control Registers
0x4006_4000 – 0x4006_4FFF	SPI3_BA	SPI3 Control Registers
0x4007_0000 – 0x4007_0FFF	UART0_BA	UART0 Control Registers
0x4007_1000 – 0x4007_1FFF	UART1_BA	UART1 Control Registers
0x4007_2000 – 0x4007_2FFF	UART2_BA	UART2 Control Registers
0x4007_3000 – 0x4007_3FFF	UART3_BA	UART3 Control Registers
0x4007_4000 – 0x4007_4FFF	UART4_BA	UART4 Control Registers
0x4007_5000 – 0x4007_5FFF	UART5_BA	UART5 Control Registers
0x4008_0000 – 0x4008_0FFF	I2C0_BA	I2C0 Control Registers
0x4008_1000 – 0x4008_1FFF	I2C1_BA	I2C1 Control Registers
0x4008_2000 – 0x4008_2FFF	I2C2_BA	I2C2 Control Registers
0x4009_0000 – 0x4009_0FFF	SC0_BA	Smartcard Host 0 Control Registers
0x4009_1000 – 0x4009_1FFF	SC1_BA	Smartcard Host 1 Control Registers
0x4009_2000 – 0x4009_2FFF	SC2_BA	Smartcard Host 2 Control Registers
0x400A_0000 – 0x400A_0FFF	CAN0_BA	CAN0 Bus Control Registers
0x400B_0000 – 0x400B_0FFF	QEI0_BA	QEI0 Control Registers
0x400B_1000 – 0x400B_1FFF	QEI1_BA	QEI1 Control Registers
0x400B_4000 – 0x400B_4FFF	ECAP0_BA	ECAP0 Control Registers
0x400B_5000 – 0x400B_5FFF	ECAP1_BA	ECAP1 Control Registers
0x400B_9000 – 0x400B_9FFF	TRNG_BA	TRNG Control Registers
0x400B_B000 – 0x400B_BFFF	LCD_BA	LCD Control Register
0x400B_D000 – 0x400B_DFFF	TAMPER_BA	Tamper Control Register (always secure)
0x400C_0000 – 0x400C_0FFF	USBD_BA	USB Device Control Register
0x400D_0000 – 0x400D_0FFF	USCI0_BA	USCI0 Control Registers
0x400D_1000 – 0x400D_1FFF	USCI1_BA	USCI1 Control Registers

Table 6.2-7 Address Space Assignments for On-Chip Controllers

Address Space	Token	Controllers
Non-secure Peripheral Controllers Space (0x5000_0000 – 0x500F_FFFF)		
0x5000_0000 – 0x5000_01FF	SYS_BA_NS	System Control Registers
0x5000_4000 – 0x5000_4FFF	GPIO_BA	GPIO Control Registers
0x5000_9000 – 0x5000_9FFF	USBH_BA	USB Host Control Registers
0x5000_D000 – 0x5000_DFFF	SDH0_BA	SDHOST0 Control Registers
0x5001_0000 – 0x5001_0FFF	EBI_BA	External Bus Interface Control Registers
0x5001_8000 – 0x5000_8FFF	PDMA1_BA	Peripheral DMA 1 Control Registers (secure or non-secure)
0x5003_1000 – 0x5003_1FFF	CRC_BA	CRC Generator Registers
0x5003_2000 – 0x5003_4FFF	CRPT_BA	Cryptographic Accelerator Registers
Non-secure APB Controllers Space (0x5004_0000 ~ 0x500F_FFFF)		
0x5004_2000 – 0x5004_2FFF	EWDT_BA	Extra Watchdog Timer Control Registers
0x5004_3000 – 0x5004_3FFF	EADC_BA	Enhanced Analog-Digital-Converter (EADC) Control Registers
0x5004_5000 – 0x5004_5FFF	ACMP01_BA	Analog Comparator 0/ 1 Control Registers
0x5004_7000 – 0x5004_7FFF	DAC_BA	DAC Control Registers
0x5004_8000 – 0x5004_8FFF	I2S0_BA	I2S0 Interface Control Registers
0x5004_D000 – 0x5004_DFFF	OTG_BA	OTG Control Registers
0x5005_1000 – 0x5005_1FFF	TMR23_BA	Timer2/Timer3 Control Registers
0x5005_2000 – 0x5005_2FFF	TMR45_BA	Timer4/Timer5 Control Registers
0x5005_8000 – 0x5005_8FFF	EPWM0_BA	EPWM0 Control Registers
0x5005_9000 – 0x5005_9FFF	EPWM1_BA	EPWM1 Control Registers
0x5005_A000 – 0x5005_AFFF	BPWM0_BA	BPWM0 Control Registers
0x5005_B000 – 0x5005_BFFF	BPWM1_BA	BPWM1 Control Registers
0x5006_0000 – 0x5006_0FFF	QSPI0_BA	QSPI0 Control Registers
0x5006_1000 – 0x5006_1FFF	SPI0_BA	SPI0 Control Registers
0x5006_2000 – 0x5006_2FFF	SPI1_BA	SPI1 Control Registers
0x5006_3000 – 0x5006_3FFF	SPI2_BA	SPI2 Control Registers
0x5006_4000 – 0x5006_4FFF	SPI3_BA	SPI3 Control Registers
0x5007_0000 – 0x5007_0FFF	UART0_BA	UART0 Control Registers
0x5007_1000 – 0x5007_1FFF	UART1_BA	UART1 Control Registers
0x5007_2000 – 0x5007_2FFF	UART2_BA	UART2 Control Registers
0x5007_3000 – 0x5007_3FFF	UART3_BA	UART3 Control Registers
0x5007_4000 – 0x5007_4FFF	UART4_BA	UART4 Control Registers
0x5007_5000 – 0x5007_5FFF	UART5_BA	UART5 Control Registers

0x5008_0000 – 0x5008_0FFF	I2C0_BA	I2C0 Control Registers
0x5008_1000 – 0x5008_1FFF	I2C1_BA	I2C1 Control Registers
0x5008_2000 – 0x5008_2FFF	I2C2_BA	I2C2 Control Registers
0x5009_0000 – 0x5009_0FFF	SC0_BA	Smartcard Host 0 Control Registers
0x5009_1000 – 0x5009_1FFF	SC1_BA	Smartcard Host 1 Control Registers
0x5009_2000 – 0x5009_2FFF	SC2_BA	Smartcard Host 2 Control Registers
0x500A_0000 – 0x500A_0FFF	CAN0_BA	CAN0 Bus Control Registers
0x500B_0000 – 0x500B_0FFF	QEI0_BA	QEI0 Control Registers
0x500B_1000 – 0x500B_1FFF	QEI1_BA	QEI1 Control Registers
0x500B_4000 – 0x500B_4FFF	ECAP0_BA	ECAP0 Control Registers
0x500B_5000 – 0x500B_5FFF	ECAP1_BA	ECAP1 Control Registers
0x500B_9000 – 0x500B_9FFF	TRNG_BA	TRNG Control Registers
0x400B_B000 – 0x400B_BFFF	LCD_BA	LCD Control Register
0x500C_0000 – 0x500C_0FFF	USBD_BA	USB Device Control Register
0x500D_0000 – 0x500D_0FFF	USCI0_BA	USCI0 Control Registers
0x500D_1000 – 0x500D_1FFF	USCI1_BA	USCI1 Control Registers

Table 6.2-2 Non-secure Address Space Assignments for On-Chip Controllers

6.2.7 Implementation Defined Attribution Unit (IDAU)

6.2.7.1 Overview

The Arm®v8-M has the new feature called TrustZone®, which adds an additional security state to allow full isolation of two security levels. The processor security state is decided by the memory definition. For example, processor is in Secure state when the code is executed in the Secure region. The memory map security state will be defined by the combination of:

- Internal Security Attribution Unit (SAU)
- Implementation Defined Attribution Unit (IDAU)

These attribution units define the memory space into four type regions:

- Secure Region: contains Secure program code or data
- Non-secure Callable Region (NSC): contains entry functions for Non-secure programs to access Secure functions
- Non-secure Region: contains Non-secure program code or data
- Exempt Region: exempt region will be exempted from security check

For each memory region defined by the SAU and IDAU has a region number generated by the SAU or by the IDAU. Region number is used for determining a group of memory share the same security attribute. Overlapping region numbers are not allow. For testing security attributes and region numbers, a new instruction “TT” (Test Target) is introduced. By using a TT instruction on the start and end addresses of the memory range, and identifying that both reside in the same region number, user can determine that the memory range is located entirely in same space. To be more specific, please refer to the Arm®v8-M Architecture Reference Manual. The M2354 IDAU memory map attributions and corresponding region numbers are shown in Figure 6.2-8. The address from 0xE000_0000 to

0xFFFF_FFFF is marked as exempt regions because the behavior of the address is fixed, so their security attributes do not control by the SAU or IDAU.

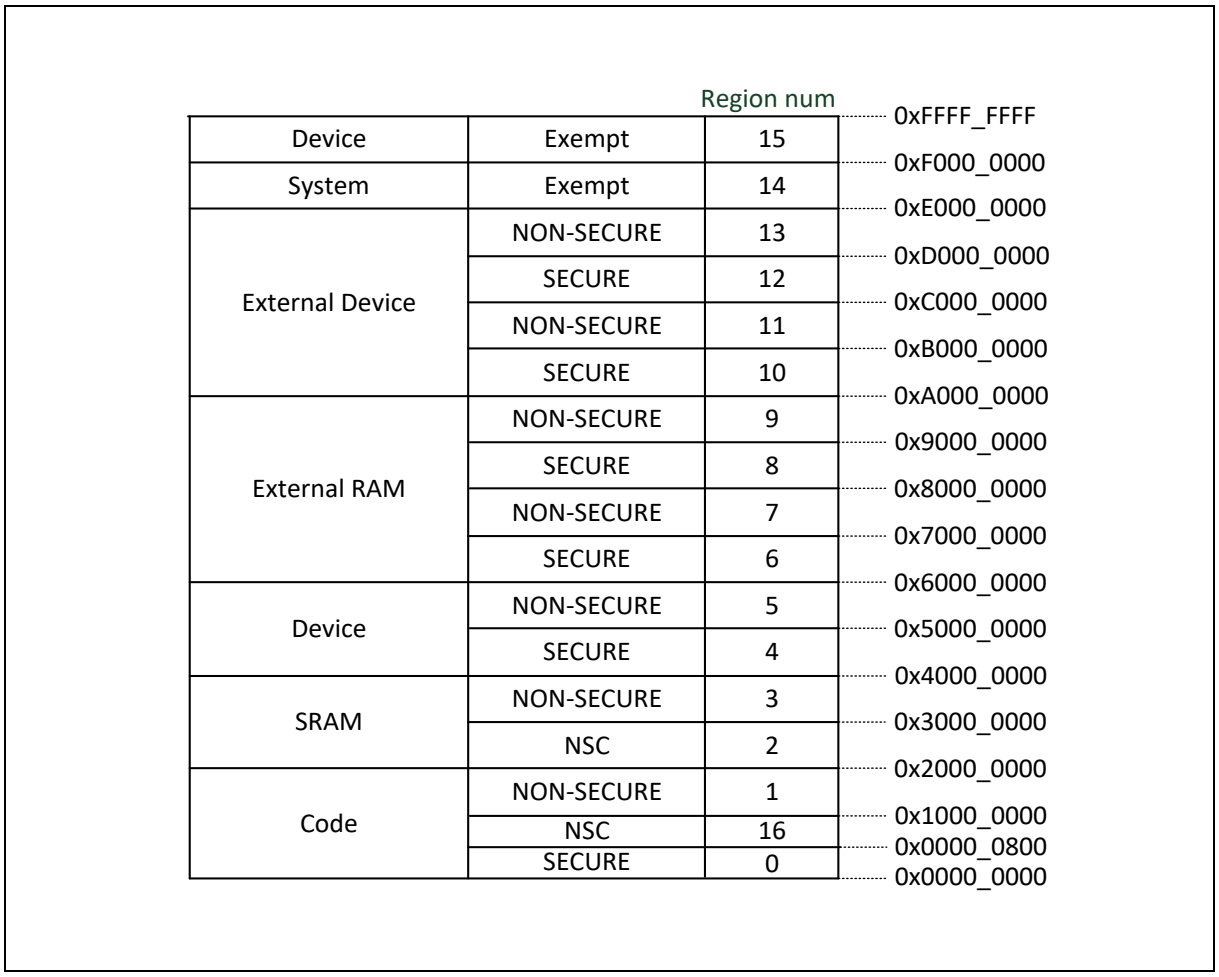


Figure 6.2-8 IDAU Memory Map

6.2.7.2 IDAU Block Diagram

The IDAU block diagram is shown in Figure 6.2-9. IDAU is security attribute unit connected outside of the processor. Both SAU and IDAU are responsible to response the security property of the address from processor, the only difference is that the memory security attribute of the SAU is configurable and the IDAU is fixed. After the processor compares the security property of the IDAU and SAU, it will take the highest security attribute applied. The hierarchy of security levels from high to low is: Secure > NSC > Non-secure.

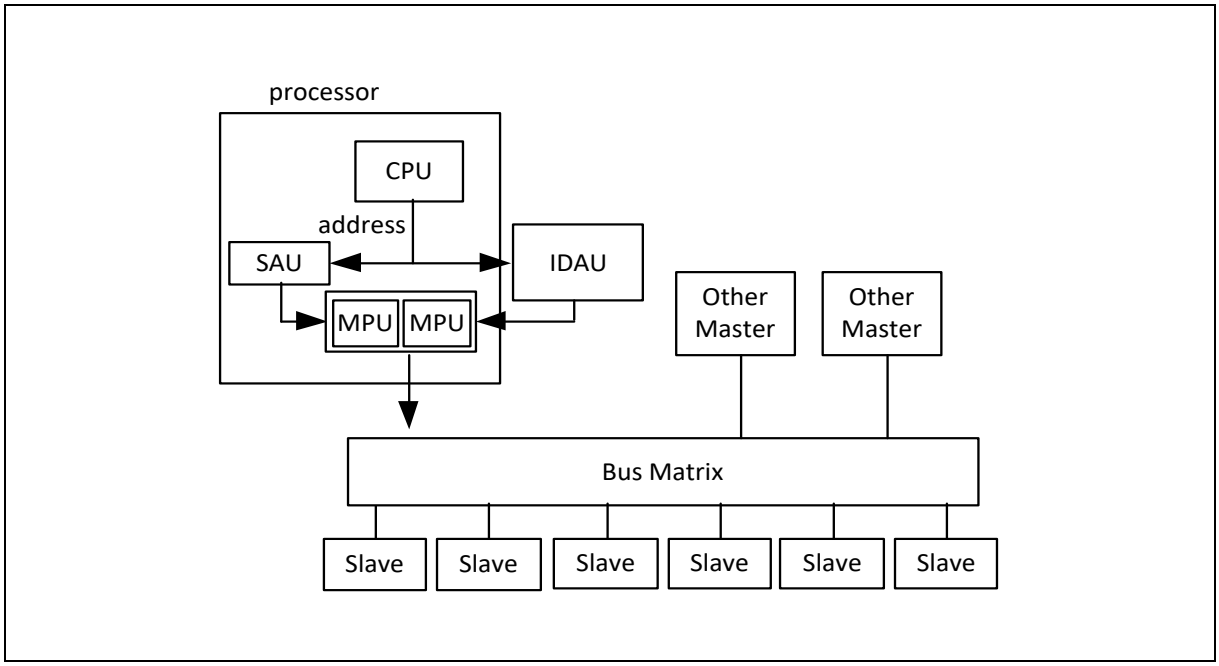


Figure 6.2-9 IDAU Block Diagram

6.2.8 SRAM Memory Organization

This chip supports embedded SRAM with a total of 256 Kbytes size and the SRAM organization is separated into three banks: SRAM bank0, SRAM bank1, and SRAM bank2. The first bank has 32 Kbytes address space, the second bank has 128Kbyte address space, and the third bank has 96Kbyte address space. These three banks address space can be accessed simultaneously. The SRAM bank0 supports parity error check to make sure the chip is operating more stable.

- Supports total 256 Kbytes SRAM
- Supports byte / half word / word write
- Supports parity error check function for SRAM bank0
- Supports oversize response error

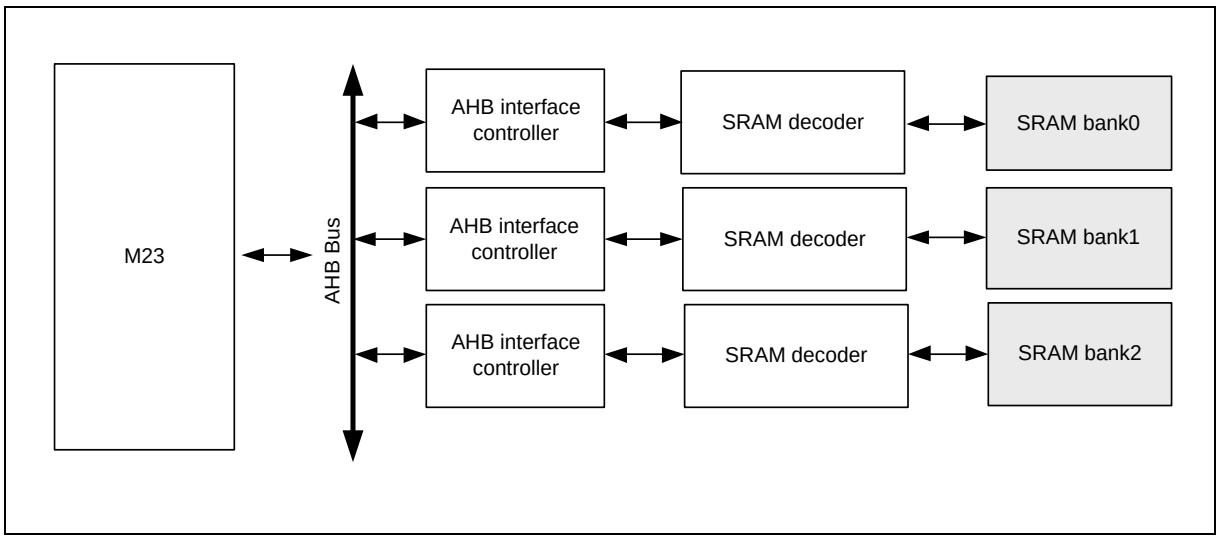


Figure 6.2-10 SRAM Block Diagram

Figure 6.2-11 shows the SRAM organization. There are three SRAM banks. The bank0 is addressed to 32 Kbytes, the bank1 is addressed to 128 Kbytes and the bank2 is addressed to 96 Kbytes. The bank0 address space is from 0x2000_0000 to 0x2000_7FFF(Secure) or 0x3000_0000 to 0x3000_7FFF(Non-secure). The bank1 address space is from 0x2000_8000 to 0x2002_7FFF (Secure) or 0x3000_8000 to 0x3002_7FFF (Non-secure). The bank2 address space is from 0x2002_8000 to 0x2003_FFFF (Secure) or 0x3002_8000 to 0x3003_FFFF (Non-secure). The address between 0x2004_0000 to 0x2FFF_FFFF(Secure) and 0x3004_0000 to 0x3FFF_FFFF(Non-secure) is illegal memory space and chip will enter hardfault if CPU accesses these illegal memory addresses.

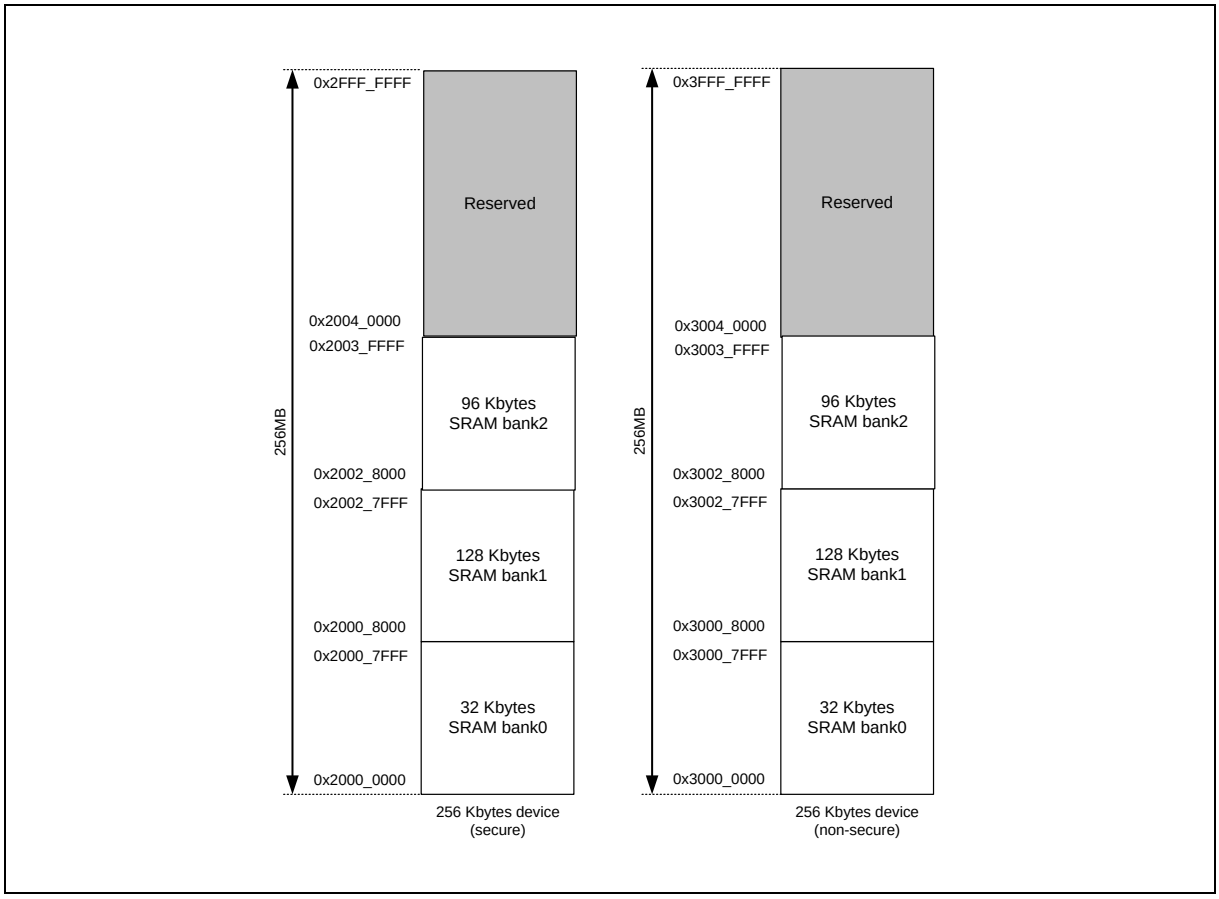


Figure 6.2-11 SRAM Memory Organization

The SRAM bank0 has byte parity error check function. When CPU is accessing SRAM bank0, the parity error checking mechanism is dynamic operating. As parity error occurs, the PERRIF (SYS_SRAMSTS[0]) will be asserted to 1 and the SYS_SRAMEADR register will recode the address with the parity error. Chip will enter interrupt when SRAM parity error occurs if PERRIEN (SYS_SRAMICTL[0]) is set to 1. When SRAM parity error occurs, chip will stop detecting SRAM parity error until user writes 1 to clear the PERRIF(SYS_SRAMSTS[0]) bit.

SRAM Power Control

The SRAM bank0 and bank1, and bank2 have marco retention and power shut down function. Each SRAM marco can be configured to retention or power shut down mode independently by SRAMxPMn(SYS_SRAMPC0 and SYS_SRAMPC1, x=0-2 n=0-7). Figure 6.2-12 shows the SRAM marco number in bank0, bank1 and bank2. When chip power down wake up, the SRAM marcos wake up in the order of marco number, from SRAM marco0 to SRAM marco17.

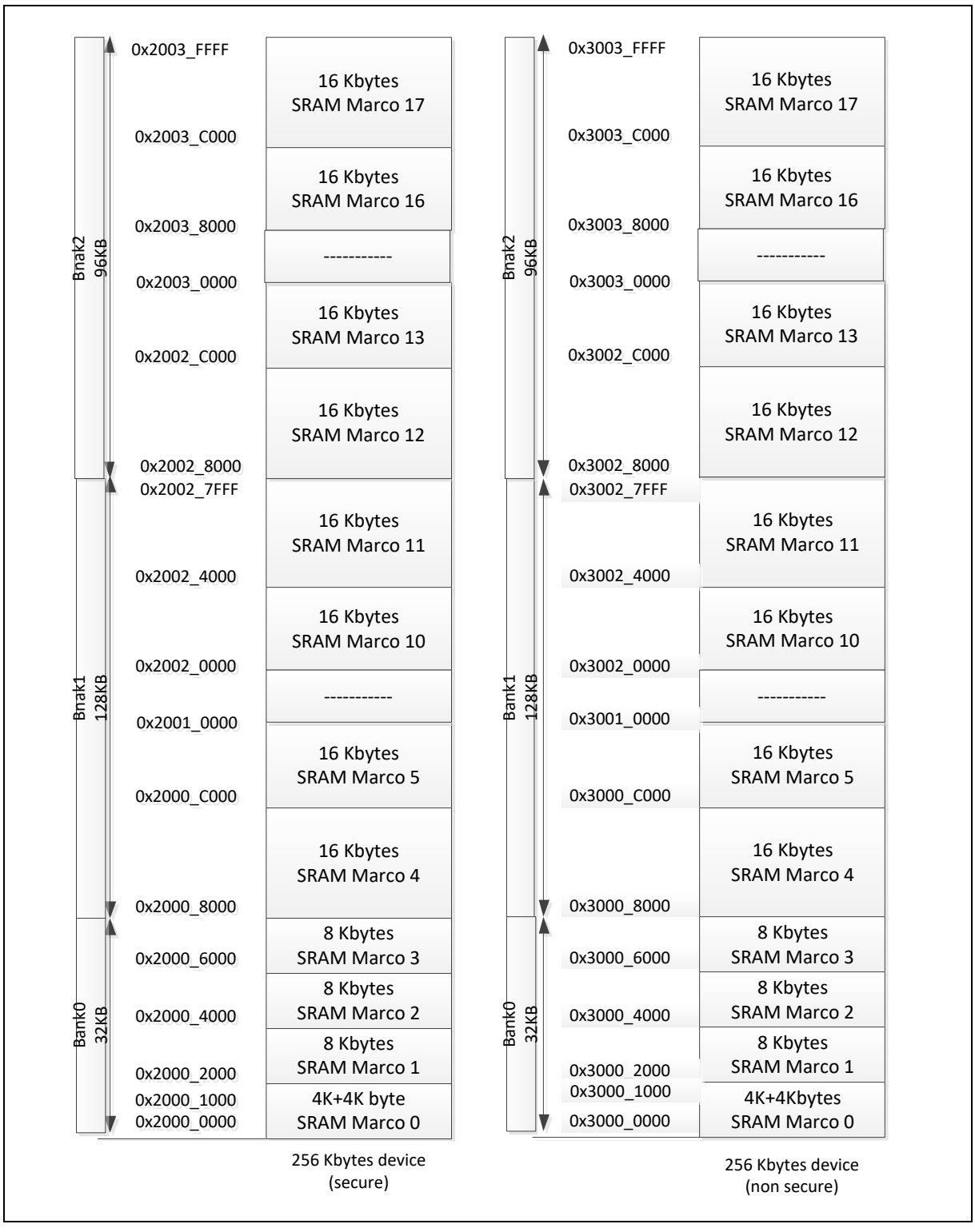


Figure 6.2-12 SRAM Marco Organization

For system SRAM (bank0, bank1, and bank2) and Key store SRAM, if the SRAM power mode is set to normal mode, it automatically changes to retention mode when system enters PD/LLPD/ULLPD/SPD

Power-down mode, and then changes back to normal mode after wake-up. System and Key store SRAM power mode do not change when system enters FWPD Power-down mode. When system enters DPD Power-down mode, system and Key store SRAM is always operating in power shut down mode and reset to normal mode after wake-up.

For other peripheral SRAM, when system enters PD/LLPD/ULLPD/SPD Power-down mode, if peripheral SRAM power mode is set to normal mode, the peripheral SRAM power mode automatically changes to retention mode, and then changes back to normal mode after wake-up, too.

But if entering SPD Power-down mode, peripheral SRAM resets to default power mode after wake-up.

When system enters DPD Power-down mode, peripheral SRAM is always operating in power shut down mode and reset to default power mode after wake-up. Peripheral SRAM power mode does not change when system enters FWPD Power-down mode.

SRAM	Power-Down Mode	SRAM Power Mode Before And After Wake-Up
SRAM bank0/1/2 Key Store SRAM	PD LLPD ULLPD SPD	1.If SRAM power mode is set to normal mode, it changes to retention mode after entering Power-down Mode, and changes back to normal mode after wake-up. 2.If SRAM power mode is set to Retention or Power shut down mode, it keeps power mode setting.
	FWPD	SRAM power mode keeps power mode setting.
	DPD	SRAM power mode is always operating in power shut down mode after entering Power-down Mode, and resets to normal mode after wake-up.
Other Peripheral SRAM	PD LLPD ULLPD	1.If SRAM power mode is set to normal mode, it changes to retention mode after entering Power-down Mode, and changes back to normal mode after wake-up. 2.If SRAM power mode is set to Retention or Power shut down mode, it keeps power mode setting.
	FWPD	SRAM power mode keeps power mode setting.
	SPD	1. If SRAM power mode is set to normal mode, it changes to retention mode after entering Power-down Mode. 2. If SRAM power mode is set to Retention or Power shut down mode, it keeps power mode setting. 3. SRAM power mode is reset to default power mode after wake-up.
	DPD	SRAM power mode is always operating in power shut down mode after entering Power-down Mode, and resets to default power mode after wake-up.

Table 6.2-8 SRAM Power Mode Behavior

6.2.9 Auto Trim

This chip supports auto-trim function: the HIRC trim (12 MHz RC oscillator, 48 MHz RC oscillator), according to the accurate external 32.768 kHz crystal oscillator or internal USB synchronous mode, to automatically get accurate HIRC output frequency, 0.25 % deviation within all temperature ranges.

For instance, the system needs an accurate 12 MHz clock. In such case, if neither using PLL as the system clock source nor soldering 32.768 kHz crystal in system, user has to set REFCKSEL (SYS_TCTL12M[10] reference clock selection) to “1”, set FREQSEL (SYS_TCTL12M[1:0] trim frequency selection) to “01”, and the auto-trim function will be enabled. Interrupt status bit FREQLOCK (SYS_TISTS12M[8] HIRC frequency lock status) “1” indicates the HIRC output frequency is accurate within 0.25% deviation. In another case, the system needs an accurate 48 MHz clock. In such case, if neither using PLL as the system clock source nor soldering 32.768 kHz crystal in system, user has to

set REFCKSEL (SYS_TCTL48M[10] reference clock selection) to “1”, set FREQSEL (SYS_TCTL48M[1:0] trim frequency selection) to “01”, and the auto-trim function will be enabled. Interrupt status bit FREQLOCK (SYS_TISTS48M[8] HIRC48 frequency lock status) “1” indicates the HIRC output frequency is accurate within 0.25% deviation.

HIRC trim can only work properly when the clock sources are stable. When the RC clock or the reference clock are not stable or the system go into power down, HIRC trim will not be enable.

6.2.10 Register Lock Control

Some of the system control registers need to be protected to avoid inadvertent write and disturb the chip operation. These write-protected system control registers, as listed in Table 6.2-9, have write-protection after the power-on reset till user disables register protection.

Before writing to these protected registers, user has to unlock the write-protected mechanism by writing a register protection disable sequence to the REGLCTL register. The register protection disable sequence is writing the data “59h”, “16h” “88h” sequentially. Any different data value, different sequence or any other write to other address during these three data writing will abort the whole sequence.

Once a register protection disable sequence is writing to the REGLCTL register successfully, These write-protected registers will be unlocked and able to accept write access. It’s recommended to locked these registers by writing any value to REGLCTL register.

6.2.10.1 Register Lock Control mechanism with Trustzone

For M2354, due to Trustzone technology, system resources are divided into secure and non-secure, leading to type of register of peripheral is either secure or non-secure. Secure registers exist in 0x4nnn_nnnn region (i.e. bit[28] is 0), while non-secure registers exist in 0x5nnn_nnnn region (i.e. bit[28] is 1).

The security types of registers are defined by which peripheral they belong to. Secure peripheral has only secure registers while non-secure peripheral has only non-secure registers. Note that shared registers in some peripherals are also defined as non-secure registers. Refer to **SCU “Memory Access Policy”** section for more details.

There are two REGLCTL registers in system. Secure REGLCTL is SYS_REGLCTL register at address 0x40000100 for secure code to unlock write-protection of both secure and non-secure registers; Non-secure REGLCTL is SYS_REGLCTLNS register at address 0x50000100, which can be seen as the non-secure alias address of SYS_REGLCTL and is used for non-secure code to unlock write-protection of non-secure registers. Note that address listed in Table 6.2-9 is the address of secure register, for non-secure register, the first nibble of the address is 0x5.

Item	Security Type	Address
SYS_IPRST0	Secure and Non-secure	0x4000_0008
SYS_BODCTL	Secure	0x4000_0018
SYS_PORCTL	Secure	0x4000_0024
SYS_VREFCTL	Secure	0x4000_0028
SYS_USBPHY	Secure	0x4000_002C
SYS_SRAMPC0	Secure	0x4000_00DC
SYS_SRAMPC1	Secure	0x4000_00E0

SYS_PORCTL1	Secure	0x4000_01EC
SYS_PSWCTL	Secure	0x4000_01F4
SYS_PLCTL	Secure	0x4000_01F8
SYS_PLSTS	Secure	0x4000_01FC
CLK_PWRCTL	Secure	0x4000_0200
CLK_APBCLK0	Secure	0x4000_0208
CLK_CLKSEL0	Secure	0x4000_0210
CLK_CLKSEL1	Secure	0x4000_0214
CLK_PLLCTL	Secure	0x4000_0240
CLK_CLKDSTS	Secure	0x4000_0274
CLK_PMUCTL	Secure	0x4000_0290
NMIEN	Secure	0x4000_0300
AHBMCTL	Secure	0x4000_0400
FMC_ISPCTL	Secure and Non-secure	0x4000_C000
FMC_ISPTRG	Secure and Non-secure	0x4000_C010
FMC_ISPSTS	Secure and Non-secure	0x4000_C040
FMC_CYCCTL	Secure	0x4000_C04C
WDT_CTL	Secure	0x4004_0000
WDT_ALTCTL	Secure	0x4004_0004
EWDT_CTL	Secure or Non-secure	0x4004_2000
EWDT_ALTCTL	Secure or Non-secure	0x4004_2004
TIMER0_CTL	Secure	0x4005_0000
TIMER1_CTL	Secure	0x4005_0100
TIMER2_CTL	Secure or Non-secure	0x4005_1000
TIMER3_CTL	Secure or Non-secure	0x4005_1100
TIMER4_CTL	Secure or Non-secure	0x4005_2000
TIMER5_CTL	Secure or Non-secure	0x4005_2100
TIMER0_PWMCTL	Secure	0x4005_0040
TIMER1_PWMCTL	Secure	0x4005_0140
TIMER2_PWMCTL	Secure or Non-secure	0x4005_1040
TIMER3_PWMCTL	Secure or Non-secure	0x4005_1140
TIMER4_PWMCTL	Secure or Non-secure	0x4005_2040
TIMER5_PWMCTL	Secure or Non-secure	0x4005_2140
TIMER0_PWMDTCTL	Secure	0x4005_0058

TIMER1_PWMDTCTL	Secure	0x4005_0158
TIMER2_PWMDTCTL	Secure or Non-secure	0x4005_1058
TIMER3_PWMDTCTL	Secure or Non-secure	0x4005_1158
TIMER0_PWMBRKCTL	Secure	0x4005_0070
TIMER1_PWMBRKCTL	Secure	0x4005_0170
TIMER2_PWMBRKCTL	Secure or Non-secure	0x4005_1070
TIMER3_PWMBRKCTL	Secure or Non-secure	0x4005_1170
TIMER0_PWMSWBRK	Secure	0x4005_007C
TIMER1_PWMSWBRK	Secure	0x4005_017C
TIMER2_PWMSWBRK	Secure or Non-secure	0x4005_107C
TIMER3_PWMSWBRK	Secure or Non-secure	0x4005_117C
TIMER0_PWMINTSTS1	Secure	0x4005_008C
TIMER1_PWMINTSTS1	Secure	0x4005_018C
TIMER2_PWMINTSTS1	Secure or Non-secure	0x4005_108C
TIMER3_PWMINTSTS1	Secure or Non-secure	0x4005_118C
EPWM_CTL0	Secure or Non-secure	0x4005_8000/0x4005_9000
EPWM_CTL1	Secure or Non-secure	0x4005_8000/0x4005_9000
EPWM_DTCTL0_1	Secure or Non-secure	0x4005_8070/0x4005_9070
EPWM_DTCTL2_3	Secure or Non-secure	0x4005_8074/0x4005_9074
EPWM_DTCTL4_5	Secure or Non-secure	0x4005_8078/0x4005_9078
EPWM_BRKCTL0_1	Secure or Non-secure	0x4005_80C8/0x4005_90C8
EPWM_BRKCTL2_3	Secure or Non-secure	0x4005_80CC/0x4005_90CC
EPWM_BRKCTL4_5	Secure or Non-secure	0x4005_80D0/0x4005_90D0
EPWM_SWBRK	Secure or Non-secure	0x4005_80DC/0x4005_90DC
EPWM_INTEN1	Secure or Non-secure	0x4005_80E4/0x4005_90E4
EPWM_INTSTS1	Secure or Non-secure	0x4005_80EC/0x4005_90EC
BPWM_CTL0	Secure or Non-secure	0x4005_A000/0x4005_B000

Table 6.2-9 List of Registers with Write Protection

6.2.11 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
SYS Base Address: **SYS_BA = 0x4000_0000** **SYS_BA_NS = 0x5000_0000**				
SYS_PDID	SYS_BA+0x00	R	Part Device Identification Number Register	0x0023_5400 [1]
SYS_RSTSTS	SYS_BA+0x04	R/W	System Reset Status Register	0x0000_0001
SYS_IPRST0	SYS_BA+0x08	R/W	Peripheral Reset Control Register 0	0x0000_0000
SYS_IPRST1	SYS_BA+0x0C	R/W	Peripheral Reset Control Register 1	0x0000_0000
SYS_IPRST2	SYS_BA+0x10	R/W	Peripheral Reset Control Register 2	0x0000_0000
SYS_BODCTL	SYS_BA+0x18	R/W	Brown-out Detector Control Register	0x008X_038X
SYS_IVSCTL	SYS_BA+0x1C	R/W	Internal Voltage Source Control Register	0x0000_0000
SYS_PORCTL0	SYS_BA+0x24	R/W	Power-on Reset Controller Register 0	0x0000_00XX
SYS_VREFCTL	SYS_BA+0x28	R/W	V_{REF} Control Register	0x0000_0200
SYS_USBPHY	SYS_BA+0x2C	R/W	USB PHY Control Register	0x0000_0007
SYS_GPA_MFPL	SYS_BA+0x30	R/W	GPIOA Low Byte Multiple Function Control Register	0x0000_0000
SYS_GPA_MFPH	SYS_BA+0x34	R/W	GPIOA High Byte Multiple Function Control Register	0x0000_0000
SYS_GPB_MFPL	SYS_BA+0x38	R/W	GPIOB Low Byte Multiple Function Control Register	0x0000_0000
SYS_GPB_MFPH	SYS_BA+0x3C	R/W	GPIOB High Byte Multiple Function Control Register	0x0000_0000
SYS_GPC_MFPL	SYS_BA+0x40	R/W	GPIOC Low Byte Multiple Function Control Register	0x0000_0000
SYS_GPC_MFPH	SYS_BA+0x44	R/W	GPIOC High Byte Multiple Function Control Register	0x0000_0000
SYS_GPD_MFPL	SYS_BA+0x48	R/W	GPIOD Low Byte Multiple Function Control Register	0x0000_0000
SYS_GPD_MFPH	SYS_BA+0x4C	R/W	GPIOD High Byte Multiple Function Control Register	0x0000_0000
SYS_GPE_MFPL	SYS_BA+0x50	R/W	GPIOE Low Byte Multiple Function Control Register	0x0000_0000
SYS_GPE_MFPH	SYS_BA+0x54	R/W	GPIOE High Byte Multiple Function Control Register	0x0000_0000
SYS_GPF_MFPL	SYS_BA+0x58	R/W	GPIOF Low Byte Multiple Function Control Register	0x0000_00EE
SYS_GPF_MFPH	SYS_BA+0x5C	R/W	GPIOF High Byte Multiple Function Control Register	0x0000_0000
SYS_GPG_MFPL	SYS_BA+0x60	R/W	GPIOG Low Byte Multiple Function Control Register	0x0000_0000
SYS_GPG_MFPH	SYS_BA+0x64	R/W	GPIOG High Byte Multiple Function Control Register	0x0000_0000

SYS_GPH_MFPL	SYS_BA+0x68	R/W	GPIOH Low Byte Multiple Function Control Register	0x0000_0000
SYS_GPH_MFPH	SYS_BA+0x6C	R/W	GPIOH High Byte Multiple Function Control Register	0x0000_0000
SYS_GPA_MFOS	SYS_BA+0x80	R/W	GPIOA Multiple Function Output Select Register	0x0000_0000
SYS_GPB_MFOS	SYS_BA+0x84	R/W	GPIOB Multiple Function Output Select Register	0x0000_0000
SYS_GPC_MFOS	SYS_BA+0x88	R/W	GPIOC Multiple Function Output Select Register	0x0000_0000
SYS_GPD_MFOS	SYS_BA+0x8C	R/W	GPIOD Multiple Function Output Select Register	0x0000_0000
SYS_GPE_MFOS	SYS_BA+0x90	R/W	GPIOE Multiple Function Output Select Register	0x0000_0000
SYS_GPF_MFOS	SYS_BA+0x94	R/W	GPIOF Multiple Function Output Select Register	0x0000_0000
SYS_GPG_MFOS	SYS_BA+0x98	R/W	GPIOG Multiple Function Output Select Register	0x0000_0000
SYS_GPH_MFOS	SYS_BA+0x9C	R/W	GPIOH Multiple Function Output Select Register	0x0000_0000
SYS_VTORSET	SYS_BA+0xA0	R/W	VTOR Setting Register	0x0000_0000
SYS_SRAMICTL	SYS_BA+0xC0	R/W	System SRAM Parity Error Interrupt Enable Control Register	0x0000_0000
SYS_SRAMSTS	SYS_BA+0xC4	R	System SRAM Parity Check Status Register	0x0000_000X
SYS_SRAMEADR	SYS_BA+0xC8	R	System SRAM Parity Error Address Register	0x0000_0000
SYS_SRAMPC0	SYS_BA+0xDC	R/W	SRAM Power Mode Control Register 0	0x0000_0000
SYS_SRAMPC1	SYS_BA+0xE0	R/W	SRAM Power Mode Control Register 1	0x0800_0000
SYS_TCTL48M	SYS_BA+0xE4	R/W	HIRC 48M Trim Control Register	0x0008_0000
SYS_TIEN48M	SYS_BA+0xE8	R/W	HIRC 48M Trim Interrupt Enable Register	0x0000_0000
SYS_TISTS48M	SYS_BA+0xEC	R/W	HIRC 48M Trim Interrupt Status Register	0x0000_0000
SYS_TCTL12M	SYS_BA+0xF0	R/W	HIRC 12M Trim Control Register	0x0008_0000
SYS_TIEN12M	SYS_BA+0xF4	R/W	HIRC 12M Trim Interrupt Enable Register	0x0000_0000
SYS_TISTS12M	SYS_BA+0xF8	R/W	HIRC 12M Trim Interrupt Status Register	0x0000_0000
SYS_REGLCTL	SYS_BA+0x100	R/W	Register Lock Control Register	0x0000_0000
SYS_REGLCTLNS	SYS_BA_NS+0x100	R/W	Register Lock Control Register for Non-secure	0x0000_0000
SYS_CPUCFG	SYS_BA+0x1D8	R/W	CPU General Configuration Register	0x0000_0000
SYS_BATLDCTL	SYS_BA+0x1DC	R/W	Battery Loss Detector Control Register	0x0000_0000
SYS_OVDCTL	SYS_BA+0x1E0	R/W	Over Voltage Detector Control Register	0x0000_0000
SYS_PORCTL1	SYS_BA+0x1EC	R/W	Power-on Reset Controller Register 1	0x0000_00XX
SYS_PSWCTL	SYS_BA+0x1F4	R/W	Power Switch Control Register	0x0000_0000
SYS_PLCTL	SYS_BA+0x1F8	R/W	Power Level Control Register	0x0000_0002

SYS_PLSTS	SYS_BA+0x1FC	R/W	Power Level Status Register	0x0000_020X

6.2.12 Register Description

Part Device Identification Number Register (SYS_PDID)

Register	Offset	R/W	Description	Reset Value
SYS_PDID	SYS_BA+0x00	R	Part Device Identification Number Register	0x0023_5400 [1]

[1] Every part number has a unique default reset value.

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

Bits	Description	
[31:0]	**PDID**	**Part Device Identification Number (Read Only)** This register reflects device part number code. Software can read this register to identify which device is used.

System Reset Status Register (SYS_RSTSTS)

This register provides specific information for software to identify this chip's reset source from last operation.

Register	Offset	R/W	Description	Reset Value
SYS_RSTSTS	SYS_BA+0x04	R/W	System Reset Status Register	0x0000_0001

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							CPULKRF
7	6	5	4	3	2	1	0
CPURF	Reserved	SYSRF	BODRF	LVRF	WDTRF	PINRF	PORF

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**CPULKRF**	**CPU Lockup Reset Flag** The CPU Lockup reset flag is set by hardware if Cortex®-M23 lockup happened. 0 = No reset from CPU lockup happened. 1 = The Cortex®-M23 lockup happened and chip is reset. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** When CPU lockup happened under ICE is connected, this flag will set to 1 but chip will not reset.
[7]	**CPURF**	**CPU Reset Flag** The CPU reset flag is set by hardware if software writes CPURST (SYS_IPRST0[1]) 1 to reset the Cortex®-M23 core and Flash Memory Controller (FMC). 0 = No reset from CPU. 1 = The Cortex®-M23 core and FMC are reset by software setting CPURST to 1. **Note:** Write 1 to clear this bit to 0.
[6]	**Reserved**	Reserved.
[5]	**SYSRF**	**System Reset Flag** The system reset flag is set by the "Reset Signal" from the Cortex®-M23 core to indicate the previous reset source. 0 = No reset from the Cortex®-M23. 1 = The Cortex®-M23 had issued the reset signal to reset the system by writing 1 to the bit SYSRESETREQ(AIRCR[2], Application Interrupt and Reset Control Register, address = 0xE000ED0C) in system control registers of Cortex®-M23 core. **Note:** Write 1 to clear this bit to 0.

Bits	Description	
[4]	**BODRF**	**BOD Reset Flag** The BOD reset flag is set by the “Reset Signal” from the Brown-out Detector to indicate the previous reset source. 0 = No reset from BOD. 1 = The BOD had issued the reset signal to reset the system. **Note:** Write 1 to clear this bit to 0.
[3]	**LVRF**	**LVR Reset Flag** The LVR reset flag is set by the “Reset Signal” from the Low Voltage Reset Controller to indicate the previous reset source. 0 = No reset from LVR. 1 = The LVR controller had issued the reset signal to reset the system. **Note:** Write 1 to clear this bit to 0.
[2]	**WDTRF**	**WDT Reset Flag** The WDT reset flag is set by the “Reset Signal” from the Watchdog Timer or Window Watchdog Timer to indicate the previous reset source. 0 = No reset from watchdog timer or window watchdog timer. 1 = The watchdog timer or window watchdog timer had issued the reset signal to reset the system. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** Watchdog Timer register RSTF(WDT_CTL[2]) bit is set if the system has been reset by WDT time-out reset. Window Watchdog Timer register WWDTRF(WWDT_STATUS[1]) bit is set if the system has been reset by WWDT time-out reset. **Note 3**: Extra Watchdog Timer register RSTF(EWDT_CTL[2]) bit is set if the system has been reset by EWDT time-out reset. Extra Window Watchdog Timer register WWDTRF(EWWDT_STATUS[1]) bit is set if the system has been reset by EWWDT time-out reset.
[1]	**PINRF**	**nRESET Pin Reset Flag** The nRESET pin reset flag is set by the “Reset Signal” from the nRESET Pin to indicate the previous reset source. 0 = No reset from nRESET pin. 1 = Pin nRESET had issued the reset signal to reset the system. **Note:** Write 1 to clear this bit to 0.
[0]	**PORF**	**POR Reset Flag** The POR reset flag is set by the “Reset Signal” from the Power-on Reset (POR) Controller or bit CHIPRST (SYS_IPRST0[0]) to indicate the previous reset source. 0 = No reset from POR or CHIPRST. 1 = Power-on Reset (POR) or CHIPRST had issued the reset signal to reset the system. **Note:** Write 1 to clear this bit to 0.

Peripheral Reset Control Register 0 (SYS_IPRST0)

Register	Offset	R/W	Description	Reset Value
SYS_IPRST0	SYS_BA+0x08	R/W	Peripheral Reset Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		PDMA1RST	Reserved				
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		KSRST	CRPTRST	Reserved			
7	6	5	4	3	2	1	0
CRCRST	SDH0RST	Reserved	USBHRST	EBIRST	PDMA0RST	CPURST	CHIPRST

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29]	**PDMA1RST**	**PDMA1 Controller Reset (Write Protect)** Setting this bit to 1 will generate a reset signal to the PDMA1. User needs to set this bit to 0 to release from reset state. 0 = PDMA1 controller normal operation. 1 = PDMA1 controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[28:14]	**Reserved**	Reserved.
[13]	**KSRST**	**Key Store Controller Reset (Write Protect)** Setting this bit to 1 will generate a reset signal to the Key Store controller. User needs to set this bit to 0 to release from the reset state. 0 = Key Store controller normal operation. 1 = Key Store controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[12]	**CRPTRST**	**CRYPTO Controller Reset (Write Protect)** Setting this bit to 1 will generate a reset signal to the CRYPTO controller. User needs to set this bit to 0 to release from the reset state. 0 = CRYPTO controller normal operation. 1 = CRYPTO controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[11:8]	**Reserved**	Reserved.
[7]	**CRCRST**	**CRC Calculation Controller Reset (Write Protect)** Set this bit to 1 will generate a reset signal to the CRC calculation controller. User needs to set this bit to 0 to release from the reset state. 0 = CRC calculation controller normal operation. 1 = CRC calculation controller reset.

		Note: This bit is write protected. Refer to the SYS_REGLCTL register.
[6]	**SDH0RST**	**SDHOST0 Controller Reset (Write Protect)** Setting this bit to 1 will generate a reset signal to the SDHOST0 controller. User needs to set this bit to 0 to release from the reset state. 0 = SDHOST0 controller normal operation. 1 = SDHOST0 controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[5]	**Reserved**	Reserved.
[4]	**USBHRST**	**USB Host Controller Reset (Write Protect)** Set this bit to 1 will generate a reset signal to the USB Host. User needs to set this bit to 0 to release from the reset state. 0 = USB Host controller normal operation. 1 = USB Host controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[3]	**EBIRST**	**EBI Controller Reset (Write Protect)** Set this bit to 1 will generate a reset signal to the EBI. User needs to set this bit to 0 to release from the reset state. 0 = EBI controller normal operation. 1 = EBI controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[2]	**PDMA0RST**	**PDMA0 Controller Reset (Write Protect)** Setting this bit to 1 will generate a reset signal to the PDMA0 (always secure). User needs to set this bit to 0 to release from reset state. 0 = PDMA0 controller normal operation. 1 = PDMA0 controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[1]	**CPURST**	**Processor Core One-shot Reset (Write Protect)** Setting this bit will only reset the processor core and Flash Memory Controller(FMC), and this bit will automatically return to 0 after the 2 clock cycles. 0 = Processor core normal operation. 1 = Processor core one-shot reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[0]	**CHIPRST**	**Chip One-shot Reset (Write Protect)** Setting this bit will reset the whole chip, including Processor core and all peripherals, and this bit will automatically return to 0 after the 2 clock cycles. The CHIPRST is same as the POR reset, all the chip controllers is reset and the chip setting from Flash are also reload. About the difference between CHIPRST and SYSRESETREQ(AIRCR[2]), please refer to section 6.2.2 0 = Chip normal operation. 1 = Chip one-shot reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

Peripheral Reset Control Register 1 (SYS_IPRST1)

Setting these bits 1 will generate asynchronous reset signals to the corresponding module controller. Users need to set these bits to 0 to release corresponding module controller from reset state.

Register	Offset	R/W	Description	Reset Value
SYS_IPRST1	SYS_BA+0x0C	R/W	Peripheral Reset Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
TRNGRST	LCDRST	I2S0RST	EADCRST	USBDRST	OTGRST	Reserved	CAN0RST
23	22	21	20	19	18	17	16
Reserved	Reserved	UART5RST	UART4RST	UART3RST	UART2RST	UART1RST	UART0RST
15	14	13	12	11	10	9	8
SPI2RST	SPI1RST	SPI0RST	QSPI0RST	Reserved	I2C2RST	I2C1RST	I2C0RST
7	6	5	4	3	2	1	0
ACMP01RST	Reserved	TMR3RST	TMR2RST	TMR1RST	TMR0RST	GPIORST	Reserved

Bits	Description	
[31]	**TRNGRST**	**TRNG Controller Reset** 0 = TRNG controller normal operation. 1 = TRNG controller reset.
[30]	**LCDRST**	**LCD Controller Reset** 0 = LCD controller normal operation. 1 = LCD controller reset.
[29]	**I2S0RST**	**I2S0 Controller Reset** 0 = I2S0 controller normal operation. 1 = I2S0 controller reset.
[28]	**EADCRST**	**EADC Controller Reset** 0 = EADC controller normal operation. 1 = EADC controller reset.
[27]	**USBDRST**	**USBD Controller Reset** 0 = USBD controller normal operation. 1 = USBD controller reset.
[26]	**OTGRST**	**OTG Controller Reset** 0 = OTG controller normal operation. 1 = OTG controller reset.
[25]	**Reserved**	Reserved.
[24]	**CAN0RST**	**CAN0 Controller Reset** 0 = CAN0 controller normal operation. 1 = CAN0 controller reset.

[23]	**Reserved**	Reserved.
[22]	**Reserved**	Reserved.
[21]	**UART5RST**	**UART5 Controller Reset** 0 = UART5 controller normal operation. 1 = UART5 controller reset.
[20]	**UART4RST**	**UART4 Controller Reset** 0 = UART4 controller normal operation. 1 = UART4 controller reset.
[19]	**UART3RST**	**UART3 Controller Reset** 0 = UART3 controller normal operation. 1 = UART3 controller reset.
[18]	**UART2RST**	**UART2 Controller Reset** 0 = UART2 controller normal operation. 1 = UART2 controller reset.
[17]	**UART1RST**	**UART1 Controller Reset** 0 = UART1 controller normal operation. 1 = UART1 controller reset.
[16]	**UART0RST**	**UART0 Controller Reset** 0 = UART0 controller normal operation. 1 = UART0 controller reset.
[15]	**SPI2RST**	**SPI2 Controller Reset** 0 = SPI2 controller normal operation. 1 = SPI2 controller reset.
[14]	**SPI1RST**	**SPI1 Controller Reset** 0 = SPI1 controller normal operation. 1 = SPI1 controller reset.
[13]	**SPI0RST**	**SPI0 Controller Reset** 0 = SPI0 controller normal operation. 1 = SPI0 controller reset.
[12]	**QSPI0RST**	**QSPI0 Controller Reset** 0 = QSPI0 controller normal operation. 1 = QSPI0 controller reset.
[11]	**Reserved**	Reserved.
[10]	**I2C2RST**	**I2C2 Controller Reset** 0 = I2C2 controller normal operation. 1 = I2C2 controller reset.
[9]	**I2C1RST**	**I2C1 Controller Reset** 0 = I2C1 controller normal operation. 1 = I2C1 controller reset.
[8]	**I2C0RST**	**I2C0 Controller Reset** 0 = I2C0 controller normal operation. 1 = I2C0 controller reset.

[7]	**ACMP01RST**	**Analog Comparator 0/1 Controller Reset** 0 = Analog Comparator 0/1 controller normal operation. 1 = Analog Comparator 0/1 controller reset.
[6]	**Reserved**	Reserved.
[5]	**TMR3RST**	**Timer3 Controller Reset** 0 = Timer3 controller normal operation. 1 = Timer3 controller reset.
[4]	**TMR2RST**	**Timer2 Controller Reset** 0 = Timer2 controller normal operation. 1 = Timer2 controller reset.
[3]	**TMR1RST**	**Timer1 Controller Reset** 0 = Timer1 controller normal operation. 1 = Timer1 controller reset.
[2]	**TMR0RST**	**Timer0 Controller Reset** 0 = Timer0 controller normal operation. 1 = Timer0 controller reset.
[1]	**GPIORST**	**GPIO Controller Reset** 0 = GPIO controller normal operation. 1 = GPIO controller reset.
[0]	**Reserved**	Reserved.

Peripheral Reset Control Register 2 (SYS_IPRST2)

Setting these bits to 1 will generate asynchronous reset signals to the corresponding module controller. Users need to set these bits to 0 to release corresponding module controller from reset state.

Register	Offset	R/W	Description	Reset Value
SYS_IPRST2	SYS_BA+0x10	R/W	Peripheral Reset Control Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				ECAP1RST	ECAP0RST	Reserved	
23	22	21	20	19	18	17	16
QEI1RST	QEI0RST	TMR5RST	TMR4RST	BPWM1RST	BPWM0RST	EPWM1RST	EPWM0RST
15	14	13	12	11	10	9	8
Reserved			DACRST	Reserved		USCI1RST	USCI0RST
7	6	5	4	3	2	1	0
Reserved	SPI3RST	Reserved			SC2RST	SC1RST	SC0RST

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27]	**ECAP1RST**	**ECAP1 Controller Reset** 0 = ECAP1 controller normal operation. 1 = ECAP1 controller reset.
[26]	**ECAP0RST**	**ECAP0 Controller Reset** 0 = ECAP0 controller normal operation. 1 = ECAP0 controller reset.
[25:24]	**Reserved**	Reserved.
[23]	**QEI1RST**	**QEI1 Controller Reset** 0 = QEI1 controller normal operation. 1 = QEI1 controller reset.
[22]	**QEI0RST**	**QEI0 Controller Reset** 0 = QEI0 controller normal operation. 1 = QEI0 controller reset.
[21]	**TMR5RST**	**Timer5 Controller Reset** 0 = Timer5 controller normal operation. 1 = Timer5 controller reset.
[20]	**TMR4RST**	**Timer4 Controller Reset** 0 = Timer4 controller normal operation. 1 = Timer4 controller reset.
[19]	**BPWM1RST**	**BPWM1 Controller Reset**

		0 = BPWM1 controller normal operation. 1 = BPWM1 controller reset.
[18]	**BPWM0RST**	**BPWM0 Controller Reset** 0 = BPWM0 controller normal operation. 1 = BPWM0 controller reset.
[17]	**EPWM1RST**	**EPWM1 Controller Reset** 0 = EPWM1 controller normal operation. 1 = EPWM1 controller reset.
[16]	**EPWM0RST**	**EPWM0 Controller Reset** 0 = EPWM0 controller normal operation. 1 = EPWM0 controller reset.
[15:13]	**Reserved**	Reserved.
[12]	**DACRST**	**DAC Controller Reset** 0 = DAC controller normal operation. 1 = DAC controller reset.
[11:10]	**Reserved**	Reserved.
[9]	**USCI1RST**	**USCI1 Controller Reset** 0 = USCI1 controller normal operation. 1 = USCI1 controller reset.
[8]	**USCI0RST**	**USCI0 Controller Reset** 0 = USCI0 controller normal operation. 1 = USCI0 controller reset.
[7]	**Reserved**	Reserved.
[6]	**SPI3RST**	**SPI3 Controller Reset** 0 = SPI3 controller normal operation. 1 = SPI3 controller reset.
[5:3]	**Reserved**	Reserved.
[2]	**SC2RST**	**SC2 Controller Reset** 0 = SC2 controller normal operation. 1 = SC2 controller reset.
[1]	**SC1RST**	**SC1 Controller Reset** 0 = SC1 controller normal operation. 1 = SC1 controller reset.
[0]	**SC0RST**	**SC0 Controller Reset** 0 = SC0 controller normal operation. 1 = SC0 controller reset.

Brown-out Detector Control Register (SYS_BODCTL)

Partial of the SYS_BODCTL control registers values are initiated by the Flash configuration and partial bits are write-protected bit.

Register	Offset	R/W	Description	Reset Value
SYS_BODCTL	SYS_BA+0x18	R/W	Brown-out Detector Control Register	0x008X_038X

31	30	29	28	27	26	25	24
WRBUSY	Reserved						
23	22	21	20	19	18	17	16
STB	Reserved				BODVL		
15	14	13	12	11	10	9	8
Reserved	LVRDGSEL			Reserved	BODDGSEL		
7	6	5	4	3	2	1	0
LVREN	BODOUT	Reserved	BODIF	BODRSTEN	Reserved		BODEN

Bits	Description	
[31]	**WRBUSY**	**Write Busy Flag (Read Only)** If SYS_BODCTL is written, this bit is asserted automatically by hardware, and is de-asserted when write procedure is finished. 0 = SYS_BODCTL register is ready for write operation. 1 = SYS_BODCTL register is busy on the last write operation. Other write operations are ignored.
[30:24]	**Reserved**	Reserved.
[23]	**STB**	**Circuit Stable Flag (Read Only)** This bit indicates LVR and BOD already stable, system cannot detect LVR and BOD event when this bit is not set.
[22:19]	**Reserved**	Reserved.
[18:16]	**BODVL**	**Brown-out Detector Threshold Voltage Selection (Write Protect)** The default value is set by Flash controller user configuration register CBOV (CONFIG0 [23:21]). 000 = Brown-out Detector threshold voltage is 1.6V. 001 = Brown-out Detector threshold voltage is 1.8V. 010 = Brown-out Detector threshold voltage is 2.0V. 011 = Brown-out Detector threshold voltage is 2.2V. 100 = Brown-out Detector threshold voltage is 2.4V. 101 = Brown-out Detector threshold voltage is 2.6V. 110 = Brown-out Detector threshold voltage is 2.8V. 111 = Brown-out Detector threshold voltage is 3.0V. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[15]	**Reserved**	Reserved.

Bits	Description	
[14:12]	**LVRDGSEL**	**LVR Output De-glitch Time Select (Write Protect)** 000 = Without de-glitch function. 001 = 4 system clock (HCLK). 010 = 8 system clock (HCLK). 011 = 16 system clock (HCLK). 100 = 32 system clock (HCLK). 101 = 64 system clock (HCLK). 110 = 128 system clock (HCLK). 111 = 256 system clock (HCLK). **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[11]	**Reserved**	Reserved.
[10:8]	**BODDGSEL**	**Brown-out Detector Output De-glitch Time Select (Write Protect)** 000 = BOD output is sampled by LIRC clock. 001 = 4 system clock (HCLK). 010 = 8 system clock (HCLK). 011 = 16 system clock (HCLK). 100 = 32 system clock (HCLK). 101 = 64 system clock (HCLK). 110 = 128 system clock (HCLK). 111 = 256 system clock (HCLK). **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[7]	**LVREN**	**Low Voltage Reset Enable Bit (Write Protect)** The LVR function resets the chip when the input power voltage is lower than LVR circuit setting. LVR function is enabled by default. 0 = Low Voltage Reset function Disabled. 1 = Low Voltage Reset function Enabled. **Note 1:** After enabling the bit, the LVR function will be active with 200us delay for LVR output stable (default). **Note 2:** This bit is write protected. Refer to the SYS_REGLCTL register.
[6]	**BODOUT**	**Brown-out Detector Output Status (Read Only)** 0 = Brown-out Detector output status is 0. It means the detected voltage is higher than BODVL setting or BODEN is 0. 1 = Brown-out Detector output status is 1. It means the detected voltage is lower than BODVL setting. If the BODEN is 0, BOD function disabled, this bit always responds 0.
[5]	**Reserved**	Reserved.
[4]	**BODIF**	**Brown-out Detector Interrupt Flag** 0 = Brown-out Detector does not detect any voltage draft at V_{DD} down through or up through the voltage of BODVL setting. 1 = When Brown-out Detector detects the V_{DD} is dropped down through the voltage of BODVL setting or the V_{DD} is raised up through the voltage of BODVL setting, this bit is set to 1 and the brown-out interrupt is requested if brown-out interrupt is enabled. **Note:** Write 1 to clear this bit to 0.

Bits	Description	
[3]	**BODRSTEN**	**Brown-out Reset Enable Bit (Write Protect)** The default value is set by Flash controller user configuration register CBORST(CONFIG0[20]) bit. 0 = Brown-out "INTERRUPT" function Enabled. 1 = Brown-out "RESET" function Enabled. **Note 1:** When the Brown-out Detector function is enabled (BODEN high) and BOD reset function is enabled (BODRSTEN high), BOD will assert a signal to reset chip when the detected voltage is lower than the threshold (BODOUT high). When the BOD function is enabled (BODEN high) and BOD interrupt function is enabled (BODRSTEN low), BOD will assert an interrupt if AV_{DD} is lower than BODVL, BOD interrupt will keep till the BODIF set to 0. BOD interrupt can be blocked by disabling the NVIC BOD interrupt or disabling BOD function (setting BODEN low). **Note 2:** This bit is write protected. Refer to the SYS_REGLCTL register.
[2:1]	**Reserved**	Reserved.
[0]	**BODEN**	**Brown-out Detector Enable Bit (Write Protect)** The default value is set by Flash controller user configuration register CBODEN (CONFIG0 [23]). 0 = Brown-out Detector function Disabled. 1 = Brown-out Detector function Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

Internal Voltage Source Control Register (SYS_IVSCTL)

Register	Offset	R/W	Description	Reset Value
SYS_IVSCTL	SYS_BA+0x1C	R/W	Internal Voltage Source Control Register	0x0000_0000

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

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**VBATUGEN**	**V_{BAT} Unity Gain Buffer Enable Bit** This bit is used to enable/disable V_{BAT} unity gain buffer function. 0 = V_{BAT} unity gain buffer function Disabled (default). 1 = V_{BAT} unity gain buffer function Enabled. **Note:** After this bit is set to 1, the value of V_{BAT} unity gain buffer output voltage can be obtained from ADC conversion result.
[0]	**VTEMPEN**	**Temperature Sensor Enable Bit** This bit is used to enable/disable temperature sensor function. 0 = Temperature sensor function Disabled (default). 1 = Temperature sensor function Enabled. **Note:** After this bit is set to 1, the value of temperature sensor output can be obtained from ADC conversion result. Please refer to ADC chapter for details.

Power-on Reset Controller Register 0 (SYS_PORCTL0)

Register	Offset	R/W	Description	Reset Value
SYS_PORCTL0	SYS_BA+0x24	R/W	Power-on Reset Controller Register 0	0x0000_00XX

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PORMASK**	**Power-on Reset Mask Enable Bit (Write Protect)** When powered on, the POR circuit generates a reset signal to reset the whole chip function, but noise on the power may cause the POR active again. User can mask internal POR signal to avoid unpredictable noise to cause chip reset by writing 0x5AA5 to this field. The POR function will be active again when this field is set to another value or chip is reset by other reset source, including: nRESET, Watchdog, LVR reset, BOD reset, ICE reset command and the software-chip reset function. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.

V_{REF} Control Register (SYS_VREFCTL)

Register	Offset	R/W	Description	Reset Value
SYS_VREFCTL	SYS_BA+0x28	R/W	V_{REF} Control Register	0x0000_0200

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:6]	**PRELOADSEL**	**Pre-load Timing Selection (Write Protect)** 00 = pre-load time is 60us for 0.1uF Capacitor. 01 = pre-load time is 310us for 1uF Capacitor. 10 = pre-load time is 2100us for 4.7uF Capacitor. 11 = pre-load time is 2850us for 10uF Capacitor. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[5]	**IBIASSEL**	**V_{REF} Bias Current Selection (Write Protect)** 0 = Bias current from MEGBIAS. 1 = Bias current from internal. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[4:0]	**VREFCTL**	**V_{REF} Control Bits (Write Protect)** 00000 = V_{REF} is from external pin. 00011 = V_{REF} is internal 1.6V. 00111 = V_{REF} is internal 2.0V. 01011 = V_{REF} is internal 2.5V. 01111 = V_{REF} is internal 3.0V Others = Reserved. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.

USB PHY Control Register (SYS_USBPHY)

Register	Offset	R/W	Description	Reset Value
SYS_USBPHY	SYS_BA+0x2C	R/W	USB PHY Control Register	0x0000_0007

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

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**OTGPHYEN**	**USB OTG PHY Enable (Write Protect)** This bit is used to enable/disable OTG PHY function. 0 = OTG PHY function Disabled (default). 1 = OTG PHY function Enabled.
[7:3]	**Reserved**	Reserved.
[2]	**SBO**	**Note: This bit must always be kept 1.** If set to 0, the result is unpredictable.
[1:0]	**USBROLE**	**USB Role Option (Write Protect)** These two bits are used to select the role of USB. 00 = Standard USB Device mode. 01 = Standard USB Host mode. 10 = ID dependent mode. 11 = On-The-Go device mode (default). **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.

GPIOA Low Byte Multiple Function Control Register (SYS_GPA_MFPL)

Register	Offset	R/W	Description	Reset Value
SYS_GPA_MFPL	SYS_BA+0x30	R/W	GPIOA Low Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PA7MFP				PA6MFP			
23	22	21	20	19	18	17	16
PA5MFP				PA4MFP			
15	14	13	12	11	10	9	8
PA3MFP				PA2MFP			
7	6	5	4	3	2	1	0
PA1MFP				PA0MFP			

Bits	Description	
[31:28]	PA7MFP	PA.7 Multi-function Pin Selection
[27:24]	PA6MFP	PA.6 Multi-function Pin Selection
[23:20]	PA5MFP	PA.5 Multi-function Pin Selection
[19:16]	PA4MFP	PA.4 Multi-function Pin Selection
[15:12]	PA3MFP	PA.3 Multi-function Pin Selection
[11:8]	PA2MFP	PA.2 Multi-function Pin Selection
[7:4]	PA1MFP	PA.1 Multi-function Pin Selection
[3:0]	PA0MFP	PA.0 Multi-function Pin Selection

GPIOA High Byte Multiple Function Control Register (SYS_GPA_MFPH).

Register	Offset	R/W	Description	Reset Value
SYS_GPA_MFPH	SYS_BA+0x34	R/W	GPIOA High Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PA15MFP				PA14MFP			
23	22	21	20	19	18	17	16
PA13MFP				PA12MFP			
15	14	13	12	11	10	9	8
PA11MFP				PA10MFP			
7	6	5	4	3	2	1	0
PA9MFP				PA8MFP			

Bits	Description	
[31:28]	PA15MFP	PA.15 Multi-function Pin Selection
[27:24]	PA14MFP	PA.14 Multi-function Pin Selection
[23:20]	PA13MFP	PA.13 Multi-function Pin Selection
[19:16]	PA12MFP	PA.12 Multi-function Pin Selection
[15:12]	PA11MFP	PA.11 Multi-function Pin Selection
[11:8]	PA10MFP	PA.10 Multi-function Pin Selection
[7:4]	PA9MFP	PA.9 Multi-function Pin Selection
[3:0]	PA8MFP	PA.8 Multi-function Pin Selection

GPIOB Low Byte Multiple Function Control Register (SYS_GPB_MFPL)

Register	Offset	R/W	Description	Reset Value
SYS_GPB_MFPL	SYS_BA+0x38	R/W	GPIOB Low Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PB7MFP				PB6MFP			
23	22	21	20	19	18	17	16
PB5MFP				PB4MFP			
15	14	13	12	11	10	9	8
PB3MFP				PB2MFP			
7	6	5	4	3	2	1	0
PB1MFP				PB0MFP			

Bits	Description	
[31:28]	PB7MFP	PB.7 Multi-function Pin Selection
[27:24]	PB6MFP	PB.6 Multi-function Pin Selection
[23:20]	PB5MFP	PB.5 Multi-function Pin Selection
[19:16]	PB4MFP	PB.4 Multi-function Pin Selection
[15:12]	PB3MFP	PB.3 Multi-function Pin Selection
[11:8]	PB2MFP	PB.2 Multi-function Pin Selection
[7:4]	PB1MFP	PB.1 Multi-function Pin Selection
[3:0]	PB0MFP	PB.0 Multi-function Pin Selection

GPIOB High Byte Multiple Function Control Register (SYS_GPB_MFPH)

Register	Offset	R/W	Description	Reset Value
SYS_GPB_MFPH	SYS_BA+0x3C	R/W	GPIOB High Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PB15MFP				PB14MFP			
23	22	21	20	19	18	17	16
PB13MFP				PB12MFP			
15	14	13	12	11	10	9	8
PB11MFP				PB10MFP			
7	6	5	4	3	2	1	0
PB9MFP				PB8MFP			

Bits	Description	
[31:28]	PB15MFP	PB.15 Multi-function Pin Selection
[27:24]	PB14MFP	PB.14 Multi-function Pin Selection
[23:20]	PB13MFP	PB.13 Multi-function Pin Selection
[19:16]	PB12MFP	PB.12 Multi-function Pin Selection
[15:12]	PB11MFP	PB.11 Multi-function Pin Selection
[11:8]	PB10MFP	PB.10 Multi-function Pin Selection
[7:4]	PB9MFP	PB.9 Multi-function Pin Selection
[3:0]	PB8MFP	PB.8 Multi-function Pin Selection

GPIOC Low Byte Multiple Function Control Register (SYS_GPC_MFPL)

Register	Offset	R/W	Description	Reset Value
SYS_GPC_MFPL	SYS_BA+0x40	R/W	GPIOC Low Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PC7MFP				PC6MFP			
23	22	21	20	19	18	17	16
PC5MFP				PC4MFP			
15	14	13	12	11	10	9	8
PC3MFP				PC2MFP			
7	6	5	4	3	2	1	0
PC1MFP				PC0MFP			

Bits	Description	
[31:28]	PC7MFP	PC.7 Multi-function Pin Selection
[27:24]	PC6MFP	PC.6 Multi-function Pin Selection
[23:20]	PC5MFP	PC.5 Multi-function Pin Selection
[19:16]	PC4MFP	PC.4 Multi-function Pin Selection
[15:12]	PC3MFP	PC.3 Multi-function Pin Selection
[11:8]	PC2MFP	PC.2 Multi-function Pin Selection
[7:4]	PC1MFP	PC.1 Multi-function Pin Selection
[3:0]	PC0MFP	PC.0 Multi-function Pin Selection

GPIOC High Byte Multiple Function Control Register (SYS_GPC_MFPH)

Register	Offset	R/W	Description	Reset Value
SYS_GPC_MFPH	SYS_BA+0x44	R/W	GPIOC High Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
PC13MFP				PC12MFP			
15	14	13	12	11	10	9	8
PC11MFP				PC10MFP			
7	6	5	4	3	2	1	0
PC9MFP				PC8MFP			

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:20]	**PC13MFP**	**PC.13 Multi-function Pin Selection**
[19:16]	**PC12MFP**	**PC.12 Multi-function Pin Selection**
[15:12]	**PC11MFP**	**PC.11 Multi-function Pin Selection**
[11:8]	**PC10MFP**	**PC.10 Multi-function Pin Selection**
[7:4]	**PC9MFP**	**PC.9 Multi-function Pin Selection**
[3:0]	**PC8MFP**	**PC.8 Multi-function Pin Selection**

GPIOD Low Byte Multiple Function Control Register (SYS_GPD_MFPL)

Register	Offset	R/W	Description	Reset Value
SYS_GPD_MFPL	SYS_BA+0x48	R/W	GPIOD Low Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PD7MFP				PD6MFP			
23	22	21	20	19	18	17	16
PD5MFP				PD4MFP			
15	14	13	12	11	10	9	8
PD3MFP				PD2MFP			
7	6	5	4	3	2	1	0
PD1MFP				PD0MFP			

Bits	Description	
[31:28]	PD7MFP	PD.7 Multi-function Pin Selection
[27:24]	PD6MFP	PD.6 Multi-function Pin Selection
[23:20]	PD5MFP	PD.5 Multi-function Pin Selection
[19:16]	PD4MFP	PD.4 Multi-function Pin Selection
[15:12]	PD3MFP	PD.3 Multi-function Pin Selection
[11:8]	PD2MFP	PD.2 Multi-function Pin Selection
[7:4]	PD1MFP	PD.1 Multi-function Pin Selection
[3:0]	PD0MFP	PD.0 Multi-function Pin Selection

GPIOD High Byte Multiple Function Control Register (SYS_GPD_MFPH)

Register	Offset	R/W	Description	Reset Value
SYS_GPD_MFPH	SYS_BA+0x4C	R/W	GPIOD High Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				PD14MFP			
23	22	21	20	19	18	17	16
Reserved				PD12MFP			
15	14	13	12	11	10	9	8
PD11MFP				PD10MFP			
7	6	5	4	3	2	1	0
PD9MFP				PD8MFP			

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:24]	**PD14MFP**	**PD.14 Multi-function Pin Selection**
[23:20]	**Reserved**	Reserved.
[19:16]	**PD12MFP**	**PD.12 Multi-function Pin Selection**
[15:12]	**PD11MFP**	**PD.11 Multi-function Pin Selection**
[11:8]	**PD10MFP**	**PD.10 Multi-function Pin Selection**
[7:4]	**PD9MFP**	**PD.9 Multi-function Pin Selection**
[3:0]	**PD8MFP**	**PD.8 Multi-function Pin Selection**

GPIOE Low Byte Multiple Function Control Register (SYS_GPE_MFPL)

Register	Offset	R/W	Description	Reset Value
SYS_GPE_MFPL	SYS_BA+0x50	R/W	GPIOE Low Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PE7MFP				PE6MFP			
23	22	21	20	19	18	17	16
PE5MFP				PE4MFP			
15	14	13	12	11	10	9	8
PE3MFP				PE2MFP			
7	6	5	4	3	2	1	0
PE1MFP				PE0MFP			

Bits	Description	
[31:28]	PE7MFP	PE.7 Multi-function Pin Selection
[27:24]	PE6MFP	PE.6 Multi-function Pin Selection
[23:20]	PE5MFP	PE.5 Multi-function Pin Selection
[19:16]	PE4MFP	PE.4 Multi-function Pin Selection
[15:12]	PE3MFP	PE.3 Multi-function Pin Selection
[11:8]	PE2MFP	PE.2 Multi-function Pin Selection
[7:4]	PE1MFP	PE.1 Multi-function Pin Selection
[3:0]	PE0MFP	PE.0 Multi-function Pin Selection

GPIOE High Byte Multiple Function Control Register (SYS_GPE_MFPH)

Register	Offset	R/W	Description	Reset Value
SYS_GPE_MFPH	SYS_BA+0x54	R/W	GPIOE High Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PE15MFP				PE14MFP			
23	22	21	20	19	18	17	16
PE13MFP				PE12MFP			
15	14	13	12	11	10	9	8
PE11MFP				PE10MFP			
7	6	5	4	3	2	1	0
PE9MFP				PE8MFP			

Bits	Description	
[31:28]	PE15MFP	PE.15 Multi-function Pin Selection
[27:24]	PE14MFP	PE.14 Multi-function Pin Selection
[23:20]	PE13MFP	PE.13 Multi-function Pin Selection
[19:16]	PE12MFP	PE.12 Multi-function Pin Selection
[15:12]	PE11MFP	PE.11 Multi-function Pin Selection
[11:8]	PE10MFP	PE.10 Multi-function Pin Selection
[7:4]	PE9MFP	PE.9 Multi-function Pin Selection
[3:0]	PE8MFP	PE.8 Multi-function Pin Selection

GPIOF Low Byte Multiple Function Control Register (SYS_GPF_MFPL)

Register	Offset	R/W	Description	Reset Value
SYS_GPF_MFPL	SYS_BA+0x58	R/W	GPIOF Low Byte Multiple Function Control Register	0x0000_00EE

31	30	29	28	27	26	25	24
PF7MFP				PF6MFP			
23	22	21	20	19	18	17	16
PF5MFP				PF4MFP			
15	14	13	12	11	10	9	8
PF3MFP				PF2MFP			
7	6	5	4	3	2	1	0
PF1MFP				PF0MFP			

Bits	Description	
[31:28]	PF7MFP	PF.7 Multi-function Pin Selection
[27:24]	PF6MFP	PF.6 Multi-function Pin Selection
[23:20]	PF5MFP	PF.5 Multi-function Pin Selection
[19:16]	PF4MFP	PF.4 Multi-function Pin Selection
[15:12]	PF3MFP	PF.3 Multi-function Pin Selection
[11:8]	PF2MFP	PF.2 Multi-function Pin Selection
[7:4]	PF1MFP	PF.1 Multi-function Pin Selection
[3:0]	PF0MFP	PF.0 Multi-function Pin Selection

GPIOF High Byte Multiple Function Control Register (SYS_GPF_MFPH)

Register	Offset	R/W	Description	Reset Value
SYS_GPF_MFPH	SYS_BA+0x5C	R/W	GPIOF High Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
PF11MFP				PF10MFP			
7	6	5	4	3	2	1	0
PF9MFP				PF8MFP			

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:12]	**PF11MFP**	**PF.11 Multi-function Pin Selection**
[11:8]	**PF10MFP**	**PF.10 Multi-function Pin Selection**
[7:4]	**PF9MFP**	**PF.9 Multi-function Pin Selection**
[3:0]	**PF8MFP**	**PF.8 Multi-function Pin Selection**

GPIOG Low Byte Multiple Function Control Register (SYS_GPG_MFPL)

Register	Offset	R/W	Description	Reset Value
SYS_GPG_MFPL	SYS_BA+0x60	R/W	GPIOG Low Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved				PG4MFP			
15	14	13	12	11	10	9	8
PG3MFP				PG2MFP			
7	6	5	4	3	2	1	0
Reserved							

Bits	Description	
[31:20]	**Reserved**	Reserved.
[19:16]	**PG4MFP**	**PG.4 Multi-function Pin Selection**
[15:12]	**PG3MFP**	**PG.3 Multi-function Pin Selection**
[11:8]	**PG2MFP**	**PG.2 Multi-function Pin Selection**
[7:0]	**Reserved**	Reserved.

GPIOG High Byte Multiple Function Control Register (SYS_GPG_MFPH)

Register	Offset	R/W	Description	Reset Value
SYS_GPG_MFPH	SYS_BA+0x64	R/W	GPIOG High Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PG15MFP				PG14MFP			
23	22	21	20	19	18	17	16
PG13MFP				PG12MFP			
15	14	13	12	11	10	9	8
PG11MFP				PG10MFP			
7	6	5	4	3	2	1	0
PG9MFP				Reserved			

Bits	Description	
[31:28]	**PG15MFP**	**PG.15 Multi-function Pin Selection**
[27:24]	**PG14MFP**	**PG.14 Multi-function Pin Selection**
[23:20]	**PG13MFP**	**PG.13 Multi-function Pin Selection**
[19:16]	**PG12MFP**	**PG.12 Multi-function Pin Selection**
[15:12]	**PG11MFP**	**PG.11 Multi-function Pin Selection**
[11:8]	**PG10MFP**	**PG.10 Multi-function Pin Selection**
[7:4]	**PG9MFP**	**PG.9 Multi-function Pin Selection**
[3:0]	**Reserved**	Reserved.

GPIOH Low Byte Multiple Function Control Register (SYS_GPH_MFPL)

Register	Offset	R/W	Description	Reset Value
SYS_GPH_MFPL	SYS_BA+0x68	R/W	GPIOH Low Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PH7MFP				PH6MFP			
23	22	21	20	19	18	17	16
PH5MFP				PH4MFP			
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							

Bits	Description	
[31:28]	**PH7MFP**	**PH.7 Multi-function Pin Selection**
[27:24]	**PH6MFP**	**PH.6 Multi-function Pin Selection**
[23:20]	**PH5MFP**	**PH.5 Multi-function Pin Selection**
[19:16]	**PH4MFP**	**PH.4 Multi-function Pin Selection**
[15:0]	**Reserved**	Reserved.

GPIOH High Byte Multiple Function Control Register (SYS_GPH_MFPH)

Register	Offset	R/W	Description	Reset Value
SYS_GPH_MFPH	SYS_BA+0x6C	R/W	GPIOH High Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
PH11MFP				PH10MFP			
7	6	5	4	3	2	1	0
PH9MFP				PH8MFP			

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:12]	**PH11MFP**	**PH.11 Multi-function Pin Selection**
[11:8]	**PH10MFP**	**PH.10 Multi-function Pin Selection**
[7:4]	**PH9MFP**	**PH.9 Multi-function Pin Selection**
[3:0]	**PH8MFP**	**PH.8 Multi-function Pin Selection**

GPIO A-H Multiple Function Output Select Register (SYS_GPx_MFOS)

Register	Offset	R/W	Description	Reset Value
SYS_GPA_MFOS	SYS_BA+0x80	R/W	GPIOA Multiple Function Output Select Register	0x0000_0000
SYS_GPB_MFOS	SYS_BA+0x84	R/W	GPIOB Multiple Function Output Select Register	0x0000_0000
SYS_GPC_MFOS	SYS_BA+0x88	R/W	GPIOC Multiple Function Output Select Register	0x0000_0000
SYS_GPD_MFOS	SYS_BA+0x8C	R/W	GPIOD Multiple Function Output Select Register	0x0000_0000
SYS_GPE_MFOS	SYS_BA+0x90	R/W	GPIOE Multiple Function Output Select Register	0x0000_0000
SYS_GPF_MFOS	SYS_BA+0x94	R/W	GPIOF Multiple Function Output Select Register	0x0000_0000
SYS_GPG_MFOS	SYS_BA+0x98	R/W	GPIOG Multiple Function Output Select Register	0x0000_0000
SYS_GPH_MFOS	SYS_BA+0x9C	R/W	GPIOH Multiple Function Output Select Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**MFOS**	**GPIOA-h Pin[n] Multiple Function Pin Output Mode Select** This bit used to select multiple function pin output mode type for Px.n pin. 0 = Multiple function pin output mode type is Push-pull mode. 1 = Multiple function pin output mode type is Open-drain mode. **Note:** Max. n=15 for port A/B/E. Max. n=13 for port C. The PC.14/ PC.15 is ignored. Max. n=14 for port D. The PD.13/ PD.15 is ignored. Max. n=12 for port F. The PF.12/ PF.13/ PF.14/ PF.15 is ignored. Max. n=15 for port G. The PG.0/ PG.1/ PG.5/ PG.6/ PG.7/ PG.8 is ignored. Max. n=11 for port H. The PH.0/ PH.1/ PH.2/ PH.3/ PH.12/ PH.13/ PH.14/ PH.15 is ignored.

VTOR Setting Register (SYS_VTORSET)

Register	Offset	R/W	Description	Reset Value
SYS_VTORSET	SYS_BA+0xA0	R/W	VTOR Setting Register	0x0000_0000

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

Bits	Description	
[31:10]	**VTORSET**	**VTOR Setting After SPD Wakeup (Write Protect)** This is the register to set the address of vector table after chip is waked up from SPD mode. The value will be loaded to Vector Table Offset Register, which is at the address 0xE000ED08, when chip wake up from SPD mode. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[9:0]	**Reserved**	Reserved.

System SRAM Parity Error Interrupt Enable Control Register (SYS_SRAMICTL)

Register	Offset	R/W	Description	Reset Value
SYS_SRAMICTL	SYS_BA+0xC0	R/W	System SRAM Parity Error Interrupt Enable Control Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**PERRIEN**	**SRAM Parity Check Error Interrupt Enable Bit** 0 = SRAM parity check error interrupt Disabled. 1 = SRAM parity check error interrupt Enabled.

System SRAM Parity Check Status Register (SYS_SRAMSTS)

Register	Offset	R/W	Description	Reset Value
SYS_SRAMSTS	SYS_BA+0xC4	R	System SRAM Parity Check Status Register	0x0000_000X

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

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	PERRIF	**SRAM Parity Check Error Flag** This bit indicates the System SRAM parity error occurred. Write 1 to clear this to 0. 0 = No System SRAM parity error. 1 = System SRAM parity error occur. **Note:** The default value depends on the content of SRAM. It is better to check this value before use it. If the default value is 1, it shall be cleared first.

System SRAM Parity Error Address Register (SYS_SRAMEADR)

Register	Offset	R/W	Description	Reset Value
SYS_SRAMEADR	SYS_BA+0xC8	R	System SRAM Parity Error Address Register	0x0000_0000

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

Bits	Description	
[31:0]	**ERRADDR**	**System SRAM Parity Error Address** This register shows system SRAM parity error byte address.

SRAM Power Mode Control Register 0 (SYS_SRAMPC0)

Register	Offset	R/W	Description	Reset Value
SYS_SRAMPC0	SYS_BA+0xDC	R/W	SRAM Power Mode Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
PCBUSY	Reserved	SRAM2PM1		SRAM2PM0		SRAM1PM7	
23	22	21	20	19	18	17	16
SRAM1PM6		SRAM1PM5		SRAM1PM4		SRAM1PM3	
15	14	13	12	11	10	9	8
SRAM1PM2		SRAM1PM1		SRAM1PM0		SRAM0PM4	
7	6	5	4	3	2	1	0
SRAM0PM3		SRAM0PM2		SRAM0PM1		SRAM0PM0	

Bits	Description	
[31]	**PCBUSY**	**Power Changing Busy Flag (Read Only)** This bit indicate SRAM power changing. 0 = SRAM power change finish. 1 = SRAM power changing.
[30]	**Reserved**	Reserved.
[29:28]	**SRAM2PM1**	**Bank2 SRAM Power Mode Select 1 (Write Protect)** This field can control bank2 sram1 (16k) power mode for range 0x2002_C000 - 0x2002_FFFF. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.
[27:26]	**SRAM2PM0**	**Bank2 SRAM Power Mode Select 0 (Write Protect)** This field can control bank2 sram0 (16k) power mode for range 0x2002_8000 - 0x2002_BFFF. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.
[25:24]	**SRAM1PM7**	**Bank1 SRAM Power Mode Select 7 (Write Protect)** This field can control bank1 sram7 (16k) power mode for range 0x2002_4000 - 0x2002_7FFF. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode.

		11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.
[23:22]	**SRAM1PM6**	**Bank1 SRAM Power Mode Select 6 (Write Protect)** This field can control bank1 sram6 (16k) power mode for range 0x2002_0000 - 0x2002_3FFF. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.
[21:20]	**SRAM1PM5**	**Bank1 SRAM Power Mode Select 5 (Write Protect)** This field can control bank1 sram5 (16k) power mode for range 0x2001_C000 - 0x2001_FFFF. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.
[19:18]	**SRAM1PM4**	**Bank1 SRAM Power Mode Select 4 (Write Protect)** This field can control bank1 sram4 (16k) power mode for range 0x2001_8000 - 0x2001_BFFF. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.
[17:16]	**SRAM1PM3**	**Bank1 SRAM Power Mode Select 3 (Write Protect)** This field can control bank1 sram3 (16k) power mode for range 0x2001_4000 - 0x2001_7FFF. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.
[15:14]	**SRAM1PM2**	**Bank1 SRAM Power Mode Select 2 (Write Protect)** This field can control bank1 sram2 (16k) power mode for range 0x2001_0000 - 0x2001_3FFF. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.
[13:12]	**SRAM1PM1**	**Bank1 SRAM Power Mode Select 1 (Write Protect)** This field can control bank1 sram1 (16k) power mode for range 0x2000_C000 - 0x2000_FFFF.

		00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.
[11:10]	**SRAM1PM0**	**Bank1 SRAM Power Mode Select 0 (Write Protect)** This field can control bank1 sram0 (16k) power mode for range 0x2000_8000 - 0x2000_BFFF. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.
[9:8]	**SRAM0PM4**	**Bank0 SRAM Power Mode Select 4 (Write Protect)** This field can control bank0 sram4 (8k) power mode for range 0x2000_6000 - 0x2000_7FFF. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.
[7:6]	**SRAM0PM3**	**Bank0 SRAM Power Mode Select 3 (Write Protect)** This field can control bank0 sram3 (8k) power mode for range 0x2000_4000 - 0x2000_5FFF. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.
[5:4]	**SRAM0PM2**	**Bank0 SRAM Power Mode Select 2 (Write Protect)** This field can control bank0 sram2 (8k) power mode for range 0x2000_2000 - 0x2000_3FFF. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1: These bits are write protected.** Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.
[3:2]	**SRAM0PM1**	**Bank0 SRAM Power Mode Select 1 (Write Protect)** This field can control bank0 sram1 (4k) power mode for range 0x2000_1000 - 0x2000_1FFF. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.

[1:0]	**SRAM0PM0**	**Bank0 SRAM Power Mode Select 0 (Write Protect)** This field can control bank0 sram0 (4k) power mode for range 0x2000_0000 - 0x2000_0FFF. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.

SRAM Power Mode Control Register 1 (SYS_SRAMPC1)

Register	Offset	R/W	Description	Reset Value
SYS_SRAMPC1	SYS_BA+0xE0	R/W	SRAM Power Mode Control Register 1	0x0800_0000

31	30	29	28	27	26	25	24
PCBUSY	Reserved	KS		RSA		FMCCACHE	
23	22	21	20	19	18	17	16
PDMA1		PDMA0		USBD		CAN	
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
SRAM2PM5		SRAM2PM4		SRAM2PM3		SRAM2PM2	

Bits	Description	
[31]	**PCBUSY**	**Power Changing Busy Flag (Read Only)** This bit indicate SRAM power changing. 0 = SRAM power change finish. 1 = SRAM power changing.
[30]	**Reserved**	Reserved.
[29:28]	**KS**	**Key Store SRAM Power Mode Select (Write Protect)** This field can control Key Store sram power mode. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.
[27:26]	**RSA**	**RSA SRAM Power Mode Select (Write Protect)** This field can control RSA sram power mode. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1. **Note 3:** If CRPTPWREN of SYS_PSWCTL is set to 1, RSA SRAM is auto set to normal mode by hardware.
[25:24]	**FMCCACHE**	**FMC SRAM Power Mode Select (Write Protect)** This field can control FMC cache sram power mode. 00 = Normal mode.

		01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.
[23:22]	**PDMA1**	**PDMA SRAM Power Mode Select (Write Protect)** This field can control PDMA1 sram power mode. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.
[21:20]	**PDMA0**	**PDMA SRAM Power Mode Select (Write Protect)** This field can control PDMA0 (always secure) sram power mode. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.
[19:18]	**USBD**	**USB Device SRAM Power Mode Select (Write Protect)** This field can control USB device sram power mode. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.
[17:16]	**CAN**	**CAN SRAM Power Mode Select (Write Protect)** This field can control CAN sram power mode. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved. **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.
[15:8]	**Reserved**	Reserved.
[7:6]	**SRAM2PM5**	**Bank2 SRAM Power Mode Select 5 (Write Protect)** This field can control bank2 sram5 (16k) power mode for range 0x2003_C000 - 0x2003_FFFF. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register.

		Note 2: Write ignore when PCBUSY is 1.
[5:4]	**SRAM2PM4**	**Bank2 SRAM Power Mode Select 4 (Write Protect)** This field can control bank2 sram4 (16k) power mode for range 0x2003_8000 - 0x2003_BFFF. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.
[3:2]	**SRAM2PM3**	**Bank2 SRAM Power Mode Select 3 (Write Protect)** This field can control bank2 sram3 (16k) power mode for range 0x2003_4000 - 0x2003_7FFF. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.
[1:0]	**SRAM2PM2**	**Bank2 SRAM Power Mode Select 2 (Write Protect)** This field can control bank2 sram2 (16k) power mode for range 0x2003_0000 - 0x2003_3FFF. 00 = Normal mode. 01 = Retention mode. 10 = Power shut down mode. 11 = Reserved (Write Ignore). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write ignore when PCBUSY is 1.

HIRC48M Trim Control Register (SYS_TCTL48M)

Register	Offset	R/W	Description	Reset Value
SYS_TCTL48M	SYS_BA+0xE4	R/W	HIRC 48M Trim Control Register	0x0008_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved			BOUNDARY				
15	14	13	12	11	10	9	8
Reserved					REFCKSEL	BOUNDEN	CESTOPEN
7	6	5	4	3	2	1	0
RETRYCNT		LOOPSEL		Reserved		FREQSEL	

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20:16]	**BOUNDARY**	**Boundary Selection** Fill the boundary range from 0x1 to 0x31, 0x0 is reserved. **Note:** This field is effective only when the BOUNDEN(SYS_TCTL48M [9]) is enable.
[15:11]	**Reserved**	Reserved.
[10]	**REFCKSEL**	**Reference Clock Selection** 0 = HIRC trim 48M reference clock is from external 32.768 kHz crystal oscillator. 1 = HIRC trim 48M reference clock is from internal USB synchronous mode.
[9]	**BOUNDEN**	**Boundary Enable Bit** 0 = Boundary function is disable. 1 = Boundary function is enable.
[8]	**CESTOPEN**	**Clock Error Stop Enable Bit** 0 = The trim operation is keep going if clock is inaccuracy. 1 = The trim operation is stopped if clock is inaccuracy.
[7:6]	**RETRYCNT**	**Trim Value Update Limitation Count** This field defines that how many times the auto trim circuit will try to update the HIRC trim value before the frequency of HIRC locked. Once the HIRC locked, the internal trim value update counter will be reset. If the trim value update counter reached this limitation value and frequency of HIRC still doesn't lock, the auto trim operation will be disabled and FREQSEL will be cleared to 00. 00 = Trim retry count limitation is 64 loops. 01 = Trim retry count limitation is 128 loops. 10 = Trim retry count limitation is 256 loops. 11 = Trim retry count limitation is 512 loops.
[5:4]	**LOOPSEL**	**Trim Calculation Loop Selection**

		This field defines that trim value calculation is based on how many reference clocks. 00 = Trim value calculation is based on average difference in 4 clocks of reference clock. 01 = Trim value calculation is based on average difference in 8 clocks of reference clock. 10 = Trim value calculation is based on average difference in 16 clocks of reference clock. 11 = Trim value calculation is based on average difference in 32 clocks of reference clock. **Note:** For example, if LOOPSEL is set as 00, auto trim circuit will calculate trim value based on the average frequency difference in 4 clocks of reference clock.
[3:2]	**Reserved**	Reserved.
[1:0]	**FREQSEL**	**Trim Frequency Selection** This field indicates the target frequency of 48 MHz internal high speed RC oscillator (HIRC) auto trim. During auto trim operation, if clock error detected with CESTOPEN is set to 1 or trim retry limitation count reached, this field will be cleared to 00 automatically. 00 = Disable HIRC auto trim function. 01 = Enable HIRC auto trim function and trim HIRC to 48 MHz. 10 = Reserved.. 11 = Reserved.

HIRC48M Trim Interrupt Enable Register (SYS_TIEN48M)

Register	Offset	R/W	Description	Reset Value
SYS_TIEN48M	SYS_BA+0xE8	R/W	HIRC 48M Trim Interrupt Enable Register	0x0000_0000

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

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**CLKEIEN**	**Clock Error Interrupt Enable Bit** This bit controls if CPU would get an interrupt while clock is inaccuracy during auto trim operation. If this bit is set to1, and CLKERRIF(SYS_TISTS48M [2]) is set during auto trim operation, an interrupt will be triggered to notify the clock frequency is inaccuracy. 0 = Disable CLKERRIF(SYS_TISTS48M [2]) status to trigger an interrupt to CPU. 1 = Enable CLKERRIF(SYS_TISTS48M [2]) status to trigger an interrupt to CPU.
[1]	**TFAILIEN**	**Trim Failure Interrupt Enable Bit** This bit controls if an interrupt will be triggered while HIRC trim value update limitation count reached and HIRC frequency still not locked on target frequency set by FREQSEL(SYS_TCTL48M[1:0]). If this bit is high and TFAILIF(SYS_TSTS48M [1]) is set during auto trim operation, an interrupt will be triggered to notify that HIRC trim value update limitation count was reached. 0 = Disable TFAILIF(SYS_TISTS48M [1]) status to trigger an interrupt to CPU. 1 = Enable TFAILIF(SYS_TISTS48M[1]) status to trigger an interrupt to CPU.
[0]	**Reserved**	Reserved.

HIRC48M Trim Interrupt Status Register (SYS_TISTS48M)

Register	Offset	R/W	Description	Reset Value
SYS_TISTS48M	SYS_BA+0xEC	R/W	HIRC 48M Trim Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				OVBDIF	CLKERRIF	TFAILIF	FREQLOCK

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**OVBDIF**	**Over Boundary Status** When the over boundary function is set, if there occurs the over boundary condition, this flag will be set. 0 = Over boundary condition did not occur. 1 = Over boundary condition occurred. **Note:** Write 1 to clear this flag.
[2]	**CLKERRIF**	**Clock Error Interrupt Status** When the frequency of 32.768 kHz external low speed crystal oscillator (LXT) or 48 MHz internal high speed RC oscillator (HIRC) is shift larger to unreasonable value, this bit will be set and to be an indicate that clock frequency is inaccuracy Once this bit is set to 1, the auto trim operation stopped and FREQSEL(SYS_TCTL48M[1:0]) will be cleared to 00 by hardware automatically if CESTOPEN(SYS_TCTL48M[8]) is set to 1. If this bit is set and CLKEIEN(SYS_TIEN48M[2]) is high, an interrupt will be triggered to notify the clock frequency is inaccuracy. Write 1 to clear this to 0. 0 = Clock frequency is accurate. 1 = Clock frequency is inaccurate.
[1]	**TFAILIF**	**Trim Failure Interrupt Status** This bit indicates that HIRC trim value update limitation count reached and the HIRC clock frequency is still not locked. Once this bit is set, the auto trim operation stopped and FREQSEL(SYS_TCTL48M[1:0]) will be cleared to 00 by hardware automatically. If this bit is set and TFAILIEN(SYS_TIEN48M[1]) is high, an interrupt will be triggered to notify that HIRC trim value update limitation count was reached. Write 1 to clear this to 0. 0 = Trim value update limitation count does not reach. 1 = Trim value update limitation count reached and HIRC frequency still not locked.
[0]	**FREQLOCK**	**HIRC Frequency Lock Status** This bit indicates the HIRC frequency is locked. This is a status bit and doesn't trigger any interrupt. Write 1 to clear this to 0. This bit will be set automatically, if the frequency is lock and the RC_TRIM is

		enabled. 0 = The internal high-speed oscillator frequency doesn't lock at 48 MHz yet. 1 = The internal high-speed oscillator frequency locked at 48 MHz.

HIRC12M Trim Control Register (SYS_TCTL12M)

Register	Offset	R/W	Description	Reset Value
SYS_TCTL12M	SYS_BA+0xF0	R/W	HIRC 12M Trim Control Register	0x0008_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved			BOUNDARY				
15	14	13	12	11	10	9	8
Reserved					REFCKSEL	BOUNDEN	CESTOPEN
7	6	5	4	3	2	1	0
RETRYCNT		LOOPSEL		Reserved		FREQSEL	

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20:12]	**BOUNDARY**	**Boundary Selection** Fill the boundary range from 0x1 to 0x31, 0x0 is reserved. **Note:** This field is effective only when the BOUNDEN(SYS_TCTL12M[9]) is enabled.
[11]	**Reserved**	Reserved.
[10]	**REFCKSEL**	**Reference Clock Selection** 0 = HIRC trim reference clock is from external 32.768 kHz crystal oscillator. 1 = HIRC trim reference clock is from internal USB synchronous mode.
[9]	**BOUNDEN**	**Boundary Enable Bit** 0 = Boundary function Disabled. 1 = Boundary function Enabled.
[8]	**CESTOPEN**	**Clock Error Stop Enable Bit** 0 = The trim operation keeps going if clock is inaccurate. 1 = The trim operation is stopped if clock is inaccurate.
[7:6]	**RETRYCNT**	**Trim Value Update Limitation Count** This field defines that how many times the auto trim circuit will try to update the HIRC trim value before the frequency of HIRC locked. Once the HIRC locked, the internal trim value update counter will be reset. If the trim value update counter reached this limitation value and frequency of HIRC still doesn't lock, the auto trim operation will be disabled and FREQSEL will be cleared to 00. 00 = Trim retry count limitation is 64 loops. 01 = Trim retry count limitation is 128 loops. 10 = Trim retry count limitation is 256 loops. 11 = Trim retry count limitation is 512 loops.
[5:4]	**LOOPSEL**	**Trim Calculation Loop Selection**

		This field defines that trim value calculation is based on how many reference clocks. 00 = Trim value calculation is based on average difference in 4 clocks of reference clock. 01 = Trim value calculation is based on average difference in 8 clocks of reference clock. 10 = Trim value calculation is based on average difference in 16 clocks of reference clock. 11 = Trim value calculation is based on average difference in 32 clocks of reference clock. **Note:** For example, if LOOPSEL is set as 00, auto trim circuit will calculate trim value based on the average frequency difference in 4 clocks of reference clock.
[3:2]	**Reserved**	Reserved.
[1:0]	**FREQSEL**	**Trim Frequency Selection** This field indicates the target frequency of 12 MHz internal high speed RC oscillator (HIRC) auto trim. During auto trim operation, if clock error detected with CESTOPEN is set to 1 or trim retry limitation count reached, this field will be cleared to 00 automatically. 00 = Disable HIRC auto trim function. 01 = Enable HIRC auto trim function and trim HIRC to 12 MHz. 10 = Reserved.. 11 = Reserved.

HIRC12M Trim Interrupt Enable Register (SYS_TIEN12M)

Register	Offset	R/W	Description	Reset Value
SYS_TIEN12M	SYS_BA+0xF4	R/W	HIRC 12M Trim Interrupt Enable Register	0x0000_0000

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

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**CLKEIEN**	**Clock Error Interrupt Enable Bit** This bit controls if CPU would get an interrupt while clock is inaccuracy during auto trim operation. If this bit is set to1, and CLKERRIF(SYS_TISTS12M[2]) is set during auto trim operation, an interrupt will be triggered to notify the clock frequency is inaccuracy. 0 = Disable CLKERRIF(SYS_TISTS12M[2]) status to trigger an interrupt to CPU. 1 = Enable CLKERRIF(SYS_TISTS12M[2]) status to trigger an interrupt to CPU.
[1]	**TFAILIEN**	**Trim Failure Interrupt Enable Bit** This bit controls if an interrupt will be triggered while HIRC trim value update limitation count reached and HIRC frequency still not locked on target frequency set by FREQSEL(SYS_TCTL12M[1:0]). If this bit is high and TFAILIF(SYS_TSTS12M[1]) is set during auto trim operation, an interrupt will be triggered to notify that HIRC trim value update limitation count was reached. 0 = Disable TFAILIF(SYS_TISTS12M[1]) status to trigger an interrupt to CPU. 1 = Enable TFAILIF(SYS_TISTS12M[1]) status to trigger an interrupt to CPU.
[0]	**Reserved**	Reserved.

HIRC12M Trim Interrupt Status Register (SYS_TISTS12M)

Register	Offset	R/W	Description	Reset Value
SYS_TISTS12M	SYS_BA+0xF8	R/W	HIRC 12M Trim Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				OVBDIF	CLKERRIF	TFAILIF	FREQLOCK

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**OVBDIF**	**Over Boundary Status** When the over boundary function is set, if there occurs the over boundary condition, this flag will be set. 0 = Over boundary condition did not occur. 1 = Over boundary condition occurred. **Note:** Write 1 to clear this flag.
[2]	**CLKERRIF**	**Clock Error Interrupt Status** When the frequency of 32.768 kHz external low speed crystal oscillator (LXT) or 12 MHz internal high speed RC oscillator (HIRC) is shift larger to unreasonable value, this bit will be set and to be an indicate that clock frequency is inaccuracy Once this bit is set to 1, the auto trim operation stopped and FREQSEL(SYS_TCTL12M[1:0]) will be cleared to 00 by hardware automatically if CESTOPEN(SYS_TCTL12M[8]) is set to 1. If this bit is set and CLKEIEN(SYS_IRCTIEN[2]) is high, an interrupt will be triggered to notify the clock frequency is inaccuracy. Write 1 to clear this to 0. 0 = Clock frequency is accurate. 1 = Clock frequency is inaccurate.
[1]	**TFAILIF**	**Trim Failure Interrupt Status** This bit indicates that HIRC trim value update limitation count reached and the HIRC clock frequency still doesn't be locked. Once this bit is set, the auto trim operation stopped and FREQSEL(SYS_TCTL12M[1:0]) will be cleared to 00 by hardware automatically. If this bit is set and TFAILIEN(SYS_TIEN12M[1]) is high, an interrupt will be triggered to notify that HIRC trim value update limitation count was reached. Write 1 to clear this to 0. 0 = Trim value update limitation count does not reach. 1 = Trim value update limitation count reached and HIRC frequency still not locked.
[0]	**FREQLOCK**	**HIRC Frequency Lock Status** This bit indicates the HIRC frequency is locked. This is a status bit and doesn't trigger any interrupt. Write 1 to clear this to 0. This bit will be set automatically, if the frequency is lock and the RC_TRIM is

		enabled. 0 = The internal high-speed oscillator frequency is not locked at 12 MHz yet. 1 = The internal high-speed oscillator frequency locked at 12 MHz.

Register Lock Control Register (SYS_REGLCTL)

Register	Offset	R/W	Description	Reset Value
SYS_REGLCTL	SYS_BA+0x100	R/W	Register Lock Control Register	0x0000_0000

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**REGLCTL**	**Register Lock Control Code** Some registers have write-protection function. Writing these registers have to disable the protected function by writing the sequence value “59h”, “16h”, “88h” to this field. After this sequence is completed, the REGLCTL bit will be set to 1 and write-protection registers can be normal write. **REGLCTL[0]** **Register Lock Control Disable Index** 0 = Write-protection Enabled for writing protected registers. Any write to the protected register is ignored. 1 = Write-protection Disabled for writing protected registers.

Register Lock Control Register for Non-secure (SYS_REGLCTLNS)

Register	Offset	R/W	Description	Reset Value
SYS_REGLCTLNS	SYS_BA_NS+0x100	R/W	Register Lock Control Register for Non-secure	0x0000_0000

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

Bits	Description	
[31:8]	Reserved	Reserved.
[7:0]	REGLCTL	**Register Lock Control Code** Some registers have write-protection function. Writing these registers have to disable the protected function by writing the sequence value “59h”, “16h”, “88h” to this field. After this sequence is completed, the REGLCTL bit will be set to 1 and write-protection registers can be normal write. **REGLCTL[0]** **Register Lock Control Disable Index** 0 = Write-protection Enabled for writing protected registers. Any write to the protected register is ignored. 1 = Write-protection Disabled for writing protected registers.

CPU General Configuration Register (SYS_CPUCFG)

Register	Offset	R/W	Description	Reset Value
SYS_CPUCFG	SYS_BA+0x1D8	R/W	CPU General Configuration Register	0x0000_0000

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

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	INTRTEN	**CPU Interrupt Realtime Enable Bit** When this bit is 0, the latency of CPU entering interrupt service routine (ISR) will be various but shorter. When this bit is 1, the latency of CPU entering ISR will be kept constant. 0 = CPU Interrupt Realtime Disabled. 1 = CPU Interrupt Realtime Enabled.

Battery Loss Detector Control Register (SYS_BATLDCTL)

Register	Offset	R/W	Description	Reset Value
SYS_BATLDCTL	SYS_BA+0x1DC	R/W	Battery Loss Detector Control Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**BATLDEN**	**Battery Loss Detector Enable Bit (Write Protect)** 0 = Battery Loss Detector Disabled. 1 = Battery Loss Detector Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

Over Voltage Detector Control Register (SYS_OVPCTL)

Register	Offset	R/W	Description	Reset Value
SYS_OVDCTL	SYS_BA+0x1E0	R/W	Over Voltage Detector Control Register	0x0000_0000

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

Bits	Description	
[31]	**OVDSTB**	**Over Voltage Detector Stable Flag (Read Only)** 0 = Over voltage detector not stable. 1 = Over voltage detector stable.
[30:1]	**Reserved**	Reserved.
[0]	**OVDEN**	**Over Voltage Detector Enable Bit (Write Protect)** 0 = Over voltage detector Disabled. 1 = Over voltage detector Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

Power-on Reset Controller Register 1 (SYS_PORCTL1)

Register	Offset	R/W	Description	Reset Value
SYS_PORCTL1	SYS_BA+0x1EC	R/W	Power-on Reset Controller Register 1	0x0000_00XX

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**POROFF**	**Power-on Reset Enable Bit (Write Protect)** When powered on, the POR circuit generates a reset signal to reset the whole chip function, but noise on the power may cause the POR active again. User can disable internal POR circuit to avoid unpredictable noise to cause chip reset by writing 0x5AA5 to this field. The POR function will be active again when this field is set to another value or chip is reset by other reset source, including: nRESET, Watchdog, LVR reset, BOD reset, ICE reset command and the software-chip reset function. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.

Power Switch Control Register (SYS_PSWCTL)

Register	Offset	R/W	Description	Reset Value
SYS_PSWCTL	SYS_BA+0x1F4	R/W	Power Switch Control Register	0x0000_0000

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

Bits	Description	
[31:13]	**Reserved**	Reserved.
[12]	**CRPTPWREN**	**Cryptographic Accelerator Power Switch Enable Bit (Write Protect)** 0 = Cryptographic accelerator power supply Disabled. 1 = Cryptographic accelerator power supply Enabled. **Note 1:** If this bit is set 1, RSA of SYS_SRAMPC1 is set to normal mode by hardware. **Note 2:** Write ignored when PCBUSY(SYS_SRAMPC1[31]) is 1. **Note 3:** This bit is write protected. Refer to the SYS_REGLCTL register.
[11:0]	**Reserved**	Reserved.

Power Level Control Register (SYS_PLCTL)

Register	Offset	R/W	Description	Reset Value
SYS_PLCTL	SYS_BA+0x1F8	R/W	Power Level Control Register	0x0000_0002

31	30	29	28	27	26	25	24
LVSPRD							
23	22	21	20	19	18	17	16
Reserved		LVSSTEP					
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
WRBUSY	Reserved		MVRS	Reserved		PLSEL	

Bits	Description	
[31:24]	**LVSPRD**	**LDO Voltage Scaling Period (Write Protect)** The LVSPRD value is the period of each LDO voltage rising step. LDO voltage scaling period = (LVSPRD + 1) * 1us. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[23:22]	**Reserved**	Reserved.
[21:16]	**LVSSTEP**	**LDO Voltage Scaling Step (Write Protect)** The LVSSTEP value is LDO voltage rising step. LDO voltage scaling step = (LVSSTEP + 1) * 10mV. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[15:8]	**Reserved**	Reserved.
[7]	**WRBUSY**	**Write Busy Flag (Read Only)** If SYS_PLCTL be written, this bit be asserted automatic by hardware, and be de-asserted when write procedure finish. 0 = SYS_PLCTL register is ready for write operation. 1 = SYS_PLCTL register is busy on the last write operation. Other write operations are ignored.
[6:5]	**Reserved**	Reserved.
[4]	**MVRS**	**Main Voltage Regulator Type Select (Write Protect)** This bit field sets main voltage regulator type. After setting main voltage regulator type to DCDC (MVRS (SYS_PLCTL[4]) = 1), system will set main voltage regulator type change busy flag MVRCBUSY(SYS_PLSTS[1]), detect inductor connection and update inductor connection status LCONS (SYS_PLSTS[3]). if inductor exist LCONS will be cleared and main voltage regulator type can switch to DCDC (CURMVR (SYS_PLSTS[12])=1). 0 = Set main voltage regulator to LDO. 1 = Set main voltage regulator to DCDC. **Note 1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note 2:** This bit not be reset when wake-up from Standby Power-down mode(SPD).

[3:2]	**Reserved**	Reserved.
[1:0]	**PLSEL**	**Power Level Select (Write Protect)** 00 = Set to Power level 0 (PL0). 01 = Set to Power level 1 (PL1). 10 = Set to Power level 2 (PL2). (default) 11 = Set to Power level 3 (PL3). **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** These bits not be reset when wake-up from Standby Power-down mode (SPD).

Power Level Status Register (SYS_PLSTS)

Register	Offset	R/W	Description	Reset Value
SYS_PLSTS	SYS_BA+0x1FC	R/W	Power Level Status Register	0x0000_020X

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved			CURMVR	Reserved		PLSTATUS	
7	6	5	4	3	2	1	0
Reserved				LCONS	MVRCERR	MVRCBUSY	PLCBUSY

Bits	Description	
[31:13]	**Reserved**	Reserved.
[12]	**CURMVR**	**Current Main Voltage Regulator Type (Read Only)** This bit field reflects current main voltage regulator type. 0 = Current main voltage regulator in active and Idle mode is LDO. 1 = Current main voltage regulator in active and Idle mode is DCDC.
[11:10]	**Reserved**	Reserved.
[9:8]	**PLSTATUS**	**Power Level Status (Read Only)** This bit field reflect the current power level. 00 = Power level is PL0. 01 = Power level is PL1. 10 = Power level is PL2. 11 = Power level is PL3.
[7:4]	**Reserved**	Reserved.
[3]	**LCONS**	**Inductor for DCDC Connect Status (Read Only)** This bit is valid when current main voltage regulator type is DCDC (CURMVR (SYS_PLSTS[12])=1). If current main voltage regulator type is LDO (CURMVR (SYS_PLSTS[12])=0) this bit is set to 1. 0 = Inductor connect between Vsw and LDO_CAP pin. 1 = No Inductor connect between Vsw and LDO_CAP pin. **Note:** This bit is 1 when main voltage regulator is LDO.
[2]	**MVRCERR**	**Main Voltage Regulator Type Change Error Bit (Write Protect)** This bit is set to 1 when main voltage regulator type change from LDO to DCDC error, the following conditions will cause change errors: ▪ System changed to DC-DC mode but LDO change voltage process not finish. ▪ Detect inductor failed. 0 = No main voltage regulator type change error.

		1 = Main voltage regulator type change to DCDC error occurred. **Note 1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note 2:** Write 1 to clear this bit to 0.
[1]	**MVRCBUSY**	**Main Voltage Regulator Type Change Busy Bit (Read Only)** This bit is set by hardware when main voltage regulator type is changing. After main voltage regulator type change is completed, this bit will be cleared automatically by hardware. 0 = Main voltage regulator type change is completed. 1 = Main voltage regulator type change is ongoing.
[0]	**PLCBUSY**	**Power Level Change Busy Bit (Read Only)** This bit is set by hardware when power level is changing. After power level change is completed, this bit will be cleared automatically by hardware. 0 = Power level change is completed. 1 = Power level change is ongoing.

6.2.13 System Timer (SysTick)

The Cortex®-M23 includes an integrated system timer, SysTick, which provides a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism. The counter can be used as a Real Time Operating System (RTOS) tick timer or as a simple counter.

When system timer is enabled, it will count down from the value in the SysTick Current Value Register (SYST_VAL) to zero, and reload (wrap) to the value in the SysTick Reload Value Register (SYST_LOAD) on the next clock cycle, and then decrement on subsequent clocks. When the counter transitions to zero, the COUNTFLAG status bit is set. The COUNTFLAG bit clears on reads.

The SYST_VAL value is UNKNOWN on reset. Software should write to the register to clear it to zero before enabling the feature. This ensures the timer will count from the SYST_LOAD value rather than an arbitrary value when it is enabled.

If the SYST_LOAD is zero, the timer will be maintained with a current value of zero after it is reloaded with this value. This mechanism can be used to disable the feature independently from the timer enable bit.

For more detailed information, please refer to the "*Arm®Cortex®-M23 Technical Reference Manual*" and "*Arm®v8-M Architecture Reference Manual*".

6.2.13.1 System Timer Control Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
SYST Base Address: **SCS_BA = 0xE000_E000**				
SYST_CTRL	SCS_BA+0x10	R/W	SysTick Control and Status Register	0x0000_0000
SYST_LOAD	SCS_BA+0x14	R/W	SysTick Reload Value Register	0xXXXX_XXXX
SYST_VAL	SCS_BA+0x18	R/W	SysTick Current Value Register	0xXXXX_XXXX

6.2.13.2 System Timer Control Register Description

SysTick Control and Status Register (SYST_CTRL)

Register	Offset	R/W	Description	Reset Value
SYST_CTRL	SCS_BA+0x10	R/W	SysTick Control and Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							COUNTFLAG
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					CLKSRC	TICKINT	ENABLE

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**COUNTFLAG**	**System Tick Counter Flag** Returns 1 if timer counted to 0 since last time this register was read. COUNTFLAG is set by a count transition from 1 to 0. COUNTFLAG is cleared on read or by a write to the Current Value register.
[15:3]	**Reserved**	Reserved.
[2]	**CLKSRC**	**System Tick Clock Source Selection** 0 = Clock source is the (optional) external reference clock. 1 = Core clock used for SysTick.
[1]	**TICKINT**	**System Tick Interrupt Enabled** 0 = Counting down to 0 does not cause the SysTick exception to be pended. Software can use COUNTFLAG to determine if a count to zero has occurred. 1 = Counting down to 0 will cause the SysTick exception to be pended. Clearing the SysTick current value register by a register write in software will not cause SysTick to be pended.
[0]	**ENABLE**	**System Tick Counter Enabled** 0 = Counter Disabled. 1 = Counter will operate in a multi-shot manner.

SysTick Reload Value Register (SYST_LOAD)

Register	Offset	R/W	Description	Reset Value
SYST_LOAD	SCS_BA+0x14	R/W	SysTick Reload Value Register	0xXXXX_XXXX

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

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**RELOAD**	**System Tick Reload Value** Value to load into the Current Value register when the counter reaches 0.

SysTick Current Value Register (SYST_VAL)

Register	Offset	R/W	Description	Reset Value
SYST_VAL	SCS_BA+0x18	R/W	SysTick Current Value Register	0xXXXX_XXXX

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

Bits	Description	
[31:24]	Reserved	Reserved.
[23:0]	CURRENT	**System Tick Current Value** Current counter value. This is the value of the counter at the time it is sampled. The counter does not provide read-modify-write protection. The register is write-clear. A software write of any value will clear the register to 0. Unsupported bits RAZ.

6.2.14 Nested Vectored Interrupt Controller (NVIC)

The NVIC and the processor core interface are closely coupled to enable low latency interrupt processing and efficient processing of late arriving interrupts. The NVIC maintains knowledge of the stacked, or nested, interrupts to enable tail-chaining of interrupts. You can only fully access the NVIC from privileged mode, Any other user mode access causes a bus fault. You can access all NVIC registers using byte, halfword, and word accesses unless otherwise stated. NVIC registers are located within the SCS (System Control Space). All NVIC registers and system debug registers are little-endian regardless of the endianness state of the processor.

The NVIC supports:

- An implementation-defined number of interrupts, in the range 1-240 interrupts.
- A programmable priority level of 0-3 for each interrupt; a higher level corresponds to a lower priority, so level 0 is the highest interrupt priority.
- Level and pulse detection of interrupt signals.
- Dynamic reprioritization of interrupts.
- Grouping of priority values into group priority.
- Interrupt tail-chaining.
- An external Non maskable Interrupt (NMI)
- WIC with Ultra-low Power Sleep mode support

The processor automatically stacks its state on exception entry and unstacks this state on exception exit, with no instruction overhead. This provides low latency exception handling.

6.2.14.1 Exception Model and System Interrupt Map

Table 6.2-10 lists the exception model supported by the M2354 series. Software can set 4 levels of priority on some of these exceptions as well as on all interrupts. The highest user-configurable priority is denoted as “0x00” and the lowest priority is denoted as “0xC0” (The 4-LSB always 0). The default priority of all the user-configurable interrupts is “0x00”. Note that priority “0” is treated as the fourth priority on the system, after three system exceptions “Reset”, “NMI” and “Hard Fault”.

When any interrupts is accepted, the processor will automatically fetch the starting address of the interrupt service routine (ISR) from a vector table in memory. On system reset, the vector table is fixed at address 0x00000000. Privileged software can write to the VTOR to relocate the vector table start address to a different memory location, in the range 0x00000080 to 0x3FFFFF80,

The vector table contains the initialization value for the stack pointer on reset, and the entry point addresses for all exception handlers. The vector number on previous page defines the order of entries in the vector table associated with exception handler entry as illustrated in previous section.

Exception Type	Vector Number	Vector Address	Priority
Reset	1	0x00000004	-4
NMI	2	0x00000008	-2
Secure Hard Fault (when AIRCR.BFHFNMINS is 1)	3	0x0000000C	-3
Non-secure Hard Fault (when AIRCR.BFHFNMINS is 0)	3	0x0000000C	-1

Reserved	4 ~ 10		Reserved
SVCall	11	0x0000002C	Configurable
Reserved	12 ~ 13		Reserved
PendSV	14	0x00000038	Configurable
SysTick	15	0x0000003C	Configurable
Interrupt (IRQ0 ~ IRQ)	16 ~ 111	0x00000000 + (Vector Number)*4	Configurable

Table 6.2-10 Exception Model

Vector Number	Interrupt Number (Bit In Interrupt Registers)	Interrupt Name	Interrupt Description
0 ~ 15	-	-	System exceptions
16	0	BODOUT	Brown-out low voltage detected interrupt
17	1	IRC_INT	IRC TRIM interrupt
18	2	PWRWU_INT	Clock controller interrupt for chip wake-up from power-down state
19	3	SRAM_PERR	SRAM parity check error interrupt
20	4	CLKFAIL	Clock fail detected interrupt
21	5	ISP_INT	FMC ISP interrupt
22	6	RTC_INT	Real time clock interrupt
23	7	RTC_TAMPER_INT	Backup register tamper interrupt
24	8	WDT_INT	Watchdog Timer interrupt
25	9	WWDT_INT	Window Watchdog Timer interrupt
26	10	EINT0	External interrupt from PA.6 or PB.5 pins
27	11	EINT1	External interrupt from PA.7 or PB.4 pins
28	12	EINT2	External interrupt from PB.3 or PC.6 pins
29	13	EINT3	External interrupt from PB.2 or PC.7 pins
30	14	EINT4	External interrupt from PA.8 or PB.6 pins
31	15	EINT5	External interrupt from PB.7 or PD.12 pins
32	16	GPA_INT	External interrupt from PA[15:0] pin
33	17	GPB_INT	External interrupt from PB[15:0] pin
34	18	GPC_INT	External interrupt from PC[15:0] pin
35	19	GPD_INT	External interrupt from PD[15:0] pin
36	20	GPE_INT	External interrupt from PE[15:0] pin

37	21	GPF_INT	External interrupt from PF[15:0] pin
38	22	QSPI0_INT	QSPI0 interrupt
39	23	SPI0_INT	SPI0 interrupt
40	24	BRAKE0_INT	EPWM0 brake interrupt
41	25	EPWM0_P0_INT	EPWM0 pair 0 interrupt
42	26	EPWM0_P1_INT	EPWM0 pair 1 interrupt
43	27	EPWM0_P2_INT	EPWM0 pair 2 interrupt
44	28	BRAKE1_INT	EPWM1 brake interrupt
45	29	EPWM1_P0_INT	EPWM1 pair 0 interrupt
46	30	EPWM1_P1_INT	EPWM1 pair 1 interrupt
47	31	EPWM1_P2_INT	EPWM1 pair 2 interrupt
48	32	TMR0_INT	Timer 0 interrupt
49	33	TMR1_INT	Timer 1 interrupt
50	34	TMR2_INT	Timer 2 interrupt
51	35	TMR3_INT	Timer 3 interrupt
52	36	UART0_INT	UART0 interrupt
53	37	UART1_INT	UART1 interrupt
54	38	I2C0_INT	I2C0 interrupt
55	39	I2C1_INT	I2C1 interrupt
56	40	PDMA0_INT	PDMA0 interrupt
57	41	DAC_INT	DAC interrupt
58	42	EADC0_INT	EADC interrupt source 0
59	43	EADC1_INT	EADC interrupt source 1
60	44	ACMP01_INT	ACMP0 and ACMP1 interrupt
61	45	Reserved	Reserved
62	46	EADC2_INT	EADC interrupt source 2
63	47	EADC3_INT	EADC interrupt source 3
64	48	UART2_INT	UART2 interrupt
65	49	UART3_INT	UART3 interrupt
66	50	Reserved	Reserved
67	51	SPI1_INT	SPI1 interrupt
68	52	SPI2_INT	SPI2 interrupt
69	53	USBD_INT	USB device interrupt
70	54	USBH_INT	USB host interrupt

71	55	USBOTG_INT	USB OTG interrupt
72	56	CAN0_INT	CAN0 interrupt
73	57	Reserved	Reserved
74	58	SC0_INT	Smart card host 0 interrupt
75	59	SC1_INT	Smart card host 1 interrupt
76	60	SC2_INT	Smart card host 2 interrupt
77	61	Reserved	Reserved
78	62	SPI3_INT	SPI3 interrupt
79	63	Reserved	Reserved
80	64	SDHOST0_INT	SD host 0 interrupt
81	65	Reserved	Reserved
82	66	Reserved	Reserved
83	67	Reserved	Reserved
84	68	I2S0_INT	I2S0 interrupt
85	69	Reserved	Reserved
86	70	Reserved	Reserved
87	71	CRYPTO	Crypto interrupt
88	72	GPG_INT	External interrupt from PG[15:0] pin
89	73	EINT6	External interrupt from PB.8 or PD.11 pins
90	74	UART4_INT	UART4 interrupt
91	75	UART5_INT	UART5 interrupt
92	76	USCI0_INT	USCI0 interrupt
93	77	USCI1_INT	USCI1 interrupt
94	78	BPWM0_INT	BPWM0 interrupt
95	79	BPWM1_INT	BPWM1 interrupt
96	80	Reserved	Reserved
97	81	Reserved	Reserved
98	82	I2C2_INT	I2C2 interrupt
99	83	Reserved	Reserved
100	84	QEI0_INT	QEI0 interrupt
101	85	QEI1_INT	QEI1 interrupt
102	86	ECAP0_INT	ECAP0 interrupt
103	87	ECAP1_INT	ECAP1 interrupt
104	88	GPH_INT	External interrupt from PH[15:0] pin

105	89	EINT7	External interrupt from PB.9 or PD.10 pins
106	90	Reserved	Reserved
107	91	Reserved	Reserved
108	92	Reserved	Reserved
109	93	Reserved	Reserved
110	94	Reserved	Reserved
111	95	Reserved	Reserved
112	96	Reserved	Reserved
113	97	Reserved	Reserved
114	98	PDMA1_INT	PDMA 1 interrupt
115	99	SCU_INT	SCU interrupt
116	100	LCD_INT	LCD interrupt
117	101	TRNG_INT	TRNG interrupt
1118	102	Reserved	Reserved
1119	103	Reserved	Reserved
120	104	Reserved	Reserved
121	105	Reserved	Reserved
122	106	Reserved	Reserved
123	107	Reserved	Reserved
124	108	Reserved	Reserved
125	109	KS_INT	Key store interrupt
126	110	TAMPER_INT	Tamper interrupt
127	111	EWDT_INT	Extra Watchdog Timer interrupt
128	112	EWWDT_INT	Extra Window Watchdog Timer interrupt
129	113	NS_ISP_INT	Non secure FMC ISP interrupt
130	114	TMR4_INT	Timer 4 interrupt
131	115	TMR5_INT	Timer 5 interrupt

Table 6.2-11 Interrupt Number Table

6.2.14.2 Operation Description

NVIC interrupts can be enabled and disabled by writing to their corresponding Interrupt Set-Enable or Interrupt Clear-Enable register bit-field. The registers use a write-1-to-enable and write-1-to-clear policy, both registers reading back the current enabled state of the corresponding interrupts. When an interrupt is disabled, interrupt assertion will cause the interrupt to become Pending, however, the interrupt will not activate. If an interrupt is Active when it is disabled, it remains in its Active state until cleared by reset or an exception return. Clearing the enable bit prevents new activations of the

associated interrupt.

NVIC interrupts can be pended/un-pended using a complementary pair of registers to those used to enable/disable the interrupts, named the Set-Pending Register and Clear-Pending Register respectively. The registers use a write-1-to-enable and write-1-to-clear policy, both registers reading back the current pended state of the corresponding interrupts. The Clear-Pending Register has no effect on the execution status of an Active interrupt.

NVIC interrupts are prioritized by updating an 8-bit field within a 32-bit register (each register supporting four interrupts).

The general registers associated with the NVIC are all accessible from a block of memory in the System Control Space and will be described in the next section.

6.2.14.3 NVIC Control Registers

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
NVIC Base Address: **NVIC_BA = 0xE000_E100**				
NVIC_ISER0	NVIC_BA+0x000	R/W	IRQ00 ~ IRQ31 Set-enable Control Register	0x0000_0000
NVIC_ISER1	NVIC_BA+0x004	R/W	IRQ32 ~ IRQ63 Set-enable Control Register	0x0000_0000
NVIC_ISER2	NVIC_BA+0x008	R/W	IRQ64 ~ IRQ95 Set-enable Control Register	0x0000_0000
NVIC_ISER3	NVIC_BA+0x00C	R/W	IRQ96 ~ IRQ115 Set-enable Control Register	0x0000_0000
NVIC_ICER0	NVIC_BA+0x080	R/W	IRQ00 ~ IRQ31 Clear-enable Control Register	0x0000_0000
NVIC_ICER1	NVIC_BA+0x084	R/W	IRQ32 ~ IRQ63 Clear-enable Control Register	0x0000_0000
NVIC_ICER2	NVIC_BA+0x088	R/W	IRQ64 ~ IRQ95 Clear-enable Control Register	0x0000_0000
NVIC_ICER3	NVIC_BA+0x08C	R/W	IRQ96 ~ IRQ115 Clear-enable Control Register	0x0000_0000
NVIC_ISPR0	NVIC_BA+0x100	R/W	IRQ00 ~ IRQ31 Set-pending Control Register	0x0000_0000
NVIC_ISPR1	NVIC_BA+0x104	R/W	IRQ32 ~ IRQ63 Set-pending Control Register	0x0000_0000
NVIC_ISPR2	NVIC_BA+0x108	R/W	IRQ64 ~ IRQ95 Set-pending Control Register	0x0000_0000
NVIC_ISPR3	NVIC_BA+0x10C	R/W	IRQ96 ~ IRQ115 Set-pending Control Register	0x0000_0000
NVIC_ICPR0	NVIC_BA+0x180	R/W	IRQ00 ~ IRQ31 Clear-pending Control Register	0x0000_0000
NVIC_ICPR1	NVIC_BA+0x184	R/W	IRQ32 ~ IRQ63 Clear-pending Control Register	0x0000_0000
NVIC_ICPR2	NVIC_BA+0x188	R/W	IRQ64 ~ IRQ95 Clear-pending Control Register	0x0000_0000
NVIC_ICPR3	NVIC_BA+0x18C	R/W	IRQ96 ~ IRQ115 Clear-pending Control Register	0x0000_0000
NVIC_IABR0	NVIC_BA+0x200	R/W	IRQ00 ~ IRQ31 Active Bit Register	0x0000_0000
NVIC_IABR1	NVIC_BA+0x204	R/W	IRQ32 ~ IRQ63 Active Bit Register	0x0000_0000
NVIC_IABR2	NVIC_BA+0x208	R/W	IRQ64 ~ IRQ95 Active Bit Register	0x0000_0000
NVIC_IABR3	NVIC_BA+0x20C	R/W	IRQ96 ~ IRQ115 Active Bit Register	0x0000_0000
NVIC_ITNS0	NVIC_BA+0x280	R/W	IRQ00 ~ IRQ31 Interrupt Target Non-secure Register	0x0000_0000
NVIC_ITNS1	NVIC_BA+0x284	R/W	IRQ32 ~ IRQ63 Interrupt Target Non-secure Register	0x0000_0000
NVIC_ITNS2	NVIC_BA+0x288	R/W	IRQ64 ~ IRQ95 Interrupt Target Non-secure Register	0x0000_0000
NVIC_ITNS3	NVIC_BA+0x28C	R/W	IRQ96 ~ IRQ115 Interrupt Target Non-secure Register	0x0000_0000
NVIC_IPRn **n=0,1..28**	NVIC_BA+0x300 +0x4*n	R/W	IRQ0 ~ IRQ115 Priority Control Register	0x0000_0000

IRQ00 ~ IRQ31 Set-enable Control Register (NVIC_ISER0)

Register	Offset	R/W	Description	Reset Value
NVIC_ISER0	NVIC_BA+0x000	R/W	IRQ00 ~ IRQ31 Set-enable Control Register	0x0000_0000

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

Bits	Description	
[31:0]	**SETENA**	**Interrupt Set Enable Bit** The NVIC_ISER0-NVIC_ISER3 registers enable interrupts, and show which interrupts are enabled. Write Operation: 0 = No effect. 1 = Interrupt Enabled. Read Operation: 0 = Interrupt Disabled. 1 = Interrupt Enabled.

IRQ32 ~ IRQ63 Set-enable Control Register (NVIC_ISER1)

Register	Offset	R/W	Description	Reset Value
NVIC_ISER1	NVIC_BA+0x004	R/W	IRQ32 ~ IRQ63 Set-enable Control Register	0x0000_0000

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

Bits	Description	
[31:0]	**SETENA**	**Interrupt Set Enable Bit** The NVIC_ISER0-NVIC_ISER3 registers enable interrupts, and show which interrupts are enabled. Write Operation: 0 = No effect. 1 = Interrupt Enabled. Read Operation: 0 = Interrupt Disabled. 1 = Interrupt Enabled.

IRQ64 ~ IRQ95 Set-enable Control Register (NVIC_ISER2)

Register	Offset	R/W	Description	Reset Value
NVIC_ISER2	NVIC_BA+0x008	R/W	IRQ64 ~ IRQ95 Set-enable Control Register	0x0000_0000

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

Bits	Description	
[31:0]	**SETENA**	**Interrupt Set Enable Bit** The NVIC_ISER0-NVIC_ISER3 registers enable interrupts, and show which interrupts are enabled. Write Operation: 0 = No effect. 1 = Interrupt Enabled. Read Operation: 0 = Interrupt Disabled. 1 = Interrupt Enabled.

IRQ96 ~ IRQ115 Set-enable Control Register (NVIC_ISER3)

Register	Offset	R/W	Description	Reset Value
NVIC_ISER3	NVIC_BA+0x00C	R/W	IRQ96 ~ IRQ115 Set-enable Control Register	0x0000_0000

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

Bits	Description	
[31:0]	**SETENA**	**Interrupt Set Enable Bit** The NVIC_ISER0-NVIC_ISER3 registers enable interrupts, and show which interrupts are enabled. Write Operation: 0 = No effect. 1 = Interrupt Enabled. Read Operation: 0 = Interrupt Disabled. 1 = Interrupt Enabled.

IRQ00 ~ IRQ31 Clear-enable Control Register (NVIC_ICER0)

Register	Offset	R/W	Description	Reset Value
NVIC_ICER0	NVIC_BA+0x080	R/W	IRQ00 ~ IRQ31 Clear-enable Control Register	0x0000_0000

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

Bits	Description	
[31:0]	**CALENA**	**Interrupt Clear Enable Bit** The NVIC_ICER0-NVIC_ICER3 registers disable interrupts, and show which interrupts are enabled. Write Operation: 0 = No effect. 1 = Interrupt Disabled. Read Operation: 0 = Interrupt Disabled. 1 = Interrupt Enabled.

IRQ32 ~ IRQ63 Clear-enable Control Register (NVIC_ICER1)

Register	Offset	R/W	Description	Reset Value
NVIC_ICER1	NVIC_BA+0x084	R/W	IRQ32 ~ IRQ63 Clear-enable Control Register	0x0000_0000

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

Bits	Description	
[31:0]	**CALENA**	**Interrupt Clear Enable Bit** The NVIC_ICER0-NVIC_ICER3 registers disable interrupts, and show which interrupts are enabled. Write Operation: 0 = No effect. 1 = Interrupt Disabled. Read Operation: 0 = Interrupt Disabled. 1 = Interrupt Enabled.

IRQ64 ~ IRQ95 Clear-enable Control Register (NVIC_ICER2)

Register	Offset	R/W	Description	Reset Value
NVIC_ICER2	NVIC_BA+0x088	R/W	IRQ64 ~ IRQ95 Clear-enable Control Register	0x0000_0000

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

Bits	Description	
[31:0]	CALENA	**Interrupt Clear Enable Bit** The NVIC_ICER0-NVIC_ICER3 registers disable interrupts, and show which interrupts are enabled. Write Operation: 0 = No effect. 1 = Interrupt Disabled. Read Operation: 0 = Interrupt Disabled. 1 = Interrupt Enabled.

IRQ96 ~ IRQ115 Clear-enable Control Register (NVIC_ICER3)

Register	Offset	R/W	Description	Reset Value
NVIC_ICER3	NVIC_BA+0x08C	R/W	IRQ96 ~ IRQ115 Clear-enable Control Register	0x0000_0000

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

Bits	Description	
[31:0]	**CALENA**	**Interrupt Clear Enable Bit** The NVIC_ICER0-NVIC_ICER3 registers disable interrupts, and show which interrupts are enabled. Write Operation: 0 = No effect. 1 = Interrupt Disabled. Read Operation: 0 = Interrupt Disabled. 1 = Interrupt Enabled.

IRQ00 ~ IRQ31 Set-pending Control Register (NVIC_ISPR0)

Register	Offset	R/W	Description	Reset Value
NVIC_ISPR0	NVIC_BA+0x100	R/W	IRQ00 ~ IRQ31 Set-pending Control Register	0x0000_0000

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

Bits	Description	
[31:0]	**SETPEND**	**Interrupt Set-pending** The NVIC_ISPR0-NVIC_ISPR3 registers force interrupts into the pending state, and show which interrupts are pending. Write Operation: 0 = No effect. 1 = Changes interrupt state to pending. Read Operation: 0 = Interrupt is not pending. 1 = Interrupt is pending.

IRQ32 ~ IRQ63 Set-pending Control Register (NVIC_ISPR1)

Register	Offset	R/W	Description	Reset Value
NVIC_ISPR1	NVIC_BA+0x104	R/W	IRQ32 ~ IRQ63 Set-pending Control Register	0x0000_0000

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

Bits	Description	
[31:0]	**SETPEND**	**Interrupt Set-pending** The NVIC_ISPR0-NVIC_ISPR3 registers force interrupts into the pending state, and show which interrupts are pending. Write Operation: 0 = No effect. 1 = Changes interrupt state to pending. Read Operation: 0 = Interrupt is not pending. 1 = Interrupt is pending.

IRQ64 ~ IRQ95 Set-pending Control Register (NVIC_ISPR2)

Register	Offset	R/W	Description	Reset Value
NVIC_ISPR2	NVIC_BA+0x108	R/W	IRQ64 ~ IRQ95 Set-pending Control Register	0x0000_0000

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

Bits	Description	
[31:0]	**SETPEND**	**Interrupt Set-pending** The NVIC_ISPR0-NVIC_ISPR3 registers force interrupts into the pending state, and show which interrupts are pending. Write Operation: 0 = No effect. 1 = Changes interrupt state to pending. Read Operation: 0 = Interrupt is not pending. 1 = Interrupt is pending.

IRQ96 ~ IRQ115 Set-pending Control Register (NVIC_ISPR3)

Register	Offset	R/W	Description	Reset Value
NVIC_ISPR3	NVIC_BA+0x10C	R/W	IRQ96 ~ IRQ115 Set-pending Control Register	0x0000_0000

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

Bits	Description	
[31:0]	**SETPEND**	**Interrupt Set-pending** The NVIC_ISPR0-NVIC_ISPR3 registers force interrupts into the pending state, and show which interrupts are pending. Write Operation: 0 = No effect. 1 = Changes interrupt state to pending. Read Operation: 0 = Interrupt is not pending. 1 = Interrupt is pending.

IRQ00 ~ IRQ31 Clear-pending Control Register (NVIC_ICPR0)

Register	Offset	R/W	Description	Reset Value
NVIC_ICPR0	NVIC_BA+0x180	R/W	IRQ00 ~ IRQ31 Clear-pending Control Register	0x0000_0000

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

Bits	Description	
[31:0]	**CALPEND**	**Interrupt Clear-pending** The NVIC_ICPR0-NVIC_ICPR3 registers remove the pending state from interrupts, and show which interrupts are pending. Write Operation: 0 = No effect. 1 = Removes pending state an interrupt. Read Operation: 0 = Interrupt is not pending. 1 = Interrupt is pending.

IRQ32 ~ IRQ63 Clear-pending Control Register (NVIC_ICPR1)

Register	Offset	R/W	Description	Reset Value
NVIC_ICPR1	NVIC_BA+0x184	R/W	IRQ32 ~ IRQ63 Clear-pending Control Register	0x0000_0000

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

Bits	Description	
[31:0]	**CALPEND**	**Interrupt Clear-pending** The NVIC_ICPR0-NVIC_ICPR3 registers remove the pending state from interrupts, and show which interrupts are pending. Write Operation: 0 = No effect. 1 = Removes pending state an interrupt. Read Operation: 0 = Interrupt is not pending. 1 = Interrupt is pending.

IRQ64 ~ IRQ95 Clear-pending Control Register (NVIC_ICPR2)

Register	Offset	R/W	Description	Reset Value
NVIC_ICPR2	NVIC_BA+0x188	R/W	IRQ64 ~ IRQ95 Clear-pending Control Register	0x0000_0000

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

Bits	Description	
[31:0]	**CALPEND**	**Interrupt Clear-pending** The NVIC_ICPR0-NVIC_ICPR3 registers remove the pending state from interrupts, and show which interrupts are pending. Write Operation: 0 = No effect. 1 = Remove pending state an interrupt. Read Operation: 0 = Interrupt is not pending. 1 = Interrupt is pending.

IRQ96 ~ IRQ115 Clear-pending Control Register (NVIC_ICPR3)

Register	Offset	R/W	Description	Reset Value
NVIC_ICPR3	NVIC_BA+0x18C	R/W	IRQ96 ~ IRQ115 Clear-pending Control Register	0x0000_0000

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

Bits	Description	
[31:0]	**CALPEND**	**Interrupt Clear-pending** The NVIC_ICPR0-NVIC_ICPR3 registers remove the pending state from interrupts, and show which interrupts are pending. Write Operation: 0 = No effect. 1 = Remove pending state an interrupt. Read Operation: 0 = Interrupt is not pending. 1 = Interrupt is pending.

IRQ00 ~ IRQ31 Active Bit Register (NVIC_IABR0)

Register	Offset	R/W	Description	Reset Value
NVIC_IABR0	NVIC_BA+0x200	R/W	IRQ00 ~ IRQ31 Active Bit Register	0x0000_0000

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

Bits	Description	
[31:0]	**ACTIVE**	**Interrupt Active Flags** The NVIC_IABR0-NVIC_IABR3 registers indicate which interrupts are active. 0 = Interrupt not active. 1 = Interrupt active.

IRQ32 ~ IRQ63 Active Bit Register (NVIC_IABR1)

Register	Offset	R/W	Description	Reset Value
NVIC_IABR1	NVIC_BA+0x204	R/W	IRQ32 ~ IRQ63 Active Bit Register	0x0000_0000

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

Bits	Description	
[31:0]	**ACTIVE**	**Interrupt Active Flags** The NVIC_IABR0-NVIC_IABR3 registers indicate which interrupts are active. 0 = interrupt not active. 1 = interrupt active.

IRQ64 ~ IRQ95 Active Bit Register (NVIC_IABR2)

Register	Offset	R/W	Description	Reset Value
NVIC_IABR2	NVIC_BA+0x208	R/W	IRQ64 ~ IRQ95 Active Bit Register	0x0000_0000

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

Bits	Description	
[31:0]	**ACTIVE**	**Interrupt Active Flags** The NVIC_IABR0-NVIC_IABR3 registers indicate which interrupts are active. 0 = interrupt not active. 1 = interrupt active.

IRQ96 ~ IRQ115 Active Bit Register (NVIC_IABR3)

Register	Offset	R/W	Description	Reset Value
NVIC_IABR3	NVIC_BA+0x20C	R/W	IRQ96 ~ IRQ115 Active Bit Register	0x0000_0000

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

Bits	Description	
[31:0]	**ACTIVE**	**Interrupt Active Flags** The NVIC_IABR0-NVIC_IABR3 registers indicate which interrupts are active. 0 = interrupt not active. 1 = interrupt active.

IRQ00 ~ IRQ31 Interrupt Target Non-secure Register (NVIC_ITNS0)

Register	Offset	R/W	Description	Reset Value
NVIC_ITNS0	NVIC_BA+0x280	R/W	IRQ00 ~ IRQ31 Interrupt Target Non-secure Register	0x0000_0000

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

Bits	Description	
[31:0]	**ITNS**	**Interrupt Target Non-secure Register** The NVIC_ITNS0-NVIC_INTS3 registers, determines whether each interrupt targets Non-secure or Secure state. 0 = Interrupt targets Secure state. 1 = Interrupt targets Non-secure state. This register is RAZ/WI when accessed as Non-secure.

IRQ32 ~ IRQ63 Interrupt Target Non-secure Register (NVIC_ITNS1)

Register	Offset	R/W	Description	Reset Value
NVIC_ITNS1	NVIC_BA+0x284	R/W	IRQ32 ~ IRQ63 Interrupt Target Non-secure Register	0x0000_0000

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

Bits	Description	
[31:0]	**ITNS**	**Interrupt Target Non-secure Register** The NVIC_ITNS0-NVIC_INTS3 registers, determines whether each interrupt targets Non-secure or Secure state. 0 = Interrupt targets Secure state. 1 = Interrupt targets Non-secure state. **Note:** This register is RAZ/WI when accessed as Non-secure.

IRQ64 ~ IRQ95 Interrupt Target Non-secure Register (NVIC_ITNS2)

Register	Offset	R/W	Description	Reset Value
NVIC_ITNS2	NVIC_BA+0x288	R/W	IRQ64 ~ IRQ95 Interrupt Target Non-secure Register	0x0000_0000

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

Bits	Description	
[31:0]	ITNS	**Interrupt Target Non-secure Register** The NVIC_ITNS0-NVIC_INTS3 registers, determines whether each interrupt targets Non-secure or Secure state. 0 = Interrupt targets Secure state. 1 = Interrupt targets Non-secure state. **Note:** This register is RAZ/WI when accessed as Non-secure.

IRQ96 ~ IRQ115 Interrupt Target Non-secure Register (NVIC_ITNS3)

Register	Offset	R/W	Description	Reset Value
NVIC_ITNS3	NVIC_BA+0x28C	R/W	IRQ96 ~ IRQ115 Interrupt Target Non-secure Register	0x0000_0000

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

Bits	Description	
[31:0]	ITNS	**Interrupt Target Non-secure Register** The NVIC_ITNS0-NVIC_INTS3 registers, determines whether each interrupt targets Non-secure or Secure state. 0 = Interrupt targets Secure state. 1 = Interrupt targets Non-secure state. **Note:** This register is RAZ/WI when accessed as Non-secure.

IRQ0 ~ IRQ115 Interrupt Priority Register (NVIC_IPRn)

Register	Offset	R/W	Description	Reset Value
NVIC_IPRn **n=0,1..28**	NVIC_BA+0x300 +0x4*n	R/W	IRQ0 ~ IRQ115 Priority Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PRI_4n_3		**Reserved**					
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PRI_4n_2		**Reserved**					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PRI_4n_1		**Reserved**					
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PRI_4n_0		**Reserved**					

Bits	Description	
[31:30]	**PRI_4n_3**	**Priority of IRQ_4n+3** "0" denotes the highest priority and "3" denotes the lowest priority.
[29:24]	**Reserved**	Reserved.
[23:22]	**PRI_4n_2**	**Priority of IRQ_4n+2** "0" denotes the highest priority and "3" denotes the lowest priority.
[21:16]	**Reserved**	Reserved.
[15:14]	**PRI_4n_1**	**Priority of IRQ_4n+1** "0" denotes the highest priority and "3" denotes the lowest priority.
[13:8]	**Reserved**	Reserved.
[7:6]	**PRI_4n_0**	**Priority of IRQ_4n+0** "0" denotes the highest priority and "3" denotes the lowest priority.
[5:0]	**Reserved**	Reserved.

Priority Value PRI_4n_X[7:6] X=0,1,2,3	Secure Priority	Non-Secure Priority When PRIS=0	Non-Secure Priority When PRIS=1
0x0	0	0	128
0x1	64	64	160
0x2	128	128	192
0x3	192	192	224

Table 6.2-12 Priority Grouping

6.2.14.4 NMI Control Registers

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
NMI Base Address: **NMI_BA = 0x4000_0300**				
NMIEN	NMI_BA+0x00	R/W	NMI Source Interrupt Enable Register	0x0000_0000
NMISTS	NMI_BA+0x04	R	NMI Source Interrupt Status Register	0x0000_0000

NMI Source Interrupt Enable Register (NMIEN)

Register	Offset	R/W	Description	Reset Value
NMIEN	NMI_BA+0x00	R/W	NMI Source Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved						**EINT7**	**EINT6**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
UART1INT	**UART0INT**	**EINT5**	**EINT4**	**EINT3**	**EINT2**	**EINT1**	**EINT0**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TAMPERINT	**RTCINT**	**Reserved**	**CLKFAIL**	**SRAMPERR**	**PWRWUINT**	**IRCINT**	**BODOUT**

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**EINT7**	**External Interrupt From PB.9 or PD.10 Pin NMI Source Enable (Write Protect)** 0 = External interrupt from PB.9 or PD.10 pin NMI source Disabled. 1 = External interrupt from PB.9 or PD.10 pin NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[16]	**EINT6**	**External Interrupt From PB.8 or PD.11 Pin NMI Source Enable (Write Protect)** 0 = External interrupt from PB.8 or PD.11 pin NMI source Disabled. 1 = External interrupt from PB.8 or PD.11 pin NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[15]	**UART1INT**	**UART1 NMI Source Enable (Write Protect)** 0 = UART1 NMI source Disabled. 1 = UART1 NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[14]	**UART0INT**	**UART0 NMI Source Enable (Write Protect)** 0 = UART0 NMI source Disabled. 1 = UART0 NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[13]	**EINT5**	**External Interrupt From PB.7 or PD.12 Pin NMI Source Enable (Write Protect)** 0 = External interrupt from PB.7 or PD.12 pin NMI source Disabled. 1 = External interrupt from PB.7 or PD.12 pin NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[12]	**EINT4**	**External Interrupt From PA.8 or PB.6 Pin NMI Source Enable (Write Protect)** 0 = External interrupt from PA.8 or PB.6 pin NMI source Disabled. 1 = External interrupt from PA.8 or PB.6 pin NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

[11]	**EINT3**	**External Interrupt From PB.2 or PC.7 Pin NMI Source Enable (Write Protect)** 0 = External interrupt from PB.2 or PC.7pin NMI source Disabled. 1 = External interrupt from PB.2 or PC.7 pin NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[10]	**EINT2**	**External Interrupt From PB.3 or PC.6 Pin NMI Source Enable (Write Protect)** 0 = External interrupt from PB.3 or PC.6 pin NMI source Disabled. 1 = External interrupt from PB.3 or PC.6 pin NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[9]	**EINT1**	**External Interrupt From PA.7 or PB.4 Pin NMI Source Enable (Write Protect)** 0 = External interrupt from PA.7 or PB.4 pin NMI source Disabled. 1 = External interrupt from PA.7 or P4.4 pin NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[8]	**EINT0**	**External Interrupt From PA.6, or PB.5 Pin NMI Source Enable (Write Protect)** 0 = External interrupt from PA.6, or PB.5 pin NMI source Disabled. 1 = External interrupt from PA.6, or PB.5 pin NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[7]	**TAMPERINT**	**Tamper Interrupt NMI Source Enable (Write Protect)** 0 = Backup register tamper detected interrupt NMI source Disabled. 1 = Backup register tamper detected interrupt NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[6]	**RTCINT**	**RTC NMI Source Enable (Write Protect)** 0 = RTC NMI source Disabled. 1 = RTC NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[5]	**Reserved**	Reserved.
[4]	**CLKFAIL**	**Clock Fail Detected NMI Source Enable (Write Protect)** 0 = Clock fail detected interrupt NMI source Disabled. 1 = Clock fail detected interrupt NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[3]	**SRAMPERR**	**SRAM Parity Check Error NMI Source Enable (Write Protect)** 0 = SRAM parity check error NMI source Disabled. 1 = SRAM parity check error NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[2]	**PWRWUINT**	**Power-down Mode Wake-up NMI Source Enable (Write Protect)** 0 = Power-down mode wake-up NMI source Disabled. 1 = Power-down mode wake-up NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[1]	**IRCINT**	**IRC TRIM NMI Source Enable (Write Protect)** 0 = IRC TRIM NMI source Disabled. 1 = IRC TRIM NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[0]	**BODOUT**	**BOD NMI Source Enable (Write Protect)** 0 = BOD NMI source Disabled.

		1 = BOD NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

NMI Source Interrupt Status Register (NMISTS)

Register	Offset	R/W	Description	Reset Value
NMISTS	NMI_BA+0x04	R	NMI Source Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						EINT7	EINT6
15	14	13	12	11	10	9	8
UART1INT	UART0INT	EINT5	EINT4	EINT3	EINT2	EINT1	EINT0
7	6	5	4	3	2	1	0
TAMPERINT	RTCINT	Reserved	CLKFAIL	SRAMPERR	PWRWUINT	IRCINT	BODOUT

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**EINT7**	**External Interrupt From PB.9 or PD.10 Pin Interrupt Flag (Read Only)** 0 = External Interrupt from PB.9 or PD.10 interrupt is deasserted. 1 = External Interrupt from PB.9 or PD.10 interrupt is asserted.
[16]	**EINT6**	**External Interrupt From PB.8 or PD.11 Pin Interrupt Flag (Read Only)** 0 = External Interrupt from PB.8 or PD.11 interrupt is deasserted. 1 = External Interrupt from PB.8 or PD.11 interrupt is asserted.
[15]	**UART1INT**	**UART1 Interrupt Flag (Read Only)** 0 = UART1 interrupt is deasserted. 1 = UART1 interrupt is asserted.
[14]	**UART0INT**	**UART0 Interrupt Flag (Read Only)** 0 = UART1 interrupt is deasserted. 1 = UART1 interrupt is asserted.
[13]	**EINT5**	**External Interrupt From PB.7 or PD.12 Pin Interrupt Flag (Read Only)** 0 = External Interrupt from PB.7 or PD.12 interrupt is deasserted. 1 = External Interrupt from PB.7 or PD.12 interrupt is asserted.
[12]	**EINT4**	**External Interrupt From PA.8 or PB.6 Pin Interrupt Flag (Read Only)** 0 = External Interrupt from PA.8 or PB.6 interrupt is deasserted. 1 = External Interrupt from PA.8 or PB.6 interrupt is asserted.
[11]	**EINT3**	**External Interrupt From PB.2 or PC.7 Pin Interrupt Flag (Read Only)** 0 = External Interrupt from PB.2 or PC.7 interrupt is deasserted. 1 = External Interrupt from PB.2 or PC.7 interrupt is asserted.
[10]	**EINT2**	**External Interrupt From PB.3 or PC.6 Pin Interrupt Flag (Read Only)** 0 = External Interrupt from PB.3 or PC.6 interrupt is deasserted.

		1 = External Interrupt from PB.3 or PC.6 interrupt is asserted.
[9]	**EINT1**	**External Interrupt From PA.7, or PB.4 Pin Interrupt Flag (Read Only)** 0 = External Interrupt from PA.7, or PB.4 interrupt is deasserted. 1 = External Interrupt from PA.7, or PB.4 interrupt is asserted.
[8]	**EINT0**	**External Interrupt From PA.6, or PB.5 Pin Interrupt Flag (Read Only)** 0 = External Interrupt from PA.6, or PB.5 interrupt is deasserted. 1 = External Interrupt from PA.6, or PB.5 interrupt is asserted.
[7]	**TAMPERINT**	**Tamper Interrupt Flag (Read Only)** 0 = Backup register tamper detected interrupt is deasserted. 1 = Backup register tamper detected interrupt is asserted.
[6]	**RTCINT**	**RTC Interrupt Flag (Read Only)** 0 = RTC interrupt is deasserted. 1 = RTC interrupt is asserted.
[5]	**Reserved**	Reserved.
[4]	**CLKFAIL**	**Clock Fail Detected Interrupt Flag (Read Only)** 0 = Clock fail detected interrupt is deasserted. 1 = Clock fail detected interrupt is asserted.
[3]	**SRAMPERR**	**SRAM Parity Check Error Interrupt Flag (Read Only)** 0 = SRAM parity check error interrupt is deasserted. 1 = SRAM parity check error interrupt is asserted.
[2]	**PWRWUINT**	**Power-down Mode Wake-up Interrupt Flag (Read Only)** 0 = Power-down mode wake-up interrupt is deasserted. 1 = Power-down mode wake-up interrupt is asserted.
[1]	**IRCINT**	**IRC TRIM Interrupt Flag (Read Only)** 0 = HIRC TRIM interrupt is deasserted. 1 = HIRC TRIM interrupt is asserted.
[0]	**BODOUT**	**BOD Interrupt Flag (Read Only)** 0 = BOD interrupt is deasserted. 1 = BOD interrupt is asserted.

6.2.14.5 AHB Bus Matrix Priority Control Register

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
AHB Base Address: **AHB_BA = 0x4000_0400**				
AHBMCTL	0x40000400	R/W	AHB Bus Matrix Priority Control Register	0x0000_0001

AHB Bus Matrix Priority Control Register (AHBMCTL)

Register	Offset	R/W	Description	Reset Value
AHBMCTL	0x40000400	R/W	AHB Bus Matrix Priority Control Register	0x0000_0001

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**INTACTEN**	**Highest AHB Bus Priority of Cortex®M23 Core Enable Bit (Write Protect)** Enable Cortex®-M23 core with highest AHB bus priority in AHB bus matrix. 0 = Round robin mode. 1 = Cortex®-M23 CPU with highest bus priority when interrupt occur. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

6.2.15 System Control Register

The Cortex®-M23 status and operation mode control are managed by System Control Registers. Including CPUID, Cortex®-M23 interrupt priority and Cortex®-M23 power management can be controlled through these system control registers.

For more detailed information, please refer to the "Arm® Cortex®-M23 Technical Reference Manual" and "Arm® v8-M Architecture Reference Manual".

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
SCR Base Address: **SCS_BA = 0xE000_E000**				
ICSR	SCS_BA+0xD04	R/W	Interrupt Control and State Register	0x0000_0000
VTOR	SCS_BA+0xD08	R/W	Vector Table Offset Register	0x0000_0000
AIRCR	SCS_BA+0xD0C	R/W	Application Interrupt and Reset Control Register	0xFA05_0000
SCR	SCS_BA+0xD10	R/W	System Control Register	0x0000_0000
SHPR2	SCS_BA+0xD1C	R/W	System Handler Priority Register 2	0x0000_0000
SHPR3	SCS_BA+0xD20	R/W	System Handler Priority Register 3	0x0000_0000
SHCSR	SCS_BA+0xD24	R/W	System Handler Control and State Register	0x0000_0000

Interrupt Control State Register (ICSR)

Register	Offset	R/W	Description	Reset Value
ICSR	SCS_BA+0xD04	R/W	Interrupt Control and State Register	0x0000_0000

31	30	29	28	27	26	25	24
NMIPENDSET	NMIPENDCLR	Reserved	PENDSVSET	PENDSVCLR	PENDSTSET	PENDSTCLR	Reserved
23	22	21	20	19	18	17	16
ISRPREEMPT	ISRPENDING	Reserved	VECTPENDING				
15	14	13	12	11	10	9	8
VECTPENDING				Reserved			VECTACTIVE
7	6	5	4	3	2	1	0
VECTACTIVE							

Bits	Description	
[31]	**NMIPENDSET**	**NMI Set-pending Bit** Write Operation: 0 = No effect. 1 = Changes NMI exception state to pending. Read Operation: 0 = NMI exception is not pending. 1 = NMI exception is pending. **Note:** If AIRCR.BFHFNMINS is 0, this bit is RAZ/WI from Non-secure state.
[30]	**NMIPENDCLR**	**NMI Bit-pending Bit** 0 = No effect. 1 = Clear pending status. **Note:** If AIRCR.BFHFNMINS is 0, this bit is RAZ/WI from Non-secure state.
[29]	**Reserved**	Reserved.
[28]	**PENDSVSET**	**PendSV Set-pending Bit** Write Operation: 0 = No effect. 1 = Changes PendSV exception state to pending. Read Operation: 0 = PendSV exception is not pending. 1 = PendSV exception is pending. **Note:** Writing 1 to this bit is the only way to set the PendSV exception state to pending.

[27]	**PENDSVCLR**	**PendSV Clear-pending Bit** Write Operation: 0 = No effect. 1 = Removes the pending state from the PendSV exception. **Note:** This is a write only bit. To clear the PENDSV bit, you must "write 0 to PENDSVSET and write 1 to PENDSVCLR" at the same time.
[26]	**PENDSTSET**	**SysTick Exception Set-pending Bit** Write Operation: 0 = No effect. 1 = Changes SysTick exception state to pending. Read Operation: 0 = SysTick exception is not pending. 1 = SysTick exception is pending.
[25]	**PENDSTCLR**	**SysTick Exception Clear-pending Bit** Write Operation: 0 = No effect. 1 = Removes the pending state from the SysTick exception. **Note:** This is a write only bit. To clear the PENDST bit, you must "write 0 to PENDSTSET and write 1 to PENDSTCLR" at the same time.
[24]	**Reserved**	Reserved.
[23]	**ISRPREEMPT**	**Interrupt Preempt Bit (Read Only)** If set, a pending exception will be serviced on exit from the debug halt state.
[22]	**ISRPENDING**	**Interrupt Pending Flag, Excluding NMI and Faults (Read Only)** 0 = Interrupt not pending. 1 = Interrupt pending.
[21]	**Reserved**	Reserved.
[20:12]	**VECTPENDING**	**Number of the Highest Pended Exception** 0 = no pending exceptions. Nonzero = the exception number of the highest priority pending enabled exception.
[11:9]	**Reserved**	Reserved.
[8:0]	**VECTACTIVE**	**Number of the Current Active Exception** 0 = Thread mode. Non-zero = The exception number of the currently active exception.

Vector Table Offset Register (VTOR)

Register	Offset	R/W	Description	Reset Value
VTOR	SCS_BA+0xD08	R/W	Vector Table Offset Register	0x0000_0000

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

Bits	Description	
[31:9]	**TBLOFF**	**Table Offset Bits** The vector table address for the selected Security state.
[8:0]	**Reserved**	Reserved.

Application Interrupt and Reset Control Register (AIRCR)

Register	Offset	R/W	Description	Reset Value
AIRCR	SCS_BA+0xD0C	R/W	Application Interrupt and Reset Control Register	0xFA05_0000

31	30	29	28	27	26	25	24
VECTORKEY							
23	22	21	20	19	18	17	16
VECTORKEY							
15	14	13	12	11	10	9	8
ENDIANNESS	PRIS	BFHFNMINS	Reserved				
7	6	5	4	3	2	1	0
Reserved				SYSRESETREQS	SYSRESETREQ	VECTCLRACTIVE	Reserved

Bits	Description	
[31:16]	**VECTORKEY**	**Register Access Key** When writing this register, this field should be 0x05FA, otherwise the write action will be ignored. The VECTORKEY filed is used to prevent accidental write to this register from resetting the system or clearing of the exception status.
[15]	**ENDIANNESS**	**Data Endianness** 0 = Little-endian. 1 = Big-endian.
[14]	**PRIS**	**Priority Secure Exceptions Bit** 0 = Priority ranges of Secure and Non-secure exceptions are identical. 1 = Non-secure exceptions are de-prioritized.
[13]	**BFHFNMINS**	**BusFault, HardFault, AndNMI Non-secure Enable Bit** 0 = BusFault, HardFault, and NMI are Secure. 1 = BusFault, Non-secure HardFault and NMI are Non-secure and exceptions can target Non-secure HardFault (Priority = -3) while Secure HardFault is secure and exception targets secure HardFault (Priority = -3).
[12:4]	**Reserved**	Reserved.
[3]	**SYSRESETREQS**	**System Reset Request Secure Only Bit** 0 = SYSRESETREQ functionality is available to both security states. 1 = SYSRESETREQ functionality is available to secure state only.
[2]	**SYSRESETREQ**	**System Reset Request Bit** Writing This Bit to 1 Will Cause A Reset Signal To Be Asserted To The Chip And Indicate A Reset Is Requested This bit is write only and self-cleared as part of the reset sequence.
[1]	**VECTCLRACTIVE**	**Exception Active Status Clear Bit** Setting This Bit To 1 Will Clears All Active State Information For Fixed And Configurable Exceptions This bit is write only and can only be written when the core is halted. **Note:** It is the debugger's responsibility to re-initialize the stack.

[0]	**Reserved**	Reserved.

System Control Register (SCR)

Register	Offset	R/W	Description	Reset Value
SCR	SCS_BA+0xD10	R/W	System Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			SEVONPEND	SLEEPDEEPS	SLEEPDEEP	SLEEPONEXIT	Reserved

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**SEVONPEND**	**Send Event on Pending** 0 = Only enabled interrupts or events can wake up the processor, while disabled interrupts are excluded. 1 = Enabled events and all interrupts, including disabled interrupts, can wake up the processor. When an event or interrupt enters pending state, the event signal wakes up the processor from WFE. If the processor is not waiting for an event, the event is registered and affects the next WFE. The processor also wakes up on execution of an SEV instruction or an external event.
[3]	**SLEEPDEEPS**	**SLEEPDEEP Bit Accessible Selection** Control whether the SLEEPDEEP bit is only accessible from the Secure state. 0 = The SLEEPDEEP bit is accessible from both security states. 1 = The SLEEPDEEP bit behaves as RAZ/WI when accessed from the Non-secure state.
[2]	**SLEEPDEEP**	**Processor Deep Sleep and Sleep Mode Selection** Control Whether the Processor Uses Sleep Or Deep Sleep as its Low Power Mode. 0 = Sleep. 1 = Deep sleep.
[1]	**SLEEPONEXIT**	**Sleep-on-exit Enable Control** This bit indicate Sleep-On-Exit when Returning from Handler Mode to Thread Mode. 0 = Do not sleep when returning to Thread mode. 1 = Enters sleep, or deep sleep, on return from an ISR to Thread mode. Setting this bit to 1 enables an interrupt driven application to avoid returning to an empty main application.
[0]	**Reserved**	Reserved.

System Handler Priority Register 2 (SHPR2)

Register	Offset	R/W	Description	Reset Value
SHPR2	SCS_BA+0xD1C	R/W	System Handler Priority Register 2	0x0000_0000

31	30	29	28	27	26	25	24
PRI_11		**Reserved**					
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:30]	**PRI_11**	**Priority of System Handler 11 – SVCall** "0" denotes the highest priority and "3" denotes the lowest priority.
[29:0]	**Reserved**	Reserved.

System Handler Priority Register 3 (SHPR3)

Register	Offset	R/W	Description	Reset Value
SHPR3	SCS_BA+0xD20	R/W	System Handler Priority Register 3	0x0000_0000

31	30	29	28	27	26	25	24
PRI_15		**Reserved**					
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PRI_14		**Reserved**					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:30]	**PRI_15**	**Priority of System Handler 15 – SysTick** "0" denotes the highest priority and "3" denotes the lowest priority.
[29:24]	**Reserved**	Reserved.
[23:22]	**PRI_14**	**Priority of System Handler 14 – PendSV** "0" denotes the highest priority and "3" denotes the lowest priority.
[21:0]	**Reserved**	Reserved.

System Handler Control and State Register (SHCSR)

Register	Offset	R/W	Description	Reset Value
SHCSR	SCS_BA+0xD24	R/W	System Handler Control and State Register	0x0000_0000

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

Bits	Description	
[31:22]	**Reserved**	Reserved.
[21]	**HARDFAULTPENDED**	**HardFault Exception Pended State** This bit indicates and allows modification of the pending state after HardFault exception corresponding to the selected Security state. This bit is banked between Security states. The possible values of this bit are: 0 = HardFault exception not pending for the selected Security state. 1 = HardFault exception pending for the selected Security state.
[20:0]	**Reserved**	Reserved.

6.2.16 Security Attribution Unit (SAU)

The Arm® Cortex®-M23 has an security attribution unit (SAU) to support hardware Arm® TrustZone® technique. The NuMicro® M2354 supports up to 8 memory regions in SAU for secure code to configure, and provides the memory alias architecture which can work only with proper setting of SAU, IDAU and SCU. IDAU has already defined all memory regions that should be non-secure (refer to the "Address Space Partition" section). Secure code should properly set these regions to non-secure by setting SAU. However, secure code should overwrite NSC regions to secure regions to prevent from being unexpectedly accessed by non-secure code.

SAU can be accessed by secure code. Non-secure access to all SAU registers will be RAZ/WI.

Register	Offset	R/W	Description	Reset Value
SCR Base Address: **SCS_BA = 0xE000_E000**				
SAU_CTRL	SCS_BA+0xDD0	R/W	SAU Control Register	0x0000_0000
SAU_TYPE	SCS_BA+0xDD4	R	SAU Type Register	0x0000_0008
SAU_RNR	SCS_BA+0xDD8	R/W	SAU Region Number Register	0x0000_00XX
SAU_RBAR	SCS_BA+0xDDC	R/W	SAU Region Base Address Register	0xXXXX_XXX0
SAU_RLAR	SCS_BA+0xDE0	R/W	SAU Region Limit Address Register	0x0000_0000

SAU Control Register (SAU_CTRL)

Register	Offset	R/W	Description	Reset Value
SAU_CTRL	SCS_BA+0xDD0	R/W	SAU Control Register	0x0000_0000

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

Bits	Description	
[31:2]	Reserved	Reserved.
[1]	ALLNS	**All Non-secure** When SAU is not enabled (ENABLE =0) this bit controls if the memory is marked as Non-secure or Secure by SAU. 0 = All Memory region is marked as Secure and is not Non-secure callable. 1 = All Memory region is marked as Non-secure by SAU, indicating the security attribute of all memory is defined by IDAU.
[0]	ENABLE	**SAU Enable Bit** 0 = SAU Disabled. 1 = SAU Enabled.

SAU Type Register (SAU_TYPE)

Register	Offset	R/W	Description	Reset Value
SAU_TYPE	SCS_BA+0xDD4	R	SAU Type Register	0x0000_0008

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

Bits	Description	
[31:8]	Reserved	Reserved.
[7:0]	SREGION	SAU Regions Indicates the number of regions implemented by the Security Attribution Unit.

SAU Region Number Register (SAU_RNR)

Register	Offset	R/W	Description	Reset Value
SAU_RNR	SCS_BA+0xDD8	R/W	SAU Region Number Register	0x0000_00XX

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**REGION**	**Current Region of SAU** Indicates the SAU region accessed by SAU_RBAR and SAU_RLAR. Set a value to select the region to be configure through SAU_RBAR and SAU_RLAR. It can be set to 0 ~ the value of SAU_TYPE -1.

SAU Region Base Address Register (SAU_RBAR)

Register	Offset	R/W	Description	Reset Value
SAU_RBAR	SCS_BA+0xDDC	R/W	SAU Region Base Address Register	0xXXXX_XXX0

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

Bits	Description	
[31:5]	**BADDR**	**Base Address of Currently Selected Region** The base address of the region selected by SAU_RNR. SAU region is 32-byte-aligned.
[4:0]	**Reserved**	Reserved.

SAU Region Limit Address Register (SAU_RLAR)

Register	Offset	R/W	Description	Reset Value
SAU_RLAR	SCS_BA+0xDE0	R/W	SAU Region Limit Address Register	0x0000_0000

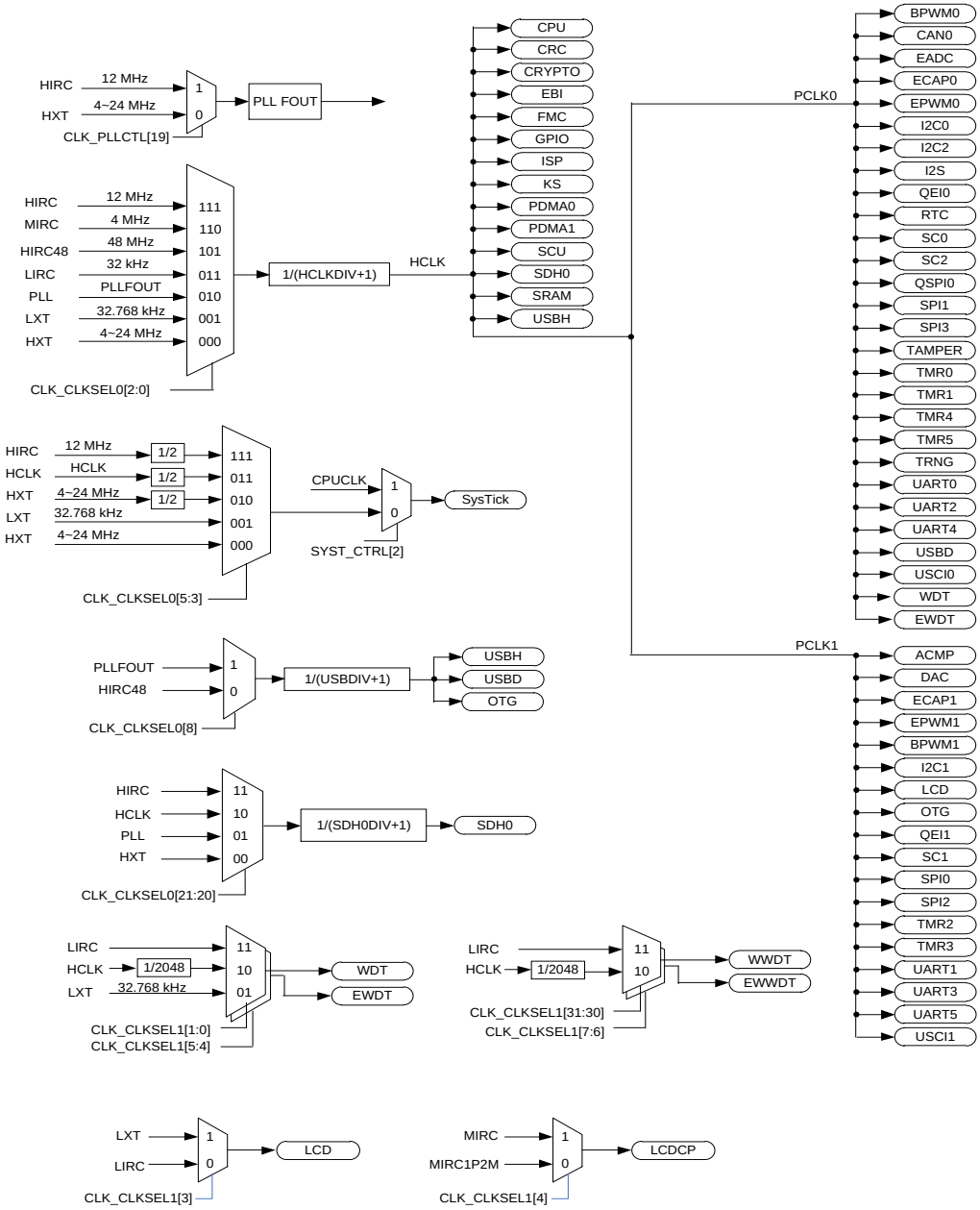
31	30	29	28	27	26	25	24
LADDR							
23	22	21	20	19	18	17	16
LADDR							
15	14	13	12	11	10	9	8
LADDR							
7	6	5	4	3	2	1	0
LADDR			Reserved			NSC	RENABLE

Bits	Description	
[31:5]	**LADDR**	**Limit Address of Currently Selected Region** The limited address of the region selected by SAU_RNR. The region of the selected SAU region is [SAU_RBAR.BADDR,5'b00000] ~ [LADDR,5'b11111]
[4:2]	**Reserved**	Reserved.
[1]	**NSC**	**Non-secure Callable Setting Bit** Controls whether Non-secure state is permitted to execute an SG instruction from this region. 0 = Region is not Non-secure callable. If RENABLE =1, the current region is Non-secure. 1 = Region is Non-secure callable. If RENABLE=1, the current region is Secure and Non-secure callable.
[0]	**RENABLE**	**Region Enable Bit** Enable or disable the currently selected region set by SAU_RNR. 0 = Disabled selected region set by SAU_RNR. 1 = Enabled selected region set by SAU_RNR.

6.3 Clock Controller

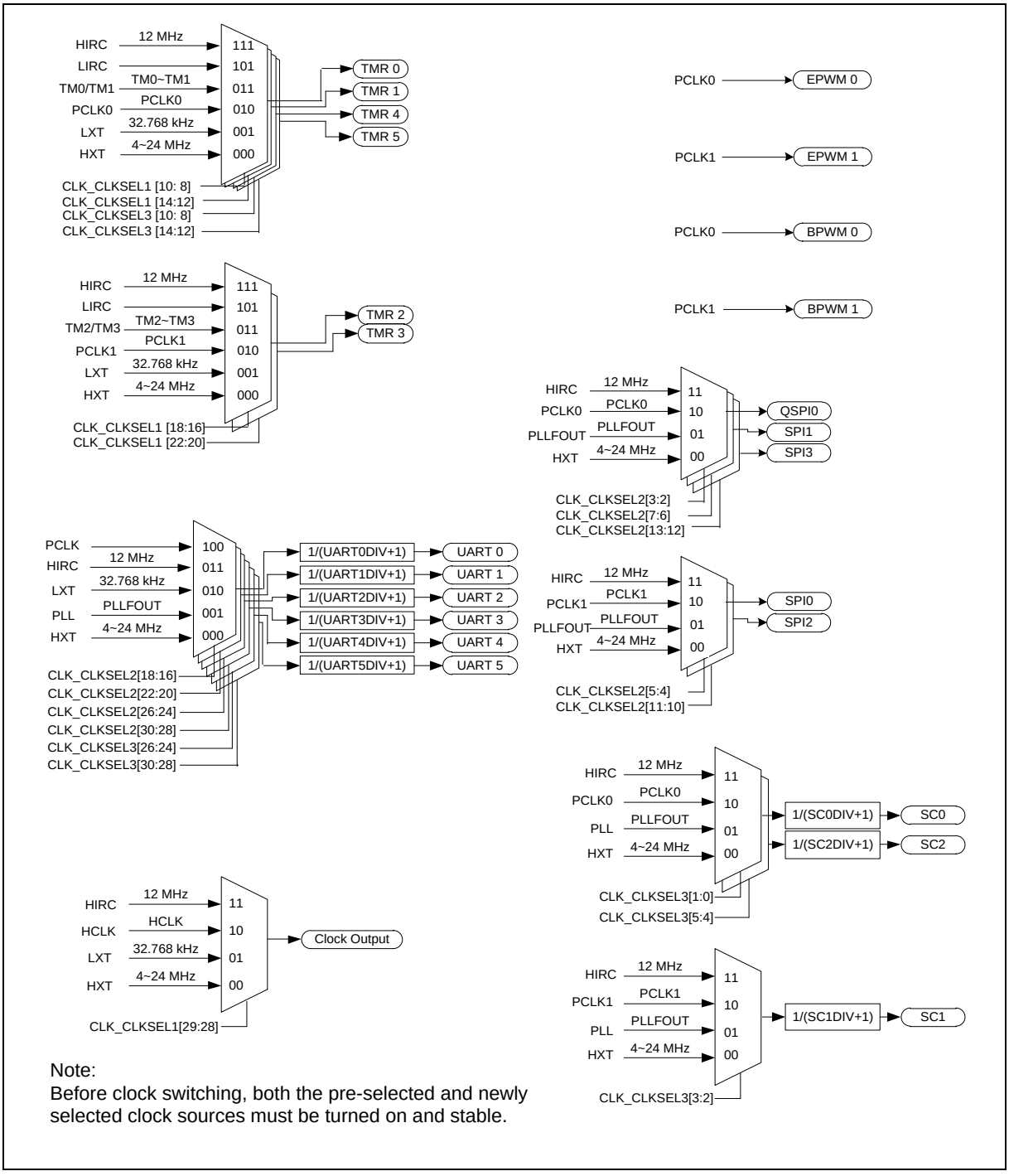
6.3.1 Overview

The clock controller generates clocks for the whole chip, including system clocks and all peripheral clocks. The clock controller also implements the power control function with the individually clock ON/OFF control, clock source selection and a clock divider. The chip will not enter Power-down mode until CPU sets the Power-down enable bit PDEN (CLK_PWRCTL[7]) and core executes the WFI instruction. After that, chip enters Power-down mode and wait for wake-up interrupt source triggered to leave Power-down mode. In Power-down mode, the clock controller turns off the 4~24 MHz external high speed crystal (HXT), 48 MHz internal high speed RC oscillator (HIRC48), 4 MHz internal medium speed RC oscillator (MIRC) and 12 MHz internal high speed RC oscillator (HIRC) to reduce the overall system power consumption. Figure 6.3-1 to Figure 6.3-3 show the clock generator and the overview of the clock source control.



Note:
Before clock switching, both the pre-selected and newly selected clock sources must be turned on and stable.

Figure 6.3-1 Clock Generator Global View Diagram (1/3)



Note:
Before clock switching, both the pre-selected and newly selected clock sources must be turned on and stable.

Figure 6.3-2 Clock Generator Global View Diagram (2/3)

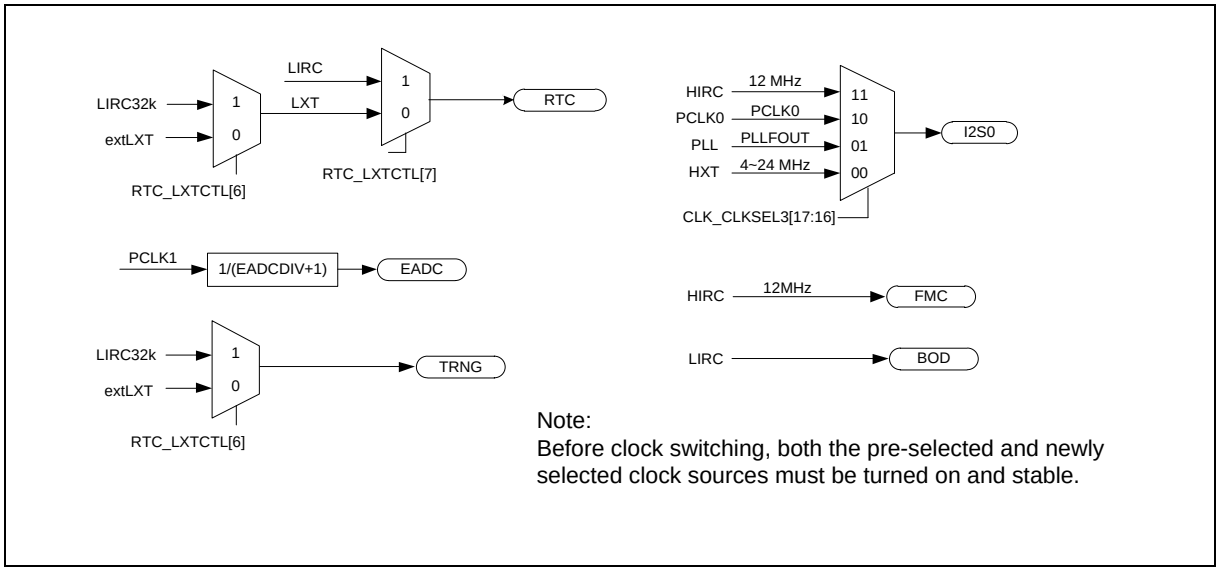
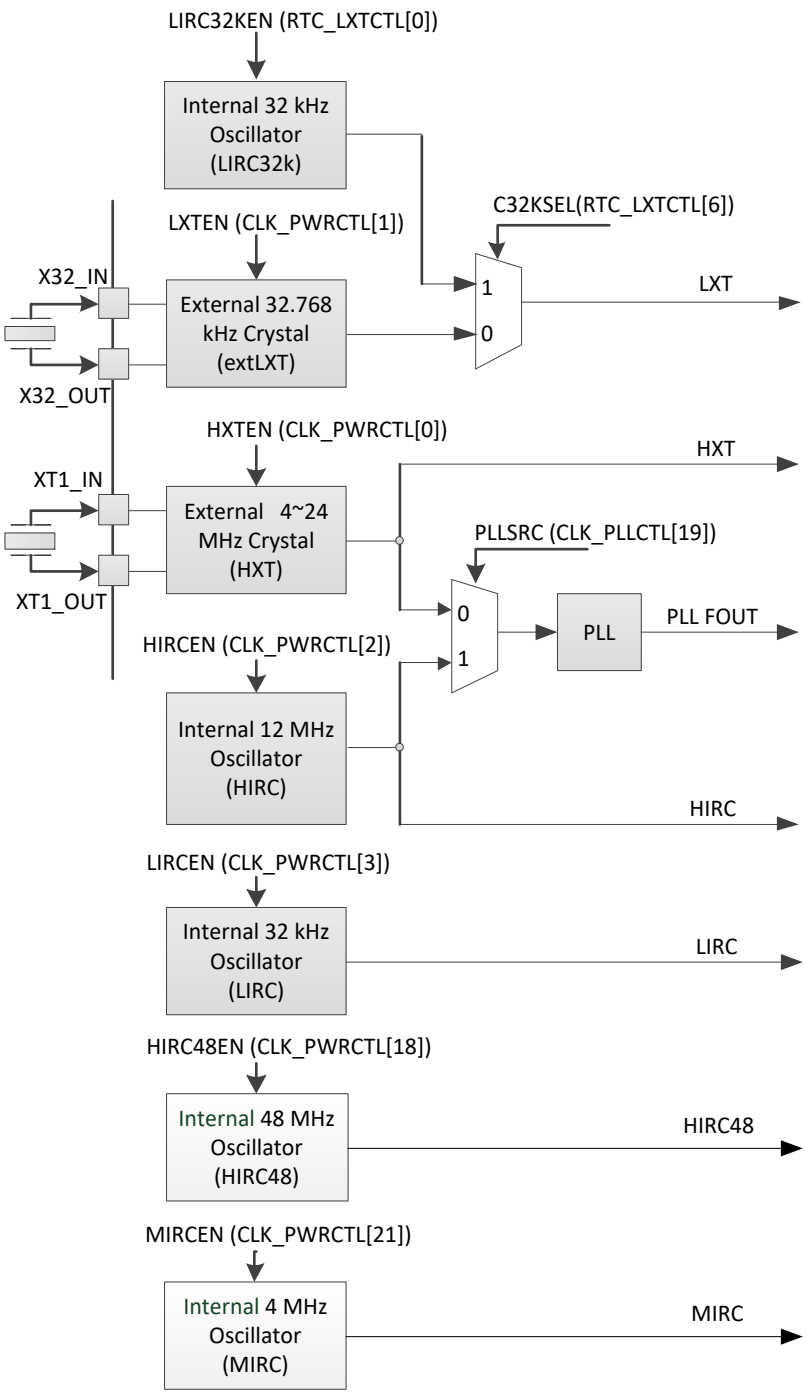


Figure 6.3-3 Clock Generator Global View Diagram (3/3)

6.3.2 Clock Generator

The clock generator consists of 7 clock sources, which are listed below:

- 32.768 kHz external low speed crystal oscillator (LXT)
- 4~24 MHz external high speed crystal oscillator (HXT)
- Programmable PLL output clock frequency (PLLFOUT), PLL source can be selected from external 4~24 MHz external high speed crystal (HXT) or 12 MHz internal high speed oscillator (HIRC)
- 12 MHz internal high speed RC oscillator (HIRC)
- 4 MHz internal medium speed RC oscillator (MIRC)
- 48 MHz internal high speed RC oscillator (HIRC48)
- 32 kHz internal low speed RC oscillator (LIRC)



Note:
Before clock switching, both the pre-selected and newly selected clock sources must be turned on and stable.

Figure 6.3-4 Clock Generator Block Diagram

Each of these clock sources has certain stable time to wait for clock operating at stable frequency. When clock source is enabled, a stable counter start counting and correlated clock stable index is set to 1 after stable counter value reach a define value.

System and peripheral can use the clock as its operating clock only when correlate clock stable index is set to 1. The clock stable index as shown in Table 6.3-1 will auto clear when user disables the clock source.

Besides, the clock stable index of HXT, HIRC, MIRC, HIRC48 and PLL will auto clear when chip enter power-down and clock stable counter will re-counting after chip wake-up if correlate clock is enabled.

Clock Source	Clock Source Enable Bit	Correlated Clock Stable Index
HXT	HXTEN (CLK_PWRCTL[0])	HXTSTB (CLK_STATUS[0])
LXT	LXTEN (CLK_PWRCTL[1]) or LIRC32KEN (RTC_LXTCTL[0])	LXTSTB (CLK_STATUS[1])
PLL	PD (CLK_PLLCTL[16])	PLLSTB (CLK_STATUS[2])
LIRC	LIRCEN (CLK_PWRCTL[3])	LIRCSTB (CLK_STATUS[3])
HIRC	HIRCEN (CLK_PWRCTL[2])	HIRCSTB (CLK_STATUS[4])
MIRC	MIRCEN (CLK_PWRCTL[21])	MIRCSTB (CLK_STATUS[5])
HIRC48	HIRC48EN (CLK_PWRCTL[18])	HIRC48STB (CLK_STATUS[6])
extLXT	LXTEN (CLK_PWRCTL[1])	EXTLXTSTB (CLK_STATUS[8])
LIRC32	LIRC32KEN (RTC_LXTCTL[0])	LIRC32STB (CLK_STATUS[9])

Table 6.3-1 Each Clock Source Enable Bit and Corresponding Stable Flag Table

6.3.3 System Clock and SysTick Clock

The system clock has 7 clock sources, which were generated from clock generator block. The clock source switch depends on the register HCLKSEL (CLK_CLKSEL0[2:0]). The block diagram is shown in Figure 6.3-5.

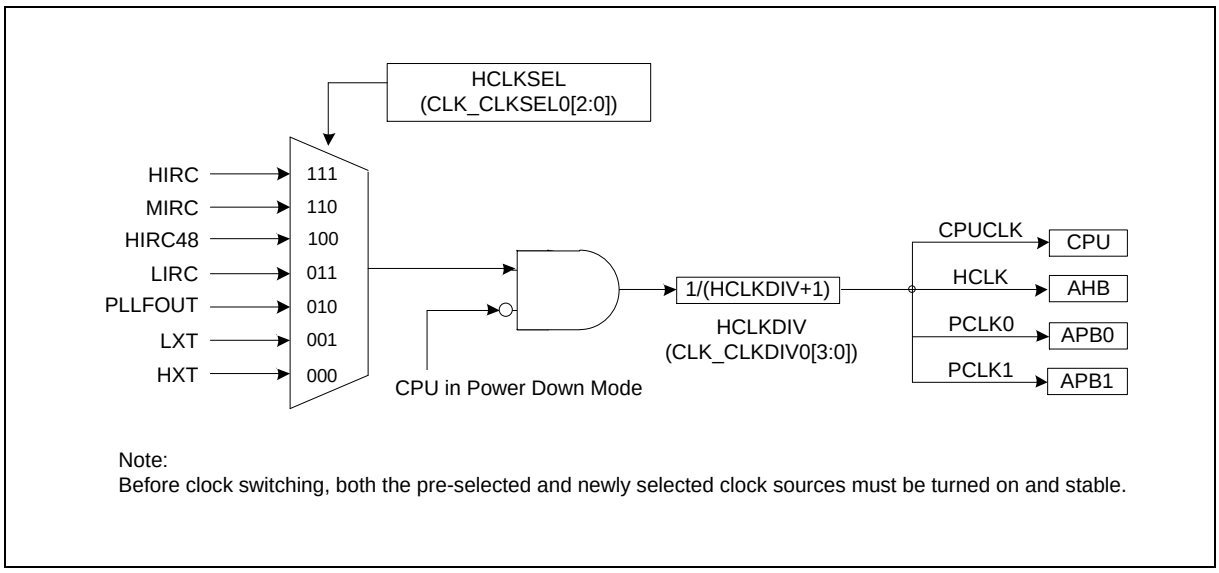


Figure 6.3-5 System Clock Block Diagram

There are two clock fail detectors to observe HXT and LXT clock source and they have individual enable and interrupt control. When HXT detector is enabled, the HIRC clock is enabled automatically. When LXT detector is enabled, the LIRC clock is enabled automatically.

When HXT clock detector is enabled, the system clock will auto switch to HIRC if HXT clock stop being detected on the following condition: system clock source comes from HXT or system clock source comes from PLL with HXT as the input of PLL. If HXT clock stop condition is detected, the HXTFIF (CLK_CLKDSTS[0]) is set to 1 and chip will enter interrupt if HXTFIE (CLK_CLKDCTL[5]) is set to 1. HXT clock source stable flag, HXTSTB (CLK_STATUS[0]), will be cleared if HXT stops when using HXT fail detector function. User can trying to recover HXT by disable HXT and enable HXT again to check if the clock stable bit is set to 1 or not. If HXT clock stable bit is set to 1, it means HXT is recover to oscillate after re-enable action and user can switch system clock to HXT again.

When LXT clock detector is enabled, the system clock will auto switch to LIRC if LXT clock stop being detected on the following condition: system clock source comes from LXT. If LXT clock stop condition is detected, the LXTFIF (CLK_CLKDSTS[1]) is set to 1 and chip will enter interrupt if LXTFIE (CLK_CLKDCTL[5]) is set to 1. LXT clock source stable flag, LXTSTB (CLK_STATUS[1]), will be cleared if LXT stops when using LXT fail detector function. User can trying to recover LXT by disable LXT and enable LXT again to check if the clock stable bit is set to 1 or not. If LXT clock stable bit is set to 1, it means LXT is recover to oscillate after re-enable action and user can switch system clock to LXT again.

The HXT clock stopping detecting and system clock switch to HIRC procedure is shown in Figure 6.3-6.

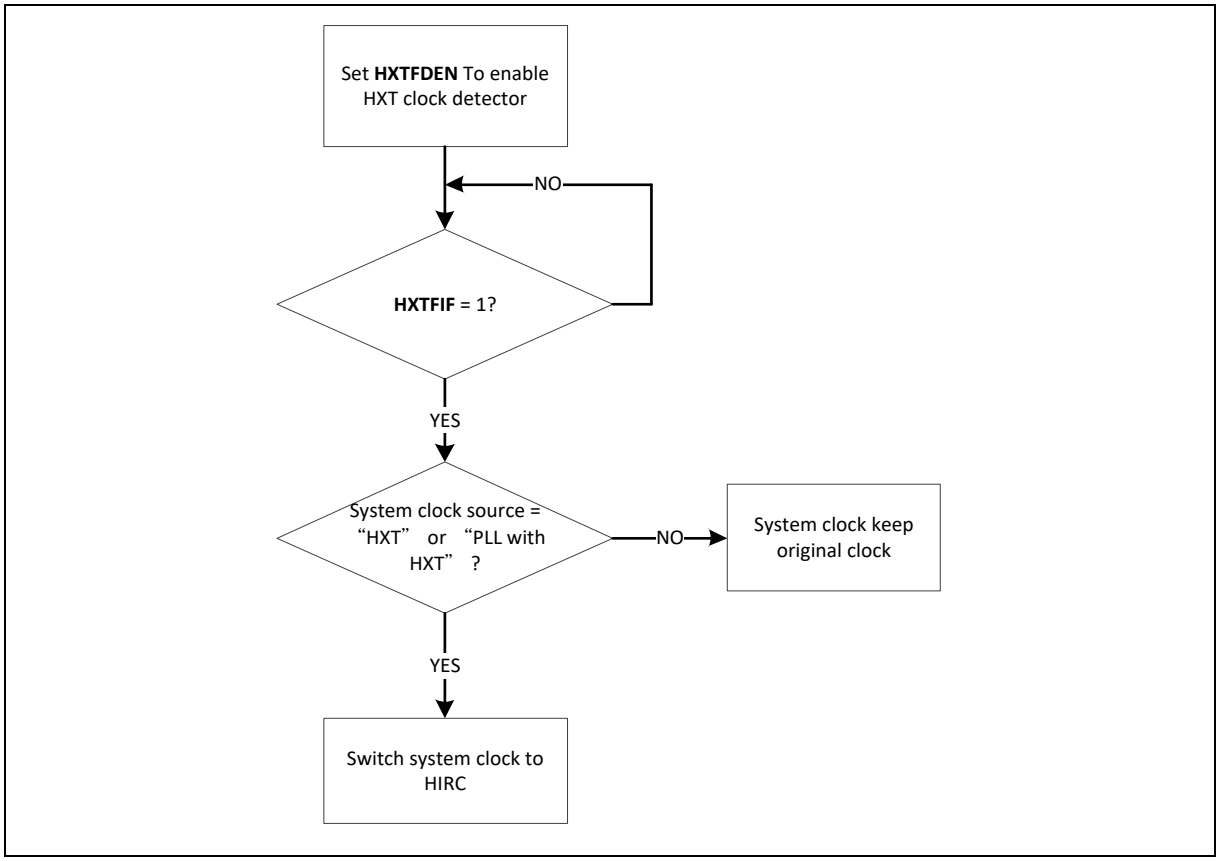
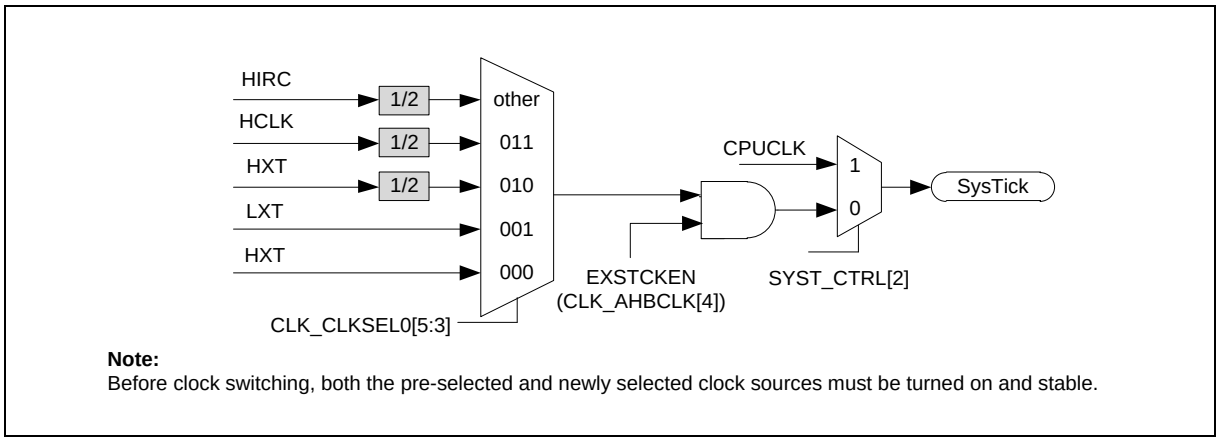


Figure 6.3-6 HXT Stop Protect Procedure



Note:
Before clock switching, both the pre-selected and newly selected clock sources must be turned on and stable.

Figure 6.3-7 SysTick Clock Control Block Diagram

The clock source of SysTick in processor can use CPU clock or external clock (SYST_CTRL[2]). If using external clock, the SysTick clock (STCLK) has 5 clock sources. The clock source switch depends on the setting of the register STCLKSEL (CLK_CLKSEL0[5:3]). The block diagram is shown in Figure 6.3-7.

6.3.4 Peripherals Clock

Each peripheral clock has its own clock source selection. Refer to the CLK_CLKSEL1, CLK_CLKSEL2

and CLK_CLKSEL3 register.

6.3.5 Power-down Mode Clock

When entering Power-down mode, system clocks, some clock sources and some peripheral clocks are disabled. Some clock sources and peripherals clock are still active in Power-down mode.

For theses clocks, which still keep active, are listed below:

- Clock Generator
  - 32 kHz internal low speed RC oscillator (LIRC) clock
  - 32.768 kHz external low speed crystal oscillator (LXT) clock
  - 4 MHz internal medium speed RC oscillator (MIRC) clock when TIMER4~5 or LCDCP select MIRC as peripheral clock source
- Peripherals Clock
  - when the modules adopt LXT or LIRC as clock source
  - when TIMER4~5 or LCDCP select MIRC as peripheral clock source

6.3.6 Clock Output

This device is equipped with a power-of-2 frequency divider that is composed by 16 chained divide-by-2 shift registers. One of the 16 shift register outputs selected by a sixteen to one multiplexer is reflected to CLKO function pin. Therefore, there are 16 options of power-of-2 divided clocks with the frequency from $F_{in}/2^1$ to $F_{in}/2^{16}$ where F_{in} is input clock frequency to the clock divider.

The output formula is $\mathbf{F_{out} = F_{in}/2^{(N+1)}}$, where F_{in} is the input clock frequency, F_{out} is the clock divider output frequency and N is the 4-bit value in FREQSEL (CLK_CLKOCTL[3:0]). When writing 1 to CLKOEN (CLK_CLKOCTL[4]), the chained counter starts to count. When writing 0 to CLKOEN (CLK_CLKOCTL[4]), the chained counter continuously runs till divided clock reaches low state and stays in low state.

If DIV1EN(CLK_CLKOCTL[5]) is set to 1, the clock output clock (CLKO_CLK) will bypass power-of-2 frequency divider. The output divider clock will be output to CLKO pin directly.

When entering Power-down mode, clock output does not output clock even if the CKO clock source is LXT.

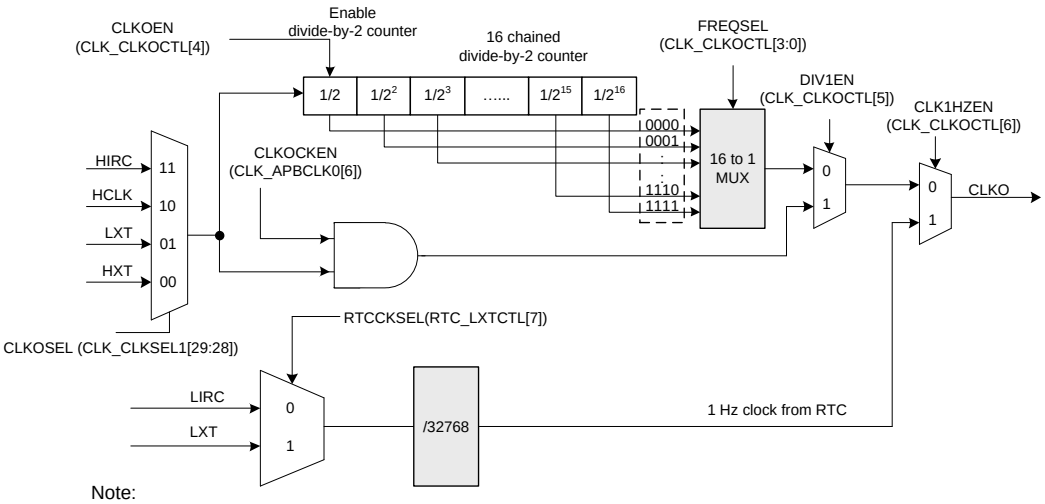


Figure 6.3-8 Clock Output Block Diagram

6.3.7 Share Registers

The clock controller shares part of register information to non-secure world with enable bits in SYSSIAEN (SCU_SINFAEN[1]) register. Shared registers are enabled by default.

Shared Register Access	Clock Controller
R/W	NA
Read only	CLK_PWRCTL, CLK_AHBCLK, CLK_APBCLK0, CLK_APBCLK1, CLK_CLKSEL0, CLK_CLKSEL1, CLK_CLKSEL2, CLK_CLKSEL3, CLK_CLKDIV0, CLK_CLKDIV1, CLK_CLKDIV4, CLK_PLLCTL, CLK_STATUS
Write only	NA

Table 6.3-2 Clock Controller Share Register list

6.3.8 Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
CLK Base Address: **CLK_BA = 0x4000_0200** **CLK_BA non-secure base address is CLK_BA + 0x1000_0000.**				
CLK_PWRCTL	CLK_BA+0x00	R/W	System Power-down Control Register	0x002X_XX08
CLK_AHBCLK	CLK_BA+0x04	R/W	AHB Devices Clock Enable Register	0x0010_8000
CLK_APBCLK0	CLK_BA+0x08	R/W	APB Devices Clock Enable Register 0	0x8000_0001
CLK_APBCLK1	CLK_BA+0x0C	R/W	APB Devices Clock Enable Register 1	0x0000_0000
CLK_CLKSEL0	CLK_BA+0x10	R/W	Clock Source Select Register 0	0x0020_011X
CLK_CLKSEL1	CLK_BA+0x14	R/W	Clock Source Select Register 1	0xA022_22B3
CLK_CLKSEL2	CLK_BA+0x18	R/W	Clock Source Select Register 2	0x4444_2BAB
CLK_CLKSEL3	CLK_BA+0x1C	R/W	Clock Source Select Register 3	0x4402_222A
CLK_CLKDIV0	CLK_BA+0x20	R/W	Clock Divider Number Register 0	0x0000_000X
CLK_CLKDIV1	CLK_BA+0x24	R/W	Clock Divider Number Register 1	0x0000_0000
CLK_CLKDIV4	CLK_BA+0x30	R/W	Clock Divider Number Register 4	0x0000_0000
CLK_PLLCTL	CLK_BA+0x40	R/W	PLL Control Register	0x0009_440A
CLK_STATUS	CLK_BA+0x50	R	Clock Status Monitor Register	0x0000_00X8
CLK_CLKOCTL	CLK_BA+0x60	R/W	Clock Output Control Register	0x0000_0000
CLK_CLKDCTL	CLK_BA+0x70	R/W	Clock Fail Detector Control Register	0x0000_0000
CLK_CLKDSTS	CLK_BA+0x74	R/W	Clock Fail Detector Status Register	0x0000_0000
CLK_CDUPB	CLK_BA+0x78	R/W	Clock Frequency Detector Upper Boundary Register	0x0000_0000
CLK_CDLOWB	CLK_BA+0x7C	R/W	Clock Frequency Detector Lower Boundary Register	0x0000_0000
CLK_PMUCTL	CLK_BA+0x90	R/W	Power Manager Control Register	0x0000_0010
CLK_PMUSTS	CLK_BA+0x94	R/W	Power Manager Status Register	0x0000_0000
CLK_SWKDBCTL	CLK_BA+0x9C	R/W	Standby Power-down Wake-up De-bounce Control Register	0x0000_0000
CLK_PASWKCTL	CLK_BA+0xA0	R/W	GPA Standby Power-down Wake-up Control Register	0x0000_0000
CLK_PBSWKCTL	CLK_BA+0xA4	R/W	GPB Standby Power-down Wake-up Control Register	0x0000_0000
CLK_PCSWKCTL	CLK_BA+0xA8	R/W	GPC Standby Power-down Wake-up Control Register	0x0000_0000
CLK_PDSWKCTL	CLK_BA+0xAC	R/W	GPD Standby Power-down Wake-up Control Register	0x0000_0000

CLK_IOPDCTL	CLK_BA+0xB0	R/W	GPIO Standby Power-down Control Register	0x0000_0000
CLK_HXTFSEL	CLK_BA+0xB4	R/W	HXT Filter Select Register	0x0000_0000

6.3.9 Register Description

System Power-down Control Register (CLK_PWRCTL)

Register	Offset	R/W	Description	Reset Value
CLK_PWRCTL	CLK_BA+0x00	R/W	System Power-down Control Register	0x002X_XX08

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		MIRCEN	MIRC1P2MEN	Reserved	HIRC48EN	Reserved	
15	14	13	12	11	10	9	8
Reserved		HXTTBEN	HXTSELTYP	HXTGAIN		Reserved	
7	6	5	4	3	2	1	0
PDEN	PDWKIF	PDWKIEN	Reserved	LIRCEN	HIRCEN	LXTEN	HXTEN

Bits	Description	
[31:22]	**Reserved**	Reserved.
[21]	**MIRCEN**	**MIRC Enable Bit (Write Protect)** 0 = 4 MHz internal medium speed RC oscillator (MIRC) Disabled. 1 = 4 MHz internal medium speed RC oscillator (MIRC) Enabled. **Note 1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note 2:** MIRC cannot be disabled and MIRCEN will read as 1 if HCLK clock source is selected from MIRC.
[20]	**MIRC1P2MEN**	**MIRC1P2M Enable Bit (Write Protect)** 0 = 1.2 MHz internal medium speed RC oscillator (MIRC1P2M) Disabled. 1 = 1.2 MHz internal medium speed RC oscillator (MIRC1P2M) Enabled. **Note 1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note 2:** This clock source only for LCD use.
[19]	**Reserved**	Reserved.
[18]	**HIRC48EN**	**HIRC48 Enable Bit (Write Protect)** 0 = 48 MHz internal high speed RC oscillator (HIRC48) Disabled. 1 = 48 MHz internal high speed RC oscillator (HIRC48) Enabled. **Note 1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note 2:** HIRC48 cannot be disabled and HIRC48EN will read as 1 if HCLK clock source is selected from HIRC48.
[17:14]	**Reserved**	Reserved.
[13]	**HXTTBEN**	**HXT Crystal TURBO Mode (Write Protect)** 0 = HXT Crystal TURBO mode Disabled. 1 = HXT Crystal TURBO mode Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

[12]	**HXTSELTYP**	**HXT Crystal Type Select Bit (Write Protect)** 0 = Select INV type. 1 = Select GM type. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[11:10]	**HXTGAIN**	**HXT Gain Control Bit (Write Protect)** Gain control is used to enlarge the gain of crystal to make sure crystal work normally. If gain control is enabled, crystal will consume more power than gain control off. 00 = HXT frequency is lower than from 8 MHz. 01 = HXT frequency is from 8 MHz to 12 MHz. 10 = HXT frequency is from 12 MHz to 16 MHz. 11 = HXT frequency is higher than 16 MHz. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[9:8]	**Reserved**	Reserved.
[7]	**PDEN**	**System Power-down Enable (Write Protect)** When this bit is set to 1, Power-down mode is enabled and the chip keeps active till the CPU sleep mode is also active and then the chip enters Power-down mode. When chip wakes up from Power-down mode, this bit is auto cleared. Users need to set this bit again for next Power-down. In Power-down mode, HXT, HIRC, HIRC48, PLL and system clock will be disabled and ignored the clock source selection. The clocks of peripheral are not controlled by Power-down mode, if the peripheral clock source is from LXT, LIRC or MIRC. 0 = Chip operating normally or chip in idle mode because of WFI command. 1 = Chip waits CPU sleep command WFI and then enters Power-down mode. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[6]	**PDWKIF**	**Power-down Mode Wake-up Interrupt Status** Set by “Power-down wake-up event”, it indicates that resume from Power-down mode” The flag is set if the EINT7~0, GPIO, UART0~5, USBH, USBD, OTG, CAN0, BOD, ACMP, WDT, EWDT, SDH0, TIMER, I2C0~2, USCI0~1, RTC, TAMPER and CLKD wake-up occurred. **Note 1:** Write 1 to clear the bit to 0. **Note 2:** This bit works only if PDWKIEN (CLK_PWRCTL[5]) set to 1.
[5]	**PDWKIEN**	**Power-down Mode Wake-up Interrupt Enable Bit (Write Protect)** 0 = Power-down mode wake-up interrupt Disabled. 1 = Power-down mode wake-up interrupt Enabled. **Note 1:** The interrupt will occur when both PDWKIF and PDWKIEN are high, after resume from power-down mode. **Note 2:** This bit is write protected. Refer to the SYS_REGLCTL register.
[4]	**Reserved**	Reserved.
[3]	**LIRCEN**	**LIRC Enable Bit (Write Protect)** 0 = 32 kHz internal low speed RC oscillator (LIRC) Disabled. 1 = 32 kHz internal low speed RC oscillator (LIRC) Enabled. **Note 1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note 2:** LIRC cannot be disabled and LIRCEN will read as 1 if HCLK clock source is selected from LIRC. **Note 3:** If CWDTEN(CONFIG0[31,4:3]) is set to 111, LIRC clock can be enabled or disabled by setting LIRCEN(CLK_PWRCTL[3]). If CWDTEN(CONFIG0[31,4:3]) is not set to 111, LIRC cannot be disabled in normal mode. In Power-down mode, LIRC clock is controlled by LIRCEN(CLK_PWRCTL[3]) and CWDTPDEN(CONFIG0[30]) setting.

[2]	**HIRCEN**	**HIRC Enable Bit (Write Protect)** 0 = 12 MHz internal high speed RC oscillator (HIRC) Disabled. 1 = 12 MHz internal high speed RC oscillator (HIRC) Enabled. **Note 1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note 2:** HIRC cannot be disabled and HIRCEN will read as 1 if HCLK clock source is selected from HIRC or PLL (clock source from HIRC).
[1]	**LXTEN**	**LXT Enable Bit (Write Protect)** 0 = 32.768 kHz external low speed crystal (extLXT) Disabled. 1 = 32.768 kHz external low speed crystal (extLXT) Enabled. **Note 1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note 2:** LXT cannot be disabled and LXTEN will read as 1 if HCLK clock source is selected from LXT when the LXT clock source is selected as extLXT by setting C32KSEL(RTC_LXTCTL[6]) to 1.
[0]	**HXTEN**	**HXT Enable Bit (Write Protect)** 0 = 4~24 MHz external high speed crystal (HXT) Disabled. 1 = 4~24 MHz external high speed crystal (HXT) Enabled. **Note 1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note 2:** HXT cannot be disabled and HXTEN will read as 1 if HCLK clock source is selected from HXT or PLL (clock source from HXT).

AHB Devices Clock Enable Control Register (CLK_AHBCLK)

The bits in this register are used to enable/disable clock for system clock, AHB bus devices clock.

Register	Offset	R/W	Description	Reset Value
CLK_AHBCLK	CLK_BA+0x04	R/W	AHB Devices Clock Enable Register	0x0010_8000

31	30	29	28	27	26	25	24
GPHCKEN	GPGCKEN	GPFCKEN	GPECKEN	GPDCKEN	GPCCKEN	GPBCKEN	GPACKEN
23	22	21	20	19	18	17	16
Reserved	SRAM2CKEN	SRAM1CKEN	SRAM0CKEN	Reserved			USBHCKEN
15	14	13	12	11	10	9	8
FMCIDLE	TRACECKEN	KSCKEN	CRPTCKEN	Reserved			
7	6	5	4	3	2	1	0
CRCCKEN	SDH0CKEN	Reserved	EXSTCKEN	EBICKEN	ISPCKEN	PDMA1CKEN	PDMA0CKEN

Bits	Description	
[31]	**GPHCKEN**	**GPIOH Clock Enable Bit** 0 = GPIOH port clock Disabled. 1 = GPIOH port clock Enabled.
[30]	**GPGCKEN**	**GPIOG Clock Enable Bit** 0 = GPIOG port clock Disabled. 1 = GPIOG port clock Enabled.
[29]	**GPFCKEN**	**GPIOF Clock Enable Bit** 0 = GPIOF port clock Disabled. 1 = GPIOF port clock Enabled.
[28]	**GPECKEN**	**GPIOE Clock Enable Bit** 0 = GPIOE port clock Disabled. 1 = GPIOE port clock Enabled.
[27]	**GPDCKEN**	**GPIOD Clock Enable Bit** 0 = GPIOD port clock Disabled. 1 = GPIOD port clock Enabled.
[26]	**GPCCKEN**	**GPIOC Clock Enable Bit** 0 = GPIOC port clock Disabled. 1 = GPIOC port clock Enabled.
[25]	**GPBCKEN**	**GPIOB Clock Enable Bit** 0 = GPIOB port clock Disabled. 1 = GPIOB port clock Enabled.
[24]	**GPACKEN**	**GPIOA Clock Enable Bit** 0 = GPIOA port clock Disabled.

		1 = GPIOA port clock Enabled.
[23]	**Reserved**	Reserved.
[22]	**SRAM2CKEN**	**SRAM Bank2 Controller Clock Enable Bit** 0 = SRAM bank2 clock Disabled. 1 = SRAM bank2 clock Enabled.
[21]	**SRAM1CKEN**	**SRAM Bank1 Controller Clock Enable Bit** 0 = SRAM bank1 clock Disabled. 1 = SRAM bank1 clock Enabled.
[20]	**SRAM0CKEN**	**SRAM Bank0 Controller Clock Enable Bit** 0 = SRAM bank0 clock Disabled. 1 = SRAM bank0 clock Enabled.
[19:17]	**Reserved**	Reserved.
[16]	**USBHCKEN**	**USB HOST 1.1 Controller Clock Enable Bit** 0 = USB HOST 1.1 peripheral clock Disabled. 1 = USB HOST 1.1 peripheral clock Enabled.
[15]	**FMCIDLE**	**Flash Memory Controller Clock Enable Bit in IDLE Mode** 0 = FMC clock Disabled when chip is under IDLE mode. 1 = FMC clock Enabled when chip is under IDLE mode.
[14]	**TRACECKEN**	**Trace Clock Enable Bit** 0 = Trace clock Disabled. 1 = Trace clock Enabled.
[13]	**KSCKEN**	**Key Store Clock Enable Bit** 0 = Key store clock Disabled. 1 = Key store clock Enabled.
[12]	**CRPTCKEN**	**Cryptographic Accelerator Clock Enable Bit** 0 = Cryptographic Accelerator clock Disabled. 1 = Cryptographic Accelerator clock Enabled.
[11:8]	**Reserved**	Reserved.
[7]	**CRCCKEN**	**CRC Generator Controller Clock Enable Bit** 0 = CRC peripheral clock Disabled. 1 = CRC peripheral clock Enabled.
[6]	**SDH0CKEN**	**SDHOST0 Controller Clock Enable Bit** 0 = SDHOST0 peripheral clock Disabled. 1 = SDHOST0 peripheral clock Enabled.
[5]	**Reserved**	Reserved.
[4]	**EXSTCKEN**	**External System Tick Clock Enable Bit** 0 = External System tick clock Disabled. 1 = External System tick clock Enabled.
[3]	**EBICKEN**	**EBI Controller Clock Enable Bit** 0 = EBI peripheral clock Disabled. 1 = EBI peripheral clock Enabled.

[2]	**ISPCKEN**	**Flash ISP Controller Clock Enable Bit** 0 = Flash ISP peripheral clock Disabled. 1 = Flash ISP peripheral clock Enabled.
[1]	**PDMA1CKEN**	**PDMA1 Controller Clock Enable Bit** 0 = PDMA1 peripheral clock Disabled. 1 = PDMA1 peripheral clock Enabled.
[0]	**PDMA0CKEN**	**PDMA0 Controller Clock Enable Bit (Secure)** 0 = PDMA0 peripheral clock Disabled. 1 = PDMA0 peripheral clock Enabled.

APB Devices Clock Enable Control Register 0 (CLK_APBCLK0)

The bits in this register are used to enable/disable clock for peripheral controller clocks.

Register	Offset	R/W	Description	Reset Value
CLK_APBCLK0	CLK_BA+0x08	R/W	APB Devices Clock Enable Register 0	0x8000_0001

31	30	29	28	27	26	25	24
EWDTCKEN	Reserved	I2S0CKEN	EADCCKEN	USBDCKEN	OTGCKEN	Reserved	CAN0CKEN
23	22	21	20	19	18	17	16
Reserved	TAMPERCKEN	UART5CKEN	UART4CKEN	UART3CKEN	UART2CKEN	UART1CKEN	UART0CKEN
15	14	13	12	11	10	9	8
SPI2CKEN	SPI1CKEN	SPI0CKEN	QSPI0CKEN	Reserved	I2C2CKEN	I2C1CKEN	I2C0CKEN
7	6	5	4	3	2	1	0
ACMP01CKEN	CLKOCKEN	TMR3CKEN	TMR2CKEN	TMR1CKEN	TMR0CKEN	RTCCKEN	WDTCKEN

Bits	Description	
[31]	**EWDTCKEN**	**Extra Watchdog Timer Clock Enable Bit (Write Protect)** 0 = Extra Watchdog timer and Windows watchdog timer clock Disabled. 1 = Extra Watchdog timer and Windows watchdog timer clock Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[30]	**Reserved**	Reserved.
[29]	**I2S0CKEN**	**I2S0 Clock Enable Bit** 0 = I2S0 Clock Disabled. 1 = I2S0 Clock Enabled.
[28]	**EADCCKEN**	**Enhanced Analog-digital-converter (EADC) Clock Enable Bit** 0 = EADC clock Disabled. 1 = EADC clock Enabled.
[27]	**USBDCKEN**	**USB Device Clock Enable Bit** 0 = USB Device clock Disabled. 1 = USB Device clock Enabled.
[26]	**OTGCKEN**	**USB OTG Clock Enable Bit** 0 = USB OTG clock Disabled. 1 = USB OTG clock Enabled.
[25]	**Reserved**	Reserved.
[24]	**CAN0CKEN**	**CAN0 Clock Enable Bit** 0 = CAN0 clock Disabled. 1 = CAN0 clock Enabled.
[23]	**Reserved**	Reserved.

[22]	**TAMPERCKEN**	**TAMPER Clock Enable Bit** 0 = Tamper clock Disabled. 1 = Tamper clock Enabled.
[21]	**UART5CKEN**	**UART5 Clock Enable Bit** 0 = UART5 clock Disabled. 1 = UART5 clock Enabled.
[20]	**UART4CKEN**	**UART4 Clock Enable Bit** 0 = UART4 clock Disabled. 1 = UART4 clock Enabled.
[19]	**UART3CKEN**	**UART3 Clock Enable Bit** 0 = UART3 clock Disabled. 1 = UART3 clock Enabled.
[18]	**UART2CKEN**	**UART2 Clock Enable Bit** 0 = UART2 clock Disabled. 1 = UART2 clock Enabled.
[17]	**UART1CKEN**	**UART1 Clock Enable Bit** 0 = UART1 clock Disabled. 1 = UART1 clock Enabled.
[16]	**UART0CKEN**	**UART0 Clock Enable Bit** 0 = UART0 clock Disabled. 1 = UART0 clock Enabled.
[15]	**SPI2CKEN**	**SPI2 Clock Enable Bit** 0 = SPI2 clock Disabled. 1 = SPI2 clock Enabled.
[14]	**SPI1CKEN**	**SPI1 Clock Enable Bit** 0 = SPI1 clock Disabled. 1 = SPI1 clock Enabled.
[13]	**SPI0CKEN**	**SPI0 Clock Enable Bit** 0 = SPI0 clock Disabled. 1 = SPI0 clock Enabled.
[12]	**QSPI0CKEN**	**QSPI0 Clock Enable Bit** 0 = QSPI0 clock Disabled. 1 = QSPI0 clock Enabled.
[11]	**Reserved**	Reserved.
[10]	**I2C2CKEN**	**I2C2 Clock Enable Bit** 0 = I2C2 clock Disabled. 1 = I2C2 clock Enabled.
[9]	**I2C1CKEN**	**I2C1 Clock Enable Bit** 0 = I2C1 clock Disabled. 1 = I2C1 clock Enabled.
[8]	**I2C0CKEN**	**I2C0 Clock Enable Bit** 0 = I2C0 clock Disabled.

		1 = I2C0 clock Enabled.
[7]	**ACMP01CKEN**	**Analog Comparator 0/1 Clock Enable Bit** 0 = Analog comparator 0/1 clock Disabled. 1 = Analog comparator 0/1 clock Enabled.
[6]	**CLKOCKEN**	**CLKO Clock Enable Bit** 0 = CLKO clock Disabled. 1 = CLKO clock Enabled.
[5]	**TMR3CKEN**	**Timer3 Clock Enable Bit** 0 = Timer3 clock Disabled. 1 = Timer3 clock Enabled.
[4]	**TMR2CKEN**	**Timer2 Clock Enable Bit** 0 = Timer2 clock Disabled. 1 = Timer2 clock Enabled.
[3]	**TMR1CKEN**	**Timer1 Clock Enable Bit** 0 = Timer1 clock Disabled. 1 = Timer1 clock Enabled.
[2]	**TMR0CKEN**	**Timer0 Clock Enable Bit** 0 = Timer0 clock Disabled. 1 = Timer0 clock Enabled.
[1]	**RTCCKEN**	**Real-time-clock APB Interface Clock Enable Bit** This bit is used to control the RTC APB clock only. The RTC peripheral clock source is selected from RTCCKSEL(RTC_LXTCTL[7]). It can be selected to 32.768 kHz external low speed crystal (LXT) or 32 kHz internal low speed RC oscillator (LIRC). 0 = RTC clock Disabled. 1 = RTC clock Enabled.
[0]	**WDTCKEN**	**Watchdog Timer Clock Enable Bit (Write Protect)** 0 = Watchdog timer and Windows watchdog timer clock Disabled. 1 = Watchdog timer and Windows watchdog timer clock Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

APB Devices Clock Enable Control Register 1 (CLK_APBCLK1)

The bits in this register are used to enable/disable clock for peripheral controller clocks.

Register	Offset	R/W	Description	Reset Value
CLK_APBCLK1	CLK_BA+0x0C	R/W	APB Devices Clock Enable Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			LCDCPCKEN	ECAP1CKEN	ECAP0CKEN	TRNGCKEN	LCDCKEN
23	22	21	20	19	18	17	16
QEI1CKEN	QEI0CKEN	Reserved		BPWM1CKEN	BPWM0CKEN	EPWM1CKEN	EPWM0CKEN
15	14	13	12	11	10	9	8
Reserved			DACCKEN	Reserved		USCI1CKEN	USCI0CKEN
7	6	5	4	3	2	1	0
Reserved	SPI3CKEN	TMR5CKEN	TMR4CKEN	Reserved	SC2CKEN	SC1CKEN	SC0CKEN

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28]	**LCDCPCKEN**	**LCD Charge Pump Clock Enable Bit** 0 = LCD charge pump clock Disabled. 1 = LCD charge pump clock Enabled.
[27]	**ECAP1CKEN**	**ECAP1 Clock Enable Bit** 0 = ECAP1 clock Disabled. 1 = ECAP1 clock Enabled.
[26]	**ECAP0CKEN**	**ECAP0 Clock Enable Bit** 0 = ECAP0 clock Disabled. 1 = ECAP0 clock Enabled.
[25]	**TRNGCKEN**	**TRNG Clock Enable Bit** 0 = TRNG clock Disabled. 1 = TRNG clock Enabled.
[24]	**LCDCKEN**	**LCD Clock Enable Bit** 0 = LCD clock Disabled. 1 = LCD clock Enabled.
[23]	**QEI1CKEN**	**QEI1 Clock Enable Bit** 0 = QEI1 clock Disabled. 1 = QEI1 clock Enabled.
[22]	**QEI0CKEN**	**QEI0 Clock Enable Bit** 0 = QEI0 clock Disabled. 1 = QEI0 clock Enabled.
[21:20]	**Reserved**	Reserved.

[19]	**BPWM1CKEN**	**BPWM1 Clock Enable Bit** 0 = BPWM1 clock Disabled. 1 = BPWM1 clock Enabled.
[18]	**BPWM0CKEN**	**BPWM0 Clock Enable Bit** 0 = BPWM0 clock Disabled. 1 = BPWM0 clock Enabled.
[17]	**EPWM1CKEN**	**EPWM1 Clock Enable Bit** 0 = EPWM1 clock Disabled. 1 = EPWM1 clock Enabled.
[16]	**EPWM0CKEN**	**EPWM0 Clock Enable Bit** 0 = EPWM0 clock Disabled. 1 = EPWM0 clock Enabled.
[15:13]	**Reserved**	Reserved.
[12]	**DACCKEN**	**DAC Clock Enable Bit** 0 = DAC clock Disabled. 1 = DAC clock Enabled.
[11:10]	**Reserved**	Reserved.
[9]	**USCI1CKEN**	**USCI1 Clock Enable Bit** 0 = USCI1 clock Disabled. 1 = USCI1 clock Enabled.
[8]	**USCI0CKEN**	**USCI0 Clock Enable Bit** 0 = USCI0 clock Disabled. 1 = USCI0 clock Enabled.
[7]	**Reserved**	Reserved.
[6]	**SPI3CKEN**	**SPI3 Clock Enable Bit** 0 = SPI3 clock Disabled. 1 = SPI3 clock Enabled.
[5]	**TMR5CKEN**	**Timer5 Clock Enable Bit** 0 = Timer5 clock Disabled. 1 = Timer5 clock Enabled.
[4]	**TMR4CKEN**	**Timer4 Clock Enable Bit** 0 = Timer4 clock Disabled. 1 = Timer4 clock Enabled.
[3]	**Reserved**	Reserved.
[2]	**SC2CKEN**	**Smart Card 2 (SC2) Clock Enable Bit** 0 = SC2 clock Disabled. 1 = SC2 clock Enabled.
[1]	**SC1CKEN**	**Smart Card 1 (SC1) Clock Enable Bit** 0 = SC1 clock Disabled. 1 = SC1 clock Enabled.
[0]	**SC0CKEN**	**Smart Card 0 (SC0) Clock Enable Bit**

		0 = SC0 clock Disabled. 1 = SC0 clock Enabled.

Clock Source Select Control Register 0 (CLK_CLKSEL0)

Register	Offset	R/W	Description	Reset Value
CLK_CLKSEL0	CLK_BA+0x10	R/W	Clock Source Select Register 0	0x0020_011X

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		SDH0SEL		Reserved			
15	14	13	12	11	10	9	8
Reserved							USBSEL
7	6	5	4	3	2	1	0
Reserved		STCLKSEL			HCLKSEL		

Bits	Description	
[31:22]	**Reserved**	Reserved.
[21:20]	**SDH0SEL**	**SDHOST0 Peripheral Clock Source Selection (Write Protect)** 00 = Clock source from HXT clock. 01 = Clock source from PLL clock. 10 = Clock source from HCLK. 11 = Clock source from HIRC clock. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[19:9]	**Reserved**	Reserved.
[8]	**USBSEL**	**USB Clock Source Selection (Write Protect)** 0 = Clock source from HIRC48. 1 = Clock source from PLL.
[7:6]	**Reserved**	Reserved.
[5:3]	**STCLKSEL**	**SysTick Clock Source Selection (Write Protect)** If SYST_CTRL[2]=0, SysTick uses listed clock source below. 000 = Clock source from HXT. 001 = Clock source from LXT. 010 = Clock source from HXT/2. 011 = Clock source from HCLK/2. 111 = Clock source from HIRC/2. Others = Reserved. **Note 1:** if SysTick clock source is not from HCLK (i.e. SYST_CTRL[2] = 0), SysTick need to enable EXSTCKEN(CLK_AHBCLK[4]) and clock frequency must less than or equal to HCLK/2. **Note 2:** These bits are write protected. Refer to the SYS_REGLCTL register.
[2:0]	**HCLKSEL**	**HCLK Clock Source Selection (Write Protect)** Before clock switching, the related clock sources (both pre-select and new-select) must be turned on.

		000 = Clock source from HXT. 001 = Clock source from LXT. 010 = Clock source from PLL. 011 = Clock source from LIRC. 100 = Reserved. 101 = Clock source from HIRC48. 110 = Clock source from MIRC. 111 = Clock source from HIRC. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.

Clock Source Select Control Register 1 (CLK_CLKSEL1)

Before clock switching, the related clock sources (pre-selected and newly-selected) must be turned on.

Register	Offset	R/W	Description	Reset Value
CLK_CLKSEL1	CLK_BA+0x14	R/W	Clock Source Select Register 1	0xA022_22B3

31	30	29	28	27	26	25	24
WWDTSEL		CLKOSEL		Reserved			
23	22	21	20	19	18	17	16
Reserved	TMR3SEL			Reserved	TMR2SEL		
15	14	13	12	11	10	9	8
Reserved	TMR1SEL			Reserved	TMR0SEL		
7	6	5	4	3	2	1	0
EWWDTSEL		EWDTSEL		LCDCPSEL	LCDSEL	WDTSEL	

Bits	Description	
[31:30]	**WWDTSEL**	**Window Watchdog Timer Clock Source Selection (Write Protect)** 10 = Clock source from HCLK/2048. 11 = Clock source from 32 kHz internal low speed RC oscillator (LIRC). Others = Reserved. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[29:28]	**CLKOSEL**	**Clock Output Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 10 = Clock source from HCLK. 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[27:23]	**Reserved**	Reserved.
[22:20]	**TMR3SEL**	**TIMER3 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 010 = Clock source from PCLK1. 011 = Clock source from external clock TM3 pin. 101 = Clock source from 32 kHz internal low speed RC oscillator (LIRC). 111 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). Others = Reserved.
[19]	**Reserved**	Reserved.
[18:16]	**TMR2SEL**	**TIMER2 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 010 = Clock source from PCLK1.

		011 = Clock source from external clock TM2 pin. 101 = Clock source from 32 kHz internal low speed RC oscillator (LIRC). 111 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). Others = Reserved.
[15]	**Reserved**	Reserved.
[14:12]	**TMR1SEL**	**TIMER1 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 010 = Clock source from PCLK0. 011 = Clock source from external clock TM1 pin. 101 = Clock source from 32 kHz internal low speed RC oscillator (LIRC). 111 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). Others = Reserved.
[11]	**Reserved**	Reserved.
[10:8]	**TMR0SEL**	**TIMER0 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 010 = Clock source from PCLK0. 011 = Clock source from external clock TM0 pin. 101 = Clock source from 32 kHz internal low speed RC oscillator (LIRC). 111 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). Others = Reserved.
[7:6]	**EWWDTSEL**	**Extra Window Watchdog Timer Clock Source Selection (Write Protect)** 10 = Clock source from HCLK/2048. 11 = Clock source from 32 kHz internal low speed RC oscillator (LIRC). Others = Reserved. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[5:4]	**EWDTSEL**	**Extra Watchdog Timer Clock Source Selection (Write Protect)** 00 = Reserved. 01 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 10 = Clock source from HCLK/2048. 11 = Clock source from 32 kHz internal low speed RC oscillator (LIRC). **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[3]	**LCDCPSEL**	**LCD Charge Pump Clock Source Selection** 0 = Clock source from 1.2 MHz internal medium speed RC oscillator (MIRC1P2M). 1 = Clock source from 4 MHz internal medium speed RC oscillator (MIRC).
[2]	**LCDSEL**	**LCD Clock Source Selection** 0 = Clock source from 32 kHz internal low speed RC oscillator (LIRC). 1 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT).
[1:0]	**WDTSEL**	**Watchdog Timer Clock Source Selection (Write Protect)** 00 = Reserved. 01 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 10 = Clock source from HCLK/2048. 11 = Clock source from 32 kHz internal low speed RC oscillator (LIRC).

		Note: These bits are write protected. Refer to the SYS_REGLCTL register.

Clock Source Select Control Register 2 (CLK_CLKSEL2)

Before clock switching, the related clock sources (pre-select and new-select) must be turned on.

Register	Offset	R/W	Description	Reset Value
CLK_CLKSEL2	CLK_BA+0x18	R/W	Clock Source Select Register 2	0x4444_2BAB

31	30	29	28	27	26	25	24
Reserved	UART3SEL			Reserved	UART2SEL		
23	22	21	20	19	18	17	16
Reserved	UART1SEL			Reserved	UART0SEL		
15	14	13	12	11	10	9	8
Reserved		SPI3SEL		SPI2SEL		BPWM1SEL	BPWM0SEL
7	6	5	4	3	2	1	0
SPI1SEL		SPI0SEL		QSPI0SEL		EPWM1SEL	EPWM0SEL

Bits	Description	
[31]	**Reserved**	Reserved.
[30:28]	**UART3SEL**	**UART3 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from PLL. 010 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 011 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). 100 = Clock source from PCLK1. Others = Reserved.
[27]	**Reserved**	Reserved.
[26:24]	**UART2SEL**	**UART2 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from PLL. 010 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 011 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). 100 = Clock source from PCLK0. Others = Reserved.
[23]	**Reserved**	Reserved.
[22:20]	**UART1SEL**	**UART1 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from PLL. 010 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 011 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). 100 = Clock source from PCLK1. Others = Reserved.

[19]	**Reserved**	Reserved.
[18:16]	**UART0SEL**	**UART0 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from PLL. 010 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 011 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). 100 = Clock source from PCLK0. Others = Reserved.
[15:14]	**Reserved**	Reserved.
[13:12]	**SPI3SEL**	**SPI3 Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL. 10 = Clock source from PCLK0. 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[11:10]	**SPI2SEL**	**SPI2 Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL. 10 = Clock source from PCLK1. 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[9]	**BPWM1SEL**	**BPWM1 Clock Source Selection (Read Only)** The peripheral clock source of BPWM1 is defined by BPWM1SEL. 1 = Clock source from PCLK1.
[8]	**BPWM0SEL**	**BPWM0 Clock Source Selection (Read Only)** The peripheral clock source of BPWM0 is defined by BPWM0SEL. 1 = Clock source from PCLK0.
[7:6]	**SPI1SEL**	**SPI1 Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL. 10 = Clock source from PCLK0. 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[5:4]	**SPI0SEL**	**SPI0 Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL. 10 = Clock source from PCLK1. 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[3:2]	**QSPI0SEL**	**QSPI0 Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL. 10 = Clock source from PCLK0. 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[1]	**EPWM1SEL**	**EPWM1 Clock Source Selection (Read Only)** The peripheral clock source of EPWM1 is defined by EPWM1SEL. 1 = Clock source from PCLK1.

[0]	**EPWM0SEL**	**EPWM0 Clock Source Selection (Read Only)** The peripheral clock source of EPWM0 is defined by EPWM0SEL. 1 = Clock source from PCLK0.

Clock Source Select Control Register 3 (CLK_CLKSEL3)

Before clock switching, the related clock sources (pre-select and new-select) must be turned on.

Register	Offset	R/W	Description	Reset Value
CLK_CLKSEL3	CLK_BA+0x1C	R/W	Clock Source Select Register 3	0x4402_222A

31	30	29	28	27	26	25	24
Reserved	**UART5SEL**			**Reserved**	**UART4SEL**		
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved						**I2S0SEL**	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved	**TMR5SEL**			**Reserved**	**TMR4SEL**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**SC2SEL**		**SC1SEL**		**SC0SEL**	

Bits	Description	
[31]	**Reserved**	Reserved.
[30:28]	**UART5SEL**	**UART5 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from PLL. 010 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 011 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). 100 = Clock source from PCLK1. Others = Reserved.
[27]	**Reserved**	Reserved.
[26:24]	**UART4SEL**	**UART4 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from PLL. 010 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 011 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). 100 = Clock source from PCLK0. Others = Reserved.
[23:18]	**Reserved**	Reserved.
[17:16]	**I2S0SEL**	**I2S0 Clock Source Selection** 00 = Clock source from HXT clock. 01 = Clock source from PLL clock. 10 = Clock source from PCLK0. 11 = Clock source from HIRC clock.
[15]	**Reserved**	Reserved.

[14:12]	**TMR5SEL**	**TIMER5 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 010 = Clock source from PCLK0. 011 = Clock source from external clock TM5 pin. 100 = Clock source from 4 MHz internal medium speed RC oscillator (MIRC). 101 = Clock source from 32 kHz internal low speed RC oscillator (LIRC). 110 = Reserved. 111 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[11]	**Reserved**	Reserved.
[10:8]	**TMR4SEL**	**TIMER4 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 010 = Clock source from PCLK0. 011 = Clock source from external clock TM4 pin. 100 = Clock source from 4 MHz internal medium speed RC oscillator (MIRC). 101 = Clock source from 32 kHz internal low speed RC oscillator (LIRC). 110 = Reserved. 111 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[7:6]	**Reserved**	Reserved.
[5:4]	**SC2SEL**	**Smart Card 2 (SC2) Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL. 10 = Clock source from PCLK0. 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[3:2]	**SC1SEL**	**Smart Card 1 (SC1) Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL. 10 = Clock source from PCLK1. 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[1:0]	**SC0SEL**	**Smart Card 0 (SC0) Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL. 10 = Clock source from PCLK0. 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).

Clock Divider Number Register 0 (CLK_CLKDIV0)

Register	Offset	R/W	Description	Reset Value
CLK_CLKDIV0	CLK_BA+0x20	R/W	Clock Divider Number Register 0	0x0000_000X

31	30	29	28	27	26	25	24
SDH0DIV							
23	22	21	20	19	18	17	16
EADCDIV							
15	14	13	12	11	10	9	8
UART1DIV				UART0DIV			
7	6	5	4	3	2	1	0
USBDIV				HCLKDIV			

Bits	Description	
[31:24]	SDH0DIV	**SDHOST0 Clock Divide Number From SDHOST0 Clock Source** SDHOST0 clock frequency = (SDHOST0 clock source frequency) / (SDH0DIV + 1).
[23:16]	EADCDIV	**EADC Clock Divide Number From EADC Clock Source** EADC clock frequency = (EADC clock source frequency) / (EADCDIV + 1).
[15:12]	UART1DIV	**UART1 Clock Divide Number From UART1 Clock Source** UART1 clock frequency = (UART1 clock source frequency) / (UART1DIV + 1).
[11:8]	UART0DIV	**UART0 Clock Divide Number From UART0 Clock Source** UART0 clock frequency = (UART0 clock source frequency) / (UART0DIV + 1).
[7:4]	USBDIV	**USB Clock Divide Number From USB Clock Source** USB clock frequency = (USB clock source frequency) / (USBDIV + 1).
[3:0]	HCLKDIV	**HCLK Clock Divide Number From HCLK Clock Source** HCLK clock frequency = (HCLK clock source frequency) / (HCLKDIV + 1).

Clock Divider Number Register 1 (CLK_CLKDIV1)

Register	Offset	R/W	Description	Reset Value
CLK_CLKDIV1	CLK_BA+0x24	R/W	Clock Divider Number Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SC2DIV							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SC1DIV							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SC0DIV							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:16]	**SC2DIV**	**Smart Card 2 (SC2) Clock Divide Number From SC2 Clock Source** SC2 clock frequency = (SC2 clock source frequency) / (SC2DIV + 1).
[15:8]	**SC1DIV**	**Smart Card 1 (SC1) Clock Divide Number From SC1 Clock Source** SC1 clock frequency = (SC1 clock source frequency) / (SC1DIV + 1).
[7:0]	**SC0DIV**	**Smart Card 0 (SC0) Clock Divide Number From SC0 Clock Source** SC0 clock frequency = (SC0 clock source frequency) / (SC0DIV + 1).

Clock Divider Number Register 4 (CLK_CLKDIV4)

Register	Offset	R/W	Description	Reset Value
CLK_CLKDIV4	CLK_BA+0x30	R/W	Clock Divider Number Register 4	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
UART5DIV				**UART4DIV**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UART3DIV				**UART2DIV**			

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:12]	**UART5DIV**	**UART5 Clock Divide Number From UART5 Clock Source** UART5 clock frequency = (UART5 clock source frequency) / (UART5DIV + 1).
[11:8]	**UART4DIV**	**UART4 Clock Divide Number From UART4 Clock Source** UART4 clock frequency = (UART4 clock source frequency) / (UART4DIV + 1).
[7:4]	**UART3DIV**	**UART3 Clock Divide Number From UART3 Clock Source** UART3 clock frequency = (UART3 clock source frequency) / (UART3DIV + 1).
[3:0]	**UART2DIV**	**UART2 Clock Divide Number From UART2 Clock Source** UART2 clock frequency = (UART2 clock source frequency) / (UART2DIV + 1).

PLL Control Register (CLK_PLLCTL)

The PLL reference clock input is from the 4~24 MHz external high speed crystal oscillator (HXT) clock input or from the 12 MHz internal high speed RC oscillator (HIRC). This register is used to control the PLL output frequency and PLL operation mode.

Programming these bits needs to write “59h”, “16h” and “88h” to address 0x4000_0100 to disable register protection. Refer to the register SYS_REGLCTL at address SYS_BA+0x100.

CLK_PLLCTL only can be modified while PD(CLK_PLLCTL[16]) is set.

Register	Offset	R/W	Description	Reset Value
CLK_PLLCTL	CLK_BA+0x40	R/W	PLL Control Register	0x0009_440A

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
STBSEL	Reserved			PLLSRC	Reserved	BP	PD
15	14	13	12	11	10	9	8
OUTDIV		INDIV					FBDIV
7	6	5	4	3	2	1	0
FBDIV							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23]	**STBSEL**	**PLL Stable Counter Selection (Write Protect)** 0 = PLL stable time is 1200 PLL source clock (suitable for source clock is equal to or less than 12 MHz). 1 = PLL stable time is 2400 PLL source clock (suitable for source clock is larger than 12 MHz). **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[22:20]	**Reserved**	Reserved.
[19]	**PLLSRC**	**PLL Source Clock Selection (Write Protect)** 0 = PLL source clock from 4~24 MHz external high-speed crystal oscillator (HXT). 1 = PLL source clock from 12 MHz internal high-speed oscillator (HIRC). **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[18]	**Reserved**	Reserved.
[17]	**BP**	**PLL Bypass Control (Write Protect)** 0 = PLL is in normal mode (default). 1 = PLL clock output is same as PLL input clock FIN. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[16]	**PD**	**Power-down Mode (Write Protect)** 0 = PLL is enable (in normal mode). 1 = PLL is disable (in Power-down mode) (default).

		Note 1: If set the PDEN bit to 1 in CLK_PWRCTL register, the PLL will enter Power-down mode, too. **Note 2:** This bit is write protected. Refer to the SYS_REGLCTL register.
[15:14]	**OUTDIV**	**PLL Output Divider Control (Write Protect)** Refer to the formulas below the table. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[13:9]	**INDIV**	**PLL Input Divider Control (Write Protect)** Refer to the formulas below the table. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[8:0]	**FBDIV**	**PLL Feedback Divider Control (Write Protect)** Refer to the formulas below the table. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.

Output Clock Frequency formula:

$$FREF = FIN \times \frac{1}{NR}$$

$$FVCO = FIN \times \frac{2 \times NF}{NR}$$

$$FOUT = FIN \times \frac{2 \times NF}{NR} \times \frac{1}{NO}$$

where FREF is the comparison frequency for the PFD (phase frequency detector).

For proper operation in normal mode, the following constraints must be satisfied:

2 MHz ≤ FREF ≤ 8 MHz

96 MHz ≤ FVCO ≤ 200 MHz

24 MHz ≤ FOUT ≤ 200 MHz

Symbol	Description
FOUT	Output Clock Frequency
FIN	Input (Reference) Clock Frequency
NR	Input Divider (INDIV + 1)
NF	Feedback Divider (FBDIV + 2)
NO	OUTDIV = “00” : NO = 1 OUTDIV = “01” : NO = 2 OUTDIV = “10” : NO = 2 OUTDIV = “11” : NO = 4

Table 6.3-3 Symbol Definition of PLL Output Frequency Formula

For example:

FOUT	FIN	PLLCTL[15:0]
48 MHz	12 MHz	0x440A
84 MHz	12 MHz	0x420C
96 MHz	12 MHz	0x040A
144 MHz	12 MHz	0x020A
168 MHz	12 MHz	0x020C
192 MHz	12 MHz	0x020E

Clock Status Monitor Register (CLK_STATUS)

The bits in this register are used to monitor if the chip clock source is stable or not, and whether the clock switch is failed.

Register	Offset	R/W	Description	Reset Value
CLK_STATUS	CLK_BA+0x50	R	Clock Status Monitor Register	0x0000_00X8

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						LIRC32STB	EXTLXTSTB
7	6	5	4	3	2	1	0
CLKSFAIL	HIRC48STB	MIRCSTB	HIRCSTB	LIRCSTB	PLLSTB	LXTSTB	HXTSTB

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**LIRC32STB**	**LIRC32 Clock Source Stable Flag (Read Only)** 0 = 32 kHz internal low speed RC oscillator (LIRC32) clock is not stable or disabled. 1 = 32 kHz internal low speed RC oscillator (LIRC32) clock is stable and enabled.
[8]	**EXTLXTSTB**	**EXTLXT Clock Source Stable Flag (Read Only)** 0 = 32.768 kHz external low speed crystal oscillator (extLXT) clock is not stable or disabled. 1 = 32.768 kHz external low speed crystal oscillator (extLXT) clock is stable and enabled.
[7]	**CLKSFAIL**	**Clock Switching Fail Flag (Read Only)** This bit is updated when software switches system clock source. If switch target clock is stable, this bit will be set to 0. If switch target clock is not stable, this bit will be set to 1. 0 = Clock switching success. 1 = Clock switching failure. **Note:** This bit is read only. After selected clock source is stable, hardware will switch system clock to selected clock automatically, and CLKSFAIL will be cleared automatically by hardware.
[6]	**HIRC48STB**	**HIRC48 Clock Source Stable Flag (Read Only)** 0 = 48 MHz internal high speed RC oscillator (HIRC48) clock is not stable or disabled. 1 = 48 MHz internal high speed RC oscillator (HIRC48) clock is stable and enabled.
[5]	**MIRCSTB**	**MIRC Clock Source Stable Flag (Read Only)** 0 = 4 MHz internal medium speed RC oscillator (MIRC) clock is not stable or disabled. 1 = 4 MHz internal medium speed RC oscillator (MIRC) clock is stable and enabled.
[4]	**HIRCSTB**	**HIRC Clock Source Stable Flag (Read Only)** 0 = 12 MHz internal high speed RC oscillator (HIRC) clock is not stable or disabled. 1 = 12 MHz internal high speed RC oscillator (HIRC) clock is stable and enabled.

[3]	**LIRCSTB**	**LIRC Clock Source Stable Flag (Read Only)** 0 = 32 kHz internal low speed RC oscillator (LIRC) clock is not stable or disabled. 1 = 32 kHz internal low speed RC oscillator (LIRC) clock is stable and enabled.
[2]	**PLLSTB**	**Internal PLL Clock Source Stable Flag (Read Only)** 0 = Internal PLL clock is not stable or disabled. 1 = Internal PLL clock is stable and enabled.
[1]	**LXTSTB**	**LXT Clock Source Stable Flag (Read Only)** LXT clock source can be selected as extLXT or LIRC32 by setting C32KSEL(RTC_LXTCTL[6]). If C32KSEL is set to 0 the LXT stable flag is set when extLXT clock source is stable. If C32KSEL is set to 1 the LXT stable flag is set when LIRC32 clock source is stable. 0 = 32.768 kHz external low speed crystal oscillator (LXT) clock is not stable or disabled. 1 = 32.768 kHz external low speed crystal oscillator (LXT) clock is stabled and enabled.
[0]	**HXTSTB**	**HXT Clock Source Stable Flag (Read Only)** 0 = 4~24 MHz external high speed crystal oscillator (HXT) clock is not stable or disabled. 1 = 4~24 MHz external high speed crystal oscillator (HXT) clock is stable and enabled.

Clock Output Control Register (CLK_CLKOCTL)

Register	Offset	R/W	Description	Reset Value
CLK_CLKOCTL	CLK_BA+0x60	R/W	Clock Output Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved	CLK1HZEN	DIV1EN	CLKOEN	FREQSEL			

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**CLK1HZEN**	**Clock Output 1Hz Enable Bit** 0 = 1 Hz clock output for 32.768 kHz frequency compensation Disabled. 1 = 1 Hz clock output for 32.768 kHz frequency compensation Enabled.
[5]	**DIV1EN**	**Clock Output Divide One Enable Bit** 0 = Clock Output will output clock with source frequency divided by FREQSEL. 1 = Clock Output will output clock with source frequency.
[4]	**CLKOEN**	**Clock Output Enable Bit** 0 = Clock Output function Disabled. 1 = Clock Output function Enabled.
[3:0]	**FREQSEL**	**Clock Output Frequency Selection** The formula of output frequency is $\mathbf{F_{out} = F_{in}/2^{(N+1)}}$. F_{in} is the input clock frequency. F_{out} is the frequency of divider output clock. N is the 4-bit value of FREQSEL[3:0].

Clock Fail Detector Control Register (CLK_CLKDCTL)

Register	Offset	R/W	Description	Reset Value
CLK_CLKDCTL	CLK_BA+0x70	R/W	Clock Fail Detector Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						HXTFQIEN	HXTFQDEN
15	14	13	12	11	10	9	8
Reserved		LXTFIEN	LXTFDEN	Reserved			
7	6	5	4	3	2	1	0
Reserved	HXTFDSEL	HXTFIEN	HXTFDEN	Reserved			

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**HXTFQIEN**	**HXT Clock Frequency Monitor Interrupt Enable Bit** 0 = 4~24 MHz external high speed crystal oscillator (HXT) clock frequency monitor fail interrupt Disabled. 1 = 4~24 MHz external high speed crystal oscillator (HXT) clock frequency monitor fail interrupt Enabled.
[16]	**HXTFQDEN**	**HXT Clock Frequency Monitor Enable Bit** 0 = 4~24 MHz external high speed crystal oscillator (HXT) clock frequency monitor Disabled. 1 = 4~24 MHz external high speed crystal oscillator (HXT) clock frequency monitor Enabled.
[15:14]	**Reserved**	Reserved.
[13]	**LXTFIEN**	**LXT Clock Fail Interrupt Enable Bit** 0 = 32.768 kHz external low speed crystal oscillator (LXT) clock fail interrupt Disabled. 1 = 32.768 kHz external low speed crystal oscillator (LXT) clock fail interrupt Enabled.
[12]	**LXTFDEN**	**LXT Clock Fail Detector Enable Bit** 0 = 32.768 kHz external low speed crystal oscillator (LXT) clock fail detector Disabled. 1 = 32.768 kHz external low speed crystal oscillator (LXT) clock fail detector Enabled.
[11:7]	**Reserved**	Reserved.
[6]	**HXTFDSEL**	**HXT Clock Fail Detector Selection** 0 = 4~24 MHz external high speed crystal oscillator (HXT) clock fail detector after HXT stable. 1 = 4~24 MHz external high speed crystal oscillator (HXT) clock fail detector bypass HXT stable. **Note:** When HXT Clock Fail Detector Selection is set, detector will keep detect whether HXT is stable or not, prevent HXT fail before stable.
[5]	**HXTFIEN**	**HXT Clock Fail Interrupt Enable Bit** 0 = 4~24 MHz external high speed crystal oscillator (HXT) clock fail interrupt Disabled. 1 = 4~24 MHz external high speed crystal oscillator (HXT) clock fail interrupt Enabled.
[4]	**HXTFDEN**	**HXT Clock Fail Detector Enable Bit**

		0 = 4~24 MHz external high speed crystal oscillator (HXT) clock fail detector Disabled. 1 = 4~24 MHz external high speed crystal oscillator (HXT) clock fail detector Enabled.
[3:0]	**Reserved**	Reserved.

Clock Fail Detector Status Register (CLK_CLKDSTS)

Register	Offset	R/W	Description	Reset Value
CLK_CLKDSTS	CLK_BA+0x74	R/W	Clock Fail Detector Status Register	0x0000_0000

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

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**HXTFQIF**	**HXT Clock Frequency Monitor Interrupt Flag (Write Protect)** 0 = 4~24 MHz external high speed crystal oscillator (HXT) clock is normal. 1 = 4~24 MHz external high speed crystal oscillator (HXT) clock frequency is abnormal. **Note 1:** Write 1 to clear the bit to 0. **Note 2:** This bit is write protected. Refer to the SYS_REGLCTL register.
[7:2]	**Reserved**	Reserved.
[1]	**LXTFIF**	**LXT Clock Fail Interrupt Flag (Write Protect)** 0 = 32.768 kHz external low speed crystal oscillator (LXT) clock is normal. 1 = 32.768 kHz external low speed crystal oscillator (LXT) stops. **Note 1:** Write 1 to clear the bit to 0. **Note 2:** This bit is write protected. Refer to the SYS_REGLCTL register.
[0]	**HXTFIF**	**HXT Clock Fail Interrupt Flag (Write Protect)** 0 = 4~24 MHz external high speed crystal oscillator (HXT) clock is normal. 1 = 4~24 MHz external high speed crystal oscillator (HXT) clock stops. **Note 1:** Write 1 to clear the bit to 0. **Note 2:** This bit is write protected. Refer to the SYS_REGLCTL register.

Clock Frequency Detector Upper Boundary Register (CLK_CDUPB)

Register	Offset	R/W	Description	Reset Value
CLK_CDUPB	CLK_BA+0x78	R/W	Clock Frequency Detector Upper Boundary Register	0x0000_0000

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

Bits	Description	
[31:10]	Reserved	Reserved.
[9:0]	UPERBD	**HXT Clock Frequency Detector Upper Boundary** The bits define the high value of frequency monitor window. When HXT frequency monitor value higher than this register, the HXT frequency detect fail interrupt flag will set to 1.

Clock Frequency Detector Lower Boundary Register (CLK_CDLOWB)

Register	Offset	R/W	Description	Reset Value
CLK_CDLOWB	CLK_BA+0x7C	R/W	Clock Frequency Detector Lower Boundary Register	0x0000_0000

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

Bits	Description	
[31:10]	Reserved	Reserved.
[9:0]	LOWERBD	**HXT Clock Frequency Detector Lower Boundary** The bits define the low value of frequency monitor window. When HXT frequency monitor value lower than this register, the HXT frequency detect fail interrupt flag will set to 1.

Power Manager Control Register (CLK_PMUCTL)

Register	Offset	R/W	Description	Reset Value
CLK_PMUCTL	CLK_BA+0x90	R/W	Power Manager Control Register	0x0000_0010

31	30	29	28	27	26	25	24
WKPINEN4		WKPINEN3		WKPINEN2		WKPINEN1	
23	22	21	20	19	18	17	16
RTCWKEN	Reserved			TAMPERWK	ACMPSPWK	WKPINEN0	
15	14	13	12	11	10	9	8
Reserved				WKTMRIS			WKTMREN
7	6	5	4	3	2	1	0
WRBUSY	Reserved		VDROPEN	Reserved	PDMSEL		

Bits	Description	
[31:30]	**WKPINEN4**	**Wake-up Pin 4 Enable Bits (Write Protect)** This is control register for GPF.6 to wake-up pin. 00 = Wake-up pin disable at Deep Power-down mode. 01 = Wake-up pin rising edge enabled at Deep Power-down mode. 10 = Wake-up pin falling edge enabled at Deep Power-down mode. 11 = Wake-up pin both edge enabled at Deep Power-down mode. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[29:28]	**WKPINEN3**	**Wake-up Pin 3 Enable Bits (Write Protect)** This is control register for GPB.12 to wake-up pin. 00 = Wake-up pin disable at Deep Power-down mode. 01 = Wake-up pin rising edge enabled at Deep Power-down mode. 10 = Wake-up pin falling edge enabled at Deep Power-down mode. 11 = Wake-up pin both edge enabled at Deep Power-down mode. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[27:26]	**WKPINEN2**	**Wake-up Pin 2 Enable Bits (Write Protect)** This is control register for GPB.2 to wake-up pin. 00 = Wake-up pin disable at Deep Power-down mode. 01 = Wake-up pin rising edge enabled at Deep Power-down mode. 10 = Wake-up pin falling edge enabled at Deep Power-down mode. 11 = Wake-up pin both edge enabled at Deep Power-down mode. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[25:24]	**WKPINEN1**	**Wake-up Pin 1 Enable Bits (Write Protect)** This is control register for GPB.0 to wake-up pin. 00 = Wake-up pin disable at Deep Power-down mode. 01 = Wake-up pin rising edge enabled at Deep Power-down mode. 10 = Wake-up pin falling edge enabled at Deep Power-down mode.

		11 = Wake-up pin both edge enabled at Deep Power-down mode. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[23]	**RTCWKEN**	**RTC Wake-up Enable Bit (Write Protect)** 0 = RTC wake-up disable at Deep Power-down mode or Standby Power-down mode. 1 = RTC wake-up enabled at Deep Power-down mode or Standby Power-down mode. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[22:20]	**Reserved**	Reserved.
[19]	**TAMPERWK**	**Tamper Standby Power-down Mode Wake-up Enable Bit (Write Protect)** 0 = Tamper wake-up disable at Standby Power-down mode. 1 = Tamper wake-up enabled at Standby Power-down mode. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[18]	**ACMPSPWK**	**ACMP Standby Power-down Mode Wake-up Enable Bit (Write Protect)** 0 = ACMP wake-up disable at Standby Power-down mode. 1 = ACMP wake-up enabled at Standby Power-down mode. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
`[17:16]	**WKPINEN0**	**Wake-up Pin 0 Enable Bits (Write Protect)** This is control register for GPC.0 to wake-up pin. 00 = Wake-up pin disable at Deep Power-down mode. 01 = Wake-up pin rising edge enabled at Deep Power-down mode. 10 = Wake-up pin falling edge enabled at Deep Power-down mode. 11 = Wake-up pin both edge enabled at Deep Power-down mode. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[15:12]	**Reserved**	Reserved.
[11:9]	**WKTMRIS**	**Wake-up Timer Time-out Interval Select Bits (Write Protect)** These bits control wake-up timer time-out interval when chip under Deep Power-down mode or Standby Power-down mode. 000 = Time-out interval is 410 LIRC clocks (12.8ms). 001 = Time-out interval is 819 LIRC clocks (25.6ms). 010 = Time-out interval is 1638 LIRC clocks (51.2ms). 011 = Time-out interval is 3277 LIRC clocks (102.4ms). 100 = Time-out interval is 13107 LIRC clocks (409.6ms). 101 = Time-out interval is 26214 LIRC clocks (819.2ms). 110 = Time-out interval is 52429 LIRC clocks (1638.4ms). 111 = Time-out interval is 209715 LIRC clocks (6553.6ms). **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[8]	**WKTMREN**	**Wake-up Timer Enable Bit (Write Protect)** 0 = Wake-up timer disable at Deep Power-down mode or Standby Power-down mode. 1 = Wake-up timer enabled at Deep Power-down mode or Standby Power-down mode. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[7]	**WRBUSY**	**Write Busy Flag (Read Only)** If CLK_PMUCTL be written, this bit be asserted automatic by hardware, and be de-asserted when write procedure finish. 0 = CLK_PMUCTL write ready. 1 = CLK_PMUCTL write ignore.

[6:5]	**Reserved**	Reserved.
[4]	**VDROPEN**	**Standby Power-down mode Regulator Output Voltage Drop Enable Bit (Write Protect)** If this bit be asserted, regulator output voltage drop to 0.9v when SPD mode. 0 = Regulator voltage auto drop function Disabled. 1 = Regulator voltage auto drop function Enabled. (default) **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[3]	**Reserved**	Reserved.
[2:0]	**PDMSEL**	**Power-down Mode Selection (Write Protect)** These bits control chip power-down mode grade selection when CPU executes WFI/WFE instruction. 000 = Power-down mode is selected (PD). 001 = Low leakage Power-down mode is selected (LLPD). 010 = Fast wake-up Power-down (FWPD). 011 = Ultra low leakage Power-down mode is selected (ULLPD). 100 = Standby Power-down mode is selected (SPD). 101 = Reserved. 110 = Deep Power-down mode is selected (DPD). 111 = Reserved. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.

Power Manager Status Register (CLK_PMUSTS)

Register	Offset	R/W	Description	Reset Value
CLK_PMUSTS	CLK_BA+0x94	R/W	Power Manager Status Register	0x0000_0000

31	30	29	28	27	26	25	24
CLRWK	Reserved						
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
TAMPERWK	ACMPWK	BODWK	LVRWK	GPDWK	GPCWK	GPBWK	GPAWK
7	6	5	4	3	2	1	0
Reserved	PINWK4	PINWK3	PINWK2	PINWK1	RTCWK	TMRWK	PINWK0

Bits	Description	
[31]	**CLRWK**	**Clear Wake-up Flag** 0 = No clear. 1 = Clear all of wake-up flag. **Note:** This bit is auto cleared by hardware.
[30:16]	**Reserved**	Reserved.
[15]	**TAMPERWK**	**Tamper Wake-up Flag (Read Only)** This flag indicates that wakeup of device from Standby Power-down mode (SPD) was requested with a Tamper event occurred. This flag is cleared when SPD mode is entered.
[14]	**ACMPWK**	**ACMP Wake-up Flag (Read Only)** This flag indicates that wakeup of device from Standby Power-down mode (SPD) was requested with a ACMP transition. This flag is cleared when SPD mode is entered.
[13]	**BODWK**	**BOD Wake-up Flag (Read Only)** This flag indicates that wakeup of device from Standby Power-down mode (SPD) was requested with a BOD happened. This flag is cleared when SPD mode is entered.
[12]	**LVRWK**	**LVR Wake-up Flag (Read Only)** This flag indicates that wakeup of device from Standby Power-down mode (SPD) was requested with a LVR happened. This flag is cleared when SPD mode is entered.
[11]	**GPDWK**	**GPD Wake-up Flag (Read Only)** This flag indicates that wake-up of chip from Standby Power-down mode (SPD) was requested by a transition of selected one GPD group pins. This flag is cleared when SPD mode is entered.
[10]	**GPCWK**	**GPC Wake-up Flag (Read Only)** This flag indicates that wake-up of chip from Standby Power-down mode (SPD) was requested by a transition of selected one GPC group pins. This flag is cleared when SPD mode is entered.
[9]	**GPBWK**	**GPB Wake-up Flag (Read Only)** This flag indicates that wake-up of chip from Standby Power-down mode (SPD) was requested by a transition of selected one GPB group pins. This flag is cleared when SPD mode is entered.

[8]	**GPAWK**	**GPA Wake-up Flag (Read Only)** This flag indicates that wake-up of chip from Standby Power-down mode (SPD) was requested by a transition of selected one GPA group pins. This flag is cleared when SPD mode is entered.
[7]	**Reserved**	Reserved.
[6]	**PINWK4**	**Pin 4 Wake-up Flag (Read Only)** This flag indicates that wake-up of chip from Deep Power-down mode (DPD) was requested by a transition of the Wake-up pin (GPF.6). This flag is cleared when DPD mode is entered.
[5]	**PINWK3**	**Pin 3 Wake-up Flag (Read Only)** This flag indicates that wake-up of chip from Deep Power-down mode (DPD) was requested by a transition of the Wake-up pin (GPB.12). This flag is cleared when DPD mode is entered.
[4]	**PINWK2**	**Pin 2 Wake-up Flag (Read Only)** This flag indicates that wake-up of chip from Deep Power-down mode (DPD) was requested by a transition of the Wake-up pin (GPB.2). This flag is cleared when DPD mode is entered.
[3]	**PINWK1**	**Pin 1 Wake-up Flag (Read Only)** This flag indicates that wake-up of chip from Deep Power-down mode (DPD) was requested by a transition of the Wake-up pin (GPB.0). This flag is cleared when DPD mode is entered.
[2]	**RTCWK**	**RTC Wake-up Flag (Read Only)** This flag indicates that wakeup of device from Deep Power-down mode (DPD) or Standby Power-down (SPD) mode was requested with a RTC alarm, tick time or tamper happened. This flag is cleared when DPD or SPD mode is entered.
[1]	**TMRWK**	**Timer Wake-up Flag (Read Only)** This flag indicates that wake-up of chip from Deep Power-down mode (DPD) or Standby Power-down (SPD) mode was requested by wakeup timer time-out. This flag is cleared when DPD or SPD mode is entered.
[0]	**PINWK0**	**Pin 0 Wake-up Flag (Read Only)** This flag indicates that wake-up of chip from Deep Power-down mode (DPD) was requested by a transition of the Wake-up pin (GPC.0). This flag is cleared when DPD mode is entered.

Standby Power-down Wake-up De-bounce Control Register (CLK_SWKDBCTL)

Register	Offset	R/W	Description	Reset Value
CLK_SWKDBCTL	CLK_BA+0x9C	R/W	Standby Power-down Wake-up De-bounce Control Register	0x0000_0000

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

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3:0]	**SWKDBCLKSEL**	**Standby Power-down Wake-up De-bounce Sampling Cycle Selection** 0000 = Sample wake-up input once per 1 clocks. 0001 = Sample wake-up input once per 2 clocks. 0010 = Sample wake-up input once per 4 clocks. 0011 = Sample wake-up input once per 8 clocks. 0100 = Sample wake-up input once per 16 clocks. 0101 = Sample wake-up input once per 32 clocks. 0110 = Sample wake-up input once per 64 clocks. 0111 = Sample wake-up input once per 128 clocks. 1000 = Sample wake-up input once per 256 clocks. 1001 = Sample wake-up input once per 2*256 clocks. 1010 = Sample wake-up input once per 4*256 clocks. 1011 = Sample wake-up input once per 8*256 clocks. 1100 = Sample wake-up input once per 16*256 clocks. 1101 = Sample wake-up input once per 32*256 clocks. 1110 = Sample wake-up input once per 64*256 clocks. 1111 = Sample wake-up input once per 128*256 clocks. **Note:** De-bounce counter clock source is the 32 kHz internal low speed RC oscillator (LIRC).

GPA Standby Power-down Wake-up Control Register (CLK_PASWKCTL)

Register	Offset	R/W	Description	Reset Value
CLK_PASWKCTL	CLK_BA+0xA0	R/W	GPA Standby Power-down Wake-up Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							DBEN
7	6	5	4	3	2	1	0
WKPSEL				Reserved	PFWKEN	PRWKEN	WKEN

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**DBEN**	**GPA Input Signal De-bounce Enable Bit** The DBEN bit is used to enable the de-bounce function for each corresponding I/O. If the input signal pulse width cannot be sampled by continuous two de-bounce sample cycle, the input signal transition is seen as the signal bounce and will not trigger the wakeup. The de-bounce clock source is the 32 kHz internal low speed RC oscillator (LIRC). 0 = Standby power-down wake-up pin De-bounce function Disabled. 1 = Standby power-down wake-up pin De-bounce function Enabled. The de-bounce function is valid only for edge triggered.
[7:4]	**WKPSEL**	**GPA Standby Power-down Wake-up Pin Select** 0000 = Reserved. 0001 = Reserved. 0010 = Reserved. 0011 = Reserved. 0100 = Reserved. 0101 = Reserved. 0110 = GPA.6 wake-up function Enabled. 0111 = GPA.7 wake-up function Enabled. 1000 = GPA.8 wake-up function Enabled. 1001 = GPA.9 wake-up function Enabled. 1010 = GPA.10 wake-up function Enabled. 1011 = GPA.11 wake-up function Enabled. 1100 = GPA.12 wake-up function Enabled. 1101 = GPA.13 wake-up function Enabled. 1110 = GPA.14 wake-up function Enabled. 1111 = GPA.15 wake-up function Enabled.
[3]	**Reserved**	Reserved.

[2]	**PFWKEN**	**Pin Falling Edge Wake-up Enable Bit** 0 = GPA group pin falling edge wake-up function Disabled. 1 = GPA group pin falling edge wake-up function Enabled.
[1]	**PRWKEN**	**Pin Rising Edge Wake-up Enable Bit** 0 = GPA group pin rising edge wake-up function Disabled. 1 = GPA group pin rising edge wake-up function Enabled.
[0]	**WKEN**	**Standby Power-down Pin Wake-up Enable Bit** 0 = GPA group pin wake-up function Disabled. 1 = GPA group pin wake-up function Enabled.

GPB Standby Power-down Wake-up Control Register (CLK_PBSWKCTL)

Register	Offset	R/W	Description	Reset Value
CLK_PBSWKCTL	CLK_BA+0xA4	R/W	GPB Standby Power-down Wake-up Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							DBEN
7	6	5	4	3	2	1	0
WKPSEL				Reserved	PFWKEN	PRWKEN	WKEN

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**DBEN**	**GPB Input Signal De-bounce Enable Bit** The DBEN bit is used to enable the de-bounce function for each corresponding I/O. If the input signal pulse width cannot be sampled by continuous two de-bounce sample cycle, the input signal transition is seen as the signal bounce and will not trigger the wakeup. The de-bounce clock source is the 32 kHz internal low speed RC oscillator (LIRC). 0 = Standby power-down wake-up pin De-bounce function Disabled. 1 = Standby power-down wake-up pin De-bounce function Enabled. **Note:** The de-bounce function is valid only for edge triggered.
[7:4]	**WKPSEL**	**GPB Standby Power-down Wake-up Pin Select** 0000 = GPB.0 wake-up function Enabled. 0001 = GPB.1 wake-up function Enabled. 0010 = GPB.2 wake-up function Enabled. 0011 = GPB.3 wake-up function Enabled. 0100 = GPB.4 wake-up function Enabled. 0101 = GPB.5 wake-up function Enabled. 0110 = GPB.6 wake-up function Enabled. 0111 = GPB.7 wake-up function Enabled. 1000 = GPB.8 wake-up function Enabled. 1001 = GPB.9 wake-up function Enabled. 1010 = GPB.10 wake-up function Enabled. 1011 = GPB.11 wake-up function Enabled. 1100 = GPB.12 wake-up function Enabled. 1101 = GPB.13 wake-up function Enabled. 1110 = GPB.14 wake-up function Enabled. 1111 = GPB.15 wake-up function Enabled.
[3]	**Reserved**	Reserved.

[2]	**PFWKEN**	**Pin Falling Edge Wake-up Enable Bit** 0 = GPB group pin falling edge wake-up function Disabled. 1 = GPB group pin falling edge wake-up function Enabled.
[1]	**PRWKEN**	**Pin Rising Edge Wake-up Enable Bit** 0 = GPB group pin rising edge wake-up function Disabled. 1 = GPB group pin rising edge wake-up function Enabled.
[0]	**WKEN**	**Standby Power-down Pin Wake-up Enable Bit** 0 = GPB group pin wake-up function Disabled. 1 = GPB group pin wake-up function Enabled.

GPC Standby Power-down Wake-up Control Register (CLK_PCSWKCTL)

Register	Offset	R/W	Description	Reset Value
CLK_PCSWKCTL	CLK_BA+0xA8	R/W	GPC Standby Power-down Wake-up Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							DBEN
7	6	5	4	3	2	1	0
WKPSEL				Reserved	PFWKEN	PRWKEN	WKEN

Bits	Description	
[31:9]	Reserved	Reserved.
[8]	DBEN	**GPC Input Signal De-bounce Enable Bit** The DBEN bit is used to enable the de-bounce function for each corresponding I/O. If the input signal pulse width cannot be sampled by continuous two de-bounce sample cycle, the input signal transition is seen as the signal bounce and will not trigger the wakeup. The de-bounce clock source is the 32 kHz internal low speed RC oscillator. 0 = Standby power-down wake-up pin De-bounce function Disabled. 1 = Standby power-down wake-up pin De-bounce function Enabled. The de-bounce function is valid only for edge triggered.
[7:4]	WKPSEL	**GPC Standby Power-down Wake-up Pin Select** 0000 = GPC.0 wake-up function Enabled. 0001 = GPC.1 wake-up function Enabled. 0010 = GPC.2 wake-up function Enabled. 0011 = GPC.3 wake-up function Enabled. 0100 = GPC.4 wake-up function Enabled. 0101 = GPC.5 wake-up function Enabled. 0110 = GPC.6 wake-up function Enabled. 0111 = GPC.7 wake-up function Enabled. 1000 = GPC.8 wake-up function Enabled. 1001 = GPC.9 wake-up function Enabled. 1010 = GPC.10 wake-up function Enabled. 1011 = GPC.11 wake-up function Enabled. 1100 = GPC.12 wake-up function Enabled. 1101 = GPC.13 wake-up function Enabled. 1110 = Reserved. 1111 = Reserved.
[3]	Reserved	Reserved.

[2]	**PFWKEN**	**Pin Falling Edge Wake-up Enable Bit** 0 = GPC group pin falling edge wake-up function Disabled. 1 = GPC group pin falling edge wake-up function Enabled.
[1]	**PRWKEN**	**Pin Rising Edge Wake-up Enable Bit** 0 = GPC group pin rising edge wake-up function Disabled. 1 = GPC group pin rising edge wake-up function Enabled.
[0]	**WKEN**	**Standby Power-down Pin Wake-up Enable Bit** 0 = GPC group pin wake-up function Disabled. 1 = GPC group pin wake-up function Enabled.

GPD Standby Power-down Wake-up Control Register (CLK_PDSWKCTL)

Register	Offset	R/W	Description	Reset Value
CLK_PDSWKCTL	CLK_BA+0xAC	R/W	GPD Standby Power-down Wake-up Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							DBEN
7	6	5	4	3	2	1	0
WKPSEL				Reserved	PFWKEN	PRWKEN	WKEN

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**DBEN**	**GPD Input Signal De-bounce Enable Bit** The DBEN bit is used to enable the de-bounce function for each corresponding I/O. If the input signal pulse width cannot be sampled by continuous two de-bounce sample cycle, the input signal transition is seen as the signal bounce and will not trigger the wakeup. The de-bounce clock source is the 32 kHz internal low speed RC oscillator. 0 = Standby power-down wake-up pin De-bounce function Disable. 1 = Standby power-down wake-up pin De-bounce function Enable. **Note:** The de-bounce function is valid only for edge triggered.
[7:4]	**WKPSEL**	**GPD Standby Power-down Wake-up Pin Select** 0000 = GPD.0 wake-up function Enabled. 0001 = GPD.1 wake-up function Enabled. 0010 = GPD.2 wake-up function Enabled. 0011 = GPD.3 wake-up function Enabled. 0100 = GPD.4 wake-up function Enabled. 0101 = GPD.5 wake-up function Enabled. 0110 = GPD.6 wake-up function Enabled. 0111 = GPD.7 wake-up function Enabled. 1000 = GPD.8 wake-up function Enabled. 1001 = GPD.9 wake-up function Enabled. 1010 = GPD.10 wake-up function Enabled. 1011 = GPD.11 wake-up function Enabled. 1100 = GPD.12 wake-up function Enabled. 1101 = Reserved. 1110 = GPD.14 wake-up function Enabled. 1111 = Reserved.
[3]	**Reserved**	Reserved.

[2]	**PFWKEN**	**Pin Falling Edge Wake-up Enable Bit** 0 = GPD group pin falling edge wake-up function Disabled. 1 = GPD group pin falling edge wake-up function Enabled.
[1]	**PRWKEN**	**Pin Rising Edge Wake-up Enable Bit** 0 = GPD group pin rising edge wake-up function Disabled. 1 = GPD group pin rising edge wake-up function Enabled.
[0]	**WKEN**	**Standby Power-down Pin Wake-up Enable Bit** 0 = GPD group pin wake-up function Disabled. 1 = GPD group pin wake-up function Enabled.

GPIO Standby Power-down Control Register (CLK_IOPDCTL)

Register	Offset	R/W	Description	Reset Value
CLK_IOPDCTL	CLK_BA+0xB0	R/W	GPIO Standby Power-down Control Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**IOHR**	**GPIO Hold Release** When GPIO enter standby power-down mode, all I/O status are hold to keep normal operating status. After chip was waked up from standby power-down mode, the I/O are still keep hold status until user set this bit to release I/O hold status. **Note:** This bit is auto cleared by hardware.

HXT Filter Select Control Register (CLK_HXTFSEL)

Register	Offset	R/W	Description	Reset Value
CLK_HXTFSEL	CLK_BA+0xB4	R/W	HXT Filter Select Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**HXTFSEL**	**HXT Filter Select** 0 = HXT cutoff frequency is 24 MHz. 1 = HXT cutoff frequency is 4 MHz.

6.4 Security Configuration Unit (SCU)

6.4.1 Overview

Security configuration unit is designed for Arm® TrustZone®, and used to configure the security and privilege attribution of SRAM, GPIO and all other peripherals. SCU also collects AHB slaves' security and privilege violation response and generates SCU interrupt. SCU is also equipped with a timer to monitor the duration of the core processor in non-secure state.

Note: For details on Arm® TrustZone®, refer to the section "Arm® TrustZone®"

6.4.2 Features

- Configure SRAM's security and privilege attribution block by block
- Configure GPIOs' security and privilege attribution pin by pin
- Configure peripherals' security and privilege attribution
- Generate secure and privilege violation interrupt
- Equipped with a 24-bit timer as a non-secure state monitor
- Monotonic firmware version counter
- Debug protection mechanism
- Product life-cycle management

6.4.3 Block Diagram

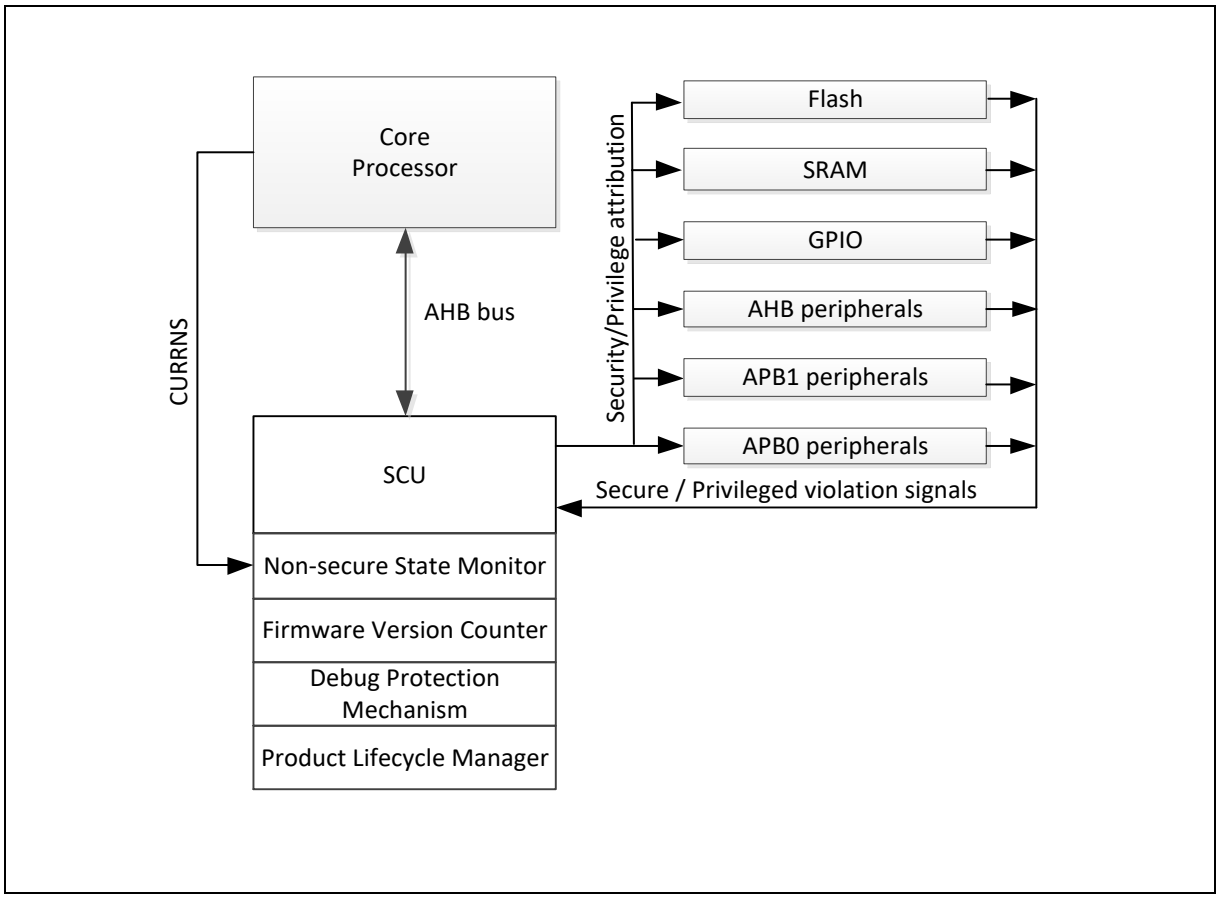


Figure 6.4-1 SCU Block Diagram

6.4.4 Functional Description

6.4.4.1 Security and Privilege Configuration

Security configuration unit (SCU) is used to configure the security and privilege attribution of all peripherals, including both masters and slaves. The security and privilege attribution of all peripherals on AHB, APB0 and APB1 bus can be configured by setting SCU registers, SCU_PNSSET0 to SCU_PNSSET6 and SCU_PNPSET0 to SCU_PNPSET6. Each bit controls the security or privilege attribution of a peripheral.

The security attribution of GPIO can be configured pin by pin through registers, SCU_IONSSET0 to SCU_IONSSET7, while the privilege attribution of GPIO is configured port by port through SCU_IONPSET register. Note that GPIO is the peripheral to configure settings for I/O pins even if these I/O pins are set to one of their multi-function and used by corresponding peripherals, so secure code should take the responsibility to set the security attribute of all GPIO pins correctly. It's recommended that only set the security attribute to non-secure for pins used as GPIO and for non-secure world. Besides, some registers in GPIO are not suitable to be controlled pin by pin, such as Px_DBCTL, in GPIO. The security attribute of these registers are always secure.

For the SRAM, the security and privilege attribution of each SRAM block can be configured through the register, SCU_SRAMNSSET and SCU_SRAMNPSET, in which each bit configures 16 Kbytes of SRAM block.

The peripherals Key Store, RTC, Tamper controller, PDMA0, timer0, timer1, WDT/WWDT and the first

Flash page are always secure and accept secure access only. It's not possible to set them to non-secure.

The security attribution of Flash data is configured by a boundary. Secure user can read the boundary through SCU_FNSADDR register. Flash data whose address is below the boundary is secure data while address upper the boundary is non-secure data. The boundary info. is stored in Flash and for setting the security attribution of Flash, refer to the FMC section.

Unlike secure code able to access all registers of SCU, non-secure code has few accessibility to SCU. However, for convenience, non-secure code can get the security attribution of SRAM regions and Flash boundary through the non-secure address of SCU_SRAMNSSET and SCU_FNSADDR registers when SCU_SINFAEN.SCUSIAEN bit is 1. For more details about SCU_SINFAEN, refer to "Shared-registers" sub-section.

Security Configuration Write Protection Mechanism

After powered on, all peripherals (including GPIO) and SRAM regions are secure and privileged. Secure code can set peripherals and blocks of SRAM to non-secure or non-privileged and further switched them back to secure from non-secure, privileged from non-privileged through corresponding registers. If the secure code intends to prevent further change of the security attribution of all peripherals and SRAM after finishing the first configuration, secure code can use the lock bit of SCU_SCWP register, which will restrict the write access to all registers related to security configuration (i.e. SCU_PNSSETn, SCU_IONSSETn, SCU_SRAMNSSET) when set. The SCU_SCWP.LOCK bit can only be cleared by a system-level reset.

6.4.4.2 Memory Access Policy

Security

This chip implements Arm® TrustZone® security extension supporting secure code as well as non-secure code running together on the chip. Secure code can access secure peripherals, secure memory regions, non-secure peripherals and non-secure memory regions while non-secure code can only access non-secure peripherals and non-secure memory regions. Refer to Arm®v8-M Architecture Reference Manual for more details.

To simply separate secure world and non-secure world, all slaves must follow the memory access policy which defines the access rule of masters and the expected response of slave.

Memory alias describes that a register can be accessed through multiple memory addresses. If a peripheral is implemented with memory alias, it will occupy two memory regions on memory map. Both regions map to the same registers in the peripheral while the only difference between these two regions is the bit 28 of the memory addresses. Bit 28 being low means it is secure region while being high means it is non-secure region. For example, suppose that a peripheral is implemented with memory alias located at 0x4nnn_nnnn, it will also occupy the region at 0x5nnn_nnnn on memory map.

Although a memory-alias-implemented peripheral occupies two memory regions, it can be accessed through only one address at any time, depending on whether the peripheral is secure or non-secure. If a peripheral is secure, it should be accessed through address bit 28 = 0 while if it's non-secure, it should be accessed through bit 28 = 1.

The access rule defined as memory access policy (Security) is listed below:

1. Secure master should access secure memory or peripheral through address bit 28 = 0.
2. Secure master should access non-secure memory or peripheral through address bit 28 = 1.
3. Non-secure master should only access non-secure memory or peripheral through address bit 28 = 1.
4. Otherwise, the slave would not respond the access and further generate security violation in

some cases.

More details about security memory access policy are shown in Table 6.4-1 and Table 6.4-2.

Master Is Core Processor			
Bit 28 of Target Address	Master is Secure	Slave is Secure	Expected Slave's Response
0	Yes	Yes	Access successful
0	Yes	No	RAZWI
0	No	Yes	Hard fault
0	No	No	Hard fault
1	Yes	Yes	RAZWI, Hard fault and security violation interrupt generated
1	Yes	No	Access successful
1	No	Yes	RAZWI, Hard fault and security violation interrupt generated
1	No	No	Access successful
*RAZWI: Slave outputs all zero data when read access while ignores the access when write access.			

Table 6.4-1 Security Memory Access Policy (Master is Core Processor)

Master Is Not Core Processor			
Bit 28 of Target Address	Master is Secure	Slave is Secure	Expected Slave's Response
0	Yes	Yes	Access successful
0	Yes	No	RAZWI
0	No	Yes	RAZWI, transfer aborted and security violation interrupt generated
0	No	No	RAZWI, transfer aborted and security violation interrupt generated
1	Yes	Yes	RAZWI, transfer aborted and security violation interrupt generated
1	Yes	No	Access successful
1	No	Yes	RAZWI, transfer aborted and security violation interrupt generated
1	No	No	Access successful

Table 6.4-2 Security Memory Access Policy (Master is Not Core Processor)

For EBI external memory, located at range from 0x6000_0000 to 0x7000_0000, the access policy check is enabled after the region is enabled, or user will get error response if accessing.

Privilege

This chip also implements the privileged access policy of AMBA HPROT protocol, which may restrict the access of the masters.

The access rule is defined as memory access policy(Privileged), which is listed below:

1. Privileged master can access privileged slave as well as non-privileged slave.
2. Non-Privileged master cannot access privileged slave but can access non-privileged slave.
3. Core processor can access SRAM, Flash and EBI external memory in spite of the privileged setting of these peripherals. If user intends to restrict process to access these regions, MPU should be used, instead.

More details about security memory access policy are shown in Table 6.4-1.

Master Is Core Processor		
Master is Privileged	Slave is Privileged	Expected Slave's Response
Yes	Yes	Access successful
Yes	No	Access successful
No	No	Access successful
No	Yes	RAZWI, Hard fault and privileged violation interrupt generated

Table 6.4-3 Privileged Memory Access Policy (Master is Core Processor)

Master Is Not Core Processor		
Master is Privileged	Slave is Privileged	Expected Slave's Response
Yes	Yes	Access successful
Yes	No	Access successful
No	No	Access successful
No	Yes	RAZWI, transfer aborted and privileged violation interrupt generated

Table 6.4-4 Privileged Memory Access Policy (Master is Not Core Processor)

Shared-registers

There are three always-secure peripherals: SCU, System, FMC, that share part of system information to non-secure world with enable bits in SCU_SINFAEN register. The shared registers of these peripherals are listed in Table 6.4-7. By default, shared registers are enabled to let non-secure code access through their non-secure address. Secure code can set corresponding bit in SCU_SINFAEN register to disable the ability of non-secure code accessing shared registers of target peripheral.

The memory access policy of these three peripherals are listed in Table 6.4-5 and Table 6.4-6, which is slightly different from general memory access policy in Table 6.4-1 and Table 6.4-2.

Master Is Core Processor				
Bit 28 of Target Address	Master is Secure	Non-Secure Access Enable (SCU_SINFAEN Register)	Expected Slave's Response (shared registers)	Expected Slave's Response (non-shared registers)
0	Yes	-	Access successful	Access successful
0	No	-	Hard fault	Hard fault
1	-	0	RAZWI	RAZWI
1	-	1	Access successful	RAZWI

Table 6.4-5 Shared Register Memory Access Policy (Master Is Core Processor)

Master Is Not Core Processor				
Bit 28 of Target Address	Master is Secure	Non-Secure Access Enable (SCU_SINFAEN Register)	Expected Slave's Response (shared registers)	Expected Slave's Response (non-shared registers)
0	Yes	-	Access successful	Access successful
0	No	-	RAZWI, transfer aborted and security violation interrupt generated	RAZWI, transfer aborted and security violation interrupt generated
1	-	0	RAZWI	RAZWI
1	-	1	Access successful	RAZWI

Table 6.4-6 Shared Register Memory Access Policy (Master Is not Core Processor)

Shared Register Types	FMC	SCU	System
R/W	FMC_ISPCTL, FMC_ISPADDR, FMC_ISPDAT, FMC_ISPCMD, FMC_ISPTRG, FMC_ISPSTS	DPM_NSCTL, DPM_NSSTS, DPM_NSPW0, DPM_NSPW1, DPM_NSPW2, DPM_NSPW3	SYS_REGLCTL_NS SYS_IPRST0 [CHIPRST]
Read only	FMC_XOMR0STS, FMC_XOMR1STS, FMC_XOMR2STS, FMC_XOMR3STS, FMC_XOMSTS	SCU_NSMSTS[CURRNS], SCU_SRAMNSSET, SCU_FNSADDR, SCU_SINFAEN	SYS_PDID, CLK_PWRCTL, CLK_AHBCLK, CLK_APBCLK0, CLK_APBCLK1, CLK_CLKSEL0, CLK_CLKSEL1, CLK_CLKSEL2, CLK_CLKSEL3, CLK_CLKDIV0, CLK_CLKDIV1, CLK_CLKDIV4, CLK_PCLKDIV, CLK_PLLCTL, CLK_STATUS

Table 6.4-7 Shared Registers List for Non-secure Access

GPIO

The security attribution of GPIO is configured pin by pin as mentioned above. Most of GPIO registers, however, control more than one GPIO pin, so GPIO requires its own access policy. As shown in Table 6.4-8, secure user can access secure GPIO bits through address bit[28] equal to 0, and also can access non-secure GPIO bits through address bit[28] equal to 1 while non-secure code can only access non-secure GPIO bits through address bit[28] equal to 1.

A special case is Px_DBCTL registers, which control function by port basis. Px_DBCTL registers accept secure access only, no matter what security attributions of all GPIO pins are.

Because the privileged attribute of GPIO is controlled by port basis, the privileged access policy of GPIO is same as shown in Table 6.4-3.

Bit 28 Of Target Address	Master Is Secure	Expected Slave's Response	Secure GPIO Bit-Field	Non-Secure GPIO Bit-Field
0	Yes	Access successful	Access successful	RAZWI
0	No	Hard fault	-	-
1	Yes	Access successful	RAZWI	Access successful
1	No	Access successful	RAZWI	Access successful

Table 6.4-8 GPIO Memory Access Policy

6.4.4.3 Security and Privileged Violation Interrupt

SCU is also the module to provide information about security and privileged violation event occurred in the system. For most cases in Table 6.4-1, when an event of violating memory access policy occurs, probably due to improper setting or malicious attack, the AHB slave that receives the access of violating memory access policy will issue a violation signal to SCU (shown in Figure 6.4-2) and report master number of the violating master (shown in Table 6.4-9) and address information of the slave been accessed. Furthermore, the slave will respond error to the master.

If the security or privileged violation interrupt enable bit of the peripheral in SCU_SVIOIEN / SCU_PVIOIEN is set, when SCU receives either the security or privileged violation event from the peripheral, it will trigger SCU interrupt to the core processor.

Because SCU is always secure, the secure code must set SCU interrupt to secure by setting ITNS register. When SCU interrupt asserts, the secure code can access SCU_SVINSTS and SCU_PVINSTS register to determine which slave generates the violation event and is violated secure or privilege policy. Then, it can get violation source from the corresponding violation source registers such as SCU_APB0VSRC, SCU_APB1VSRC, SCU_GPIOVSRC. Also, it can get violation address from the corresponding violation address registers such as SCU_APB0VA, SCU_APB1VA, SCU_GPIOVA.

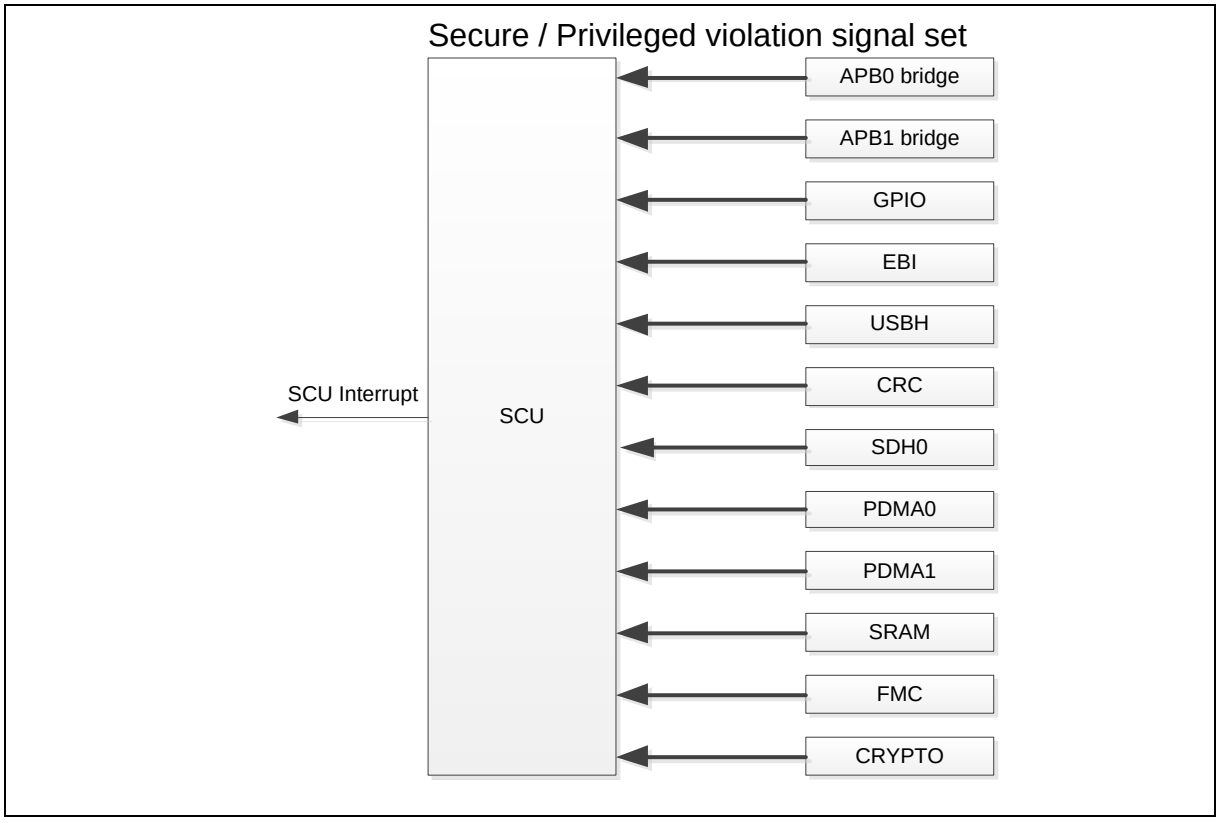


Figure 6.4-2 Security and Privileged Violation Signal Block Diagram

Master ID	Description
0	Core processor

3	PDMA0
4	Secure Digital Host Controller (SDH0)
5	Cryptographic Accelerator (CRYPTO)
6	USB Host Controller (USBH)
B	PDMA1

Table 6.4-9 Master ID Number List

6.4.4.4 Non-Secure State Monitor

SCU is equipped with a down-count counter for monitoring the core processor's non-secure state. The counter can function as the Non-secure state monitor by default, called monitor mode, or a simple timer, called free-counting mode, by setting the TMRMOD bit of SCU_NSMCTL register.

The value of the counter can be read from the SCU_NSMVAL register, but will be cleared to 0 by writing any value to this register which will also clear the interrupt flag in SCU_NSMSTS register. If the counter is enabled, the reload value saved in the SCU_NSMLOAD register will be loaded to the counter when the value of the counter is cleared to 0 or counts down to 0.

The PRESCALE value of the SCU_NSMCTL register controls the counting speed and the counter enable/disable. When PRESCALE is set to 0, the counter is disabled and the value of the counter is fixed. The PRESCALE value other than 0 indicates the ratio of the divided clock of the counter to the system clock.

When the counter counts down to 0, the interrupt flag, NSMIF, in SCU_NSMSTS register rises, which will trigger the SCU interrupt if the interrupt enable bit, NSMIEN, in SCU_NSMCTL register is on. The flag should be cleared by writing 1 to the bit of SCU_NSMSTS register.

The DBGON and IDLEON in SCU_NSMCTL register control whether the counter should keep counting or pause when the chip enters debug mode and idle mode respectively.

A typical flow of using the non-secure state monitor is similar to that of using Systick timer and is described below:

1. Write reload value to SCU_NSMLOAD
2. Set SCU_NSMVAL to clear current value and interrupt flag
3. Set SCU_NSMCTL to enable the monitor

Note that for monitoring the core processor runs at non-secure state for N cycles of the divided clock of the counter, set SCU_NSMLOAD to N.

Monitor Mode

When TMRMOD is 0, which is the default value, the non-secure state monitor works in monitor mode. Under this mode, the counter will count down only when the core processor runs in non-secure state and trigger NSMIF when counting down to 0. When the core processor runs in secure state, the counter will stop counting, and the value of the counter will be fixed or be reloaded to the reload value depending on AUTORLD bit of SCU_NSMCTL register.

If DBGON bit is 0, the counter will stop counting when the chip is in debug mode no matter the core processor is in the secure state or not. Also, if IDLEON bit is 0, the counter will stop counting when the chip is in idle mode. By default the counter should stop counting down when the core processor is in non-secure state and the chip enters idle mode.

Free-counting Mode

When TMRMOD is 1, the counter working in free-counting mode functions as a simple timer. That is,

the counter starts counting if PRESCALE is not 0 no matter the core processor is in secure state or non-secure state. However, whether the counter does counting when the chip is in debug mode or idle mode still depends on the DBGON and IDLEON settings.

6.4.4.5 Firmware Version Counter

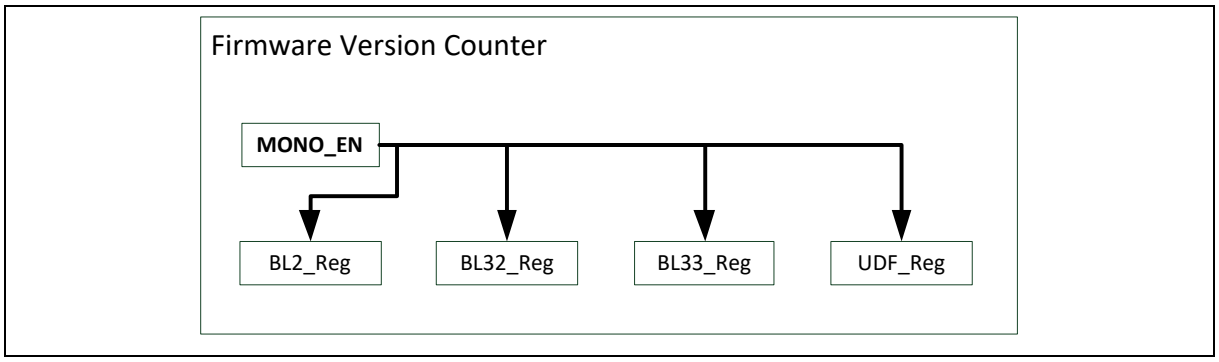


Figure 6.4-3 Firmware Version Counter Block Diagram

Firmware version counter, FVC, aims at providing a monotonic mechanism to handle the version of firmware stored in non-volatile memory. FVC provides 4 registers, FVC_NVC0, FVC_NVC1, FVC_NVC4 and FVC_NVC5 for user to record 4 individual firmware. FVC_NVC0 and FVC_NVC1 can record number of version up to 64 (0~63), while FVC_NVC4, FVC_NVC5 can record that up to 256 (0~255).

After reset, FVC will be at **Reset** state, where all firmware version in FVC_NVC0, FVC_NVC1, FVC_NVC4 and FVC_NVC5 are not valid yet and all registers do not accept write request except FVC_CTL.INIT bit. If user is going to use FVC, user should set FVC_CTL.INIT to make FVC enters **Init** state, which is indicated by FVC_STS.BUSY equal to 1 and FVC_STS.RDY equal to 0. After the init process is finished, FVC enters **Ready** state, where user can read the firmware version by reading the corresponding register, i.e. FVC_NVC0, FVC_NVC1, FVC_NVC4 or FVC_NVC5, or update the firmware version by writing a new value with VERIFY code to the corresponding registers. When FVC is updating the firmware version, the FVC is in **Write** state, and ignores any write operation. The state flow of FVC can be checked through Figure 6.4-4.

For the convenience of development, firmware versions stored in FVC registers are not monotonic before MONOEN bit in FVC_CTL register are set. User can set any value to firmware version registers if MONOEN is not set. However, after user set the MONOEN bit, the firmware version registers accept only value greater than current firmware version. For example, if the MONOEN bit is set and the current value of FVC_NVC0 is 3, writing value like 4, 5…63 to the register will be accepted, while writing other value such as 0,1,2, 65, 66 will be ignored.

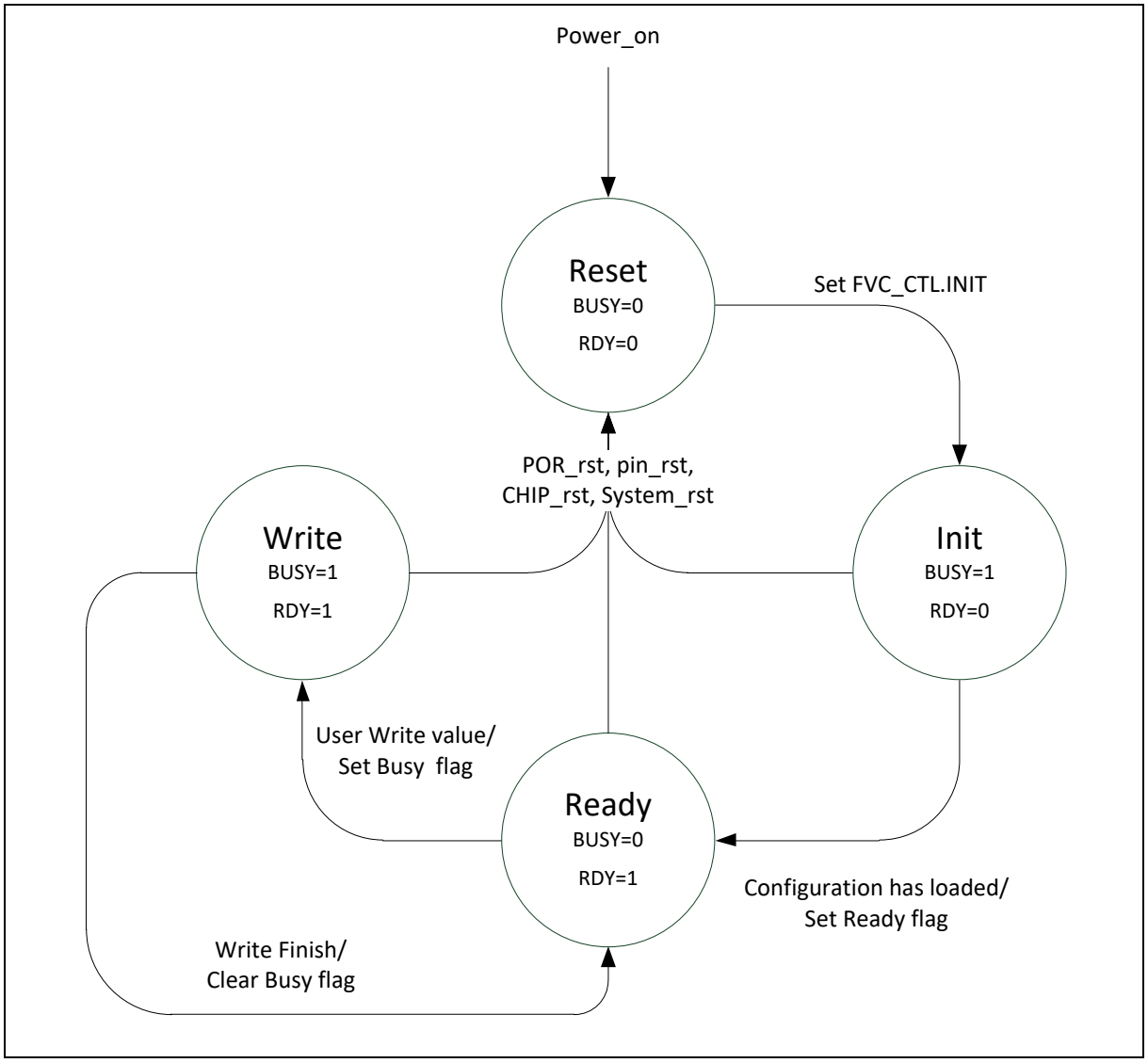


Figure 6.4-4 FVC State Flow

6.4.4.6 Debug Protection Mechanism (DPM)

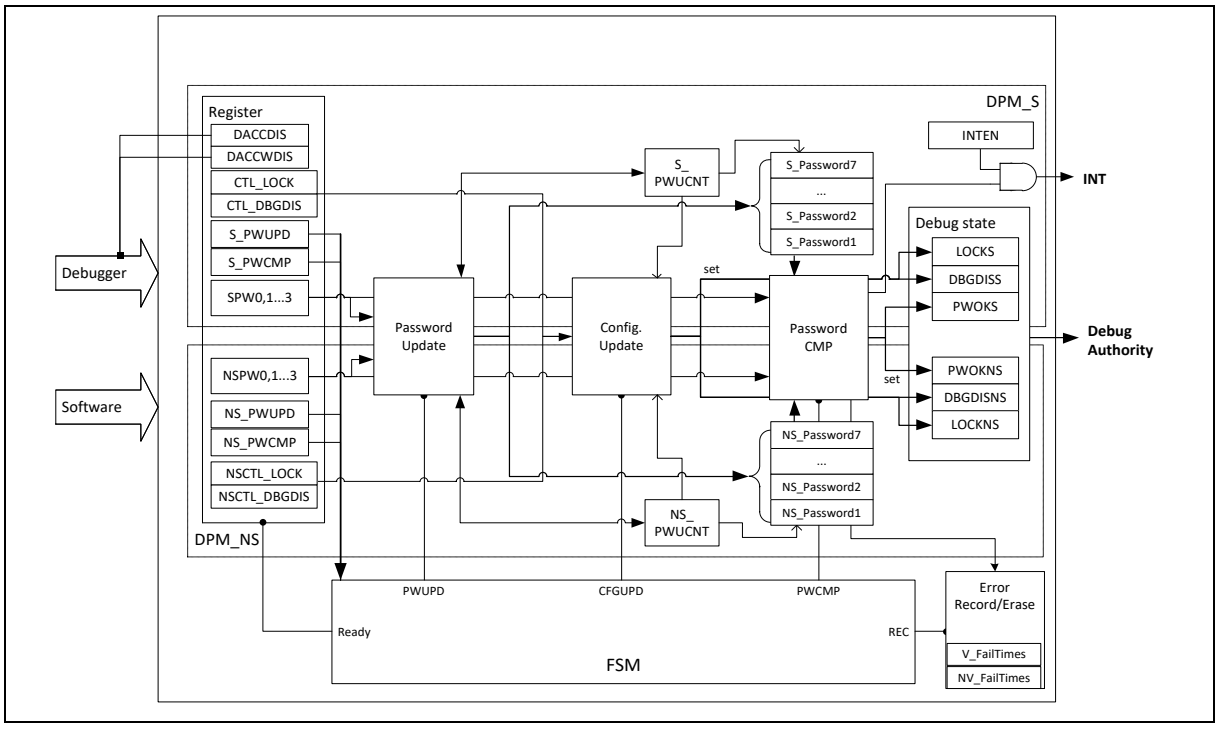


Figure 6.4-5 Block Diagram of Debug Protection Mechanism Unit

Debug protection mechanism, DPM, is used to manage the configuration, permission, authentication and authority of the external debugger. DPM provides a AHB interface for user to manage the debug access control,.

Due to TrustZone®, DPM is divided into secure DPM and non-secure DPM. Registers of secure DPM: DPM_CTL, DPM_STS, DPM_SPW…etc., accept secure software access only and reject non-secure software access, while registers of non-secure DPM: DPM_NSCTL, DPM_NSSTS, DPM_NSPW…etc., are shared registers accepting secure software access and, if SCU shared information is enabled (SCUSIAEN (SCU_SINFAEN[0])=1), accepting non-secure software access through non-secure alias address (bit[28] of address equal to 1).

DPM defines the debug state and divides it into secure debug state and non-secure debug state, each of which is defined by the password-based authentication mechanism and two configuration bits: LOCK and DBGDIS. These configuration bits are non-volatile and will be kept even after system power-off.

DPM is reset by cold reset only.

There are 3 levels of the debug authority: **Secure**, **Non-secure only**, **No debug**. Combining both secure and non-secure debug state, which should be at one of **RESET**, **LOCKED**, **DEFAULT**, **CLOSE** or **OPEN** state, the debug authority is determined.

For the debug state and debug authority, refer to the **Debug Access Control** section.

For the password-based mechanism to authenticate external debugger, refer to the **Mechanism of Debug Authentication** section.

DPM Operating State

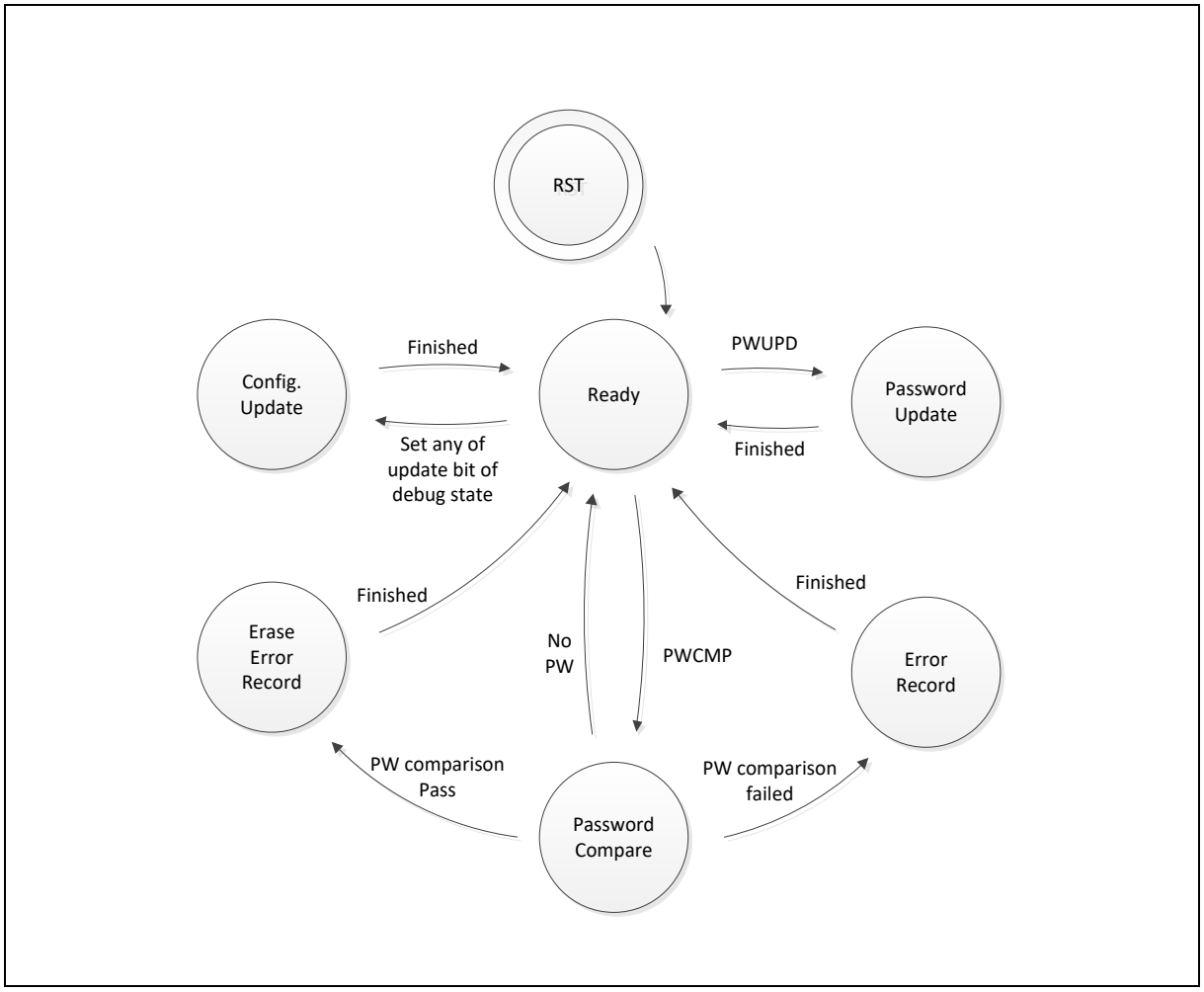


Figure 6.4-6 DPM Main State

As Figure 6.4-6, DPM's operating follows a state flow. After reset, the DPM enters **Ready** state indicated by the BUSY bit in DPM_STS and DPM_NSSTS register equal to 0. All DPM registers accept write request only at **Ready** state, so user should check if the BUSY bit is 0 before any write access.

The DPM enters **Password Update** state to set a new password when user make a valid request to do password update. Secure user can update password of both secure and non-secure DPM through DPM_CTL and DPM_NSCTL while non-secure user can only update non-secure password through DPM_NSCTL with bit[28] of address equal to one. User should follow steps described in Password Change section.

The DPM goes to **Password Compare** state and do password comparison when user triggers PWCMP bit in either DPM_CTL or DPM_NSCTL register. Note that the comparison request will be ignored if no password has been set yet. After the comparison finished, DPM goes to **Error Record** or **Erase Error Record** state depending on the result of the comparison.

Secure user can enable the DPM interrupt by setting INTEN bit in DPM_CTL register to get the announcement if the result of the authentication, i.e. password comparison, is fail. The DPM interrupt status can be checked through INT bit in DPM_STS. This bit is cleared after PWCERR flag of both

DPM_STS and DPM_NSSTS are 0.

User can change the configuration bits of DPM to change the debug state through DPM_CTL and DPM_NSCTL register. DPM_CTL is for changing secure debug state while DPM_NSCTL is for non-secure debug state. After user set a valid updated configuration to either DPM_CTL or DPM_NSCTL register, DPM will enter **Config**. **Update** state and update the configuration. The updated configuration will not work immediately after the update process finished but work after a cold reset (pin reset or chip reset).

Debug Access Control

The DPM uses 2 configuration bit, **LOCK** bit and **DBGDIS** bit, plus the **PWOK** bit related to password-based authentication mechanism to define the debug state and control the debugger's authority. After a successful password comparison, the **PWOK** bit will be set to one to indicate the debug authentication check is pass. These bits define the debug state to be one of "**DEFAULT**", "**CLOSE**", "**OPEN**", "**LOCKED**" states. The possible debug states are listed below:

- **RESET** state: system is under reset.
- LOCKED state: LOCK bit is set.
- DEFAULT state: Both LOCK and DBGDIS are not set.
- CLOSE state: LOCK is not set, DBGDIS is set but PWOK is 0
- OPEN state: LOCK is not set, DBGDIS is set and PWOK is 1

Due to Trustzone, secure DPM controls secure debug state using **LOCKS**, **DBGDISS** and **PWOKS** bits, which are indicated by LOCK, DBGDIS, PWOK bit in DPM_STS registers separately; while non-secure DPM controls non-secure debug state using **LOCKNS**, **DBGDISNS** and **PWOKNS** bits, indicated by corresponding LOCK, DBGDIS, PWOK bit in DPM_NSSTS registers.

Combining both secure and non-secure debug state, the debug authority is defined. There are 3 levels of debug authority: Secure, Non-secure only and No debug. The description of each level are listed below.

- **Secure debug** – User can debug secure and non-secure code, and debugger can access both secure and non-secure region.
- **Non-secure debug only** – User can only debug non-secure code, and debugger can access non-secure region only.
- **No debug allowed** – User can debug neither secure nor non-secure code, and debugger can access neither secure nor non-secure region only.

When reset, both secure and non-secure state are in **RESET** state where no debug is allowed. After reset released, secure debug state, controlled by **LOCKS**, **DBGDISS,** and non-secure debug state, controlled by **LOCKNS**, **DBGDISNS** bits, go to one of **LOCKED**, **DEFAULT** or **CLOSE** state separately. Refer to the Table 6.4-10 for more detail.

By default, both debug state will go to **DEFAULT** state where secure debug is allowed. If secure debug state is in either **CLOSE** or **LOCKED** state and non-secure debug state remains in **DEFAULT** state, the debug authority is non-secure debug only. If non-secure debug state is in either **CLOSE** or **LOCKED** state, no matter what state the secure debug state is, the debug authority is no debug allowed. When the debug state is in **CLOSE** state, user can enter PASSWORD and trigger the compare process. If the comparison result is pass, the PWOK bit is set to indicate the debug state enters **OPEN** state.

The behavior of debug state in **OPEN** state is similar to that in **DEFAULT** state. That is, if both debug state is in either **OPEN** or **DEFAULT** state, the secure debug is allowed. If secure debug state is in either **CLOSE** or **LOCKED** state and non-secure debug state is in either OPEN or DEFAULT state, the non-secure debug only is the authority.

When DPM is ready, which is indicated by the BUSY bit of DPM_STS or DPM_NSSTS register, user can check DPM_STS[18:16] for secure debug state while DPM_NSSTS[18:16] for non-secure debug state, and, then, infer the debug authority.

To meet Armv8-M specification, the debug authority of “Secure debug” can be controlled by software rather than the secure DPM. When the debug authority is Non-secure debug only, user can use DAUTHCTRL register, which is located at 0xE000EE04 and is only accessible by secure code, to switch the control of “Secure debug” authority to software. When DAUTHCTRL.SPIDENSEL is set to 1, secure debug is controlled by DAUTHCTRL.INTSPIDEN rather than Secure DPM. Refer to Armv8-M architecture reference manual for more detail.

LOCKNS	DBGDISNS	PWOKNS	LOCKS	DBGDISS	PWOKS	Debug State		Debug Authority
						Nonsecure DPM	Secure DPM	
-	-	-	-	-	-	RESET	RESET	No debug allowed (System is under reset)
1	X	X	X	X	X	LOCKED	X	No debug allowed
0	1	0	X	X	X	CLOSE	X	No debug allowed
0	1	1	1	X	X	OPEN	LOCKED	Non-secure debug only
0	0	X	1	X	X	DEFAULT	LOCKED	Non-secure debug only
0	1	1	0	1	X	OPEN	CLOSE	Non-secure debug only
0	0	X	0	1	X	DEFAULT	CLOSE	Non-secure debug only
0	1	1	0	1	1	OPEN	OPEN	Secure debug
0	0	X	0	1	1	DEFAULT	OPEN	Secure debug
0	1	1	0	0	X	OPEN	DEFAULT	Secure debug
0	0	X	0	0	X	DEFAULT	DEFAULT	Secure debug

Table 6.4-10 Debug State and Debug Authority

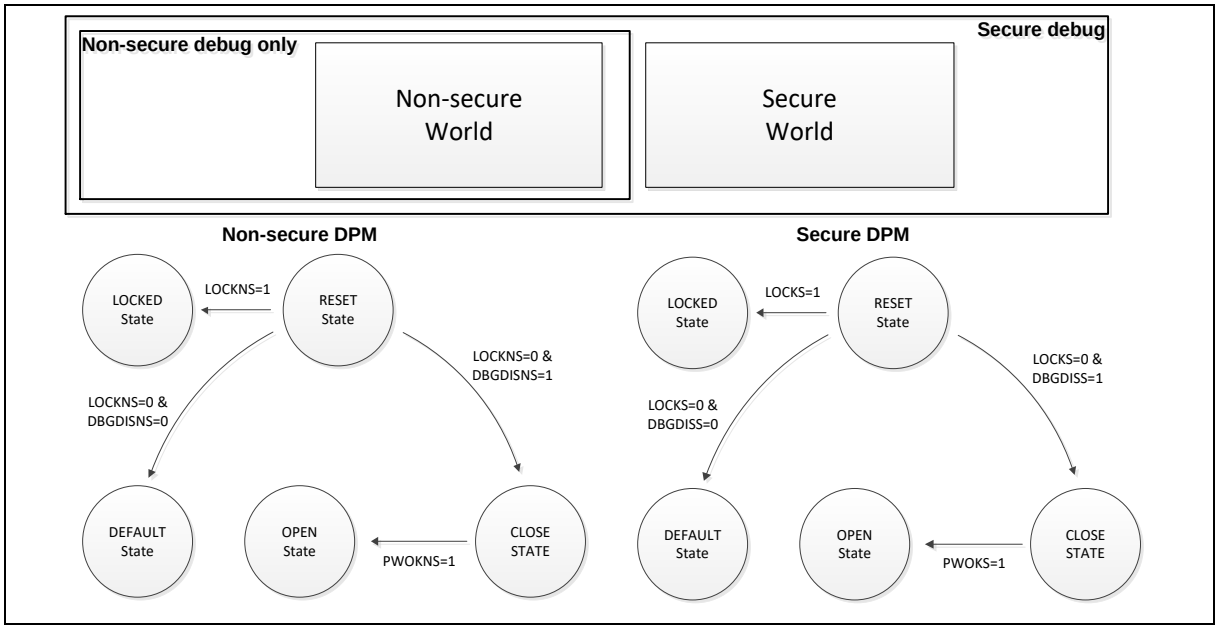


Figure 6.4-7 Debug Authority and Debug State Flow

Debug State Update

When DPM is in Ready state, user can update debug state by setting corresponding LOCK, DBGDIS bit(s) in DPM_CTL and/or DPM_NSCTL register. Secure user can update secure and non-secure debug state by DPM_CTL[1:0] and DPM_NSCTL[1:0] separately. When user request a valid write to set LOCK and/or DBGDIS bit in DPM_CTL or DPM_NSCTL register, DPM will enters **Config. Update** state and the BUSY bit in DPM_STS and DPM_NSSTS register will be high. After a successful update, the DPM will be back to **Ready** state. Thus, user can check the update request is in process, ignored or finished by firstly reading the corresponding LOCK, DBGDIS bits in DPM_CTL, DPM_NSCTL register that is/are going to be updated to check if the request is ignored, and reading the BUSY bit goes low, which indicates the process is finished.

Note that the new configuration will not take effect immediately, so user will not read updated configuration through DPM_STS[17:16] bits and DPM_STS[17:16]. User should do a cold reset (chip reset or pin reset) to make it take effect.

In order to set the debug state to **LOCKED** state, user just needs to set the corresponding LOCK bit. For non-secure debug state, set the LOCK bit in DPM_NSCTL register; while for secure debug state, set the LOCK bit in DPM_CTL register. Nevertheless, to set debug state to **CLOSE** state, at least one password of the corresponding debug state should be inserted before making debug state to enter **CLOSE** state, or the update is ignored. User can check if the password is existed through checking if the **PWUCNT** bits in DPM_STS, DPM_NSSTS registers not equal to 0.

Mechanism of Debug Authentication

External debugger's authentication is based on a password-based authentication mechanism. That is, the user with correct password(s) can be authenticated to do debug. By default, there is no password on the chip and the debug function is open-to-use to everyone. The user who wants to restrict debug function can set the password, which is confidential and only known by the user, and disable the debug function of the chip. The user can re-open the debug function by entering the password. The user can also share the password to someone, who has been authenticated to do debug on the chip, and then, that one can do debug if entering the password correctly to the chip.

DPM provides registers for user to set/change password, which is 128-bit long and non-volatile.

Secure DPM provides DPM_CTL.PWUPD, DPM_STS, DPM_SPW0~3 registers to manage secure debug authentication and non-secure DPM provides DPM_NSCTL.PWUPD, DPM_NSSTS, DPM_NSPW0~3 to manage non-secure debug authentication.. For how to set and update password, refer to Password Change section for more detail.

Debug function is managed by both secure and non-secure debug state. When user intends to disable the debug function and re-open by password, user should update the debug state to **CLOSE** state, after setting the password.

DPM also provides registers for user to compare the password when one user wants to re-open the debug function, which is DPM_CTL.PWCMP, DPM_SPW0~3 and DPM_NSCTL.PWCMP, DPM_NSPW0~3. After the compare is successful, the debug state will go to **OPEN** state and re-open the debug function if the debug state is at **CLOSE** state rather than **LOCKED** state. Refer to Password Comparison section for more detail.

Password Update

DPM provides registers for software to set and update the password for secure debug state and non-secure debug state. DPM_SPW0~3 registers are for updating password for secure debug state, while DPM_NSPW0~3 register are for updating password for non-secure debug state. Note that registers for entering password, DPM_SPW0~3 and DPM_NSPW0~3, are write-only, read-as-all-one.

To set a new password, only privileged secure software is possible. User can set the password when DPM is in **Ready** state. The following is the update flow:

1. Write new password to DPM_SPW0~3 registers for updating password for secure debug state and DPM_NSPW0~3 for updating password for non-secure debug state.
2. Write 1 to PWUPD bit of DPM_CTL register for updating password for secure debug or that of DPM_NSCTL for updating password for non-secure debug state, to trigger the update process. Note that this write access should be along with PWCMP bit as 0.
3. A valid write to PWUPD will make DPM enters **Password Update** state and make the BUSY bit high.
4. After a success update, the corresponding PWUCNT bits in DPM_STS or DPM_NSSTS will be added 1 and the corresponding PWUOK bit in DPM_STS or DPM_NSSTS will also be set to 1, which can be cleared by write one to the bit.

If user updates password not under debug mode but by software code, it is recommended that when software is going to update the password, the **DACCWDIS** bit in DPM_CTL should be set before updating password to prevent interference caused by external debugger's access. Non-secure user can use DACCWDIS bit in DPM_NSCTL, if using DPM_NSPW registers.

When first time setting the password, it is common to update debug state to **CLOSE** state to disable debug function. For updating debug state, refer to Debug State Update section.

Password of both secure and non-secure debug state can be updated finite times. The maximum time is **7** times. The number of updated times can be read through PWUCNT bits in DPM_STS and DPM_NSSTS register, so no password can be update if PWUCNT bits reaches 7.

Password Comparison

The DPM provides an interface for software to compare password, which allows software/firmware developer to provide its own debug authentication interface, and an interface for external debugger to do authentication.

Secure DPM provides DPM_CTL.PWCMP, DPM_STS, DPM_SPW0~3 for software to do authentication for secure debug state. To do the password comparison of secure state, follows these steps:

1. Read DPM_STS.PWUCNT to check if there is password existed.

2. If DPM_STS.PWUCNT is not 0, there is password existed. Enter the password to register DPM_SPW0~3 to do authentication.
3. Trigger the comparison process by setting 1 to DPM_CTL.PWCMP. Then, DPM will enter **Password Compare** state where the BUSY bit goes to 1.
4. Check DPM_STS.PWCERR after the BUSY bit, which indicates the comparison is finished, is 0. If the comparison result is failed, the DPM_STS.PWCERR is set to 1; If the comparison is successful, the DPM_STS.PWCERR is 0.

5. Non-secure DPM is similar to secure DPM and provides DPM_NSCTL.PWCMP, DPM_NSSTS, DPM_NSPW0~3 register for both secure and non-secure software to do password comparison.. Secure software can control the accessibility of non-secure software to non-secure DPM registers by setting SCU_SINFAEN if the security is concerned. The flow of password comparison of non-secure state is also similar to above steps.

PWOK bits in DPM_STS and DPM_NSSTS register are set to 1 if there is at least one successful password comparison since last reset and used to indicate the current debug step is **OPEN** state.

Due to simple password, the limit of retry times mechanism is implemented to mitigate brute-force attack. DPM limits the try time to at most **7** times fail without a system reset. DPM will also record the fail time to non-volatile memory, which is counted 1 only once for all fail try without system reset, and the limits try time of non-volatile memory is at most 7 times, too.

The failed try times record will be set when DPM is at Error Record state, and clear when DPM is at Erase Error Record state. If the result of password comparison is failed, DPM goes to **Error Record** state; If the result is pass, DPM goes to **Erase Error Record** state, where the failed try time, including volatile record and non-volatile record, will be erased to 0.

6.4.4.7 Product Life-cycle Manager

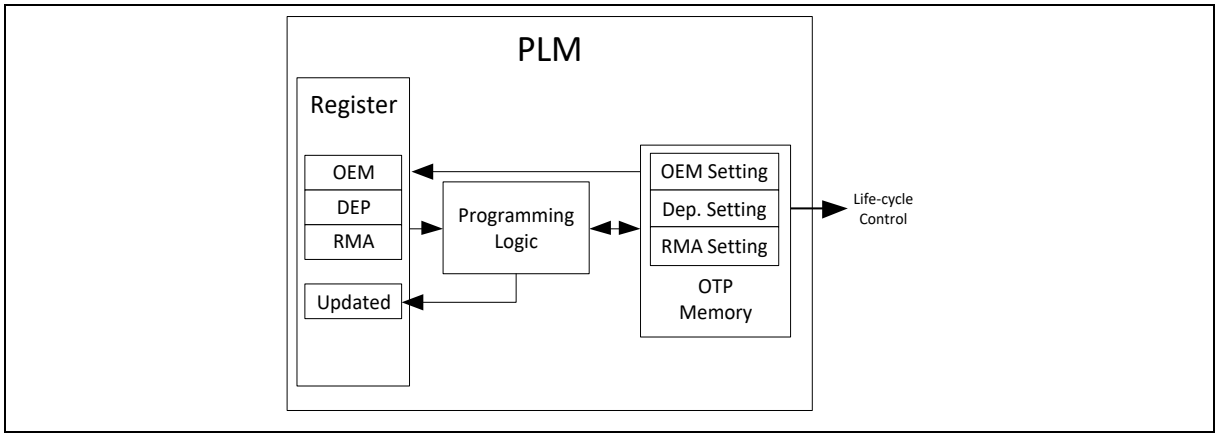


Figure 6.4-8 Product Life-cycle Manager Block Diagram

Product life-cycle manager, PLM, takes charge of recording the life-cycle stage of the chip.

The PLM divides the life-cycle of the chip into 4 stages: Vendor → OEM → Deployed → RMA. The life-cycle flow is one-way, so the configuration bits, OEM, Deployed and RMA in PLM register is allowed to set to 1 but not allowed to be cleared to 0. It makes sure that after entering the next stage, no rollback is possible.

The life-cycle stages settings are non-volatile. Privileged secure software can access PLM_STS register to check the current life-cycle stage through STAGE bits and can also access PLM_CTL register to update the stage. User should write target stage plus Write Verify code, which is 0x475A, to PLM_CTL to update the PLM. A valid write to PLM_CTL register will make PLM program the setting. After the process is finished, the **DIRTY** bit in PLM_STS register will be set, but the new setting of PLM

will not take effect before a **cold reset** (pin_rst, chip_rst, wdt_rst). After a cold reset, user can read the updated PLM stage from PLM_STS register.

PLM_CTL accepts update request only once without a reset, i.e. after a valid write to PLM_CTL to make PLM start the update progress, PLM_CTL will not accept any new request without a cold reset.

The PLM provides some hardware-based control based on the lifecycle stage the PLM is. The lifecycle stage, corresponding bit values and functions are listed in Table 6.4-11.

Lifecycle Stage	PLM_CTL.STAGE	Function
OEM	3'b001	ICE/ICP/Writer Accessibility – NI (Controlled by DPM) Mass Erase – NI
Deployed	3'b011	ICE/ICP/Writer Accessibility – OFF Mass Erase – OFF
RMA	3'b111	ICE/ICP/Writer Accessibility – NI (Controlled by DPM) Mass Erase – OFF Keystore revokes all NVM keys
ERROR	Others, i.e. 3'b100, 3'b110, 3'b101, 3'b010	ICE/ICP/Writer Accessibility – OFF Mass Erase – OFF
※NI: No influence		

Table 6.4-11 Function of Stage of PLM

6.4.5 Register Map

R: read only, **W**: write only, **R/W**: both read and write, **C**: Only value 0 can be written

Register	Offset	R/W	Description	Reset Value
SCU Base Address: **SCU_BA = 0x4002_F000** **FVC_BA = 0x4002_F500** **DPM_BA = 0x4002_F600** **PLM_BA = 0x4002_F700** **SCU non-secure base address is SCU_BA + 0x1000_0000.**				
SCU_PNSSET0	SCU_BA+0x00	R/W	Peripheral Non-secure Attribution Set Register0 (0x4000_0000~0x4001_FFFF)	0x0000_0000
SCU_PNSSET1	SCU_BA+0x04	R/W	Peripheral Non-secure Attribution Set Register1 (0x4002_0000~0x4003_FFFF)	0x0000_0000
SCU_PNSSET2	SCU_BA+0x08	R/W	Peripheral Non-secure Attribution Set Register2 (0x4004_0000~0x4005_FFFF)	0x0000_0000
SCU_PNSSET3	SCU_BA+0x0C	R/W	Peripheral Non-secure Attribution Set Register3 (0x4006_0000~0x4007_FFFF)	0x0000_0000
SCU_PNSSET4	SCU_BA+0x10	R/W	Peripheral Non-secure Attribution Set Register4 (0x4008_0000~0x4009_FFFF)	0x0000_0000
SCU_PNSSET5	SCU_BA+0x14	R/W	Peripheral Non-secure Attribution Set Register5 (0x400A_0000~0x400B_FFFF)	0x0000_0000

SCU_PNSSET6	SCU_BA+0x18	R/W	Peripheral Non-secure Attribution Set Register6 (0x400C_0000~0x400D_FFFF)	0x0000_0000
SCU_SRAMNSSET	SCU_BA+0x24	R/W	SRAM Non-secure Attribution Set Register	0x0000_0000
SCU_FNSADDR	SCU_BA+0x28	R	Flash Non-secure Boundary Address Register	0xXXXX_XXXX
SCU_SVIOIEN	SCU_BA+0x2C	R/W	Security Violation Interrupt Enable Register	0x0000_0000
SCU_SVINTSTS	SCU_BA+0x30	R/W	Security Violation Interrupt Status Register	0x0000_0000
SCU_APB0VSRC	SCU_BA+0x34	R	APB0 Security Policy Violation Source	0x0000_000X
SCU_APB0VA	SCU_BA+0x38	R	APB0 Violation Address	0xXXXX_XXXX
SCU_APB1VSRC	SCU_BA+0x3C	R	APB1 Security Policy Violation Source	0x0000_000X
SCU_APB1VA	SCU_BA+0x40	R	APB1 Violation Address	0xXXXX_XXXX
SCU_GPIOVSRC	SCU_BA+0x44	R	GPIO Security Policy Violation Source	0x0000_000X
SCU_GPIOVA	SCU_BA+0x48	R	GPIO Violation Address	0xXXXX_XXXX
SCU_EBIVSRC	SCU_BA+0x4C	R	EBI Security Policy Violation Source	0x0000_000X
SCU_EBIVA	SCU_BA+0x50	R	EBI Violation Address	0xXXXX_XXXX
SCU_USBHVSRC	SCU_BA+0x54	R	USBH Security Policy Violation Source	0x0000_000X
SCU_USBHVA	SCU_BA+0x58	R	USBH Violation Address	0xXXXX_XXXX
SCU_CRCVSRC	SCU_BA+0x5C	R	CRC Security Policy Violation Source	0x0000_000X
SCU_CRCVA	SCU_BA+0x60	R	CRC Violation Address	0xXXXX_XXXX
SCU_SD0VSRC	SCU_BA+0x64	R	SDH0 Security Policy Violation Source	0x0000_000X
SCU_SD0VA	SCU_BA+0x68	R	SDH0 Violation Address	0xXXXX_XXXX
SCU_PDMA0VSRC	SCU_BA+0x74	R	PDMA0 Security Policy Violation Source	0x0000_000X
SCU_PDMA0VA	SCU_BA+0x78	R	PDMA0 Violation Address	0xXXXX_XXXX
SCU_PDMA1VSRC	SCU_BA+0x7C	R	PDMA1 Security Policy Violation Source	0x0000_000X
SCU_PDMA1VA	SCU_BA+0x80	R	PDMA1 Violation Address	0xXXXX_XXXX
SCU_SRAM0VSRC	SCU_BA+0x84	R	SRAM0 Security Policy Violation Source	0x0000_000X
SCU_SRAM0VA	SCU_BA+0x88	R	SRAM0 Violation Address	0xXXXX_XXXX
SCU_SRAM1VSRC	SCU_BA+0x8C	R	SRAM1 Security Policy Violation Source	0x0000_000X
SCU_SRAM1VA	SCU_BA+0x90	R	SRAM1 Violation Address	0xXXXX_XXXX
SCU_FMCVSRC	SCU_BA+0x94	R	FMC Security Policy Violation Source	0x0000_000X
SCU_FMCVA	SCU_BA+0x98	R	FMC Violation Address	0xXXXX_XXXX
SCU_FLASHVSRC	SCU_BA+0x9C	R	Flash Security Policy Violation Source	0x0000_000X

SCU_FLASHVA	SCU_BA+0xA0	R	Flash Violation Address	0xXXXX_XXXX
SCU_SCUVSRC	SCU_BA+0xA4	R	SCU Security Policy Violation Source	0x0000_000X
SCU_SCUVA	SCU_BA+0xA8	R	SCU Violation Address	0xXXXX_XXXX
SCU_SYSVSRC	SCU_BA+0xAC	R	System Security Policy Violation Source	0x0000_000X
SCU_SYSVA	SCU_BA+0xB0	R	System Violation Address	0xXXXX_XXXX
SCU_CRPTVSRC	SCU_BA+0xB4	R	Crypto Security Policy Violation Source	0x0000_000X
SCU_CRPTVA	SCU_BA+0xB8	R	Crypto Violation Address	0xXXXX_XXXX
SCU_KSVSRC	SCU_BA+0xBC	R	KS Security Policy Violation Source	0x0000_000X
SCU_KSVA	SCU_BA+0xC0	R	KS Violation Address	0xXXXX_XXXX
SCU_SRAM2VSRC	SCU_BA+0xC4	R	SRAM2 Security Policy Violation Source	0x0000_000X
SCU_SRAM2VA	SCU_BA+0xC8	R	SRAM2 Violation Address	0xXXXX_XXXX
SCU_SINFAEN	SCU_BA+0xF0	R/W	Shared Information Access Enable Register	0x0000_0007
SCU_PNPSET0	SCU_BA+0x100	R/W	Peripheral Non-privileged Attribution Set Register0 (0x4000_0000~0x4001_FFFF)	0x0000_0000
SCU_PNPSET1	SCU_BA+0x104	R/W	Peripheral Non-privileged Attribution Set Register1 (0x4002_0000~0x4003_FFFF)	0x0000_0000
SCU_PNPSET2	SCU_BA+0x108	R/W	Peripheral Non-privileged Attribution Set Register2 (0x4004_0000~0x4005_FFFF)	0x0000_0000
SCU_PNPSET3	SCU_BA+0x10C	R/W	Peripheral Non-privileged Attribution Set Register3 (0x4006_0000~0x4007_FFFF)	0x0000_0000
SCU_PNPSET4	SCU_BA+0x110	R/W	Peripheral Non-privileged Attribution Set Register4 (0x4008_0000~0x4009_FFFF)	0x0000_0000
SCU_PNPSET5	SCU_BA+0x114	R/W	Peripheral Non-privileged Attribution Set Register5 (0x400A_0000~0x400B_FFFF)	0x0000_0000
SCU_PNPSET6	SCU_BA+0x118	R/W	Peripheral Non-privileged Attribution Set Register6 (0x400C_0000~0x400D_FFFF)	0x0000_0000
SCU_IONPSET	SCU_BA+0x120	R/W	I/O Non-privileged Attribution Set Register	0x0000_0000
SCU_SRAMNPSET	SCU_BA+0x124	R/W	SRAM Non-privileged Attribution Set Register	0x0000_0000
SCU_MEMNPSET	SCU_BA+0x128	R/W	Other Memory Non-privileged Attribution Set Register	0x0000_0000
SCU_PVIOIEN	SCU_BA+0x12C	R/W	Privileged Violation Interrupt Enable Register	0x0000_0000
SCU_PVINTSTS	SCU_BA+0x130	R/W	Privileged Violation Interrupt Status Register	0x0000_0000
SCU_SCWP	SCU_BA+0x134	R/W	Security Configuration Write Protection Register	0x0000_0000
SCU_IONSSET0	SCU_BA+0x140	R/W	I/O Non-secure Attribution Set Register0	0x0000_0000
SCU_IONSSET1	SCU_BA+0x144	R/W	I/O Non-secure Attribution Set Register1	0x0000_0000
SCU_IONSSET2	SCU_BA+0x148	R/W	I/O Non-secure Attribution Set Register2	0x0000_0000

SCU_IONSSET3	SCU_BA+0x14C	R/W	I/O Non-secure Attribution Set Register3	0x0000_0000
SCU_IONSSET4	SCU_BA+0x150	R/W	I/O Non-secure Attribution Set Register4	0x0000_0000
SCU_IONSSET5	SCU_BA+0x154	R/W	I/O Non-secure Attribution Set Register5	0x0000_0000
SCU_IONSSET6	SCU_BA+0x158	R/W	I/O Non-secure Attribution Set Register6	0x0000_0000
SCU_IONSSET7	SCU_BA+0x15C	R/W	I/O Non-secure Attribution Set Register7	0x0000_0000
SCU_NSMCTL	SCU_BA+0x200	R/W	Non-secure State Monitor Control Register	0x0000_1000
SCU_NSMLOAD	SCU_BA+0x204	R/W	Non-secure State Monitor Reload Value Register	0x00FF_FFFF
SCU_NSMVAL	SCU_BA+0x208	R/W	Non-secure State Monitor Counter Value Register	0x00FF_FFFF
SCU_NSMSTS	SCU_BA+0x20C	R/W	Non-secure State Monitor Status Register	0x0000_0000
FVC_CTL	FVC_BA+0x00	R/W	FVC Control Register	0x0000_000X
FVC_STS	FVC_BA+0x04	R	FVC Status Register	0x0000_000X
FVC_NVC0	FVC_BA+0x10	R/W	Non-volatile Version Counter Control Register0	0x0000_XXXX
FVC_NVC1	FVC_BA+0x14	R/W	Non-volatile Version Counter Control Register1	0x0000_XXXX
FVC_NVC4	FVC_BA+0x20	R/W	Non-volatile Version Counter Control Register4	0x0000_XXXX
FVC_NVC5	FVC_BA+0x24	R/W	Non-volatile Version Counter Control Register5	0x0000_XXXX
DPM_CTL	DPM_BA+0x00	R/W	Secure DPM Control Register	0xA500_000X
DPM_STS	DPM_BA+0x04	R/W	Secure DPM Status Register	0x000X_0X00
DPM_SPW0	DPM_BA+0x10	W	Secure DPM Password 0	0xFFFF_FFFF
DPM_SPW1	DPM_BA+0x14	W	Secure DPM Password 1	0xFFFF_FFFF
DPM_SPW2	DPM_BA+0x18	W	Secure DPM Password 2	0xFFFF_FFFF
DPM_SPW3	DPM_BA+0x1C	W	Secure DPM Password 3	0xFFFF_FFFF
DPM_NSCTL	DPM_BA+0x50	R/W	Non-secure DPM Control Register	0xA500_000X
DPM_NSSTS	DPM_BA+0x54	R/W	Non-secure DPM Status Register	0x000X_0X00
DPM_NSPW0	DPM_BA+0x60	W	Non-secure DPM Password 0	0xFFFF_FFFF
DPM_NSPW1	DPM_BA+0x64	W	Non-secure DPM Password 1	0xFFFF_FFFF
DPM_NSPW2	DPM_BA+0x68	W	Non-secure DPM Password 2	0xFFFF_FFFF
DPM_NSPW3	DPM_BA+0x6C	W	Non-secure DPM Password 3	0xFFFF_FFFF
PLM_CTL	PLM_BA+0x00	R/W	Product Life-cycle Control Register	0x0000_000X
PLM_STS	PLM_BA+0x04	R	Product Life-cycle Status Register	0x0000_000X

6.4.6 Register Description

Peripheral Non-secure Attribution Set Register0 (SCU_PNSSET0)

Register	Offset	R/W	Description	Reset Value
SCU_PNSSET0	SCU_BA+0x00	R/W	Peripheral Non-secure Attribution Set Register0 (0x4000_0000~0x4001_FFFF)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							PDMA1
23	22	21	20	19	18	17	16
Reserved							EBI
15	14	13	12	11	10	9	8
Reserved		SDH0	Reserved			USBH	Reserved
7	6	5	4	3	2	1	0
Reserved							

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24]	**PDMA1**	**Set PDMA1 to Non-secure State** Write 1 to set PDMA1 to non-secure state. 0 = PDMA1 is a secure module (default). 1 = PDMA1 is a non-secure module.
[23:17]	**Reserved**	Reserved.
[16]	**EBI**	**Set EBI to Non-secure State** Write 1 to set EBI to non-secure state. 0 = EBI is a secure module (default). 1 = EBI is a non-secure module.
[15:14]	**Reserved**	Reserved.
[13]	**SDH0**	**Set SDH0 to Non-secure State** Write 1 to set SDH0 to non-secure state. 0 = SDH0 is a secure module (default). 1 = SDH0 is a non-secure module.
[12:10]	**Reserved**	Reserved.
[9]	**USBH**	**Set USBH to Non-secure State** Write 1 to set USBH to non-secure state. 0 = USBH is a secure module (default). 1 = USBH is a non-secure module.
[8:0]	**Reserved**	Reserved.

Peripheral Non-secure Attribution Set Register1 (SCU_PNSSET1)

Register	Offset	R/W	Description	Reset Value
SCU_PNSSET1	SCU_BA+0x04	R/W	Peripheral Non-secure Attribution Set Register1 (0x4002_0000~0x4003_FFFF)	0x0000_0000

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

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18]	**CRPT**	**Set CRPT to Non-secure State** 0 = CRPT is a secure module (default). 1 = CRPT is a non-secure module.
[17]	**CRC**	**Set CRC to Non-secure State** Write 1 to set CRC to non-secure state. 0 = CRC is a secure module (default). 1 = CRC is a non-secure module.
[16:0]	**Reserved**	Reserved.

Peripheral Non-secure Attribution Set Register2 (SCU_PNSSET2)

Register	Offset	R/W	Description	Reset Value
SCU_PNSSET2	SCU_BA+0x08	R/W	Peripheral Non-secure Attribution Set Register2 (0x4004_0000~0x4005_FFFF)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				BPWM1	BPWM0	EPWM1	EPWM0
23	22	21	20	19	18	17	16
Reserved					TMR45	TMR23	Reserved
15	14	13	12	11	10	9	8
Reserved		OTG	Reserved				I2S0
7	6	5	4	3	2	1	0
DAC	Reserved	ACMP01	Reserved	EADC	EWDT	Reserved	

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27]	**BPWM1**	**Set BPWM1 to Non-secure State** Write 1 to set BPWM1 to non-secure state. 0 = BPWM1 is a secure module (default). 1 = BPWM1 is a non-secure module.
[26]	**BPWM0**	**Set BPWM0 to Non-secure State** Write 1 to set BPWM0 to non-secure state. 0 = BPWM0 is a secure module (default). 1 = BPWM0 is a non-secure module.
[25]	**EPWM1**	**Set EPWM1 to Non-secure State** Write 1 to set EPWM1 to non-secure state. 0 = EPWM1 is a secure module (default). 1 = EPWM1 is a non-secure module.
[24]	**EPWM0**	**Set EPWM0 to Non-secure State** Write 1 to set EPWM0 to non-secure state. 0 = EPWM0 is a secure module (default). 1 = EPWM0 is a non-secure module.
[23:19]	**Reserved**	Reserved.
[18]	**TMR45**	**Set TMR45 to Non-secure State** Write 1 to set TMR45 to non-secure state. 0 = TMR45 is a secure module (default). 1 = TMR45 is a non-secure module.

Bits	Description	
[17]	**TMR23**	**Set TMR23 to Non-secure State** Write 1 to set TMR23 to non-secure state. 0 = TMR23 is a secure module (default). 1 = TMR23 is a non-secure module.
[16:14]	**Reserved**	Reserved.
[13]	**OTG**	**Set OTG to Non-secure State** Write 1 to set OTG to non-secure state. 0 = OTG is a secure module (default). 1 = OTG is a non-secure module.
[12:9]	**Reserved**	Reserved.
[8]	**I2S0**	**Set I2S0 to Non-secure State** Write 1 to set I2S0 to non-secure state. 0 = I2S0 is a secure module (default). 1 = I2S0 is a non-secure module.
[7]	**DAC**	**Set DAC to Non-secure State** Write 1 to set DAC to non-secure state. 0 = DAC is a secure module (default). 1 = DAC is a non-secure module.
[6]	**Reserved**	Reserved.
[5]	**ACMP01**	**Set ACMP01 to Non-secure State** Write 1 to set ACMP0, ACMP1 to non-secure state. 0 = ACMP0, ACMP1 are secure modules (default). 1 = ACMP0, ACMP1 are non-secure modules.
[4]	**Reserved**	Reserved.
[3]	**EADC**	**Set EADC to Non-secure State** Write 1 to set EADC to non-secure state. 0 = EADC is a secure module (default). 1 = EADC is a non-secure module.
[2]	**EWDT**	**Set EWDT to Non-secure State** Write 1 to set EWDT to non-secure state. 0 = EWDT is a secure module (default). 1 = EWDT is a non-secure module.
[1:0]	**Reserved**	Reserved.

Peripheral Non-secure Attribution Set Register3 (SCU_PNSSET3)

Register	Offset	R/W	Description	Reset Value
SCU_PNSSET3	SCU_BA+0x0C	R/W	Peripheral Non-secure Attribution Set Register3 (0x4006_0000~0x4007_FFFF)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		UART5	UART4	UART3	UART2	UART1	UART0
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			SPI3	SPI2	SPI1	SPI0	QSPI0

Bits	Description	
[31:22]	**Reserved**	Reserved.
[21]	**UART5**	**Set UART5 to Non-secure State** Write 1 to set UART5 to non-secure state. 0 = UART5 is a secure module (default). 1 = UART5 is a non-secure module.
[20]	**UART4**	**Set UART4 to Non-secure State** Write 1 to set UART4 to non-secure state. 0 = UART4 is a secure module (default). 1 = UART4 is a non-secure module.
[19]	**UART3**	**Set UART3 to Non-secure State** Write 1 to set UART3 to non-secure state. 0 = UART3 is a secure module (default). 1 = UART3 is a non-secure module.
[18]	**UART2**	**Set UART2 to Non-secure State** Write 1 to set UART2 to non-secure state. 0 = UART2 is a secure module (default). 1 = UART2 is a non-secure module.
[17]	**UART1**	**Set UART1 to Non-secure State** Write 1 to set UART1 to non-secure state. 0 = UART1 is a secure module (default). 1 = UART1 is a non-secure module.
[16]	**UART0**	**Set UART0 to Non-secure State** Write 1 to set UART0 to non-secure state. 0 = UART0 is a secure module (default). 1 = UART0 is a non-secure module.

Bits	Description	
[15:5]	**Reserved**	Reserved.
[4]	**SPI3**	**Set SPI3 to Non-secure State** Write 1 to set SPI3 to non-secure state. 0 = SPI3 is a secure module (default). 1 = SPI3 is a non-secure module.
[3]	**SPI2**	**Set SPI2 to Non-secure State** Write 1 to set SPI2 to non-secure state. 0 = SPI2 is a secure module (default). 1 = SPI2 is a non-secure module.
[2]	**SPI1**	**Set SPI1 to Non-secure State** Write 1 to set SPI1 to non-secure state. 0 = SPI1 is a secure module (default). 1 = SPI1 is a non-secure module.
[1]	**SPI0**	**Set SPI0 to Non-secure State** Write 1 to set SPI0 to non-secure state. 0 = SPI0 is a secure module (default). 1 = SPI0 is a non-secure module.
[0]	**QSPI0**	**Set QSPI0 to Non-secure State** Write 1 to set QSPI0 to non-secure state. 0 = QSPI0 is a secure module (default). 1 = QSPI0 is a non-secure module.

Peripheral Non-secure Attribution Set Register4 (SCU_PNSSET4)

Register	Offset	R/W	Description	Reset Value
SCU_PNSSET4	SCU_BA+0x10	R/W	Peripheral Non-secure Attribution Set Register4 (0x4008_0000~0x4009_FFFF)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					SC2	SC1	SC0
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					I2C2	I2C1	I2C0

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18]	**SC2**	**Set SC2 to Non-secure State** Write 1 to set SC2 to non-secure state. 0 = SC2 is a secure module (default). 1 = SC2 is a non-secure module.
[17]	**SC1**	**Set SC1 to Non-secure State** Write 1 to set SC1 to non-secure state. 0 = SC1 is a secure module (default). 1 = SC1 is a non-secure module.
[16]	**SC0**	**Set SC0 to Non-secure State** Write 1 to set SC0 to non-secure state. 0 = SC0 is a secure module (default). 1 = SC0 is a non-secure module.
[15:3]	**Reserved**	Reserved.
[2]	**I2C2**	**Set I2C2 to Non-secure State** Write 1 to set I2C2 to non-secure state. 0 = I2C2 is a secure module (default). 1 = I2C2 is a non-secure module.
[1]	**I2C1**	**Set I2C1 to Non-secure State** Write 1 to set I2C1 to non-secure state. 0 = I2C1 is a secure module (default). 1 = I2C1 is a non-secure module.

Bits	Description	
[0]	**I2C0**	**Set I2C0 to Non-secure State** Write 1 to set I2C0 to non-secure state. 0 = I2C0 is a secure module (default). 1 = I2C0 is a non-secure module.

Peripheral Non-secure Attribution Set Register5 (SCU_PNSSET5)

Register	Offset	R/W	Description	Reset Value
SCU_PNSSET5	SCU_BA+0x14	R/W	Peripheral Non-secure Attribution Set Register5 (0x400A_0000~0x400B_FFFF)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				LCD	Reserved	TRNG	Reserved
23	22	21	20	19	18	17	16
Reserved		ECAP1	ECAP0	Reserved		QEI1	QEI0
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							CAN0

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27]	**LCD**	**Set LCD to Non-secure State** Write 1 to set LCD to non-secure state. 0 = LCD is a secure module (default). 1 = LCD is a non-secure module.
[26]	**Reserved**	Reserved.
[25]	**TRNG**	**Set TRNG to Non-secure State** Write 1 to set TRNG to non-secure state. 0 = TRNG is a secure module (default). 1 = TRNG is a non-secure module.
[24:22]	**Reserved**	Reserved.
[21]	**ECAP1**	**Set ECAP1 to Non-secure State** Write 1 to set ECAP1 to non-secure state. 0 = ECAP1 is a secure module (default). 1 = ECAP1 is a non-secure module.
[20]	**ECAP0**	**Set ECAP0 to Non-secure State** Write 1 to set ECAP0 to non-secure state. 0 = ECAP0 is a secure module (default). 1 = ECAP0 is a non-secure module.
[19:18]	**Reserved**	Reserved.
[17]	**QEI1**	**Set QEI1 to Non-secure State** Write 1 to set QEI1 to non-secure state. 0 = QEI1 is a secure module (default). 1 = QEI1 is a non-secure module.

Bits	Description	
[16]	**QEI0**	**Set QEI0 to Non-secure State** Write 1 to set QEI0 to non-secure state. 0 = QEI0 is a secure module (default). 1 = QEI0 is a non-secure module.
[15:1]	**Reserved**	Reserved.
[0]	**CAN0**	**Set CAN0 to Non-secure State** Write 1 to set CAN0 to non-secure state. 0 = CAN0 is a secure module (default). 1 = CAN0 is a non-secure module.

Peripheral Non-secure Attribution Set Register6 (SCU_PNSSET6)

Register	Offset	R/W	Description	Reset Value
SCU_PNSSET6	SCU_BA+0x18	R/W	Peripheral Non-secure Attribution Set Register6 (0x400C_0000~0x400D_FFFF)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						USCI1	USCI0
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							USBD

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**USCI1**	**Set USCI1 to Non-secure State** Write 1 to set USCI1 to non-secure state. 0 = USCI1 is a secure module (default). 1 = USCI1 is a non-secure module.
[16]	**USCI0**	**Set USCI0 to Non-secure State** Write 1 to set USCI0 to non-secure state. 0 = USCI0 is a secure module (default). 1 = USCI0 is a non-secure module.
[15:1]	**Reserved**	Reserved.
[0]	**USBD**	**Set USBD to Non-secure State** Write 1 to set USBD to non-secure state. 0 = USBD is a secure module (default). 1 = USBD is a non-secure module.

SRAM Non-secure Attribution Set Register (SCU_SRAMNSSET)

Register	Offset	R/W	Description	Reset Value
SCU_SRAMNSSET	SCU_BA+0x24	R/W	SRAM Non-secure Attribution Set Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8
7	6	5	4	3	2	1	0
SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0

Bits	Description	
[31:15]	**Reserved**	Reserved.
[14:0]	**SECn**	**Set SRAM Section n to Non-secure State** Write 1 to set SRAM section **n** to non-secure state. Write 0 is ignored. 0 = SRAM Section **n** is secure (default). 1 = SRAM Section **n** is non-secure. Size per section is 16 Kbytes. Secure SRAM section **n** is 0x2000_0000+0x4000***n** to 0x2000_0000+0x4000*(**n**+1)-0x1 Non-secure SRAM section **n** is 0x3000_0000+0x4000***n** to 0x3000_0000+0x4000*(**n**+1)-0x1

Flash Non-secure Boundary Address Register (SCU_FNSADDR)

Register	Offset	R/W	Description	Reset Value
SCU_FNSADDR	SCU_BA+0x28	R	Flash Non-secure Boundary Address Register	0xXXXX_XXXX

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

Bits	Description	
[31:0]	FNSADDR	**Flash Non-secure Boundary Address** Indicate the base address of Non-secure region set in user configuration. Refer to FMC section for more details.

Security Violation Interrupt Enable Register (SCU_SVIOIEN)

Register	Offset	R/W	Description	Reset Value
SCU_SVIOIEN	SCU_BA+0x2C	R/W	Security Violation Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved			SRAM2IEN	KSIEN	CRPTIEN	SYSIEN	SCUIEN
15	14	13	12	11	10	9	8
FLASHIEN	FMCIEN	SRAM1IEN	SRAM0IEN	PDMA1IEN	PDMA0IEN	Reserved	SDH0IEN
7	6	5	4	3	2	1	0
CRCIEN	USBHIEN	EBIIEN	GPIOIEN	Reserved		APB1IEN	APB0IEN

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20]	**SRAM2IEN**	**SRAM2 Security Violation Interrupt Enable Bit** 0 = Interrupt triggered from security violation of SRAM2 Disabled. 1 = Interrupt triggered from security violation of SRAM2 Enabled.
[19]	**KSIEN**	**KS Security Violation Interrupt Enable Bit** 0 = Interrupt triggered from security violation of keystore Disabled. 1 = Interrupt triggered from security violation of keystore Enabled.
[18]	**CRPTIEN**	**CRPT Security Violation Interrupt Enable Bit** 0 = Interrupt triggered from security violation of crypto Disabled. 1 = Interrupt triggered from security violation of crypto Enabled.
[17]	**SYSIEN**	**SYS Security Violation Interrupt Enable Bit** 0 = Interrupt triggered from security violation of system manager Disabled. 1 = Interrupt triggered from security violation of system manager Enabled.
[16]	**SCUIEN**	**SCU Security Violation Interrupt Enable Bit** 0 = Interrupt triggered from security violation of SCU Disabled. 1 = Interrupt triggered from security violation of SCU Enabled.
[15]	**FLASHIEN**	**FLASH Security Violation Interrupt Enable Bit** 0 = Interrupt triggered from security violation of Flash data Disabled. 1 = Interrupt triggered from security violation of Flash data Enabled.
[14]	**FMCIEN**	**FMC Security Violation Interrupt Enable Bit** 0 = Interrupt triggered from security violation of FMC Disabled. 1 = Interrupt triggered from security violation of FMC Enabled.
[13]	**SRAM1IEN**	**SRAM Bank 1 Security Violation Interrupt Enable Bit** 0 = Interrupt triggered from security violation of SRAM bank1 Disabled. 1 = Interrupt triggered from security violation of SRAM bank1 Enabled.

Bits	Description	
[12]	**SRAM0IEN**	**SRAM Bank 0 Security Violation Interrupt Enable Bit** 0 = Interrupt triggered from security violation of SRAM bank0 Disabled. 1 = Interrupt triggered from security violation of SRAM bank0 Enabled.
[11]	**PDMA1IEN**	**PDMA1 Security Violation Interrupt Enable Bit** 0 = Interrupt triggered from security violation of PDMA1 Disabled. 1 = Interrupt triggered from security violation of PDMA1 Enabled.
[10]	**PDMA0IEN**	**PDMA0 Security Violation Interrupt Enable Bit** 0 = Interrupt triggered from security violation of PDMA0 Disabled. 1 = Interrupt triggered from security violation of PDMA0 Enabled.
[9]	**Reserved**	Reserved.
[8]	**SDH0IEN**	**SDH0 Security Violation Interrupt Enable Bit** 0 = Interrupt triggered from security violation of SD host 0 Disabled. 1 = Interrupt triggered from security violation of SD host 0 Enabled.
[7]	**CRCIEN**	**CRC Security Violation Interrupt Enable Bit** 0 = Interrupt triggered from security violation of CRC Disabled. 1 = Interrupt triggered from security violation of CRC Enabled.
[6]	**USBHIEN**	**USBH Security Violation Interrupt Enable Bit** 0 = Interrupt triggered from security violation of USB host Disabled. 1 = Interrupt triggered from security violation of USB host Enabled.
[5]	**EBIIEN**	**EBI Security Violation Interrupt Enable Bit** 0 = Interrupt triggered from security violation of EBI Disabled. 1 = Interrupt triggered from security violation of EBI Enabled.
[4]	**GPIOIEN**	**GPIO Security Violation Interrupt Enable Bit** 0 = Interrupt triggered from security violation of GPIO Disabled. 1 = Interrupt triggered from security violation of GPIO Enabled.
[3:2]	**Reserved**	Reserved.
[1]	**APB1IEN**	**APB1 Security Violation Interrupt Enable Bit** 0 = Interrupt triggered from security violation of APB1 Disabled. 1 = Interrupt triggered from security violation of APB1 Enabled.
[0]	**APB0IEN**	**APB0 Security Violation Interrupt Enable Bit** 0 = Interrupt triggered from security violation of APB0 Disabled. 1 = Interrupt triggered from security violation of APB0 Enabled.

Security Violation Interrupt Status Register (SCU_SVINTSTS)

Register	Offset	R/W	Description	Reset Value
SCU_SVINTSTS	SCU_BA+0x30	R/W	Security Violation Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved			SRAM2IF	KSIF	CRPTIF	SYSIF	SCUIF
15	14	13	12	11	10	9	8
FLASHIF	FMCIF	SRAM1IF	SRAM0IF	PDMA1IF	PDMA0IF	Reserved	SDH0IF
7	6	5	4	3	2	1	0
CRCIF	USBHIF	EBIIF	GPIOIF	Reserved		APB1IF	APB0IF

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20]	**SRAM2IF**	**SRAM Bank 2 Security Violation Interrupt Status** 0 = No SRAM Bank 2 violation interrupt event. 1 = There is at least a SRAM Bank 2 violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[19]	**KSIF**	**KS Security Violation Interrupt Status** 0 = No KS violation interrupt event. 1 = There is at least a KS violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[18]	**CRPTIF**	**CRPT Security Violation Interrupt Status** 0 = No CRPT violation interrupt event. 1 = There is at least a CRPT violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[17]	**SYSIF**	**SYS Security Violation Interrupt Status** 0 = No SYS violation interrupt event. 1 = There is at least a SYS violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[16]	**SCUIF**	**SCU Security Violation Interrupt Status** 0 = No SCU violation interrupt event. 1 = There is at least a SCU violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[15]	**FLASHIF**	**FLASH Security Violation Interrupt Status** 0 = No FLASH violation interrupt event. 1 = There is at least a FLASH violation interrupt event. **Note:** Write 1 to clear the interrupt flag.

Bits	Description	
[14]	**FMCIF**	**FMC Security Violation Interrupt Status** 0 = No FMC violation interrupt event. 1 = There is at least a FMC violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[13]	**SRAM1IF**	**SRAM Bank 1 Security Violation Interrupt Status** 0 = No SRAM Bank 1 violation interrupt event. 1 = There is at least a SRAM Bank 1 violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[12]	**SRAM0IF**	**SRAM Bank 0 Security Violation Interrupt Status** 0 = No SRAM Bank 0 violation interrupt event. 1 = There is at least a SRAM Bank 0 violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[11]	**PDMA1IF**	**PDMA1 Security Violation Interrupt Status** 0 = No PDMA1 violation interrupt event. 1 = There is at least a PDMA1 violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[10]	**PDMA0IF**	**PDMA0 Security Violation Interrupt Status** 0 = No PDMA0 violation interrupt event. 1 = There is at least a PDMA0 violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[9]	**Reserved**	Reserved.
[8]	**SDH0IF**	**SDH0 Security Violation Interrupt Status** 0 = No SDH0 violation interrupt event. 1 = There is at least a SDH0 violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[7]	**CRCIF**	**CRC Security Violation Interrupt Status** 0 = No CRC violation interrupt event. 1 = There is at least a CRC violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[6]	**USBHIF**	**USBH Security Violation Interrupt Status** 0 = No USBH violation interrupt event. 1 = There is at least a USBH violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[5]	**EBIIF**	**EBI Security Violation Interrupt Status** 0 = No EBI violation interrupt event. 1 = There is at least a EBI violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[4]	**GPIOIF**	**GPIO Security Violation Interrupt Status** 0 = No GPIO violation interrupt event. 1 = There is at least a GPIO violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[3:2]	**Reserved**	Reserved.

Bits	Description	
[1]	**APB1IF**	**APB1 Security Violation Interrupt Status** 0 = No APB1 violation interrupt event. 1 = There is at least a APB1 violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[0]	**APB0IF**	**APB0 Security Violation Interrupt Status** 0 = No APB0 violation interrupt event. 1 = There is at least a APB0 violation interrupt event. **Note:** Write 1 to clear the interrupt flag.

Peripheral Security Violation Master Register (SCU_PVSRC)

Register	Offset	R/W	Description	Reset Value
SCU_APB0VSRC	SCU_BA+0x34	R	APB0 Security Policy Violation Source	0x0000_000X
SCU_APB1VSRC	SCU_BA+0x3C	R	APB1 Security Policy Violation Source	0x0000_000X
SCU_GPIOVSRC	SCU_BA+0x44	R	GPIO Security Policy Violation Source	0x0000_000X
SCU_EBIVSRC	SCU_BA+0x4C	R	EBI Security Policy Violation Source	0x0000_000X
SCU_USBHVSRC	SCU_BA+0x54	R	USBH Security Policy Violation Source	0x0000_000X
SCU_CRCVSRC	SCU_BA+0x5C	R	CRC Security Policy Violation Source	0x0000_000X
SCU_SD0VSRC	SCU_BA+0x64	R	SDH0 Security Policy Violation Source	0x0000_000X
SCU_PDMA0VSRC	SCU_BA+0x74	R	PDMA0 Security Policy Violation Source	0x0000_000X
SCU_PDMA1VSRC	SCU_BA+0x7C	R	PDMA1 Security Policy Violation Source	0x0000_000X
SCU_SRAM0VSRC	SCU_BA+0x84	R	SRAM0 Security Policy Violation Source	0x0000_000X
SCU_SRAM1VSRC	SCU_BA+0x8C	R	SRAM1 Security Policy Violation Source	0x0000_000X
SCU_FMCVSRC	SCU_BA+0x94	R	FMC Security Policy Violation Source	0x0000_000X
SCU_FLASHVSRC	SCU_BA+0x9C	R	Flash Security Policy Violation Source	0x0000_000X
SCU_SCUVSRC	SCU_BA+0xA4	R	SCU Security Policy Violation Source	0x0000_000X
SCU_SYSVSRC	SCU_BA+0xAC	R	System Security Policy Violation Source	0x0000_000X
SCU_CRPTVSRC	SCU_BA+0xB4	R	Crypto Security Policy Violation Source	0x0000_000X
SCU_KSVSRC	SCU_BA+0xBC	R	KS Security Policy Violation Source	0x0000_000X
SCU_SRAM2VSRC	SCU_BA+0xC4	R	SRAM2 Security Policy Violation Source	0x0000_000X

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

Bits	Description

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3:0]	**MASTER**	**Master Violating Security Policy** Indicate which master invokes the security violation. 0x0 = core processor. 0x3 = PDMA0. 0x4 = SDH0. 0x5 = CRYPTO. 0x6 = USH. 0xB = PDMA1. Others is undefined.

Peripheral Security Violation Address Register (SCU_PVA)

Register	Offset	R/W	Description	Reset Value
SCU_APB0VA	SCU_BA+0x38	R	APB0 Violation Address	0xXXXX_XXXX
SCU_APB1VA	SCU_BA+0x40	R	APB1 Violation Address	0xXXXX_XXXX
SCU_GPIOVA	SCU_BA+0x48	R	GPIO Violation Address	0xXXXX_XXXX
SCU_EBIVA	SCU_BA+0x50	R	EBI Violation Address	0xXXXX_XXXX
SCU_USBHVA	SCU_BA+0x58	R	USBH Violation Address	0xXXXX_XXXX
SCU_CRCVA	SCU_BA+0x60	R	CRC Violation Address	0xXXXX_XXXX
SCU_SD0VA	SCU_BA+0x68	R	SDH0 Violation Address	0xXXXX_XXXX
SCU_PDMA0VA	SCU_BA+0x78	R	PDMA0 Violation Address	0xXXXX_XXXX
SCU_PDMA1VA	SCU_BA+0x80	R	PDMA1 Violation Address	0xXXXX_XXXX
SCU_SRAM0VA	SCU_BA+0x88	R	SRAM0 Violation Address	0xXXXX_XXXX
SCU_SRAM1VA	SCU_BA+0x90	R	SRAM1 Violation Address	0xXXXX_XXXX
SCU_FMCVA	SCU_BA+0x98	R	FMC Violation Address	0xXXXX_XXXX
SCU_FLASHVA	SCU_BA+0xA0	R	Flash Violation Address	0xXXXX_XXXX
SCU_SCUVA	SCU_BA+0xA8	R	SCU Violation Address	0xXXXX_XXXX
SCU_SYSVA	SCU_BA+0xB0	R	System Violation Address	0xXXXX_XXXX
SCU_CRPTVA	SCU_BA+0xB8	R	Crypto Violation Address	0xXXXX_XXXX
SCU_KSVA	SCU_BA+0xC0	R	KS Violation Address	0xXXXX_XXXX
SCU_SRAM2VA	SCU_BA+0xC8	R	SRAM2 Violation Address	0xXXXX_XXXX

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

Bits	Description

Bits	Description	
[31:0]	**VIOADDR**	**Violation Address** Indicate the target address of the access, which invokes the security violation.

Shared Information Access Enable Register (SCU_SINFAEN)

Register	Offset	R/W	Description	Reset Value
SCU_SINFAEN	SCU_BA+0xF0	R/W	Shared Information Access Enable Register	0x0000_0007

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

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**FMCSIAEN**	**FMC Shared Information Access Enable Bit** 0 = Non-secure CPU access FMC Shared information Disabled. 1 = Non-secure CPU access FMC Shared information Enabled.
[1]	**SYSSIAEN**	**SYS Shared Information Access Enable Bit** 0 = Non-secure CPU access SYS Shared information Disabled. 1 = Non-secure CPU access SYS Shared information Enabled. **Note:** Include clock information.
[0]	**SCUSIAEN**	**SCU Shared Information Access Enable Bit** 0 = Non-secure CPU access SCU Shared information Disabled. 1 = Non-secure CPU access SCU Shared information Enabled.

Peripheral Non-privileged Attribution Set Register0 (SCU_PNPSET0)

Register	Offset	R/W	Description	Reset Value
SCU_PNPSET0	SCU_BA+0x100	R/W	Peripheral Non-privileged Attribution Set Register0 (0x4000_0000~0x4001_FFFF)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							PDMA1
23	22	21	20	19	18	17	16
Reserved							EBI
15	14	13	12	11	10	9	8
Reserved		SDH0	FMC	Reserved		USBH	PDMA0
7	6	5	4	3	2	1	0
Reserved							SYS

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24]	**PDMA1**	**Set PDMA1 to Non-privileged State** 0 = PDMA1 is a privileged module (default). 1 = PDMA1 is a non-privileged module.
[23:17]	**Reserved**	Reserved.
[16]	**EBI**	**Set EBI to Non-privileged State** 0 = EBI is a privileged module (default). 1 = EBI is a non-privileged module.
[15:14]	**Reserved**	Reserved.
[13]	**SDH0**	**Set SDH0 to Non-privileged State** 0 = SDH0 is a privileged module (default). 1 = SDH0 is a non-privileged module.
[12]	**FMC**	**Set FMC to Non-privileged State** 0 = FMC is a privileged module (default). 1 = FMC is a non-privileged module.
[11:10]	**Reserved**	Reserved.
[9]	**USBH**	**Set USBH to Non-privileged State** 0 = USBH is a privileged module (default). 1 = USBH is a non-privileged module.
[8]	**PDMA0**	**Set PDMA0 to Non-privileged State** 0 = PDMA0 is a privileged module (default). 1 = PDMA0 is a non-privileged module.
[7:1]	**Reserved**	Reserved.

Bits	Description	
[0]	**SYS**	**Set SYS to Non-privileged State** 0 = SYS is a privileged module (default). 1 = SYS is a non-privileged module.

Peripheral Non-privileged Attribution Set Register1 (SCU_PNPSET1)

Register	Offset	R/W	Description	Reset Value
SCU_PNPSET1	SCU_BA+0x104	R/W	Peripheral Non-privileged Attribution Set Register1 (0x4002_0000~0x4003_FFFF)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		KS	Reserved		CRPT	CRC	Reserved
15	14	13	12	11	10	9	8
SCU	Reserved						
7	6	5	4	3	2	1	0
Reserved							

Bits	Description	
[31:22]	**Reserved**	Reserved.
[21]	**KS**	**Set KS to Non-privileged State** 0 = KS is a privileged module (default). 1 = KS is a non-privileged module.
[:19]	**Reserved**	Reserved.
[18]	**CRPT**	**Set CRPT to Non-privileged State** 0 = CRPT is a privileged module (default). 1 = CRPT is a non-privileged module.
[17]	**CRC**	**Set CRC to Non-privileged State** 0 = CRC is a privileged module (default). 1 = CRC is a non-privileged module.
[16]	**Reserved**	Reserved.
[15]	**SCU**	**Set SCU to Non-privileged State** 0 = SCU is a privileged module (default). 1 = SCU is a non-privileged module.
[14:0]	**Reserved**	Reserved.

Peripheral Non-privileged Attribution Set Register2 (SCU_PNPSET2)

Register	Offset	R/W	Description	Reset Value
SCU_PNPSET2	SCU_BA+0x108	R/W	Peripheral Non-privileged Attribution Set Register2 (0x4004_0000~0x4005_FFFF)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				BPWM1	BPWM0	EPWM1	EPWM0
23	22	21	20	19	18	17	16
Reserved					TMR45	TMR23	TMR01
15	14	13	12	11	10	9	8
Reserved		OTG	Reserved				I2S0
7	6	5	4	3	2	1	0
DAC	Reserved	ACMP01	Reserved	EADC	EWDT	RTC	WDT

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27]	**BPWM1**	**Set BPWM1 to Non-privileged State** 0 = BPWM1 is a privileged module (default). 1 = BPWM1 is a non-privileged module.
[26]	**BPWM0**	**Set BPWM0 to Non-privileged State** 0 = BPWM0 is a privileged module (default). 1 = BPWM0 is a non-privileged module.
[25]	**EPWM1**	**Set EPWM1 to Non-privileged State** 0 = EPWM1 is a privileged module (default). 1 = EPWM1 is a non-privileged module.
[24]	**EPWM0**	**Set EPWM0 to Non-privileged State** 0 = EPWM0 is a privileged module (default). 1 = EPWM0 is a non-privileged module.
[23:19]	**Reserved**	Reserved.
[18]	**TMR45**	**Set TMR45 to Non-privileged State** 0 = TMR45 is a privileged module (default). 1 = TMR45 is a non-privileged module.
[17]	**TMR23**	**Set TMR23 to Non-privileged State** 0 = TMR23 is a privileged module (default). 1 = TMR23 is a non-privileged module.
[16:14]	**TMR01**	**Set TMR01 to Non-privileged State** 0 = TMR01 is a privileged module (default). 1 = TMR01 is a non-privileged module.

Bits	Description	
[13]	**OTG**	**Set OTG to Non-privileged State** 0 = OTG is a privileged module (default). 1 = OTG is a non-privileged module.
[12:9]	**Reserved**	Reserved.
[8]	**I2S0**	**Set I2S0 to Non-privileged State** 0 = I2S0 is a privileged module (default). 1 = I2S0 is a non-privileged module.
[7]	**DAC**	**Set DAC to Non-privileged State** 0 = DAC is a privileged module (default). 1 = DAC is a non-privileged module.
[6]	**Reserved**	Reserved.
[5]	**ACMP01**	**Set ACMP01 to Non-privileged State** 0 = ACMP0, ACMP1 are privileged modules (default). 1 = ACMP0, ACMP1 are non-privileged modules.
[4]	**Reserved**	Reserved.
[3]	**EADC**	**Set EADC to Non-privileged State** 0 = EADC is a privileged module (default). 1 = EADC is a non-privileged module.
[2]	**EWDT**	**Set EWDT to Non-privileged State** 0 = EWDT is a privileged module (default). 1 = EWDT is a non-privileged module.
[1]	**RTC**	**Set RTC to Non-privileged State** 0 = RTC is a privileged module (default). 1 = RTC is a non-privileged module.
[0]	**WDT**	**Set WDT to Non-privileged State** 0 = WDT is a privileged module (default). 1 = WDT is a non-privileged module.

Peripheral Non-privileged Attribution Set Register3 (SCU_PNPSET3)

Register	Offset	R/W	Description	Reset Value
SCU_PNPSET3	SCU_BA+0x10C	R/W	Peripheral Non-privileged Attribution Set Register3 (0x4006_0000~0x4007_FFFF)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		UART5	UART4	UART3	UART2	UART1	UART0
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			SPI3	SPI2	SPI1	SPI0	QSPI0

Bits	Description	
[31:22]	**Reserved**	Reserved.
[21]	**UART5**	**Set UART5 to Non-privileged State** 0 = UART5 is a privileged module (default). 1 = UART5 is a non-privileged module.
[20]	**UART4**	**Set UART4 to Non-privileged State** 0 = UART4 is a privileged module (default). 1 = UART4 is a non-privileged module.
[19]	**UART3**	**Set UART3 to Non-privileged State** 0 = UART3 is a privileged module (default). 1 = UART3 is a non-privileged module.
[18]	**UART2**	**Set UART2 to Non-privileged State** 0 = UART2 is a privileged module (default). 1 = UART2 is a non-privileged module.
[17]	**UART1**	**Set UART1 to Non-privileged State** 0 = UART1 is a privileged module (default). 1 = UART1 is a non-privileged module.
[16]	**UART0**	**Set UART0 to Non-privileged State** 0 = UART0 is a privileged module (default). 1 = UART0 is a non-privileged module.
[15:5]	**Reserved**	Reserved.
[4]	**SPI3**	**Set SPI3 to Non-privileged State** 0 = SPI3 is a privileged module (default). 1 = SPI3 is a non-privileged module.

Bits	Description	
[3]	**SPI2**	**Set SPI2 to Non-privileged State** 0 = SPI2 is a privileged module (default). 1 = SPI2 is a non-privileged module.
[2]	**SPI1**	**Set SPI1 to Non-privileged State** 0 = SPI1 is a privileged module (default). 1 = SPI1 is a non-privileged module.
[1]	**SPI0**	**Set SPI0 to Non-privileged State** 0 = SPI0 is a privileged module (default). 1 = SPI0 is a non-privileged module.
[0]	**QSPI0**	**Set QSPI0 to Non-privileged State** 0 = QSPI0 is a privileged module (default). 1 = QSPI0 is a non-privileged module.

Peripheral Non-privileged Attribution Set Register4 (SCU_PNPSET4)

Register	Offset	R/W	Description	Reset Value
SCU_PNPSET4	SCU_BA+0x110	R/W	Peripheral Non-privileged Attribution Set Register4 (0x4008_0000~0x4009_FFFF)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					SC2	SC1	SC0
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					I2C2	I2C1	I2C0

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18]	**SC2**	**Set SC2 to Non-privileged State** 0 = SC2 is a privileged module (default). 1 = SC2 is a non-privileged module.
[17]	**SC1**	**Set SC1 to Non-privileged State** 0 = SC1 is a privileged module (default). 1 = SC1 is a non-privileged module.
[16]	**SC0**	**Set SC0 to Non-privileged State** 0 = SC0 is a privileged module (default). 1 = SC0 is a non-privileged module.
[15:3]	**Reserved**	Reserved.
[2]	**I2C2**	**Set I2C2 to Non-privileged State** 0 = I2C2 is a privileged module (default). 1 = I2C2 is a non-privileged module.
[1]	**I2C1**	**Set I2C1 to Non-privileged State** 0 = I2C1 is a privileged module (default). 1 = I2C1 is a non-privileged module.
[0]	**I2C0**	**Set I2C0 to Non-privileged State** 0 = I2C0 is a privileged module (default). 1 = I2C0 is a non-privileged module.

Peripheral Non-privileged Attribution Set Register5 (SCU_PNPSET5)

Register	Offset	R/W	Description	Reset Value
SCU_PNPSET5	SCU_BA+0x114	R/W	Peripheral Non-privileged Attribution Set Register5 (0x400A_0000~0x400B_FFFF)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		TAMPER	Reserved	LCD	Reserved	TRNG	Reserved
23	22	21	20	19	18	17	16
Reserved		ECAP1	ECAP0	Reserved		QEI1	QEI0
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							CAN0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29]	**TAMPER**	**Set TAMPER to Non-privileged State** 0 = TAMPER is a privileged module (default). 1 = TAMPER is a non-privileged module.
[28]	**Reserved**	Reserved.
[27]	**LCD**	**Set LCD to Non-privileged State** 0 = LCD is a privileged module (default). 1 = LCD is a non-privileged module.
[26]	**Reserved**	Reserved.
[25]	**TRNG**	**Set TRNG to Non-privileged State** 0 = TRNG is a privileged module (default). 1 = TRNG is a non-privileged module.
[24:22]	**Reserved**	Reserved.
[21]	**ECAP1**	**Set ECAP1 to Non-privileged State** 0 = ECAP1 is a privileged module (default). 1 = ECAP1 is a non-privileged module.
[20]	**ECAP0**	**Set ECAP0 to Non-privileged State** 0 = ECAP0 is a privileged module (default). 1 = ECAP0 is a non-privileged module.
[19:18]	**Reserved**	Reserved.
[17]	**QEI1**	**Set QEI1 to Non-privileged State** 0 = QEI1 is a privileged module (default). 1 = QEI1 is a non-privileged module.

Bits	Description	
[16]	**QEI0**	**Set QEI0 to Non-privileged State** 0 = QEI0 is a privileged module (default). 1 = QEI0 is a non-privileged module.
[15:1]	**Reserved**	Reserved.
[0]	**CAN0**	**Set CAN0 to Non-privileged State** 0 = CAN0 is a privileged module (default). 1 = CAN0 is a non-privileged module.

Peripheral Non-privileged Attribution Set Register6 (SCU_PNPSET6)

Register	Offset	R/W	Description	Reset Value
SCU_PNPSET6	SCU_BA+0x118	R/W	Peripheral Non-privileged Attribution Set Register6 (0x400C_0000~0x400D_FFFF)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						USCI1	USCI0
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							USBD

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**USCI1**	**Set USCI1 to Non-privileged State** 0 = USCI1 is a privileged module (default). 1 = USCI1 is a non-privileged module.
[16]	**USCI0**	**Set USCI0 to Non-privileged State** 0 = USCI0 is a privileged module (default). 1 = USCI0 is a non-privileged module.
[15:1]	**Reserved**	Reserved.
[0]	**USBD**	**Set USBD to Non-privileged State** 0 = USBD is a privileged module (default). 1 = USBD is a non-privileged module.

I/O Non-privileged Attribution Set Register (SCU_IONPSET)

Register	Offset	R/W	Description	Reset Value
SCU_IONPSET	SCU_BA+0x120	R/W	I/O Non-privileged Attribution Set Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
PH	PG	PF	PE	PD	PC	PB	PA

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**PH**	**Set GPIO Port H to Non-privileged State** 0 = GPIO port H is privileged (default). 1 = GPIO port H is non-privileged.
[6]	**PG**	**Set GPIO Port G to Non-privileged State** 0 = GPIO port G is privileged (default). 1 = GPIO port G is non-privileged.
[5]	**PF**	**Set GPIO Port F to Non-privileged State** 0 = GPIO port F is privileged (default). 1 = GPIO port F is non-privileged.
[4]	**PE**	**Set GPIO Port E to Non-privileged State** 0 = GPIO port E is privileged (default). 1 = GPIO port E is non-privileged.
[3]	**PD**	**Set GPIO Port D to Non-privileged State** 0 = GPIO port D is privileged (default). 1 = GPIO port D is non-privileged.
[2]	**PC**	**Set GPIO Port C to Non-privileged State** 0 = GPIO port C is privileged (default). 1 = GPIO port C is non-privileged.
[1]	**PB**	**Set GPIO Port B to Non-privileged State** 0 = GPIO port B is privileged (default). 1 = GPIO port B is non-privileged.
[0]	**PA**	**Set GPIO Port a to Non-privileged State** 0 = GPIO port A is privileged (default). 1 = GPIO port A is non-privileged.

SRAM Non-privileged Attribution Set Register (SCU_SRAMNPSET)

Register	Offset	R/W	Description	Reset Value
SCU_SRAMNPSET	SCU_BA+0x124	R/W	SRAM Non-privileged Attribution Set Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SEC15	**SEC14**	**SEC13**	**SEC12**	**SEC11**	**SEC10**	**SEC9**	**SEC8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SEC7	**SEC6**	**SEC5**	**SEC4**	**SEC3**	**SEC2**	**SEC1**	**SEC0**

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**SECn**	**Set SRAM Section n to Non-privileged State** 0 = SRAM Section **n** is privileged (default). 1 = SRAM Section **n** is non-privileged. Size per section is 16 Kbytes. Secure SRAM section **n** is 0x2000_0000+0x4000***n** to 0x2000_0000+0x4000*(**n**+1)-0x1 Non-secure SRAM section **n** is 0x3000_0000+0x4000***n** to 0x3000_0000+0x4000*(**n**+1)-0x1

Other Memory Non-privileged Attribution Set Register (SCU_MEMNPSET)

Register	Offset	R/W	Description	Reset Value
SCU_MEMNPSET	SCU_BA+0x128	R/W	Other Memory Non-privileged Attribution Set Register	0x0000_0000

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

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**EXTMEM**	**Set External Memory (EBI Memory) to Non-privileged State** Set the privileged state of memory ranging from 0x6000_0000 to 0x7FFF_FFFF. 0 = External Memory is set to privileged (default). 1 = External Memory is set to non-privileged.
[0]	**FLASH**	**Set Flash to Non-privileged State** Set the privileged state of memory ranging from 0x0000_0000 to 0x1FFF_FFFF. 0 = Flash is set to privileged (default). 1 = Flash is set to non-privileged.

Privileged Violation Interrupt Enable Register (SCU_PVIOIEN)

Register	Offset	R/W	Description	Reset Value
SCU_PVIOIEN	SCU_BA+0x12C	R/W	Privileged Violation Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved			SRAM2IEN	KSIEN	CRPTIEN	SYSIEN	SCUIEN
15	14	13	12	11	10	9	8
FLASHIEN	FMCIEN	SRAM1IEN	SRAM0IEN	PDMA1IEN	PDMA0IEN	Reserved	SDH0IEN
7	6	5	4	3	2	1	0
CRCIEN	USBHIEN	EBIIEN	GPIOIEN	Reserved		APB1IEN	APB0IEN

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20]	**SRAM2IEN**	**SRAM2 Privileged Violation Interrupt Enable Bit** 0 = Interrupt triggered from privileged violation of SRAM2 Disabled. 1 = Interrupt triggered from privileged violation of SRAM2 Enabled.
[19]	**KSIEN**	**KS Privileged Violation Interrupt Enable Bit** 0 = Interrupt triggered from privileged violation of keystore Disabled. 1 = Interrupt triggered from privileged violation of keystore Enabled.
[18]	**CRPTIEN**	**CRPT Privileged Violation Interrupt Enable Bit** 0 = Interrupt triggered from privileged violation of crypto Disabled. 1 = Interrupt triggered from privileged violation of crypto Enabled.
[17]	**SYSIEN**	**SYS Privileged Violation Interrupt Enable Bit** 0 = Interrupt triggered from privileged violation of system manager Disabled. 1 = Interrupt triggered from privileged violation of system manager Enabled.
[16]	**SCUIEN**	**SCU Privileged Violation Interrupt Enable Bit** 0 = Interrupt triggered from privileged violation of SCU Disabled. 1 = Interrupt triggered from privileged violation of SCU Enabled.
[15]	**FLASHIEN**	**FLASH Privileged Violation Interrupt Enable Bit** 0 = Interrupt triggered from privileged violation of Flash data Disabled. 1 = Interrupt triggered from privileged violation of Flash data Enabled.
[14]	**FMCIEN**	**FMC Privileged Violation Interrupt Enable Bit** 0 = Interrupt triggered from privileged violation of FMC Disabled. 1 = Interrupt triggered from privileged violation of FMC Enabled.
[13]	**SRAM1IEN**	**SRAM Bank 1 Privileged Violation Interrupt Enable Bit** 0 = Interrupt triggered from privileged violation of SRAM bank1 Disabled. 1 = Interrupt triggered from privileged violation of SRAM bank1 Enabled.

Bits	Description	
[12]	**SRAM0IEN**	**SRAM Bank 0 Privileged Violation Interrupt Enable Bit** 0 = Interrupt triggered from privileged violation of SRAM bank0 Disabled. 1 = Interrupt triggered from privileged violation of SRAM bank0 Enabled.
[11]	**PDMA1IEN**	**PDMA1 Privileged Violation Interrupt Enable Bit** 0 = Interrupt triggered from privileged violation of PDMA1 Disabled. 1 = Interrupt triggered from privileged violation of PDMA1 Enabled.
[10]	**PDMA0IEN**	**PDMA0 Privileged Violation Interrupt Enable Bit** 0 = Interrupt triggered from privileged violation of PDMA0 Disabled. 1 = Interrupt triggered from privileged violation of PDMA0 Enabled.
[9]	**Reserved**	Reserved.
[8]	**SDH0IEN**	**SDH0 Privileged Violation Interrupt Enable Bit** 0 = Interrupt triggered from privileged violation of SD host 0 Disabled. 1 = Interrupt triggered from privileged violation of SD host 0 Enabled.
[7]	**CRCIEN**	**CRC Privileged Violation Interrupt Enable Bit** 0 = Interrupt triggered from privileged violation of CRC Disabled. 1 = Interrupt triggered from privileged violation of CRC Enabled.
[6]	**USBHIEN**	**USBH Privileged Violation Interrupt Enable Bit** 0 = Interrupt triggered from privileged violation of USB host Disabled. 1 = Interrupt triggered from privileged violation of USB host Enabled.
[5]	**EBIIEN**	**EBI Privileged Violation Interrupt Enable Bit** 0 = Interrupt triggered from privileged violation of EBI Disabled. 1 = Interrupt triggered from privileged violation of EBI Enabled.
[4]	**GPIOIEN**	**GPIO Privileged Violation Interrupt Enable Bit** 0 = Interrupt triggered from privileged violation of GPIO Disabled. 1 = Interrupt triggered from privileged violation of GPIO Enabled.
[3:2]	**Reserved**	Reserved.
[1]	**APB1IEN**	**APB1 Privileged Violation Interrupt Enable Bit** 0 = Interrupt triggered from privileged violation of APB1 Disabled. 1 = Interrupt triggered from privileged violation of APB1 Enabled.
[0]	**APB0IEN**	**APB0 Privileged Violation Interrupt Enable Bit** 0 = Interrupt triggered from privileged violation of APB0 Disabled. 1 = Interrupt triggered from privileged violation of APB0 Enabled.

Privileged Violation Interrupt Status Register (SCU_PVINTSTS)

Register	Offset	R/W	Description	Reset Value
SCU_PVINTSTS	SCU_BA+0x130	R/W	Privileged Violation Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved			SRAM2IF	KSIF	CRPTIF	SYSIF	SCUIF
15	14	13	12	11	10	9	8
FLASHIF	FMCIF	SRAM1IF	SRAM0IF	PDMA1IF	PDMA0IF	Reserved	SDH0IF
7	6	5	4	3	2	1	0
CRCIF	USBHIF	EBIIF	GPIOIF	Reserved		APB1IF	APB0IF

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20]	**SRAM2IF**	**SRAM Bank 2 Privileged Violation Interrupt Status** 0 = No SRAM Bank 2 violation interrupt event. 1 = There is SRAM Bank 2 violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[19]	**KSIF**	**KS Privileged Violation Interrupt Status** 0 = No KS violation interrupt event. 1 = There is KS violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[18]	**CRPTIF**	**CRPT Privileged Violation Interrupt Status** 0 = No CRPT violation interrupt event. 1 = There is CRPT violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[17]	**SYSIF**	**SYS Privileged Violation Interrupt Status** 0 = No SYS violation interrupt event. 1 = There is SYS violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[16]	**SCUIF**	**SCU Privileged Violation Interrupt Status** 0 = No SCU violation interrupt event. 1 = There is SCU violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[15]	**FLASHIF**	**FLASH Privileged Violation Interrupt Status** 0 = No FLASH violation interrupt event. 1 = There is FLASH violation interrupt event. **Note:** Write 1 to clear the interrupt flag.

Bits	Description	
[14]	**FMCIF**	**FMC Privileged Violation Interrupt Status** 0 = No FMC violation interrupt event. 1 = There is FMC violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[13]	**SRAM1IF**	**SRAM Bank 1 Privileged Violation Interrupt Status** 0 = No SRAM1 violation interrupt event. 1 = There is SRAM1 violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[12]	**SRAM0IF**	**SRAM0 Privileged Violation Interrupt Status** 0 = No SRAM0 violation interrupt event. 1 = There is SRAM0 violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[11]	**PDMA1IF**	**PDMA1 Privileged Violation Interrupt Status** 0 = No PDMA1 violation interrupt event. 1 = There is PDMA1 violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[10]	**PDMA0IF**	**PDMA0 Privileged Violation Interrupt Status** 0 = No PDMA0 violation interrupt event. 1 = There is PDMA0 violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[9]	**Reserved**	Reserved.
[8]	**SDH0IF**	**SDH0 Privileged Violation Interrupt Status** 0 = No SDH0 violation interrupt event. 1 = There is SDH0 violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[7]	**CRCIF**	**CRC Privileged Violation Interrupt Status** 0 = No CRC violation interrupt event. 1 = There is CRC violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[6]	**USBHIF**	**USBH Privileged Violation Interrupt Status** 0 = No USBH violation interrupt event. 1 = There is USBH violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[5]	**EBIIF**	**EBI Privileged Violation Interrupt Status** 0 = No EBI violation interrupt event. 1 = There is EBI violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[4]	**GPIOIF**	**GPIO Privileged Violation Interrupt Status** 0 = No GPIO violation interrupt event. 1 = There is GPIO violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[3:2]	**Reserved**	Reserved.

Bits	Description	
[1]	**APB1IF**	**APB1 Privileged Violation Interrupt Status** 0 = No APB1 violation interrupt event. 1 = There is APB1 violation interrupt event. **Note:** Write 1 to clear the interrupt flag.
[0]	**APB0IF**	**APB0 Privileged Violation Interrupt Status** 0 = No APB0 violation interrupt event. 1 = There is APB0 violation interrupt event. **Note:** Write 1 to clear the interrupt flag.

Security Configuration Write Protection Register (SCU_SCWP)

Register	Offset	R/W	Description	Reset Value
SCU_SCWP	SCU_BA+0x134	R/W	Security Configuration Write Protection Register	0x0000_0000

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

Bits	Description	
[31:16]	**WVCODE**	**Write Verify Code** Read operation: Reserved, all zeros. Write operation: 0x475A = The write verify code, 0x475A, is needed to do a valid write to SCU_SCWP. Others = Invalid write verify code.
[15:2]	**Reserved**	Reserved.
[1]	**LOCK**	**Enable Write Protection Lock Bit (Write One Only)** 0 = Write protection lock Disabled. 1 = Write protection Enabled and locked. **Note:** This bit cannot be cleared to 0 without a system-level reset after set to one.
[0]	**Reserved**	Reserved.

I/O Non-secure Attribution Set Register0 (SCU_IONSSET0)

Register	Offset	R/W	Description	Reset Value
SCU_IONSSET0	SCU_BA+0x140	R/W	I/O Non-secure Attribution Set Register0	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PA**	**Set GPIO Port A to Non-secure State** Write 1 to set PA to non-secure state. Each bit configures one pin in GPIO port A. 0 = GPIO port A is secure (default). 1 = GPIO port A is non-secure.

I/O Non-secure Attribution Set Register1 (SCU_IONSSET1)

Register	Offset	R/W	Description	Reset Value
SCU_IONSSET1	SCU_BA+0x144	R/W	I/O Non-secure Attribution Set Register1	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PB**	**Set GPIO Port B to Non-secure State** Write 1 to set PB to non-secure state. Each bit configures one pin in GPIO port B. 0 = GPIO port B is secure (default). 1 = GPIO port B is non-secure.

I/O Non-secure Attribution Set Register2 (SCU_IONSSET2)

Register	Offset	R/W	Description	Reset Value
SCU_IONSSET2	SCU_BA+0x148	R/W	I/O Non-secure Attribution Set Register2	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PC**	**Set GPIO Port C to Non-secure State** Write 1 to set PC to non-secure state. Each bit configures one pin in GPIO port C. 0 = GPIO port C is secure (default). 1 = GPIO port C is non-secure.

I/O Non-secure Attribution Set Register3 (SCU_IONSSET3)

Register	Offset	R/W	Description	Reset Value
SCU_IONSSET3	SCU_BA+0x14C	R/W	I/O Non-secure Attribution Set Register3	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PD**	**Set GPIO Port D to Non-secure State** Write 1 to set PD to non-secure state. Each bit configures one pin in GPIO port D. 0 = GPIO port D is secure (default). 1 = GPIO port D is non-secure.

I/O Non-secure Attribution Set Register4 (SCU_IONSSET4)

Register	Offset	R/W	Description	Reset Value
SCU_IONSSET4	SCU_BA+0x150	R/W	I/O Non-secure Attribution Set Register4	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PE**	**Set GPIO Port E to Non-secure State** Write 1 to set PE to non-secure state. Each bit configures one pin in GPIO port E. 0 = GPIO port E is secure (default). 1 = GPIO port E is non-secure.

I/O Non-secure Attribution Set Register5 (SCU_IONSSET5)

Register	Offset	R/W	Description	Reset Value
SCU_IONSSET5	SCU_BA+0x154	R/W	I/O Non-secure Attribution Set Register5	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PF**	**Set GPIO Port F to Non-secure State** Write 1 to set PF to non-secure state. Each bit configures one pin in GPIO port F. 0 = GPIO port F is secure (default). 1 = GPIO port F is non-secure.

I/O Non-secure Attribution Set Register6 (SCU_IONSSET6)

Register	Offset	R/W	Description	Reset Value
SCU_IONSSET6	SCU_BA+0x158	R/W	I/O Non-secure Attribution Set Register6	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PG**	**Set GPIO Port G to Non-secure State** Write 1 to set PG to non-secure state. Each bit configures one pin in GPIO port G. 0 = GPIO port G is secure (default). 1 = GPIO port G is non-secure.

I/O Non-secure Attribution Set Register7 (SCU_IONSSET7)

Register	Offset	R/W	Description	Reset Value
SCU_IONSSET7	SCU_BA+0x15C	R/W	I/O Non-secure Attribution Set Register7	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PH**	**Set GPIO Port H to Non-secure State** Write 1 to set PH to non-secure state. Each bit configures one pin in GPIO port H. 0 = GPIO port H is secure (default). 1 = GPIO port H is non-secure.

Non-secure State Monitor Control Register (SCU_NSMCTL)

Register	Offset	R/W	Description	Reset Value
SCU_NSMCTL	SCU_BA+0x200	R/W	Non-secure State Monitor Control Register	0x0000_1000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		DBGON	IDLEON	Reserved	TMRMOD	AUTORLD	NSMIEN
7	6	5	4	3	2	1	0
PRESCALE							

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**DBGON**	**Monitor Counter Keep Counting When the Chip Is in Debug Mode Enable Bit** 0 = The counter will be halted when the core processor is halted by ICE. (default) 1 = The counter will keep counting when the core processor is halted by ICE.
[12]	**IDLEON**	**Monitor Counter Keep Counting When the Chip Is in Idle Mode Enable Bit** 0 = The counter will be halted when the chip is in idle mode. 1 = The counter will keep counting when the chip is in idle mode. (default) **Note:** In monitor mode, the counter is always halted when the core processor is in secure state.
[11]	**Reserved**	Reserved.
[10]	**TMRMOD**	**Non-secure Monitor Mode Enable Bit** 0 = Monitor mode. The counter will count down when the core processor is in non-secure state. (default) 1 = Free-counting mode. The counter will keep counting no matter the core processor is in secure or non-secure state.
[9]	**AUTORLD**	**Auto Reload Non-secure State Monitor Counter When CURRNS Changing to 1** 0 = Disable clearing non-secure state monitor counter automatically (default). 1 = Enable clearing non-secure state monitor counter automatically when the core processor changes from secure state to non-secure state.
[8]	**NSMIEN**	**Non-secure State Monitor Interrupt Enable Bit** 0 = Non-secure state monitor interrupt Disabled. 1 = Non-secure state monitor interrupt Enabled.
[7:0]	**PRESCALE**	**Pre-scale Value of Non-secure State Monitor Counter** 0 = Counter Disabled. Others = Counter Enabled and the counter clock source = HCLK/PRESCALE.

Non-secure State Monitor Reload Value Register (SCU_NSMLOAD)

Register	Offset	R/W	Description	Reset Value
SCU_NSMLOAD	SCU_BA+0x204	R/W	Non-secure State Monitor Reload Value Register	0x00FF_FFFF

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

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**RELOAD**	**Reload Value for Non-secure State Monitor Counter** The RELOAD value will be reloaded to the counter whenever the counter counts down to 0.

Non-secure State Monitor Counter Value Register (SCU_NSMVAL)

Register	Offset	R/W	Description	Reset Value
SCU_NSMVAL	SCU_BA+0x208	R/W	Non-secure State Monitor Counter Value Register	0x00FF_FFFF

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

Bits	Description	
[31:24]	Reserved	Reserved.
[23:0]	VALUE	**Counter Value of Non-secure State Monitor Counter** Current value of non-secure state monitor counter. This is down counter and counts down only when CURRNS = 1. When counting down to 0, VALUE will automatically be reloaded from NSMLOAD register. A write of any value clears the VALUE to 0 and also clears NSMIF.

Non-secure State Monitor Status Register (SCU_NSMSTS)

Register	Offset	R/W	Description	Reset Value
SCU_NSMSTS	SCU_BA+0x20C	R/W	Non-secure State Monitor Status Register	0x0000_0000

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

Bits	Description	
[31:2]	Reserved	Reserved.
[1]	NSMIF	**Non-secure State Monitor Interrupt Flag** 0 = Counter does not count down to 0 since the last NSMIF has been cleared. 1 = Counter counts down to 0. **Note:** This bit is cleared by writing 1.
[0]	CURRNS	**Current Core Processor Secure/Non-secure State (Read Only)** 0 = Core processor is in secure state. 1 = Core processor is in non-secure state. **Note:** This bit can be used to monitor the current secure/non-secure state of the core processor, even if the non-secure state monitor counter is disabled.

FVC Control Register (FVC_CTL)

Register	Offset	R/W	Description	Reset Value
FVC_CTL	FVC_BA+0x00	R/W	FVC Control Register	0x0000_000X

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

Bits	Description	
[31:16]	**WVCODE**	**Verification Code** When **written**, this field must be **0x7710**
[15:2]	**Reserved**	Reserved.
[1]	**MONOEN**	**Monotonic Enable Bit** Set to 1 to enable the monotonic mechanism of FVC. **Note:** This bit can be set to1 but cannot be cleared to 0.
[0]	**INIT**	**FVC Init Bit** Set to 1 to enable FVC This bit is writable when FVC is at Reset state. **Note:** After set to 1, this bit is cleared to 0 automatically when FVC is back to Reset state.

FVC Status Register (FVC_STS)

Register	Offset	R/W	Description	Reset Value
FVC_STS	FVC_BA+0x04	R	FVC Status Register	0x0000_000X

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

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**RDY**	**FVC Ready Bit** Indicates the FVC is ready after the initial process.
[0]	**BUSY**	**FVC Busy Bit** Indicates the FVC is at busy state.

Non-volatile Version Counter Control Register0 (FVC_NVC0)

Register	Offset	R/W	Description	Reset Value
FVC_NVC0	FVC_BA+0x10	R/W	Non-volatile Version Counter Control Register0	0x0000_XXXX

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

Bits	Description	
[31:16]	**WVCODE**	**Verification Code** When **written**, this field must be the **current firmware version number**
[15:0]	**FWVER**	**Firmware Version** **Read**: Indicates the current firmware version of NVC0. **Write**: Updates the firmware version of NVC0. The maximum value of this field is 63.

Non-volatile Version Counter Control Register1 (FVC_NVC1)

Register	Offset	R/W	Description	Reset Value
FVC_NVC1	FVC_BA+0x14	R/W	Non-volatile Version Counter Control Register1	0x0000_XXXX

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

Bits	Description	
[31:16]	**WVCODE**	**Verification Code** When **written**, this field must be the **current firmware version number**
[15:0]	**FWVER**	**Firmware Version** **Read**: Indicates the current firmware version ofNVC1. **Write**: Updates the firmware version of NVC1. The maximum value of this field is 63.

Non-volatile Version Counter Control Register4 (FVC_NVC4)

Register	Offset	R/W	Description	Reset Value
FVC_NVC4	FVC_BA+0x20	R/W	Non-volatile Version Counter Control Register4	0x0000_XXXX

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

Bits	Description	
[31:16]	**WVCODE**	**Verification Code** When **written**, this field must be the **current firmware version number**
[15:0]	**FWVER**	**Firmware Version** **Read**: Indicates the current firmware version of NVC4. **Write**: Updates the firmware version of NVC4. The maximum value of this field is 255.

Non-volatile Version Counter Control Register5 (FVC_NVC5)

Register	Offset	R/W	Description	Reset Value
FVC_NVC5	FVC_BA+0x24	R/W	Non-volatile Version Counter Control Register5	0x0000_XXXX

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

Bits	Description	
[31:16]	**WVCODE**	**Verification Code** When **written**, this field must be the **current firmware version number**
[15:0]	**FWVER**	**Firmware Version** **Read**: Indicates the current firmware version of NVC5. **Write**: Updates the firmware version of NVC5. The maximum value of this field is 255.

Secure DPM Control Register (DPM_CTL)

Register	Offset	R/W	Description	Reset Value
DPM_CTL	DPM_BA+0x00	R/W	Secure DPM Control Register	0xA500_000X

31	30	29	28	27	26	25	24
RWVCODE							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		DACCDIS	DACCWDIS	Reserved			INTEN
7	6	5	4	3	2	1	0
Reserved				PWUPD	PWCMP	LOCK	DBGDIS

Bits	Description	
[31:24]	**RWVCODE**	**Write Verify Code and Read Verify Code** Read operation: 0xA5 = The read access for DPM_CTL is correct. Others = The read access for DPM_CTL is incorrect. Write operation: 0x5A = The write verify code, 0x5A, is needed to do a valid write to DPM_CTL. Others = Invalid write verify code.
[23:14]	**Reserved**	Reserved.
[13]	**DACCDIS**	**Debug Access Disable Bit** This bit disables the accessibility of external debugger to all DPM registers. 0 = External debugger can read/write DPM registers. 1 = External debugger cannot read/write DPM registers.
[12]	**DACCWDIS**	**Secure DPM Debug Write Access Disable Bit** This bit disables the ability of external debugger to set Secure DPM registers for debug authentication. 0 = External debugger can set Secure DPM registers. 1 = External debugger cannot set Secure DPM registers.
[11:9]	**Reserved**	Reserved.
[8]	**INTEN**	**DPM Interrupt Enable Bit** 0 = DPM interrupt function Enabled. 1 = DPM interrupt function Disabled.
[7:4]	**Reserved**	Reserved.

[3]	**PWUPD**	**Secure DPM Password Update Bit** Set to enter the process of updating Secure DPM password. 0 = No operation. 1 = Update Secure DPM password. **Note 1:** This bit should be set with PWCMP equal to 0. **Note 2:** This bit will be cleared after the update process is finished.
[2]	**PWCMP**	**Secure DPM Password Compare Bit** Set to enter the process of compare Secure DPM password. 0 = No operation. 1 = Compare Secure DPM password. **Note:** This bit will be cleared after the comparison process is finished.
[1]	**LOCK**	**Set Secure DPM Debug Lock Bit** When this bit is read as 0, it can be written to 1 to configure the Secure DPM LOCK bit (LOCKS). When written: 0 = No operation. 1 = Trigger the process to set LOCKS configuration bit. **Note:** This bit can be set to 1 but cannot be cleared to 0.
[0]	**DBGDIS**	**Set Secure DPM Debug Disable Bit** When this bit is read as 0, it can be written to 1 to configure the Secure DPM DBGDIS bit (DBGDISS). When written: 0 = No operation. 1 = Trigger the process to set DBGDISS configuration bit. **Note:** This bit can be set to 1 but cannot be cleared to 0.

Secure DPM Status Register (DPM_STS)

Register	Offset	R/W	Description	Reset Value
DPM_STS	DPM_BA+0x04	R/W	Secure DPM Status Register	0x000X_0X00

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					PWOK	LOCK	DBGDIS
15	14	13	12	11	10	9	8
Reserved					PWUCNT		
7	6	5	4	3	2	1	0
Reserved	PWFMAX	PWUOK	PWCERR	Reserved		INT	BUSY

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18]	**PWOK**	**Secure Password OK Flag (Read Only)** This bit indicates the Secure DPM password has been checked and is correct. 0 = The Secure DPM password has not been checked pass, yet. 1 = The Secure DPM password has been checked pass since last cold reset.
[17]	**LOCK**	**Secure Debug Lock Flag (Read Only)** This bit indicates the current value of LOCKS bit.
[16]	**DBGDIS**	**Secure Debug Disable Flag (Read Only)** This bit indicates the current value of DBGDISS bit. {PWOK, LOCK, DBGDIS} bits define the current state of DPM. x00 = **DEFAULT** state. x1x = **LOCKED** state. 001 = **CLOSE** state. 101 = **OPEN** state. Others = Unknown.
[15:11]	**Reserved**	Reserved.
[10:8]	**PWUCNT**	**Secure DPM Password Updated Times (Read Only)** This bit indicates how many times of secure password has been updated. The max value is 7. If PWUCNT reached the max value, Secure DPM password cannot be updated anymore.
[7]	**Reserved**	Reserved.
[6]	**PWFMAX**	**Secure DPM Password Fail Times Maximum Reached Flag (Read Only)** This bit indicates if the fail times of comparing Secure DPM password reached max times. 0 = Max time has not reached and Secure DPM password comparison can be triggered. 1 = Max time reached and Secure DPM password comparison cannot be processed anymore.

[5]	**PWUOK**	**Secure DPM Password Updated Flag** This bit indicates Secure DPM password has been updated successfully. When read: 0 = No successful updating process has happened. 1 = There is at least one successful updating process since last clearing of this bit. **Note:** This flag is write-one-clear.
[4]	**PWCERR**	**Secure DPM Password Compared Error Flag** This bit indicates the result of Secure DPM password comparison. When read: 0 = The result of Secure DPM password is correct. 1 = The result of Secure DPM password is incorrect. **Note:** This flag is write-one-clear.
[3:2]	**Reserved**	Reserved.
[1]	**INT**	**DPM Interrupt Flag (Read Only)** This bit indicates the interrupt is triggered. 0 = Interrupt is not enabled or no password comparison flag is set. 1 = Interrupt is enabled and PWCERR flag in either DPM_STS or DPM_NSSTS register is not cleared. **Note:** This bit is cleared automatically when PWCERR flag in both DPM_STS and DPM_NSSTS are 0.
[0]	**BUSY**	**DPM Busy Flag (Read Only)** This bit indicates the DPM is busy. 0 = DPM is not busy and writing to any register is accepted. 1 = DPM is busy and other bits in DPM_STS register are not valid and writing to any register is ignored.

Secure DPM Password 0 (DPM_SPW0)

Register	Offset	R/W	Description	Reset Value
DPM_SPW0	DPM_BA+0x10	W	Secure DPM Password 0	0xFFFF_FFFF

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

Bits	Description	
[31:0]	**PW**	**Password** Write password[31:0] to this register to update or compare Secure DPM password. It is write-only and always read as 0xFFFFFFFF.

Secure DPM Password 1 (DPM_SPW1)

Register	Offset	R/W	Description	Reset Value
DPM_SPW1	DPM_BA+0x14	W	Secure DPM Password 1	0xFFFF_FFFF

31	30	29	28	27	26	25	24
PW[63:56]							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PW[55:48]							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PW[47:40]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PW[39:32]							

Bits	Description	
[31:0]	**PW**	**Password** Write password[63:32] to this register to update or compare Secure DPM password. It is write-only and always read as 0xFFFFFFFF.

Secure DPM Password 2 (DPM_SPW2)

Register	Offset	R/W	Description	Reset Value
DPM_SPW2	DPM_BA+0x18	W	Secure DPM Password 2	0xFFFF_FFFF

31	30	29	28	27	26	25	24
PW[95:88]							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PW[87:80]							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PW[79:72]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PW[71:64]							

Bits	Description	
[31:0]	**PW**	**Password** Write password[95:64] to this register to update or compare Secure DPM password. It is write-only and always read as 0xFFFFFFFF.

Secure DPM Password 3 (DPM_SPW3)

Register	Offset	R/W	Description	Reset Value
DPM_SPW3	DPM_BA+0x1C	W	Secure DPM Password 3	0xFFFF_FFFF

31	30	29	28	27	26	25	24
PW[127:120]							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PW[119:112]							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PW[111:104]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PW[103:96]							

Bits	Description	
[31:0]	**PW**	**Password** Write password[127:96] to this register to update or compare Secure DPM password. It is write-only and always read as 0xFFFFFFFF.

Non-secure DPM Control Register (DPM_NSCTL)

Register	Offset	R/W	Description	Reset Value
DPM_NSCTL	DPM_BA+0x50	R/W	Non-secure DPM Control Register	0xA500_000X

31	30	29	28	27	26	25	24
RWVCODE							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved			DACCWDIS	Reserved			
7	6	5	4	3	2	1	0
Reserved				PWUPD	PWCMP	LOCK	DBGDIS

Bits	Description	
[31:24]	**RWVCODE**	**Write Verify Code and Read Verify Code** Read operation: 0xA5 = The read access for DPM_NSCTL is correct. Others = The read access for DPM_NSCTL is incorrect. Write operation: 0x5A = The write verify code, 0x5A, is needed to do a valid write to DPM_NSCTL. Others = Invalid write verify code.
[23:13]	**Reserved**	Reserved.
[12]	**DACCWDIS**	**Debug Write Access Disable Bit** This bit disables the ability of external debugger to set Non-secure DPM registers for debug authentication. 0 = External debugger can set Non-secure DPM registers. 1 = External debugger cannot set Non-secure DPM registers.
[11:4]	**Reserved**	Reserved.
[3]	**PWUPD**	**Non-secure DPM Password Update Bit** Set to enter the process of updating Non-secure DPM password. 0 = No operation. 1 = Update Non-secure DPM password. **Note 1:** This bit should be set with PWCMP equal to 0. **Note 2:** This bit will be cleared after the update process is finished.
[2]	**PWCMP**	**Non-secure DPM Password Compare Bit** Set to enter the process of compare Non-secure DPM password. 0 = No operation. 1 = Compare Non-secure DPM password. **Note:** This bit will be cleared after the comparison process is finished.

[1]	**LOCK**	**Set Non-secure DPM Debug Lock Bit** When this bit is read as 0, it can be written to 1 to configure the Non-secure DPM LOCK bit (LOCKNS). When written: 0 = No operation. 1 = Trigger the process to set LOCKNS configuration bit. **Note:** This bit can be set to 1 but cannot be cleared to 0.
[0]	**DBGDIS**	**Set Non-secure DPM Debug Disable Bit** When this bit is read as 0, it can be written to 1 to configure the Non-secure DPM DBGDIS bit (DBGDISNS). When written: 0 = No operation. 1 = Trigger the process to set DBGDISNS configuration bit. **Note:** This bit can be set to 1 but cannot be cleared to 0.

Non-secure DPM Status Register (DPM_NSSTS)

Register	Offset	R/W	Description	Reset Value
DPM_NSSTS	DPM_BA+0x54	R/W	Non-secure DPM Status Register	0x000X_0X00

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					PWOK	LOCK	DBGDIS
15	14	13	12	11	10	9	8
Reserved					PWUCNT		
7	6	5	4	3	2	1	0
Reserved	PWFMAX	PWUOK	PWCERR	Reserved			BUSY

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18]	**PWOK**	**Non-secure Password OK Flag (Read Only)** This bit indicates the Non-secure DPM password has been checked and is correct. 0 = The Non-secure DPM password has not been checked pass, yet. 1 = The Non-secure DPM password has been checked pass since last cold reset.
[17]	**LOCK**	**Non-secure Debug Lock Flag (Read Only)** This bit indicates the current value of LOCKNS bit.
[16]	**DBGDIS**	**Non-secure Debug Disable Flag (Read Only)** This bit indicates the current value of DBGDISNS bit. {PWOK, LOCK, DBGDIS} bits define the current state of DPM. x00 = **DEFAULT** state. x1x = **LOCKED** state. 001 = **CLOSE** state. 101 = **OPEN** state. Others = Unknown.
[15:11]	**Reserved**	Reserved.
[10:8]	**PWUCNT**	**Non-secure DPM Password Updated Times (Read Only)** This bit indicates how many times of non-secure password has been updated. The max value is 7. If PWUCNT reached the max value, Non-secure DPM password cannot be updated anymore.
[7]	**Reserved**	Reserved.
[6]	**PWFMAX**	**Non-secure DPM Password Fail Times Maximum Reached Flag (Read Only)** This bit indicates if the fail times of comparing Non-secure DPM password reached max times. 0 = Max time has not reached and Non-secure DPM password comparison can be triggered. 1 = Max time reached and Non-secure DPM password comparison cannot be processed anymore.

[5]	**PWUOK**	**Non-secure DPM Password Updated Flag** This bit indicates Non-secure DPM password has been updated correctly. When read: 0 = No successful updating process has happened. 1 = There is at least one successful updating process since last clearing of this bit. **Note:** This flag is write-one-clear.
[4]	**PWCERR**	**Non-secure DPM Password Compared Error Flag** This bit indicates the result of Non-secure DPM password comparison. 0 = The result of Non-secure DPM password is correct. 1 = The result of Non-secure DPM password is incorrect. **Note:** This flag is write-one-clear.
[3:1]	**Reserved**	Reserved.
[0]	**BUSY**	**DPM Busy Flag (Read Only)** This bit indicates the DPM is busy. 0 = DPM is not busy and writing to any register is accepted. 1 = DPM is busy and other bits in DPM_NSSTS register are not valid and writing to any register is ignored.

Non-secure DPM Password 0 (DPM_NSPW0)

Register	Offset	R/W	Description	Reset Value
DPM_NSPW0	DPM_BA+0x60	W	Non-secure DPM Password 0	0xFFFF_FFFF

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

Bits	Description	
[31:0]	**PW**	**Password** Write password[31:0] to this register to update or compare Non-secure DPM password. It is write-only and always read as 0xFFFFFFFF.

Non-secure DPM Password 1 (DPM_NSPW1)

Register	Offset	R/W	Description	Reset Value
DPM_NSPW1	DPM_BA+0x64	W	Non-secure DPM Password 1	0xFFFF_FFFF

31	30	29	28	27	26	25	24
PW[63:56]							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PW[55:48]							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PW[47:40]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PW[39:32]							

Bits	Description	
[31:0]	**PW**	**Password** Write password[63:32] to this register to update or compare Non-secure DPM password. It is write-only and always read as 0xFFFFFFFF.

Non-secure DPM Password 2 (DPM_NSPW2)

Register	Offset	R/W	Description	Reset Value
DPM_NSPW2	DPM_BA+0x68	W	Non-secure DPM Password 2	0xFFFF_FFFF

31	30	29	28	27	26	25	24
PW[95:88]							
23	22	21	20	19	18	17	16
PW[87:80]							
15	14	13	12	11	10	9	8
PW[79:72]							
7	6	5	4	3	2	1	0
PW[71:64]							

Bits	Description	
[31:0]	PW	**Password** Write password[95:64] to this register to update or compare Non-secure DPM password. It is write-only and always read as 0xFFFFFFFF.

Non-secure DPM Password 3 (DPM_NSPW3)

Register	Offset	R/W	Description	Reset Value
DPM_NSPW3	DPM_BA+0x6C	W	Non-secure DPM Password 3	0xFFFF_FFFF

31	30	29	28	27	26	25	24
PW[127:120]							
23	22	21	20	19	18	17	16
PW[119:112]							
15	14	13	12	11	10	9	8
PW[111:104]							
7	6	5	4	3	2	1	0
PW[103:96]							

Bits	Description	
[31:0]	PW	**Password** Write password[127:96] to this register to update or compare Non-secure DPM password. It is write-only and always read as 0xFFFFFFFF.

Product Life-cycle Control Register (PLM_CTL)

Register	Offset	R/W	Description	Reset Value
PLM_CTL	PLM_BA+0x00	R/W	Product Life-cycle Control Register	0x0000_000X

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

Bits	Description	
[31:16]	**WVCODE**	**Write Verify Code** The code is 0x475A for a valid write to this register.
[15:3]	**Reserved**	Reserved.
[2:0]	**STAGE**	**Life-cycle Stage Update Bits** Bits to update PLM stage. All bits can be set to one but cannot be cleared to 0. 001 = progress to OEM stage. 011 = progress to Deployed stage. 111 = progress to RMA stage. Other value will be ignored.

Product Life-cycle Status Register (PLM_STS)

Register	Offset	R/W	Description	Reset Value
PLM_STS	PLM_BA+0x04	R	Product Life-cycle Status Register	0x0000_000X

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

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**DIRTY**	**DIRTY Bit (Read Only)** Indicate the life-cycle stage has been progressed after last cold-reset. Value of STAGE bits is not Current stage of PLM. It needs a cold reset to make it work.
[7:3]	**Reserved**	Reserved.
[2:0]	**STAGE**	**Life-cycle Stage (Read Only)** Indicates the current stage of PLM. 000 = Vendor Stage. 001 = OEM Stage. 011 = Deployed Stage. 111 = RMA Stage. Others = ERROR Stage.

6.5 Arm® TrustZone®

The Arm® TrustZone® can be considered as a physical partition that divides the microcontroller into **Secure** (Trusted) and **Non-secure** (Non-trusted) worlds according to memory address. The secure world is an isolated execution environment, code and data loaded inside are protected and cannot be accessed from Non-secure world. Code running at secure world is called secure code that can access both secure and non-secure memories and peripherals; while code running at non-secure world is called non-secure code that can only access non-secure memories and peripherals.

Figure 6.5-1 shows an example of a system divided into the secure world and non-secure world. Green blocks indicate secure components, Red blocks indicate non-secure components and white ones are both/either secure and/or non-secure accessible. When the core processor is in secure state (left side of the figure), it belongs to secure world, which has its own MSP, PSP and VTOR registers and can access the green, red, white blocks. Contrarily, when the core processor is in non-secure state (right side of the figure), it belongs to non-secure world, which also has its own MSP, PSP and VTOR registers, but, it can only access red and white blocks so that non-secure world components are not able to impact secure world.

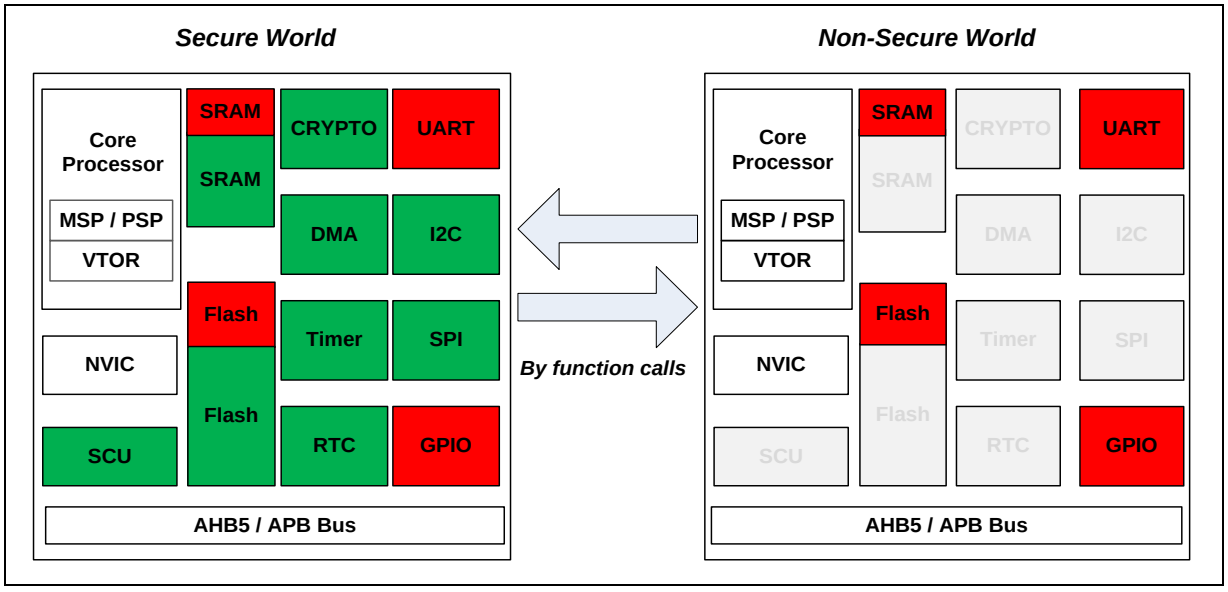


Figure 6.5-1 Secure World View and Non-secure World View on a Chip

In order to support TrustZone® to set up both secure world and non-secure world, Cortex®-M23 provides three security attributes. Each memory address is assigned with one of the security attributes. These security attributes are listed below.

- Non-secure (NS)

  Addresses used for non-secure memory or non-secure peripheral's registers.

- Secure (S)

  Addresses used for secure memory or secure peripheral's registers.

- Non-secure Callable (NSC)

  A special type of secure memory region which can contain SG instructions. The SG instruction allows a non-secure function calls to a secure function.

The address space partitioning is completed by Implementation Define Attribution Unit (IDAU) and Security Attribution Unit (SAU) together. The IDAU is non-programmable, which defines static partition of address space. The static partition specifies the default security attribute of a memory region. In

contrast with IDAU, the SAU is programmable which provides dynamic partition of address space. The dynamic partition is given by software programmer to specify the security attribute of a memory region. The core processor is in secure state when executing instructions from secure memory. Otherwise, the core processor is in non-secure state when executing instructions from non-secure memory. For setting IDAU and SAU, refer to sections “Implementation Defined Attribution Unit (IDAU)” and “Security Attribution Unit (SAU)” in “System Manager” chapter for more details.

The security attribute of Flash, SRAM and peripherals are assigned by TrustZone® related control units. The NSCBA register in FMC is used to divide the APROM into two parts, one is secure and the other is non-secure. The security attribute of SRAM and peripherals are assigned by programming Secure Configuration Unit (SCU).

Whenever being reset, the M2354 is in secure state, that is, the core processor, Flash, SRAM and peripherals are all in secure state. Therefore, the system boots in secure state. The boot code is responsible to set up TrustZone® related control units in M2354 to partition address space and assign non-secure resources that can be directly accessed from non-secure world.

6.5.1 Address Space Partition

The SAU and IDAU are the control units used to define security attribute of memory addresses. The IDAU defines default partition of secure and non-secure addresses, while the SAU is programmable to change the security attribute defined by IDAU.

6.5.1.1 Implementation Define Attribution Unit (IDAU)

The IDAU uses address bit 28 to distinguish between secure and non-secure world, i.e. the bit 28 of a secure address is always 0, and the bit 28 of a non-secure address is always 1, except regions above 0xE000_0000.

The partition of 4GB address space is shown as Figure 6.5-2. Each region consists of a secure (bit 28 is 0) and a non-secure (bit 28 is 1) sub-regions, the size of a sub-region is 256 Mbytes. In order to store entry functions for non-secure code, the security attribute of secure SRAM region is assigned as non-secure callable (**NSC**). Similarly, the secure “Code” region is assigned as NSC but has an exception at first 2 KB area. This first 2 KB area is defined as secure only to avoid accidental **SG** instruction after power on.

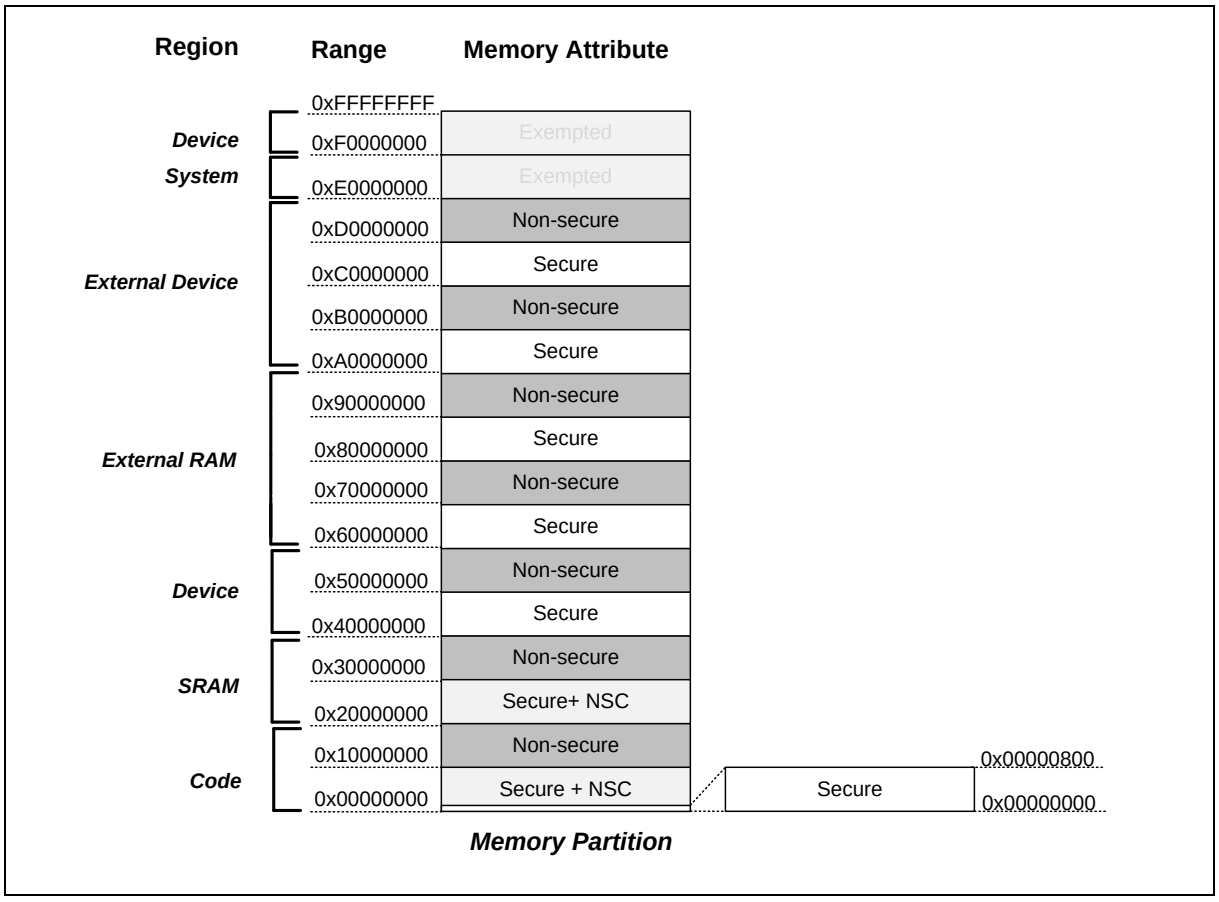


Figure 6.5-2 The 4 GB Memory Map Divided Into Secure and Non-secure Regions by IDAU

6.5.1.2 Security Attribution Unit (SAU)

The SAU is a MPU-like function unit inside Cortex®-M23. Up to 8 memory regions can be defined by programming control registers of SAU.

Memory regions are enabled individually by programming SAU_RNR, SAU_RBAR and SAU_RLAR. The memory region is enabled once RENABLE (SAU_RLAR[0]) is set to 1, and the security attribute is defined by NSC (SAU_RLAR[1]):

- NSC = 0, the memory region is Non-secure (NS).
- NSC = 1, the memory region is Secure and Non-secure callable (NSC).

The security attribute of each memory region defined by SAU is either NS or NSC. Those memory addresses not defined by SAU regions are treated as Secure. After all memory regions are set, SAU_CTRL[0] should be set to 1 to enable SAU.

Both IDAU and SAU define the security attribute of a memory address. If the definitions are different, the more secure attribute will be used for the memory address. The priority of the security attribute from high to low is Secure > NSC > NS.

When the core processor attempts to access a target, e.g. a memory or peripheral register, the security attribute of the target is decided by checking IDAU and SAU. If the core processor is non-secure but the target is secure, a HardFault exception will be generated. Because non-specified memory addresses are treated as secure, non-secure memory regions need to be defined for the core

processor to access non-secure memory and non-secure peripheral registers. Besides, whole secure code and SRAM regions are defined as NSC by IDAU. The size of NSC regions can be changed according to the NSC entry functions included in application code. The example usage of SAU regions is shown as Figure 6.5-3.

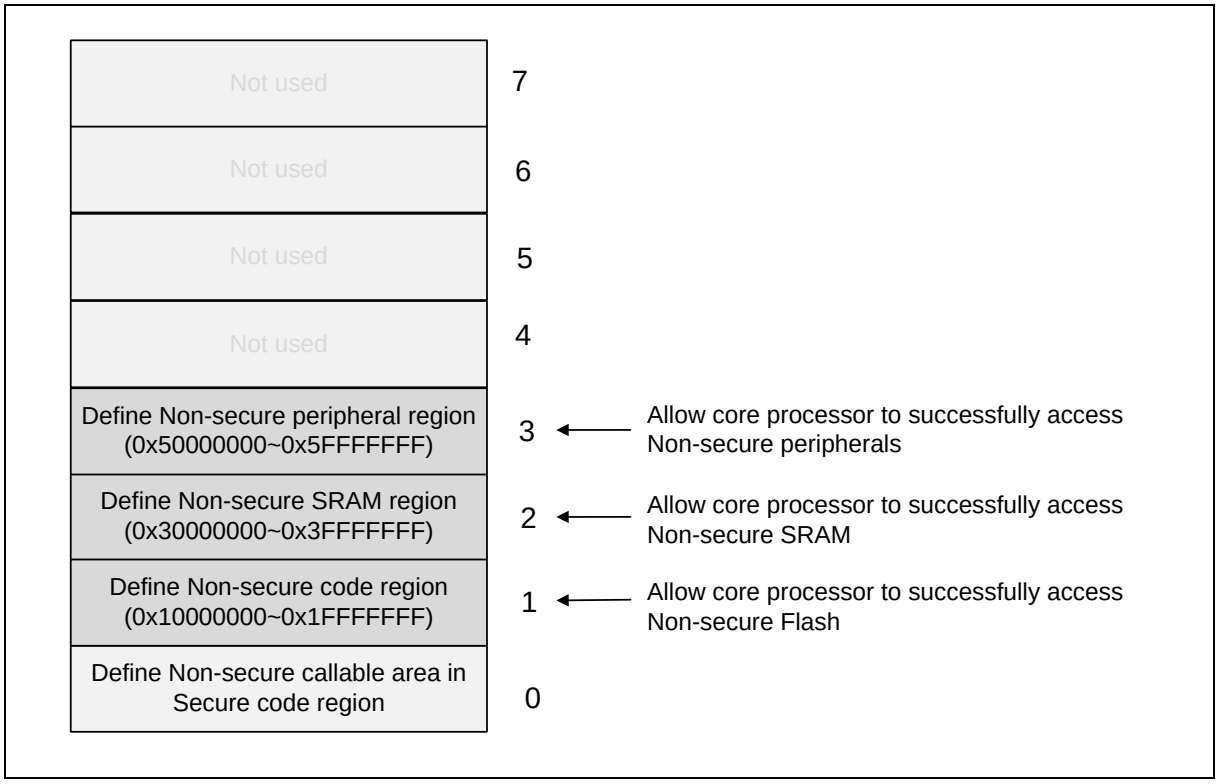


Figure 6.5-3 Typical Setting of SAU

6.5.2 Security Attribute Configuration

The previous section describes how to divide the address space of core processor view into secure world and non-secure world. For M2354, the memory and peripherals can be assigned to either secure or non-secure world during system initialization. The M2354 is designed to start execution in secure state after reset. In other words, core processor and all system resources including Flash, SRAM and peripherals are secure after reset. Then, the system initialization code may change some parts of the system resources to be non-secure.

6.5.2.1 Security Attribute Configuration of Flash

The M2354 Flash memory is split into a number of different regions such as LDROM, APROM and others. Most of the Flash regions are always secure and cannot be changed. The only one can be changed is the APROM region. Non-secure APROM region is set by programming a special control register, NSCBA (Non-secure base address). The NSCBA[23:0] indicates the starting address of non-secure APROM and its value should be aligned with a Flash page size. The secure APROM region starts from address 0x0 and ends at NSCBA[23:0] – 1, while the non-secure APROM region ranges from NSCBA[23:0] to the end of APROM. For setting NSCBA, refer to FMC section for more details.

6.5.2.2 Security Attribute Configuration of SRAM and Peripherals

The secure state of SRAM blocks and all peripherals can be configured by Security Configuration Unit (SCU), which contains a set of control registers used to assign the security attribute. Besides, the SCU monitors bus transfers to detect unsecure access. The unsecure access is one of the following

conditions.

- Non-secure master peripheral tries to access a secure address (address bit 28 = 0).
- Secure code or secure master peripheral uses non-secure address (address bit 28 = 1) to access secure SRAM or peripheral.

When an unsecure access is detected, SCU blocks the access operation and generates a secure alarm interrupt.

For more details, refer to the Security Configuration Unit (SCU) chapter.

6.5.3 System Address Map and Access Scheme

In the M2354 series, the Flash, SRAM and most peripherals can be assigned to be Secure or Non-secure, but each of them can be accessed through either Secure address or Non-secure address depending on its security attribute configuration. Core processor and master peripherals should use correct address to access resources, i.e. the secure resource should be accessed by using secure address. Similarly, the non-secure resource should be accessed by using non-secure address.

6.5.3.1 Permanent Secure Peripherals

The security attribute of some peripherals are always secure and cannot be changed for safety and security. If necessary, the secure code should manage and provide functions for non-secure code to access these peripherals. Table 6.5-1 lists these secure peripherals.

Peripheral	Function	Address
SYS	System Control Registers	0x4000_0000 – 0x4000_01FF
CLK	Clock Control Registers	0x4000_0200 – 0x4000_02FF
NMI	NMI Control Registers	0x4000_0300 – 0x4000_03FF
PDMA0	Peripheral DMA 0 Control Registers	0x4000_8000 – 0x4000_8FFF
FMC	Flash Memory Control Registers	0x4000_C000 – 0x4000_CFFF
SCU	Security Configuration Unit Registers	0x4002_F000 – 0x4002_FFFF
WDT	Watchdog Timer Control Registers	0x4004_0000 – 0x4004_0FFF
TMR01	Timer0/Timer1 Control Registers	0x4005_0000 – 0x4005_0FFF

Table 6.5-1 Peripherals and Regions that are Always Secure

6.5.3.2 Secure Address vs. Non-secure Address

A memory or a peripheral register may have secure and non-secure address in system address map, but the memory or register only responds to the address that is consistent with its security attribute. The different access modes of secure and non-secure target are illustrated in Figure 6.5-4.

Suppose that SRAM block 0, 2, and 4 are in secure state, they will respond to an access when address bit 28 is 0 (secure address), but will not respond to an access with address bit 28 is 1 (non-secure address). In this example, SRAM block 1 and 3 are in non-secure state. Hence, these blocks will only respond to an access when the address bit 28 is 1.

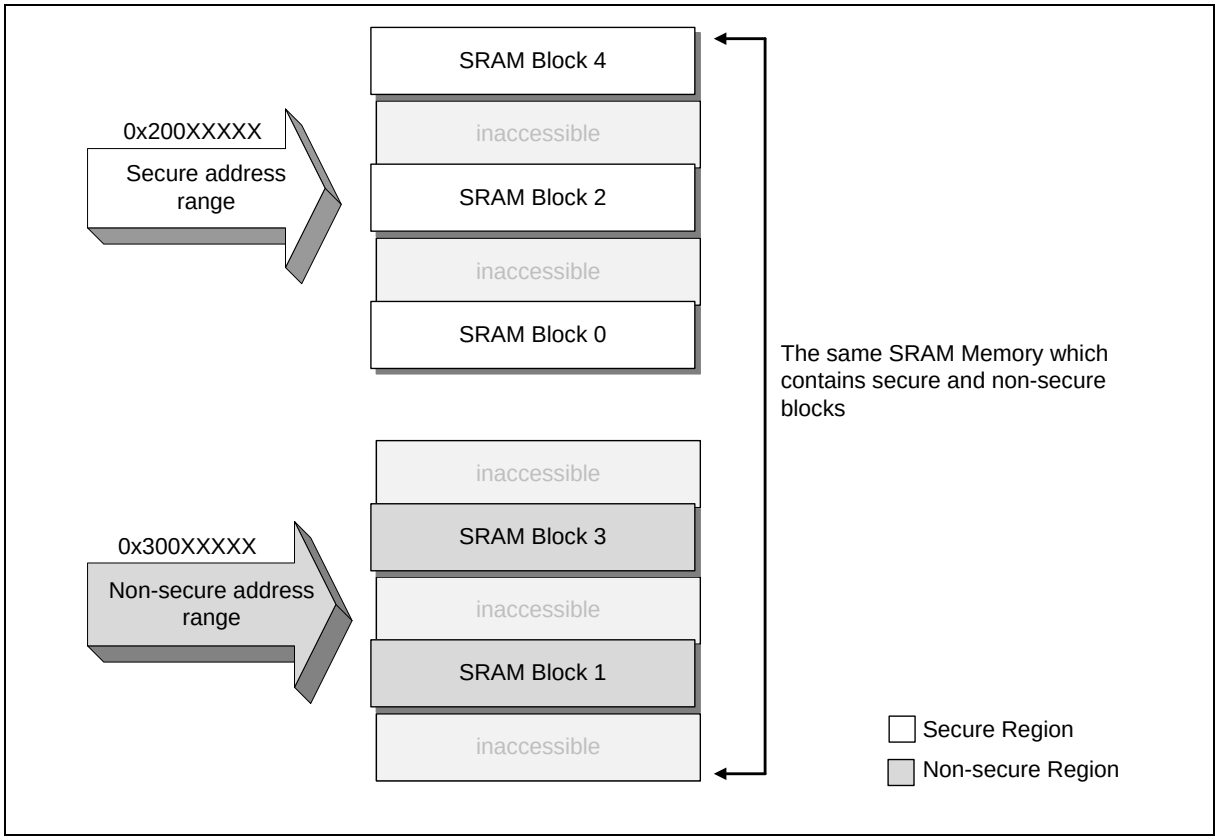


Figure 6.5-4 Example of SRAM Divided Into Secure Block and Non-secure Block

6.5.3.3 Valid Access vs. Invalid Access

When core processor or a master peripheral is trying to access (read or write) a memory or register, the result depends on the following conditions.

- Non-secure code or master peripheral is not allowed to access a secure memory or register.
- A memory or register only responds to the related address which is consistent with its security attribute.

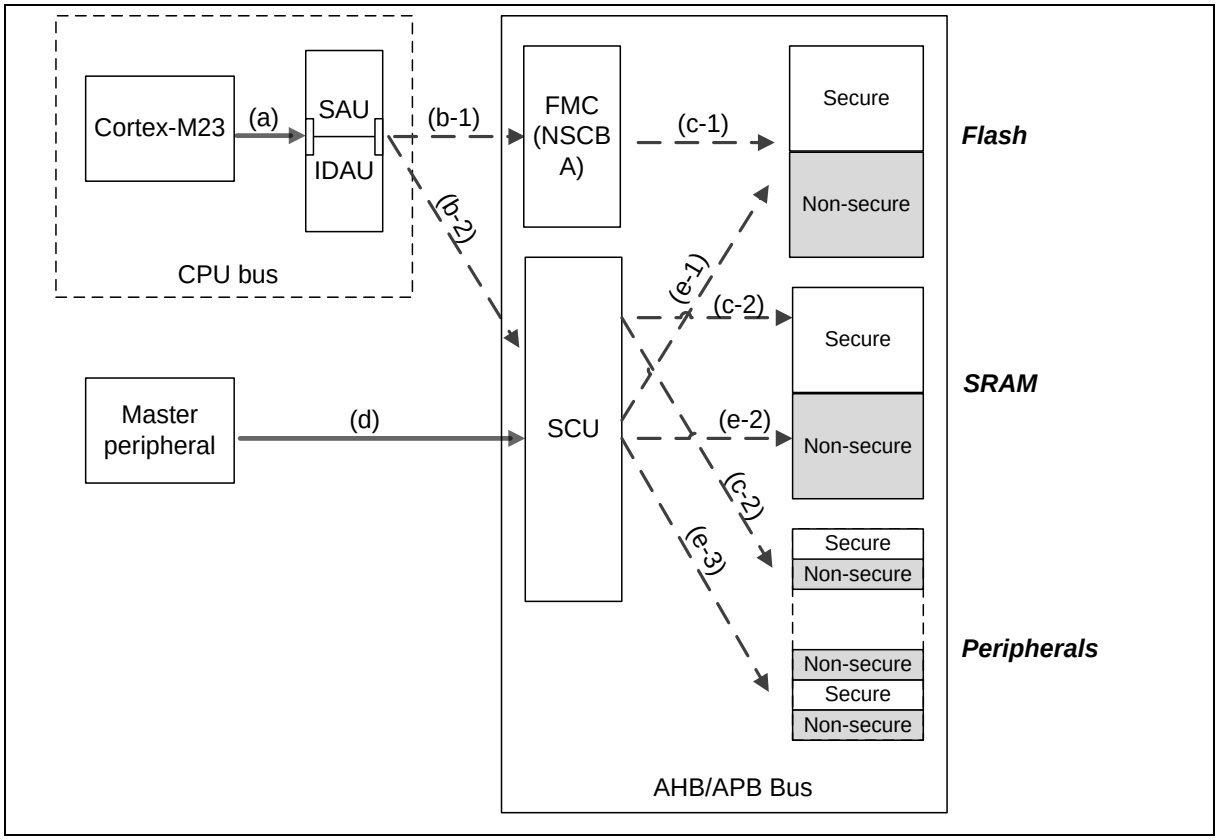


Figure 6.5-5 Checking Point of Accesses

Figure 6.5-5 illustrates how the above conditions are checked by TrustZone® related control units.

When the core processor tries to fetch instructions or access data, the security attribute of the core processor and target address are verified by SAU and IDAU (refer to (a)). If the core processor is in non-secure state and target address is secure, a hard fault exception will be generated. The other cases will go to next checkpoints (refer to (b-1) and (b-2)). If the non-secure code tries to read/write a secure memory or register, the access will be blocked and a secure violation interrupt (SCU interrupt) can be generated. If a secure code uses non-secure address to access a secure memory or register, the operation has no effect. (refer to (c-1) and (c-2))

When a master peripheral tries to read/write a memory or register, the SCU will verify the access (refer to (d)). When a non-secure master peripheral wants to access a secure memory or register, the access will be blocked and a secure violation interrupt (SCU interrupt) can be generated. If a secure master peripheral uses non-secure address to read/write a secure memory or register, the operation has no effect.

The responses of the accesses from the core processor and master peripherals follow the rule called memory access policy, which is described in the “Memory Access Policy (MAP)” section of “Security Configuration Unit (SCU)” chapter.

6.6 Flash Memory Controller (FMC)

6.6.1 Overview

The FMC is equipped with dual-bank on-chip embedded Flash (BANK0 and BANK1) for application. Both BANK0 and BANK1 have 256/512 Kbytes space. Thus, the total size of application ROM(APROM) is 512/1024 Kytes. A User Configuration block provides for system initiation in BANK0. A 16 Kbytes loader ROM (LDROM) is used for In-System-Programming (ISP) function in BANK0. A 3 Kbytes one-time-program ROM (OTP) is used for recording one-time-program data in BANK1. A 16 Kbytes Secure Bootloader is used to check boot code integrity and authenticity, and consists of native ISP functions. A 4 Kbytes cache with zero wait cycle is used to improve Flash access performance. This chip also supports In-Application-Programming (IAP) function. User switches the code executing without chip reset after the embedded Flash is updated.

6.6.2 Features

- Supports dual-bank Flash macro for safe firmware upgrade
- Supports dual-bank remapping
- Supports 512/1024 Kbytes application ROM (APROM)
- Supports 16 Kbytes loader ROM (LDROM)
- Supports 4 XOM (Execution Only Memory) regions to conceal user program in APROM.
- Supports 8 Kbytes Data Flash
- Supports 16 bytes User Configuration block to control system initiation
- Supports 3 Kbytes one-time-program ROM (OTP)
- Supports 2 Kbytes page erase for all embedded Flash
- Supports bank erase for APROM, except XOM regions.
- Supports two level locks for protecting secure region and non-sec region.
- Supports Secure Bootloader with native In-System-Programming (ISP) functions
- Supports Secure Boot function for check boot code integrity and authenticity
- Supports 32-bit/64-bit and multi-word Flash programming function
- Supports fast Flash programming verification function
- Supports CRC32 checksum calculation function
- Supports Flash all one verification function
- Supports In-System-Programming (ISP) / In-Application-Programming (IAP) to update embedded Flash memory
- Supports Non-Secure In-System-Programming (NS ISP) to update embedded Non-Secure Flash memory
- Supports cache memory to improve Flash access performance and reduce power consumption

6.6.3 Block Diagram

The Flash memory controller (FMC) consists of AHB slave interface, cache memory controller, boot loader, Flash control registers, Flash initialization controller, Flash operation control and embedded Flash memory. The block diagram of Flash memory controller is shown in Figure 6.6-1.

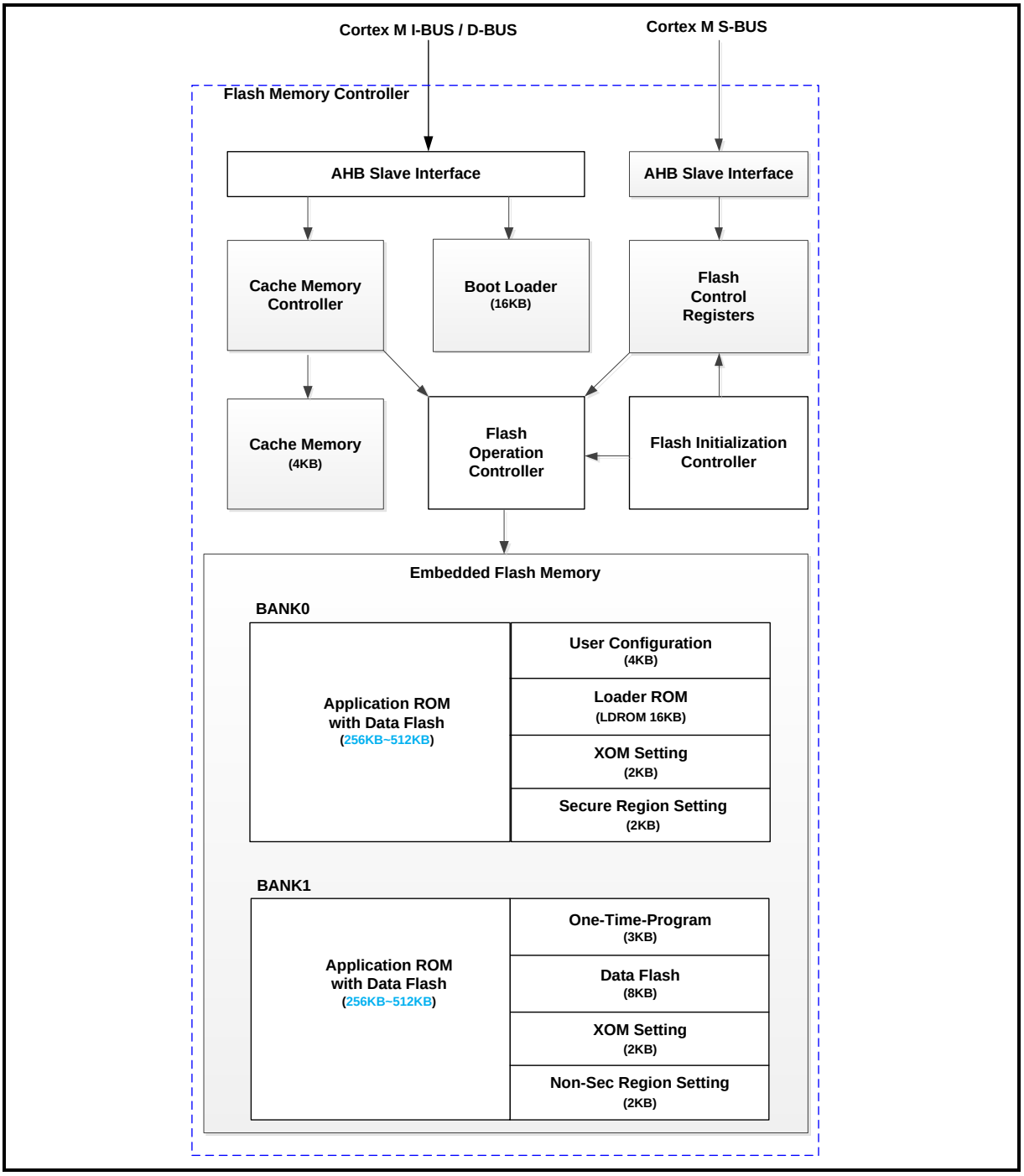


Figure 6.6-1 Flash Memory Controller Block Diagram

AHB Slave Interface

There are two AHB slave interfaces in Flash memory controller, one is from both Cortex®- M23 Bus for the instruction and data fetch;and the other is from Cortex®-M23 Bus for Flash control registers access including ISP registers.

Cache Memory Controller

A 4 Kbytes cache with zero wait cycle is implemented between Cortex®-M23 CPU and embedded

Flash memory. This cache memory controller improves the Flash access performance and reduces power consumption of the embedded Flash memory.

Secure Bootloader

The Secure Bootloader is 16 Kbytes. The Secure Bootloader content is read only, but not programmable

Flash Control Registers

All of ISP control and status registers are in the Flash control registers. The detail registers description is in the Register Description section. ISP Control Register Space can only be accessed by “Secure Master”.

Flash Initialization Controller

When chip is powered on or active from reset, the Flash initialization controller will start to access Flash automatically and check the Flash stability, and also reload User Configuration content to the Flash control registers for system initiation.

Flash Operation Controller

The Flash operations, such as Flash erase, Flash program, and Flash read operation, have specific control timing for embedded Flash memory. The Flash operation controller generates those control timing by requested from the cache memory controller, the Flash control registers and the Flash initialization controller.

Embedded Flash Memory

The embedded Flash memory is the main memory for user application code and parameters. It is consists of the user configuration block, 16 Kbytes LDROM, two 2 Kbytes XOM setting pages, 3 Kbytes OTP, 2 Kbytes Secure Region Setting, 2 Kbytes Non-Sec Region Setting, and 512/1024 Kbytes APROM The page erase Flash size is 2 Kbytes, and minimum program bit size is 32 bits.

6.6.4 Functional Description

FMC functions include the memory organization, boot selection, secure boot, IAP, ISP, the embedded Flash programming, and checksum calculation. The Flash memory map and system memory map are also introduced in the memory organization.

6.6.4.1 Memory Organization

The FMC memory consists of the dual-bank embedded Flash memory and Secure Bootloader. The dual-bank embedded Flash memory is programmable, and includes APROM, LDROM, the User Configuration block, OTP, XOM setting pages, Secure Region Setting page and Non-Sec Region Setting page. Secure Bootloader is a Mask ROM with ISP boot codes to support firmware download, boot control, security control and firmware execution. The address map includes Flash memory map and three system address maps: LDROM with IAP, APROM with IAP, and Boot Loader with IAP functions.

BANK	Flash Memory Block	Address Rang
0	APROM with 1024 Kbytes	0x00_0000 ~ 0x07_ffff
	APROM with 512 Kbytes	0x00_0000 ~ 0x03_ffff
	User Configuration	0x30_0000 ~ 0x30_000f
	LDROM	0x10_0000 ~ 0x10_3fff
	XOM Setting	0x20_0000 ~ 0x20_07ff
	Secure Region Setting	0x20_0800 ~ 0x20_0fff
1	APROM with 1024 Kbytes	0x08_0000 ~ 0x0f_ffff
	APROM with 512 Kbytes	0x04_0000 ~ 0x07_ffff
	OTP	0x31_0000 ~ 0x31_0bff
	Data Flash	0x11_0000 ~ 0x11_1fff
	Non-Secure Region Setting	0x21_0800 ~ 0x21_0fff

Table 6.6-1 Dual-Bank Block Address Range

6.6.4.2 LDROM and APROM

LDROM is designed for a loader to implement In-System-Programming (ISP) function by user. LDROM is a 16 Kbytes embedded Flash memory, the Flash address range is from 0x0010_0000 to 0x0010_3FFF. APROM is main memory for user applications. APROM size is 512/1024 Kbytes. All of the embedded Flash memory is 2 Kbytes page erased.

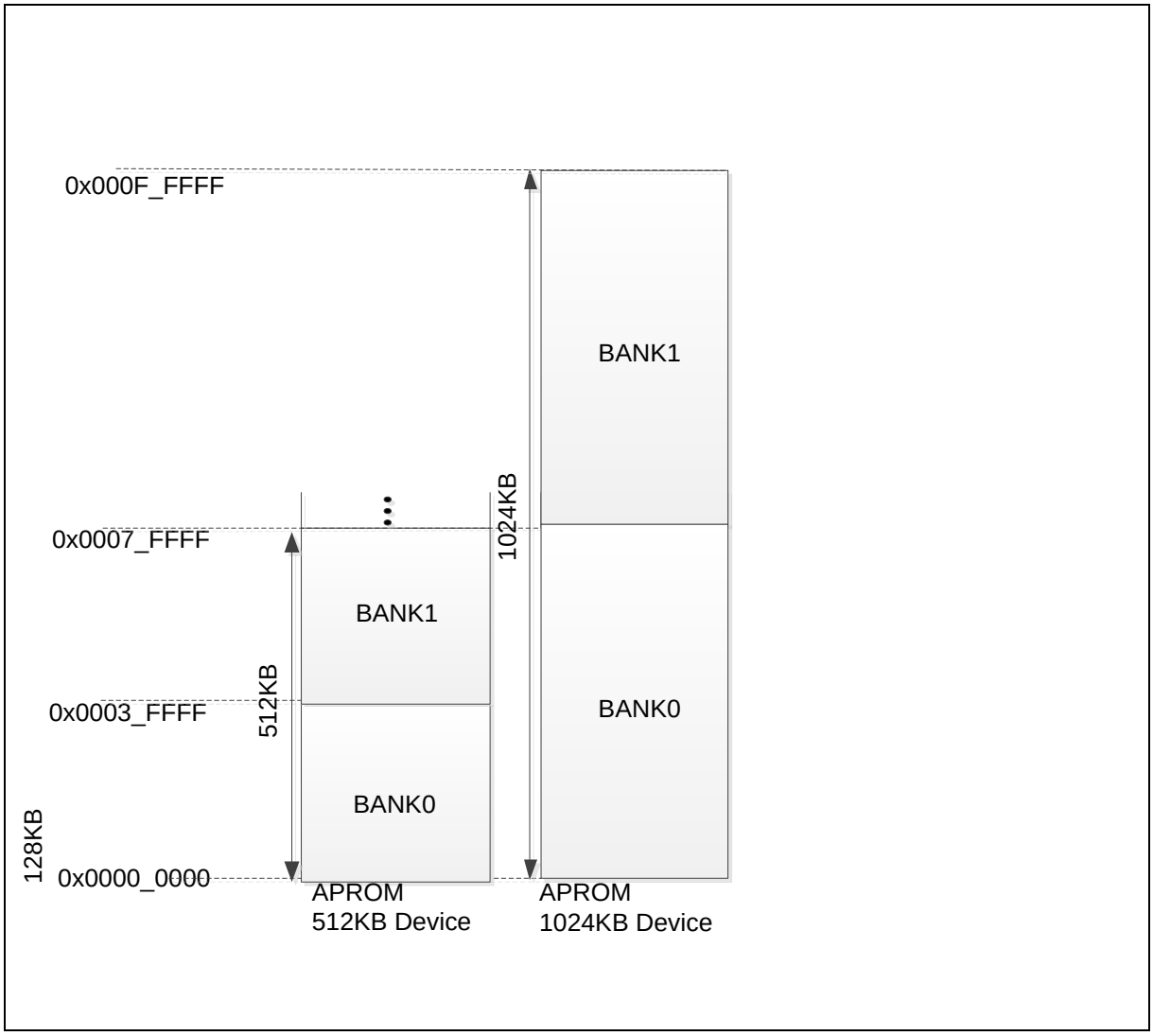


Figure 6.6-2 APROM Examples (512/1024 Kbytes)

6.6.4.3 Physical and Virtual Address Concept

For OTA application, the memory space concept is a little different form FMC commands and CPU viewpoint.

- From FMC command viewpoint, physical address concept is adopted. All of FMC commands regard APROM memory as physical address except “remapping” command. It means APROM Flash memory will be flat memory model regardless of memory remapping state.
- From Data Flash viewpoint, virtual address concept is adopted.
- From CPU viewpoint, virtual address concept is adopted. APROM memory space is separated into 2 segments (banks). As runtime stage, the virtual address depends on what address operation model (OP0, OP1) is, as shown in Figure 6.6-3.

6.6.4.4 APROM Reboot Address Operation Model Selection

To activate APROM reboot with different address operation model, use address operation model selection command 0x2C with FMC_ISPDAT “0x5AA5_5AA5” and FMC_ISPADDR ”0x0”(Address

OP0) or ”0x1”(Address OP1) to select address operation model.

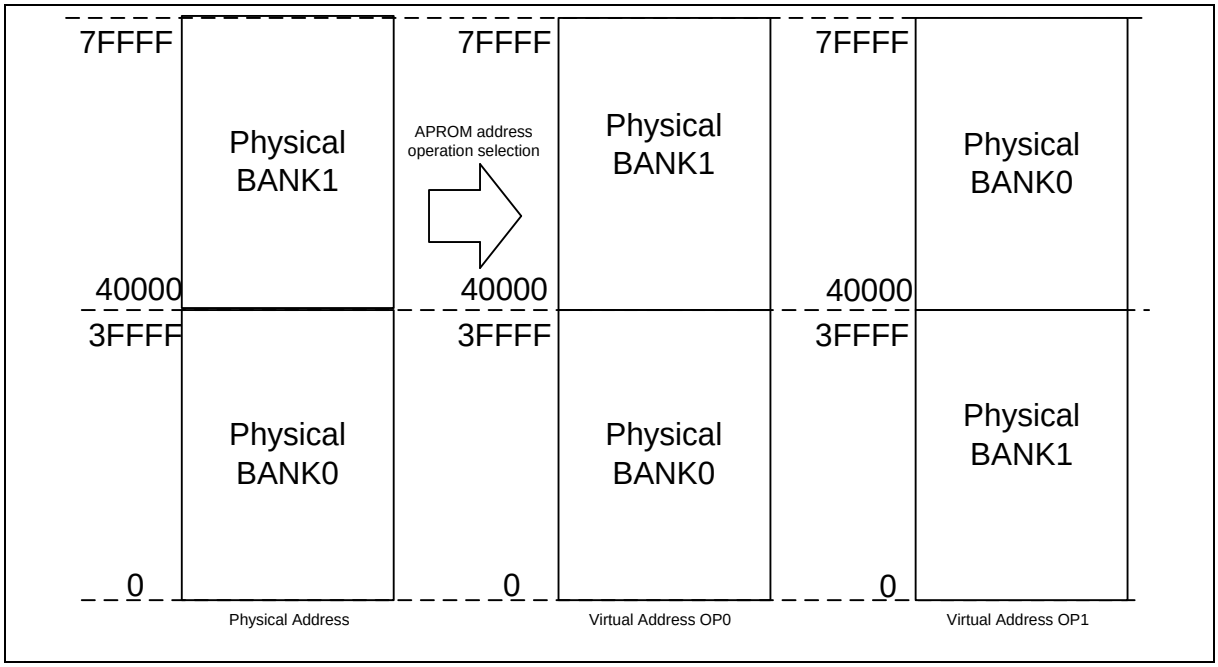


Figure 6.6-3 Address Operation Model

Since the M2354 has the TrustZone®, combined with bank remapping function, the mirror security boundary bit is supported make secure region be non-secure region after the bank address is remapped. The mirror security boundary bit is at NSCBA[31](0x210800). If NSCBA[31] =1 both bank0 and bank1 have each secure region. For example, if security boundary is set to 0x30000, NSCBA[31]=1 is set, Bank0 from address 0x0 to 0x2FFFF is secure region, and it is non-sec region from 0x30000 to 7FFFF. Bank1 from 0x80000 to 0xAFFFF is secure region, and it is non-sec region from 0xB0000 to FFFFF. If NSCBA = 80000 or 100000, all the region is secure region.

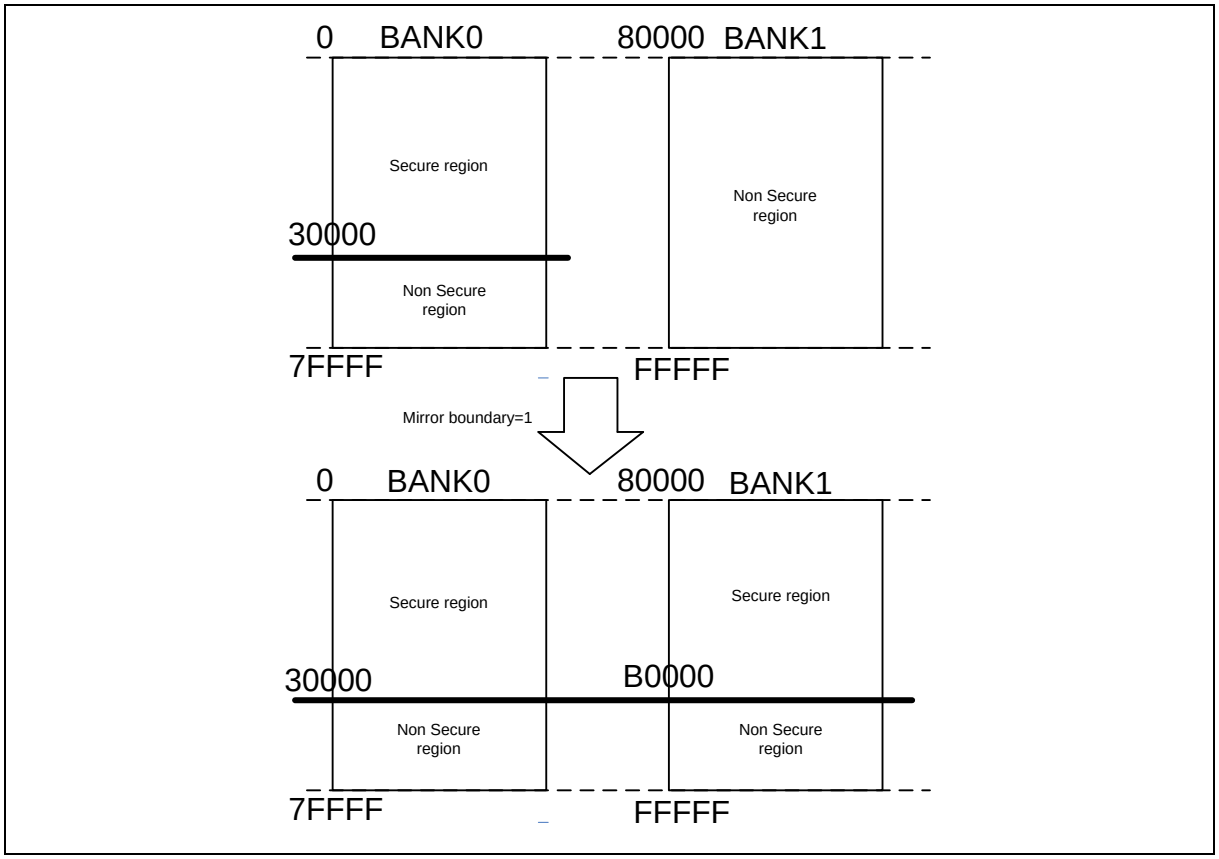


Figure 6.6-4 APROM REMAP Examples

6.6.4.5 Data Flash

Data Flash supports silent access and data scrambling for protection of user data. User should enter the 32-bit scrambling key through FMC_SCRKEY and then set SCRAMEN (FMC_DFCTL[0]) to enable data scrambling function in Data Flash. When data scrambling is enable, data in Data Flash is scrambled when written and de-scrambled when read. The 32-bit scrambling key is written only and read as zero.

The silent access is enabled by SILENTEN (FMC_DFCTL[1]), and the logical size is divided by two to store each word of data and its complement to reduce Data Flash reading noise. Therefore, the logical size of Data Flash is from 8 Kbytes to 4 Kbytes (4 pages of 1 Kbytes) while the silent access is enabled. Data scrambling and silent access operations can be performed on the same data in Data Flash. When both Data scrambling and silent access are enabled, the process is serial starting with scrambling, followed by silent access on writes and the reverse on reads. Note that the 64-bit programming, multi-word programming, and vector remap operations are not supported in Data Flash.

The Data Flash automatic programming function is used to clear the original user data in Data Flash when system detect the tamper attack event, and it is triggered by Tamper controller IP. When Data Flash automatic programming is enabled and tamper event occur, the physical 8 Kbytes size of Data Flash will be programmed with zero data. The status bit TMPCLRBUSY (FMC_DFSTS[1]) indicate whether the Data Flash automatic programming is under proceeding or not. TMPCLRDONE (FMC_DFSTS[0]) is set when Data Flash automatic programming is finished, and user can write 1 to clear it. In silence or scramble, CRC or all one check will fail, response not defined.

6.6.4.6 User Configuration Block

User Configuration block is internal programmable configuration area for boot options, such as boot

select and brown-out voltage level. It works like a fuse for power on setting. It is loaded from Flash memory to its corresponding control registers during chip power on. User can set these bits according to different application requests. User Configuration block can be updated by ISP function and located at 0x0030_0000 with four 32 bits words (CONFIG0, CONFIG1, CONFIG2 and CONFIG3). Any change on User Configuration block will take effect after system reboot.

CONFIG0 (Address = 0x0030_0000)

31	30	29	28	27	26	25	24
CWDTEN[2]	CWDTPDEN	Reserved		CFGXT1	Reserved		
23	22	21	20	19	18	17	16
CBOV			CBORST	CBODEN	Reserved		
15	14	13	12	11	10	9	8
Reserved				ICELOCK	CIOINI	Reserved	
7	6	5	4	3	2	1	0
CBS	Reserved	Reserved	CWDTEN[1:0]		Reserved		

Bits	Descriptions	
[31]	**CWDTEN[2]**	**Watchdog Timer Hardware Enable Bit** When watchdog timer hardware enable function is enabled, the watchdog enable bit WDTEN (WDT_CTL[7]) and watchdog reset enable bit RSTEN (WDT_CTL[1]) is set to 1 automatically after power on. The clock source of watchdog timer is force at LIRC and LIRC cannot be disabled. **CWDTEN[2:0]** is CONFIG0[31][4][3], 011 = WDT hardware enable function is active. WDT clock is always on except chip enters Power-down mode. When chip enter Power-down mode, WDT clock is always on if CWDTPDEN is 0 or WDT clock is controlled by LIRCEN (CLK_PWRCTL[3]) if CWDTPDEN is 1. Please refer to bit field description of CWDTPDEN. 111 = WDT hardware enable function is inactive. Others = WDT hardware enable function is active. WDT clock is always on.
[30]	**CWDTPDEN**	**Watchdog Clock Power-down Enable Bit** 0 = Watchdog Timer clock kept enabled when chip enters Power-down. 1 = Watchdog Timer clock is controlled by LIRCEN (CLK_PWRCTL[3]) when chip enters Power-down. **Note:** This bit only works if CWDTEN[2:0] is set to 011
[29:28]	**Reserved**	Reserved.
[27]	**CFGXT1**	**HXT Mode Selection** 0 = select HXT in external clock source mode. 1 = select HXT in crystal mode.
[26:24]	**Reserved**	Reserved.
[23:21]	**CBOV**	**Brown-out Voltage Selection** 000 = Brown-out voltage is 1.6V. 001 = Brown-out voltage is 1.8V. 010 = Brown-out voltage is 2.0V. 011 = Brown-out voltage is 2.2V. 100 = Brown-out voltage is 2.4V. 101 = Brown-out voltage is 2.6V. 110 = Brown-out voltage is 2.8V. 111 = Brown-out voltage is 3.0V.

[20]	**CBORST**	**Brown-out Reset Enable Bit** 0 = Brown-out reset Enabled after powered on. 1 = Brown-out reset Disabled after powered on.
[19]	**CBODEN**	**Brown-out Detector Enable Bit** 0= Brown-out detect Enabled after powered on. 1= Brown-out detect Disabled after powered on.
[18:12]	**Reserved**	Reserved.
[11]	**ICELOCK**	**ICE Lock Bit** 0 = ICE function Disabled. 1 = ICE function Enabled.
[10]	**CIOINI**	**I/O Initial State Selection** 0 = All GPIO set as Quasi-bidirectional mode after chip powered on. 1 = All GPIO set as input tri-state mode after powered on.
[9:8]	**Reserved**	Reserved.
[7]	**CBS**	**Chip Booting Selection** This bit is used for bootloader to perform normal boot from LDROM or APROM after MCU POR or cold reset. 0 = Bootloader performs normal boot from LDROM. 1 = Bootloader performs normal boot from APROM..
[6:5]	**Reserved**	Reserved.
[4:3]	**CWDTEN[1:0]**	**Watchdog Timer Hardware Enable Bit** When watchdog timer hardware enable function is enabled, the watchdog enable bit WDTEN (WDT_CTL[7]) and watchdog reset enable bit RSTEN (WDT_CTL[1]) is set to 1 automatically after power on. The clock source of watchdog timer is force at LIRC and LIRC cannot be disabled. **CWDTEN[2:0]** is CONFIG0[31][4][3], 011 = WDT hardware enable function is active. WDT clock is always on except chip enter Power-down mode. When chip enter Power-down mode, WDT clock is always on if CWDTPDEN is 0 or WDT clock is controlled by LIRCEN (CLK_PWRCTL[3]) if CWDTPDEN is 1. Please refer to bit field description of CWDTPDEN. 111 = WDT hardware enable function is inactive. Others = WDT hardware enable function is active. WDT clock is always on.
[2:0]	**Reserved**	Reserved.

Note: The config bits should be 1 if reserved.

CONFIG1 (Address = 0x0030_0004)

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

Bits	Description	
[31:0]	**Reserved**	Reserved.

Note: The config bits should be 1 if reserved.

CONFIG2 (Address = 0x0030_0008)

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

Bits	Description	
[31:0]	**Reserved** **Note:** After Mass erase or page erase CFG the value is 0xffff_fff5a	Reserved.

Note: The config bits should be 1 if reserved.

CONFIG3 (Address = 0x0030_000C)

31	30	29	28	27	26	25	24
TMPPD							
23	22	21	20	19	18	17	16
TMPPD							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					UART1PSL		

Bits	Descriptions	
[31:16]	**TMPPD**	**Tamper Domain Power Down Enable Bits** 0x5AA5 = Tamper domain power down. others = Tamper domain operating in normal mode.
[15:3]	**Reserved**	Reserved.
[2:0]	**UART1PSL**	**Bootloader UART1 Multi-function Pin Select** 000 = UART1_TXD (PB.7), UART1_RXD (PB.6). 001 = UART1_TXD (PA.9), UART1_RXD (PA.8). 010 = UART1_TXD (PF.0), UART1_RXD (PF.1). 011 = UART1_TXD (PB.3), UART1_RXD (PB.2). others = UART1_TXD (PA.3), UART1_RXD (PA.2).

Note: The config bits should be 1 if reserved.

6.6.4.7 Execution Only Memory (XOM)

The execution only memory (XOM) is used to store instructions for security application which are not allowed for data access via AHB-Bus. There are four XOM regions in APROM at most. To define a new XOM region, user must complete XOM settings in XOM setting page (0x20_0000 ~ 0x20_0008 for XOM region 0; 0x20_0010 ~ 0x20_0018 for XOM region 1; 0x20_0020 ~ 0x20_0028 for XOM region 2 ; 0x20_0030 ~ 0x20_0038 for XOM region 3) via In-System-Programming (ISP) function. After setting and restarting Flash initialization, user can check XOM status from FMC_XOMSTS and check each XOM range (i.e., base and size) from FMC_XOMR0STS~FMC_XOMR3STS. User needs to do chip reset after setting or erasing XOM. User can also use NS-ISP to setting XOM at Non-secure region.

When using XOM setting page and XOM regions, users must pay attention the following limitations:

- All XOM regions can be located in either secure region or non-secure region
- The base address of all XOM cannot be set as zero (first page)
- The XOM setting cannot exceed the APROM range; otherwise, unexpected errors will occur.
- Any XOM region must not include address 0xFF800~0xFFFFF.
- Any XOM region cannot be programmed after finishing its XOM setting
- Any XOM setting cannot be changed during runtime
- Only use Flash 32-bit program command to program XOM setting page
- Clear XOM setting and XOM region by mass erase command or XOM page erase function (a special erase command for XOM)
- User must restart Flash initialization to activate any new XOM setting
- ICE cannot step in XOM only if debug mode is enabled

Once the XOM region is active, user can only adopt mass erase command or XOM page erase function to clear XOM region and its corresponding XOM setting. To execute XOM page erase function, user must perform FLASH Page Erase (0x22) by writing the base address of the target XOM region in FMC_ISPADDR and the constant value 0x0055aa03 in FMC_ISPDAT. Note that the read-while-write function in dual-bank architecture will stop when executing XOM page erase function. When setting the XOM region, user needs to set Base address first and then set XOM page size. User should not set XOM at region where chip can boot from. User should not set the XOMs overlap each other because it would cause XOM cannot be normal active.

Example:

XOM0 base = 0x5000

XOM0 size = 3

XOM1 base = 0x4000

XOM1 size = 3 ---> ISPFF

XOM1 will not be active

In this case XOM1 base is already set, user can use XOM page erase to erase XOM1 base to clear the wrong setting.

XOMR0BASE (Address = 0x0020_0000)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
XOMR0BASE							
15	14	13	12	11	10	9	8
XOMR0BASE							
7	6	5	4	3	2	1	0
XOMR0BASE							

Bits	Descriptions	
[31:24]	**Reserved**	Reserved.
[23:0]	**XOMR0BASE**	**Base Address of XOM Region 0** XOMR0BASE must be page-aligned, and it cannot be set as zero. XOMR0BASE can be read and needs page alignment. **Note:** Page size is 2 Kbytes.

XOMR0SIZE (Address = 0x0020_0004)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
XOMR0SIZE							

Bits	Descriptions	
[31:8]	**Reserved**	Reserved.
[7:0]	**XOMR0SIZE**	**Page Number of XOM Region 0** XOMR0SIZE must be page-aligned and less than 255 pages. The minimum XOMR0SIZE value is 1 page. If XOMR0SIZE is written to 0, the register value will be set to 1. **Note:** Page size is 2 Kbytes.

XOMR0CTRL (Address = 0x0020_0008)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
XOMR0CTRL							

Bits	Descriptions	
[31:8]	**Reserved**	Reserved.
[7:0]	**XOMR0CTRL**	**Control of XOM Region 0** 0x5a = XOM region 0 is off. 0x50= XOM is in debug mode. The others = XOM region 0 is active. **Note:** But for Flash behavior,user cannot write 0 to 1.User needs to check the lock value is effective to change 0x5A(0101_1010). **Note:** The active state has come into effect after power-on or reset cycle.

XOMR1BASE (Address = 0x0020_0010)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
XOMR1BASE							
15	14	13	12	11	10	9	8
XOMR1BASE							
7	6	5	4	3	2	1	0
XOMR1BASE							

Bits	Descriptions	
[31:24]	**Reserved**	Reserved.
[23:0]	**XOMR1BASE**	**Base Address of XOM Region 1** XOMR0BASE must be page-aligned, and it cannot be set as zero. XOMR0BASE can be read and needs page alignment. **Note:** Page size is 2 Kbytes.

XOMR1SIZE (Address = 0x0020_0014)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
XOMR1SIZE							

Bits	Descriptions	
[31:8]	**Reserved**	Reserved.
[7:0]	**XOMR1SIZE**	**Page Number of XOM Region 1** XOMR1SIZE must be page-aligned and less than 255 pages. The minimum XOMR1SIZE value is 1 page. If XOMR1SIZE is written to 0, the register value will be set to 1. **Note:** Page size is 2 Kbytes.

XOMR1CTRL (Address = 0x0020_0018)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
XOMR1CTRL							

Bits	Descriptions	
[31:8]	**Reserved**	Reserved.
[7:0]	**XOMR1CTRL**	**Control of XOM Region 1 (Write Only)** 0x5a = XOM region 1 is off. 0x50= XOM is in debug mode. The others = XOM region 1 is active. **Note 1:** But for Flash behavior, user can not write 0 to 1. User needs to check the lock value is effective to change 0x5A(0101_1010). **Note 2:** The active state has come into effect after power-on or reset cycle.

XOMR2BASE (Address = 0x0020_0020)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
XOMR2BASE							
15	14	13	12	11	10	9	8
XOMR2BASE							
7	6	5	4	3	2	1	0
XOMR2BASE							

Bits	Descriptions	
[31:24]	**Reserved**	Reserved.
[23:0]	**XOMR2BASE**	**Base Address of XOM Region 2** XOMR0BASE must be page-aligned, and it cannot be set as zero. XOMR0BASE can be read and needs page alignment. **Note:** Page size is 2 Kbytes.

XOMR2SIZE (Address = 0x0020_0024)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
XOMR2SIZE							

Bits	Descriptions	
[31:8]	**Reserved**	Reserved.
[7:0]	**XOMR2SIZE**	**Page Number of XOM Region 2** XOMR2SIZE must be page-aligned and less than 255 pages. The minimum XOMR2SIZE value is 1 page. If XOMR2SIZE is written to 0, the register value will be set to 1. **Note:** Page size is 2 Kbytes.

XOMR2CTRL (Address = 0x0020_0028)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
XOMR2CTRL							

Bits	Descriptions	
[31:8]	**Reserved**	Reserved.
[7:0]	**XOMR2CTRL**	**Control of XOM Region 2 (Write Only)** 0x5a = XOM region 2 is off. 0x50= XOM is in debug mode. The others = XOM region 2 is active. **Note 1:** But for Flash behavior,user cannot write 0 to 1.User needs to check the lock value is effective to change 0x5A(0101_1010). **Note 2:** The active state has come into effect after power-on or reset cycle.

XOMR3BASE (Address = 0x0020_0030)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
XOMR3BASE							
15	14	13	12	11	10	9	8
XOMR3BASE							
7	6	5	4	3	2	1	0
XOMR3BASE							

Bits	Descriptions	
[31:24]	**Reserved**	Reserved.
[23:0]	**XOMR3BASE**	**Base Address of XOM Region 3** XOMR0BASE must be page-aligned, and it cannot be set as zero. XOMR0BASE can be read and needs page alignment. **Note:** Page size is 2 Kbytes.

XOMR3SIZE (Address = 0x0020_0034)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
XOMR3SIZE							

Bits	Descriptions	
[31:8]	**Reserved**	Reserved.
[7:0]	**XOMR3SIZE**	**Page Number of XOM Region 3** XOMR3SIZE must be page-aligned and less than 255 pages. The minimum XOMR3SIZE value is 1 page. If XOMR3SIZE is written to 0, the register value will be set to 1. **Note:** Page size is 2 Kbytes.

XOMR3CTRL (Address = 0x0020_0038)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
XOMR3CTRL							

Bits	Descriptions	
[31:8]	**Reserved**	Reserved.
[7:0]	**XOMR3CTRL**	**Control of XOM Region 3** 0x5a = XOM region 3 is off. 0x50= XOM is in debug mode. The others = XOM region 3 is active. **Note 1:** But for Flash behavior,user cannot write 0 to 1.User needs to check the lock value is effective to change 0x5A(0101_1010). **Note 2:** The active state has come into effect after power-on or reset cycle.

6.6.4.8 Two Level Locks (SCRLOCK and ARLOCK)

To meet the Arm® TrustZone® technology, FMC provides the Secure region and Non-secure region for secure world and non-secure world in the TrustZone® environment, respectively. The Non-secure region is shared with APROM and its size is configurable. The base address of Non-secure region is determined by Non-Sec boundary value (NSCBA in 0x0021_0800).

The NSCBA ~ 0x0007_FFFF/0x000F_FFFF is the Non-secure region for the Arm® TrustZone® technology when APROM size is 512/1024 Kbytes, respectively. That is, the range of secure region is from 0x0 to (NSCBA - 4) in APROM. To protect the secure region when ICE/TWICP/WRITER connects, the Secure Region Lock in 0x0061_0000 is used to start the read/write protection of secure region (i.e., SCRLOCK ≠ 0x5a). When SRLOCK is active, ICE/TWICP/WRITER only read 0xFFFF_FFFF and program ignored at the secure region.

In addition, FMC provide the All Region Lock (ARLOCK in 0x0061_0008) to avoid any access operation in all regions when ARLOCK≠0x5a and ICE/TWICP/WRITER connects.

OTP is not affected by Secure Region Lock and All Region Lock, and XOM is depend on its location which is secure region or non-secure region.

The NSCBA, SCRLOCK and Secure boot keys are located in the same page “Secure Region Setting” that is only erased by FLASH Mass Erase ISP CMD (0x26), as shown in Secure Boot Key Setup Flow. But Non-Sec boundary value (NSCBA in 0x0021_0800) can be erased by FLASH Page Erase (0x22) in the page “Non-Secure Region Setting”.

SCRLOCK (Address = 0x0061_0000)

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

Bits	Descriptions	
[31:8]	**Reserved**	Reserved.
[7:0]	**SCRLOCK**	**Secure Region Lock** 0x5a = The content of secure region is unlocked. The others = The content of secure region is read and write protection for ICE/TWICP/WRITER. **Note:** The lock/unlock state has come into effect after power-on or reset cycle.

NSCBA (Address = 0x0021_0800)

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

Bits	Descriptions	
[31]	**MirrBoundary**	**Mirror Boundary** 0= Mirror Boundary Disabled. 1= Mirror Boundary Enabled.
[30:24]	**Reserved**	Reserved.
[23:0]	**NSCBA**	**Non-sec Base Address (Page Aligned)** NSCBA is the start address of Non-secure region. That is, it is the boundary between Secure region and Non-Sec region. **Note:** The boundary has come into active after power-on or reset cycle. If user sets boundary 0x0, chip is automatically mapped to 0x800 to make sure the first page is secure. The minimum value is 0x800 and < 100000.

ARLOCK (Address = 0x0061_0008)

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

Bits	Descriptions	
[31:8]	**Reserved**	Reserved.
[7:0]	**ARLOCK**	**All Region Lock** 0x5a = The content of all regions include secure and non-secure regions are unlocked. The others = The content of all regions are read and write protection for ICE/TWICP/WRITER. **Note:** The lock/unlock state has come into effect after power-on or reset cycle.

6.6.4.9 One Time Program Memory (OTP)

The One time program memory (OTP) is used to store the important information that are not allowed user to modify them twice. The OTP is 3 Kbytes with location address 0x31_0000 ~ 0x31_0BFF. The maximum size of OTP data are 2 Kbytes from 0x31_0000 ~ 0x31_07FF. Every 64 bits of OTP data has one 32 bits “LOCK BIT” from 0x31_0800 to 0x31_0BFF, as shown in Fig. 6.4-4.

The purpose of “LOCK BIT” is recording if the programed address had been locked (LOCK BIT≠0xFFFF_FFFF) or not (LOCK BIT= 0xFFFF_FFFF) in OTP. For example, when LOCK BIT0 is not “FFFFFFFF”, OPT0 cannot be programmed again, regardless its content is all 1 or not. The “Flash page erase /mass erase command/ FLASH 64-bit Program/ FLASH Multi-Word Program” are never allowed to be executed in OTP. Before updating the content of OTP from 0x31_0000 to 0x31_07FF, users must check their “LOCK BIT” firstly. After finishing programming OTP data, users must write a non-0xFFFF_FFFF in the “LOCK BIT” of the above programed data to make sure that no one can modify them; only make sure OTP data cannot be changed. In some cases OTP data would be read. For example, when a chip is in SCRLOCK or ARLOCK, it cannot be read by NON-SEC code. Then doing chip erase SCRLOCK and ARLOCK would be unlock, and OTP data would not be erased.

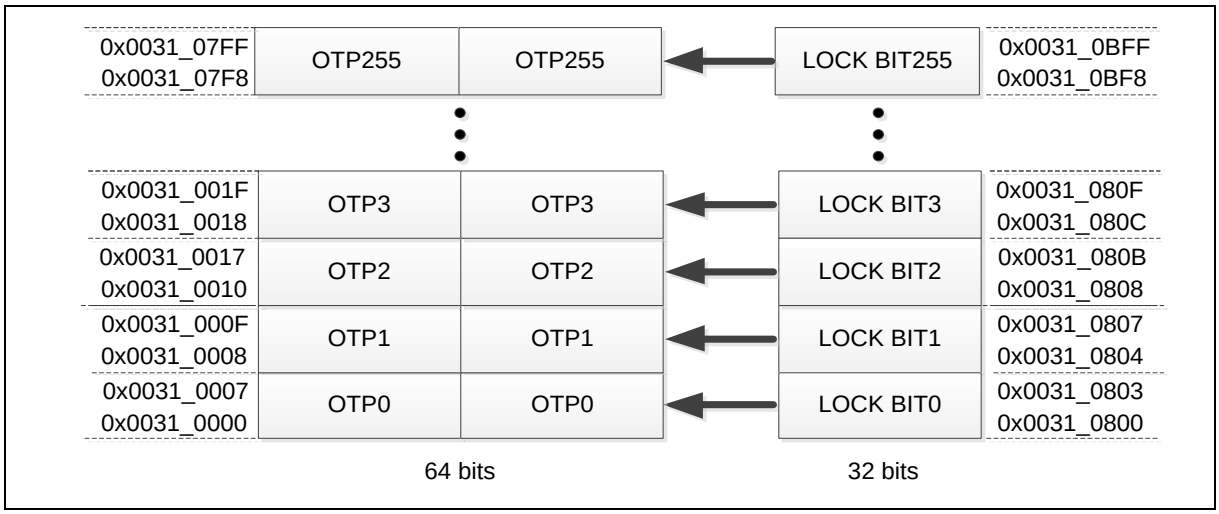


Figure 6.6-5 OTP Memory Map

6.6.4.10 Secure Boot Loader

Secure Boot Loader is used to store secure boot code. Secure Boot Loader is 16 Kbytes with location address 0x80_0000 ~ 0x80_3FFF. The Secure Boot Loader content is read only, but not programmable. CPU cannot access Secure Boot Loader after secure boot is finished. Secure Boot Loader can be accessed after POR, which causes Flash initial, WDT reset and PIN reset.

6.6.4.11 Flash Memory Map

The Flash memory map is different from system memory map. The system memory map is used by CPU fetch code or data from FMC memory. The Flash memory map is used for ISP function to read, program or erase FMC memory. The Flash memory map is as Figure 6.6-6.

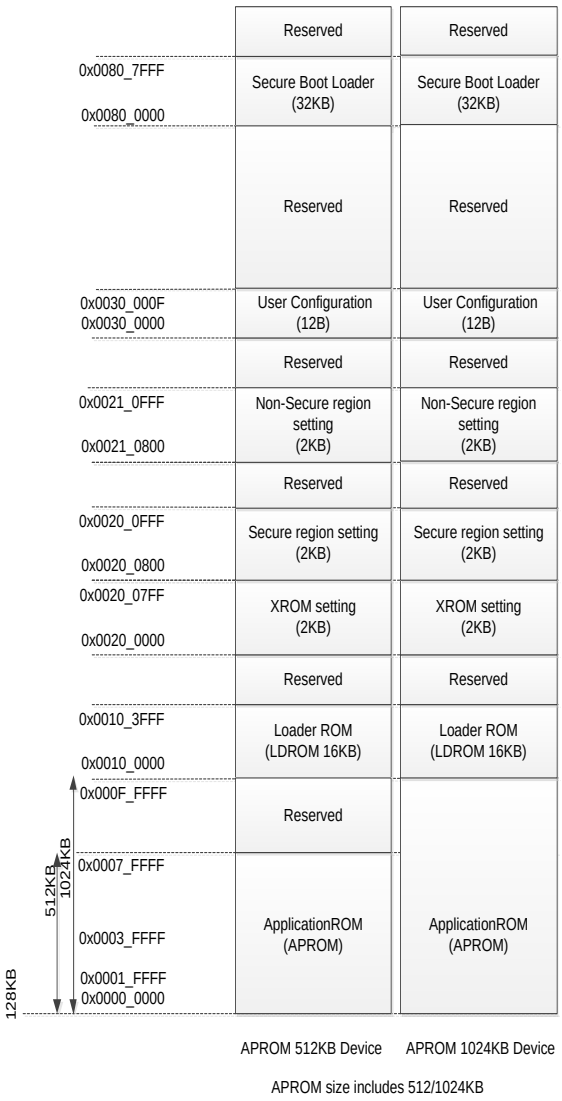


Figure 6.6-6 Flash Memory Map

6.6.4.12 System Memory Map with IAP Mode

The system memory map is used by CPU to fetch code or data from FMC memory. Boot Loader(0x0080_0000~0x0080_7FFF) and LDROM(0x0010_0000~0x0010_3FFF) address map are the same as in the Flash memory map. The 0x0000_0200~ 0x0007_FFFF/0x000F_FFFF) is APROM region for Cortex®-M23 data access when APROM size is 512/1024 Kbytes, respectively.

The address from 0x0000_0000 to 0x0000_01FF is called system memory vector.APROM, LDROM and Secure Bootloader can map to the system memory vector for CPU start up. There are three kinds of system memory map with IAP mode when chip booting: (1) LDROM with IAP, (2) APROM with IAP, and (3) Secure Bootloader with IAP.

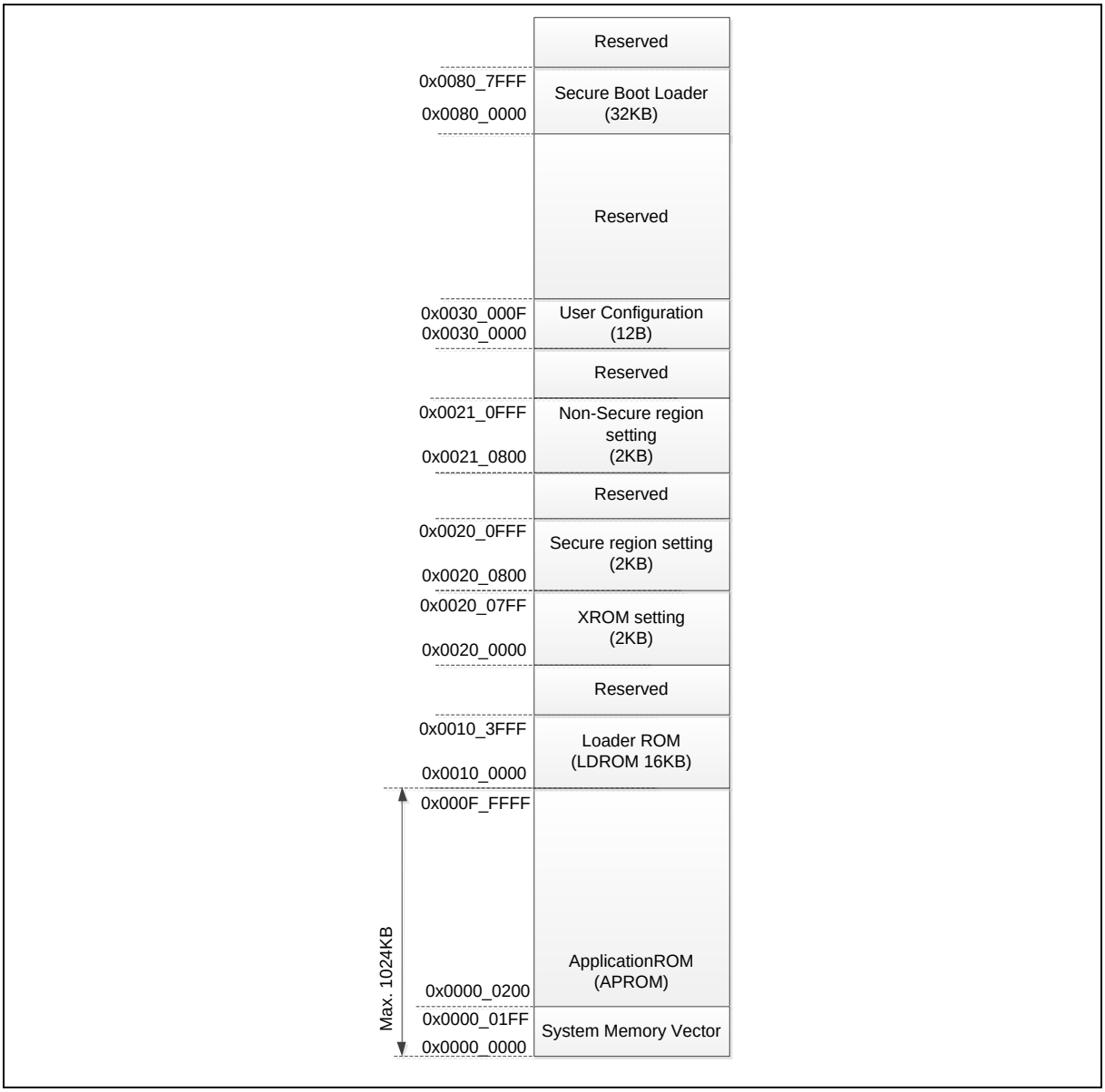


Figure 6.6-7 System Memory Map with IAP Mode

In LDROM with IAP mode, the LDROM (0x0010_0000~0x0010_01FF) is mapping to the system memory vector for Cortex®-M instruction or data access.

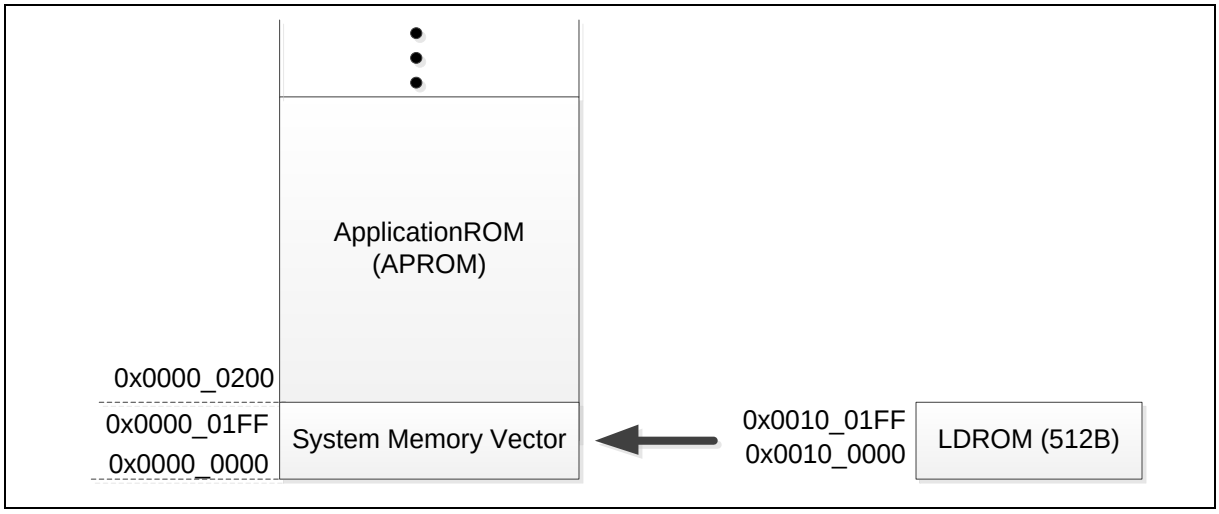


Figure 6.6-8 LDROM with IAP Mode

In APROM with IAP mode, the APROM (0x0000_0000~0x0000_01FF) is mapping to the system memory vector for Cortex®-M instruction or data access.

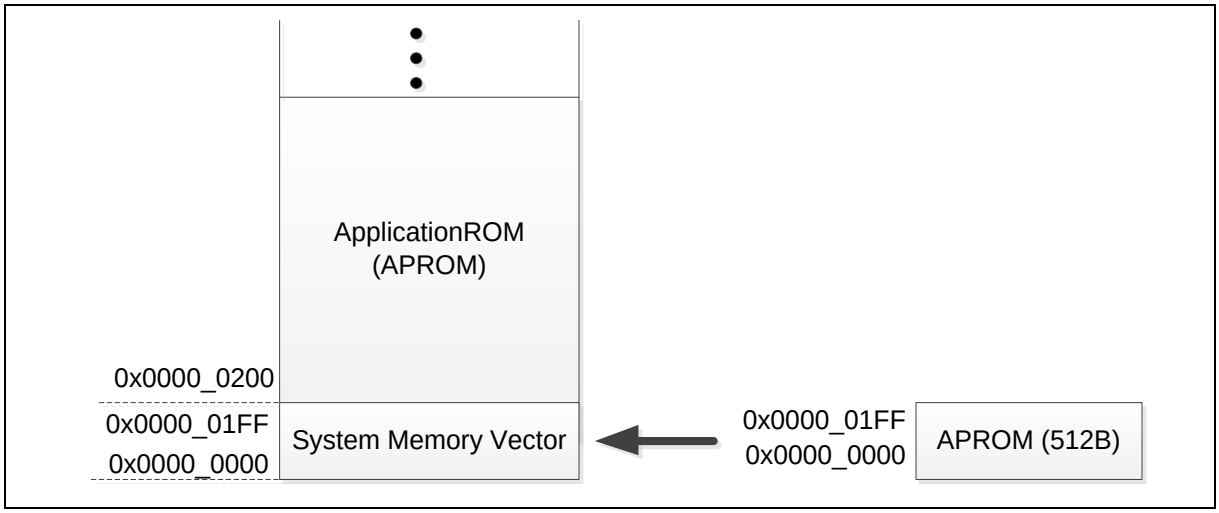


Figure 6.6-9 APROM with IAP Mode

In Secure Bootloader with IAP mode, the Secure Bootloader (0x0080_0000~0x0080_01FF) is mapping to the system memory vector for Cortex®-M instruction or data access.

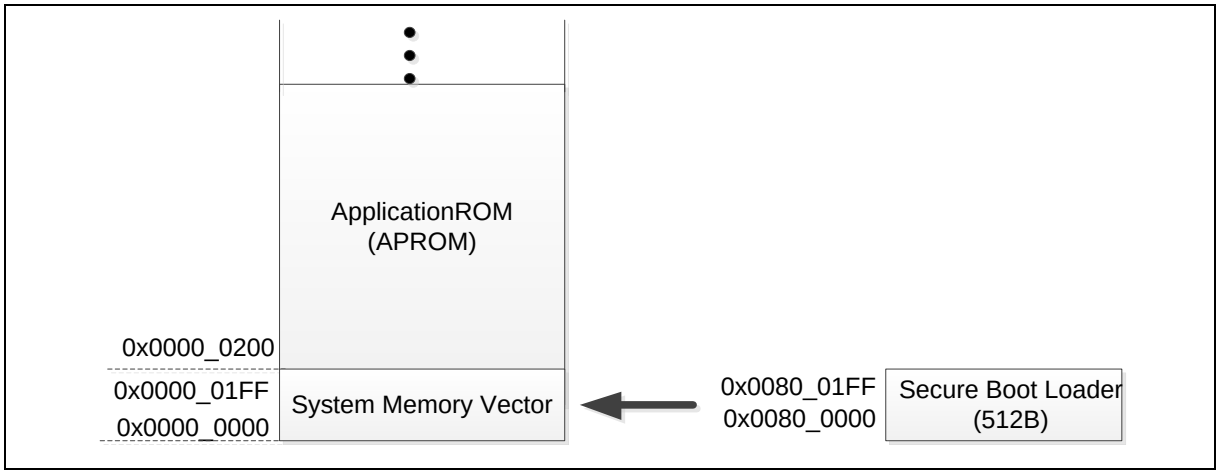


Figure 6.6-10 Boot Loader with IAP Mode

In system memory map with IAP mode, APROM, LDROM, APROM and Secure Bootloader can remap to the system memory vector when CPU running. User can write the target remap address to FMC_ISPADDR register and then trigger ISP procedure with the “Vector Remap” command (0x2E). In VECMAP (FMC_ISPSTS[23:9]), shows the finial system memory vector mapping address. When bank is remapped, both the source and destination addresses are remapped (0~512) mapped to 80000~80200 and so is the mapped source

6.6.4.13 In-Application-Programming (IAP)

The NuMicro® M2354 series provides In-Application-Programming (IAP) function for user to switch the system memory vector code executing between APROM, LDROM and Secure Bootloader. When chip boots with IAP function is enabled, any executable code (align to 512 bytes) is allowed to map to the system memory vector any time. User can change the remap address to FMC_ISPADDR and then trigger ISP procedure with the “Vector Remap” command.

6.6.4.14 In-System-Programming (ISP)

The M2354 series supports In-System-Programming (ISP) function allowing the embedded Flash memory to be reprogrammed under software control. ISP is performed without removing the microcontroller from the system through the firmware and on-chip connectivity interface, such as UART, USB, I^2C, SPI, and CAN (depended on chip feature). The target Flash memory space that ISP function operates cannot be across banks lists all ISP commands, except for XOM page erase function.

The ISP provides the following functions for embedded Flash memory.

- Supports Flash page erase function
- Supports Flash data program function
- Supports Flash data read function
- Supports company ID read function
- Supports device ID read function
- Supports unique ID read function
- Supports memory CRC32 checksum calculation function
- Supports Flash all one verification function
- Supports system memory vector remap function

ISP Commands

ISP Command	FMC_ISPCMD	FMC_ISPADDR	FMC_ISPDAT FMC_MPDAT0~FMC_MPDAT3
Flash Page Erase (XOM Page Erase)	0x22	Valid address of Flash memory organization. It must be page (2 Kbytes; 1 Kbytes in silent access protection of Data Flash) alignment. Note that FMC_ISPADDR[10:0] will be ignored.	FMC_ISPDAT = 0x0055aa03 : XOM page erase function Others : normal page erase function
Flash Bank Erase	0x23	Valid address of APROM of the target bank. Note that FMC_ISPADDR[15:0] will be ignored.	N/A
Flash Mass Erase (This command is only valid while MERASE(CONFIG0[13]) bit = 0.)	0x26	0x0000_0000	N/A
Flash 32-bit Program	0x21	Valid address of Flash memory organization	FMC_ISPDAT :Programming Data FMC_MPDAT0~FMC_MPDAT3 : N/A
Flash 64-bit Program	0x61	Valid address of Flash memory organization	FMC_ISPDAT :N/A FMC_MPDAT0: LSB Programming Data FMC_MPDAT1: MSB Programming Data FMC_MPDAT2~FMC_MPDAT3: N/A
Flash Multi-Word Program	0x27	Valid address of Flash memory organization in APROM and LDROM	FMC_ISPDAT :N/A FMC_MPDAT0: 1'st Programming Data FMC_MPDAT1: 2'nd Programming Data FMC_MPDAT2: 3'rd Programming Data FMC_MPDAT3: 4'th Programming Data
Flash Read	0x00	Valid address of Flash memory organization	FMC_ISPDAT: Return Data FMC_MPDAT0~FMC_MPDAT3 : N/A
Flash 64-bit Read	0x40	Valid address of Flash memory organization	FMC_ISPDAT: Return Data in FMC_ISPADDR FMC_MPDAT0: Return Data in FMC_ISPADDR FMC_MPDAT1: Return Data in FMC_ISPADDR+4 FMC_MPDAT2~FMC_MPDAT3 : N/A
Read Company ID	0x0B	0x0000_0000	FMC_ISPDAT: 0x0000_00DA FMC_MPDAT0~FMC_MPDAT3 : N/A
Read Device ID	0x0C	0x0000_0000	FMC_ISPDAT: Return Device ID FMC_MPDAT0~FMC_MPDAT3

			: N/A
Read CRC32 Checksum	0x0D	0x0000_0000	FMC_ISPDAT: Return Checksum FMC_MPDAT0~FMC_MPDAT3 : N/A
Bank REMAP	0x2C	0x0 (switch to bank0),0x1(switch to bank1)	0x5AA5_5AA5
Run CRC32 Checksum Calculation	0x2D	Valid start address of memory organization It must be 2 Kbytes page alignment and can't cross the bank	FMC_ISPDAT: Size It must be 2 Kbytes alignment FMC_MPDAT0~FMC_MPDAT3 : N/A
Read Flash All One Result	0x08	Keep address of "Run Flash All One Verification"	FMC_ISPDAT: Return Result 0xA110_0000 : Flash is not all one 0xA11F_FFFF: Flash is all one. FMC_MPDAT0~FMC_MPDAT3 : N/A
Run Flash All One Verification	0x28	Valid start address of memory organization It must be 2 Kbytes page alignment and can't cross the bank	FMC_ISPDAT: Size It must be 2 Kbytes alignment FMC_MPDAT0~FMC_MPDAT3 : N/A
Read Unique ID	0x04	0x0000_0000	FMC_ISPDAT: Unique ID Word 0 FMC_MPDAT0~FMC_MPDAT3 : N/A
		0x0000_0004	FMC_ISPDAT: Unique ID Word 1 FMC_MPDAT0~FMC_MPDAT3 : N/A
		0x0000_0008	FMC_ISPDAT: Unique ID Word 2 FMC_MPDAT0~FMC_MPDAT3 : N/A
Vector Remap	0x2E	Valid address in APROM,LDROM or boot loader It must be 512 bytes alignment	N/A

Table 6.6-2 ISP Command List

ISP Procedure

The FMC controller provides embedded Flash memory read, erase and program operation. Several control bits of FMC control register are write-protected, thus it is necessary to unlock before setting.

After unlocking the protected register bits, user needs to set the FMC_ISPCTL control register to decide to update LDROM, APROM or user configuration block, and then set ISPEN (FMC_ISPCTL[0]) to enable ISP function.Note that,when FMC is doing ISP command，user cannot enter any Power-down mode!

Once the FMC_ISPCTL register is set properly, user can set FMC_ISPCMD (refer to Table 6.6-2 ISP command list) for specify operation. Set FMC_ISPADDR for target Flash memory based on Flash memory organization. FMC_ISPDAT can be used to set the data to program or used to return the read data according to FMC_ISPCMD.

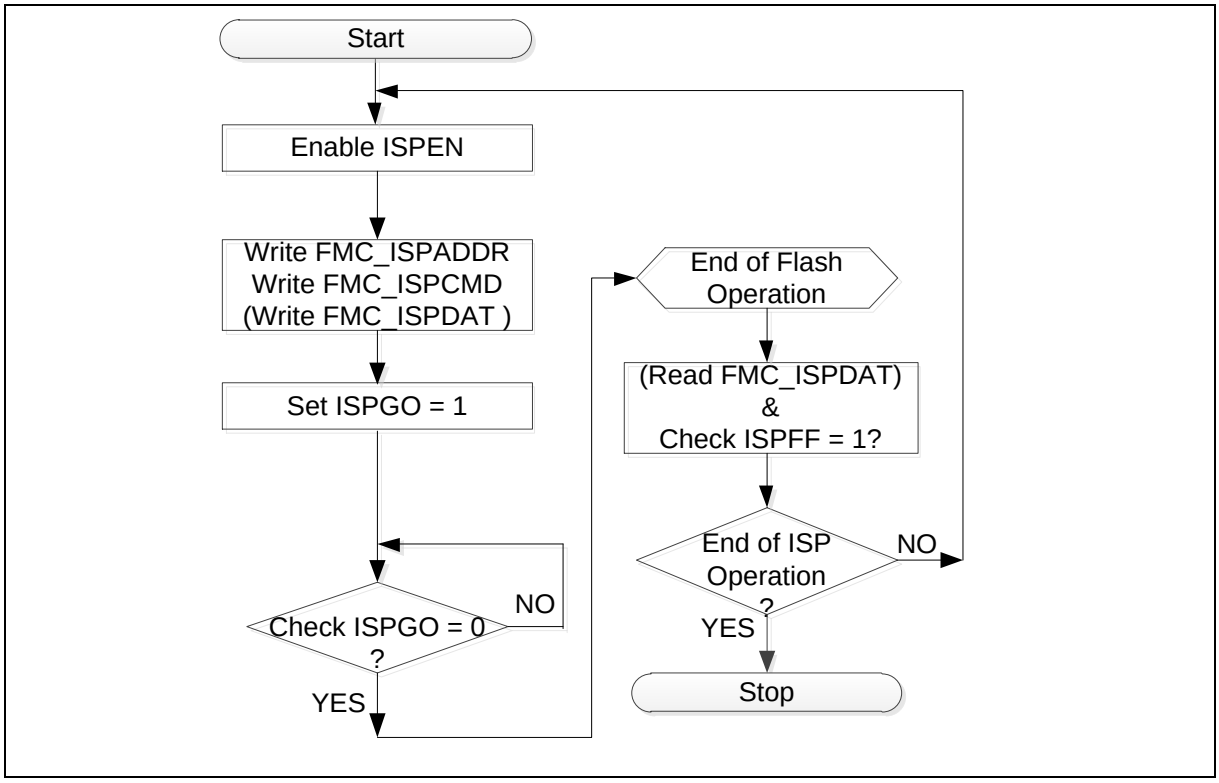


Figure 6.6-11 ISP Procedure Example

Finally, set the ISPGO (FMC_ISPTRG[0]) register to perform the relative ISP function. When ISP function is active, the ISPBUSY(FMC_ISPSTS[0]) and MPBUSY(FMC_MPSTS[0]) be set to 1. The ISPGO(FMC_ISPTRG[0]), ISPBUSY(FMC_ISPSTS[0]) and MPBUSY(FMC_MPSTS[0]) are self-cleared when ISP function has been done.

Several error conditions will be checked after ISP is completed. If an error condition occurs, ISP operation is not started and the ISP fail flag will be set instead. ISPFF(FMC_ISPSTS[6]) flag can only be cleared by software. The next ISP procedure can be started even ISPFF(FMC_ISPSTS[6]) bit is kept as 1. Therefore, it is recommended to check the ISPFF(FMC_ISPSTS[6]) bit and clear it after each ISP operation if it is set to 1.

When the ISPGO(FMC_ISPTRG[0]) bit is set and then CPU access the same bank, CPU will wait for ISP operation to finish during this period; the peripheral still keeps working as usual. If any interrupt request occurs, CPU will not service it till ISP operation is finished. When ISP operation is finished, the ISPGO bit will be cleared by hardware automatically. User can check whether ISP operation is finished or not by the ISPGO(FMC_ISPTRG[0]) bit.

When CPU access operation and ISP command are executed in different bank, CPU and ISP command can operate in parallel.

6.6.4.15 Non-secure In-System-Programming(NS-ISP)

The M2354 series supports Non-secure In-System-Programming (NS-ISP) function allowing the embedded Flash memory to be reprogrammed under non-secure software control. NS-ISP is performed without removing the microcontroller from the system through the firmware and on-chip connectivity interface, such as UART, USB, I^2C, SPI, and CAN (depending on chip feature). NS-ISP can only operate at non-secure region. Others remain the same as with the original ISP. The target Flash memory space that ISP function operates cannot be across banks, except for XOM page erase function.

The NS-ISP provides the following functions for embedded Flash memory.

- Supports Flash page erase function
- Supports Flash data program function
- Supports Flash data read function
- Supports company ID read function
- Supports device ID read function
- Supports unique ID read function
- Supports memory CRC32 checksum calculation function
- Supports Flash all one verification function

NS-ISP Commands

ISP Command	FMC_ISPCMD	FMC_ISPADDR	FMC_ISPDAT FMC_MPDAT0~FMC_MPDAT3
Flash Page Erase (XOM Page Erase)	0x22	Valid address of Flash memory organization. It must be page (2 Kbytes; 1 Kbytes in silent access protection of Data Flash) alignment. Note that FMC_ISPADDR[10:0] will be ignored.	FMC_ISPDAT = 0x0055aa03 : XOM page erase function Others : normal page erase function
Flash 32-bit Program	0x21	Valid address of Flash memory organization	FMC_ISPDAT :Programming Data FMC_MPDAT0~FMC_MPDAT3 : N/A
Flash Read	0x00	Valid address of Flash memory organization	FMC_ISPDAT: Return Data FMC_MPDAT0~FMC_MPDAT3 : N/A
Read Company ID	0x0B	0x0000_0000	FMC_ISPDAT: 0x0000_00DA FMC_MPDAT0~FMC_MPDAT3 : N/A
Read Device ID	0x0C	0x0000_0000	FMC_ISPDAT: Return Device ID FMC_MPDAT0~FMC_MPDAT3 : N/A
Read CRC32 Checksum	0x0D	0x0000_0000	FMC_ISPDAT: Return Checksum FMC_MPDAT0~FMC_MPDAT3 : N/A
Run CRC32 Checksum Calculation	0x2D	Valid start address of memory organization It must be 2 Kbytes page alignment and can't cross the bank	FMC_ISPDAT: Size It must be 2 Kbytes alignment FMC_MPDAT0~FMC_MPDAT3 : N/A
Read Flash All One Result	0x08	Keep address of "Run Flash All One Verification"	FMC_ISPDAT: Return Result 0xA110_0000 : Flash is not all one 0xA11F_FFFF: Flash is all one. FMC_MPDAT0~FMC_MPDAT3 : N/A
Run Flash All One Verification	0x28	Valid start address of memory organization It must be 2 Kbytes page alignment and	FMC_ISPDAT: Size It must be 2 Kbytes alignment

		can cross the bank	FMC_MPDAT0~FMC_MPDAT3 : N/A
Read Unique ID	0x04	0x0000_0000	FMC_ISPDAT: Unique ID Word 0 FMC_MPDAT0~FMC_MPDAT3 : N/A
		0x0000_0004	FMC_ISPDAT: Unique ID Word 1 FMC_MPDAT0~FMC_MPDAT3 : N/A
		0x0000_0008	FMC_ISPDAT: Unique ID Word 2 FMC_MPDAT0~FMC_MPDAT3 : N/A

Table 6.6-3 NS-ISP Command List

6.6.4.16 Embedded Flash Memory Programming

The NuMicro® M2354 series provides 32-bit, 64-bit and multi-word Flash memory programming function to speed up Flash updated procedure. Table 6.6-4 lists required FMC control registers in each embedded Flash programming function.

Register	Description	32-Bit Programming	64-Bit Programming	Multi-Word Programming
FMC_ISPCTL	ISP Control Register	√	√	√
FMC_ISPADDR	ISP Address Register	√	√	√
FMC_ISPDAT	ISP Data Register	√	N/A	N/A
FMC_ISPCMD	ISP CMD Register	0x21	0x61	0x27
FMC_ISPTRG	ISP Trigger Register	√	√	√
FMC_ISPSTS	ISP Status Register	√	√	N/A
FMC_MPDAT0	ISP Data0 Register	N/A	√	√
FMC_MPDAT1	ISP Data1 Register	N/A	√	√
FMC_MPDAT2	ISP Data2 Register	N/A	N/A	√
FMC_MPDAT3	ISP Data3 Register	N/A	N/A	√
FMC_MPSTA	ISP Multi-Program status	N/A	N/A	√
FMC_MPADDR	ISP Multi-Program Address	N/A	N/A	√

Table 6.6-4 FMC Control Registers for Flash Programming

64-bit Programming

The M2354 series 64-bit programming function is faster than 32-bit programming. FMC_ISPDAT is used for 32-bit programming data register. In 64-bit programming, there are two programming data registers, one is FMC_MPDAT0 for LSB word, and the other is FMC_MPDAT1 for MSB word, and ISP command is 0x61, the other registers are the same as 32-bit programming. Figure 6.6-12 and Figure 6.6-13 show the ISP 32-bit / 64-bit programing procedure flow.

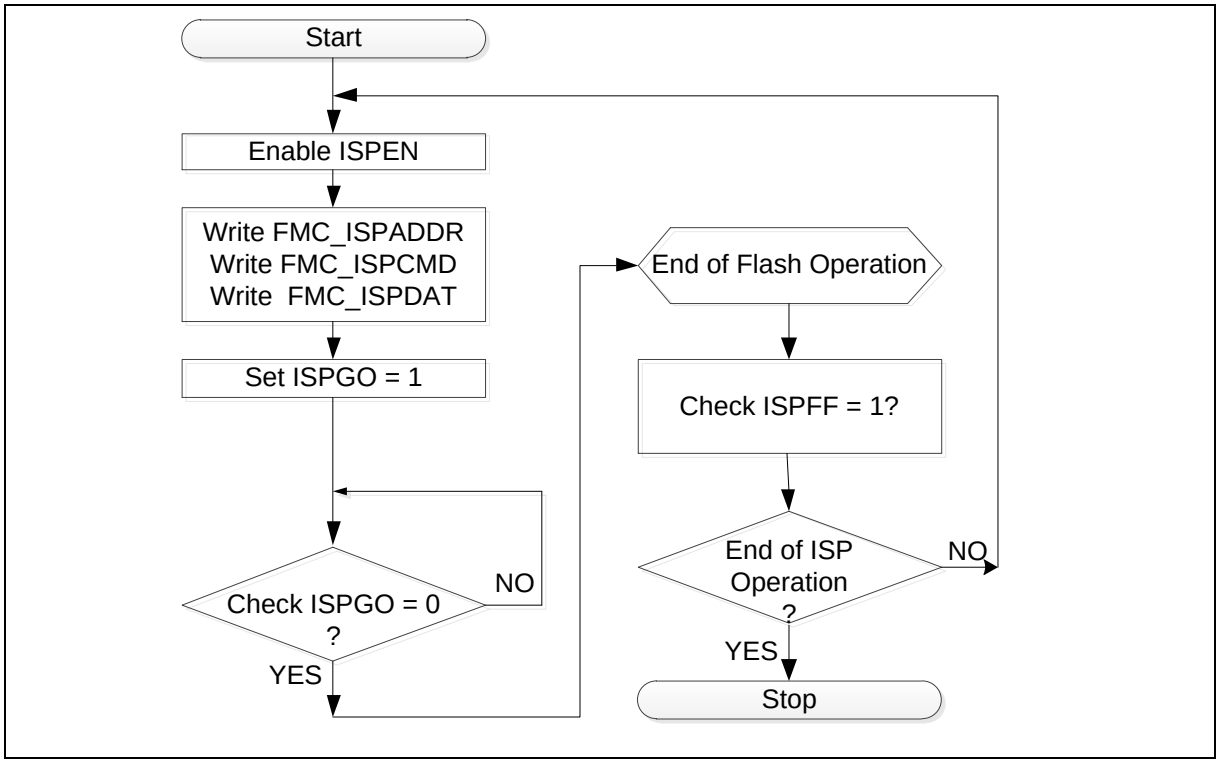


Figure 6.6-12 ISP 32-bit Programming Procedure

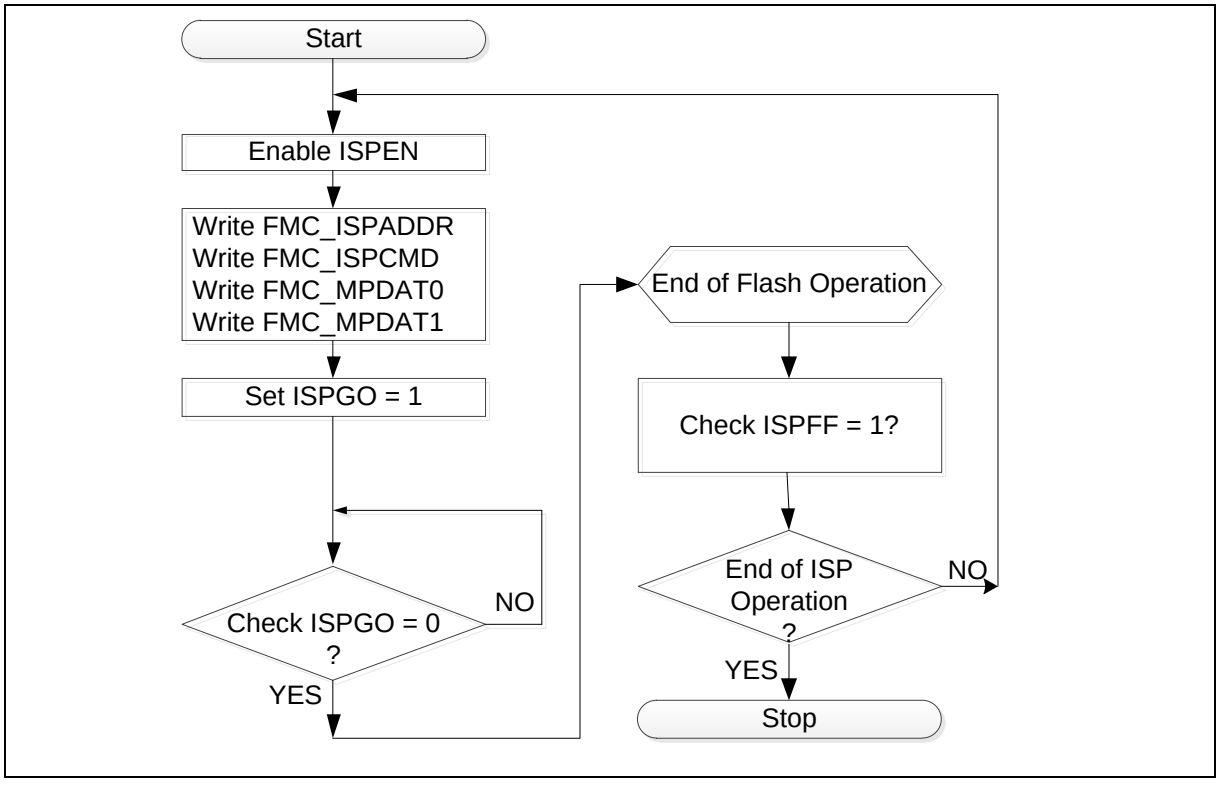


Figure 6.6-13 ISP 64-bit Programming Procedure

Multi-word Programming

The M2354 supports multi-word programming function to speed up Flash updated procedure. The maximum programming length is up to 256 bytes, and the minimum programming length is 8 bytes (2 words). The multi-word programming is the fastest programming function if the programming words more than 8 bytes, because only one set of Flash setup time and hold time is needed for one time operation.

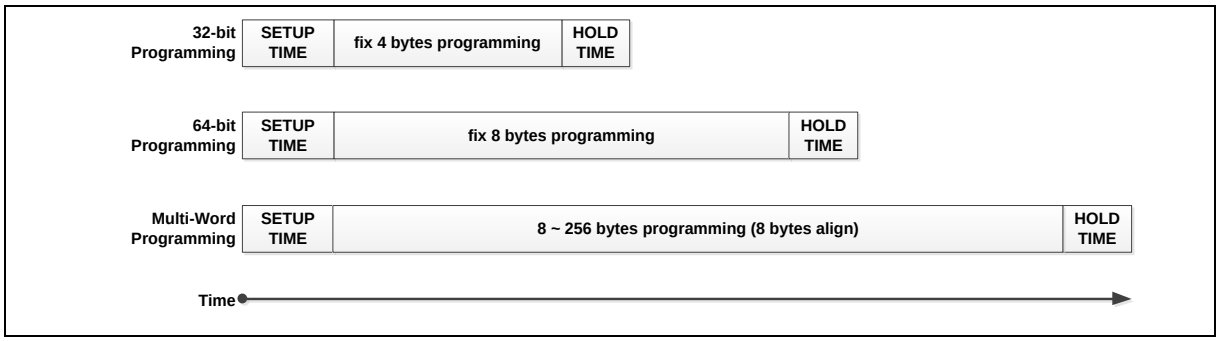


Figure 6.6-14 Multi-word Programming Time

In multi-word programming operation, the Cortex®-M23 CPU has to monitor the empty status of the programming buffer. CPU has to prepare the next data for programming continuity. The multi-program firmware should not be located in APROM or LDROM, because CPU instruction fetch cannot be hold. The firmware has to be located in Boot loader or embedded SRAM of chip to avoid CPU hold.

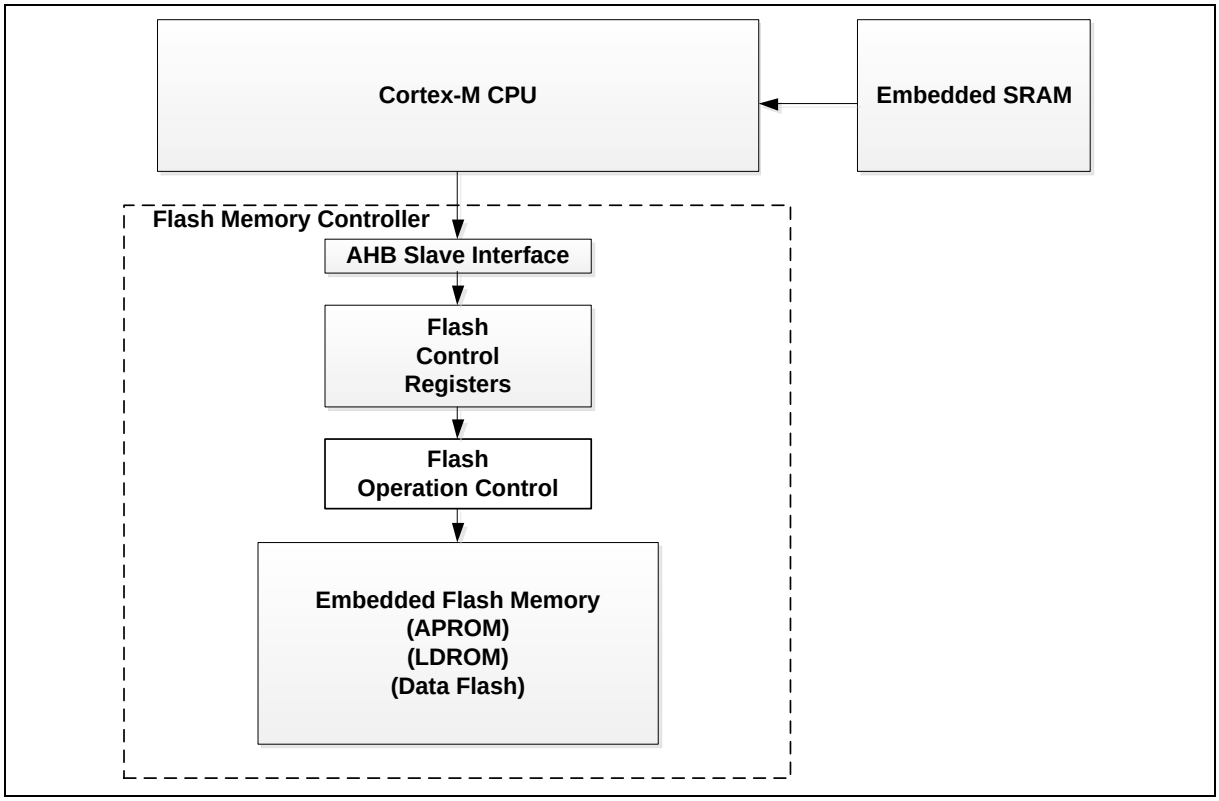


Figure 6.6-15 Firmware in SRAM for Multi-word Programming

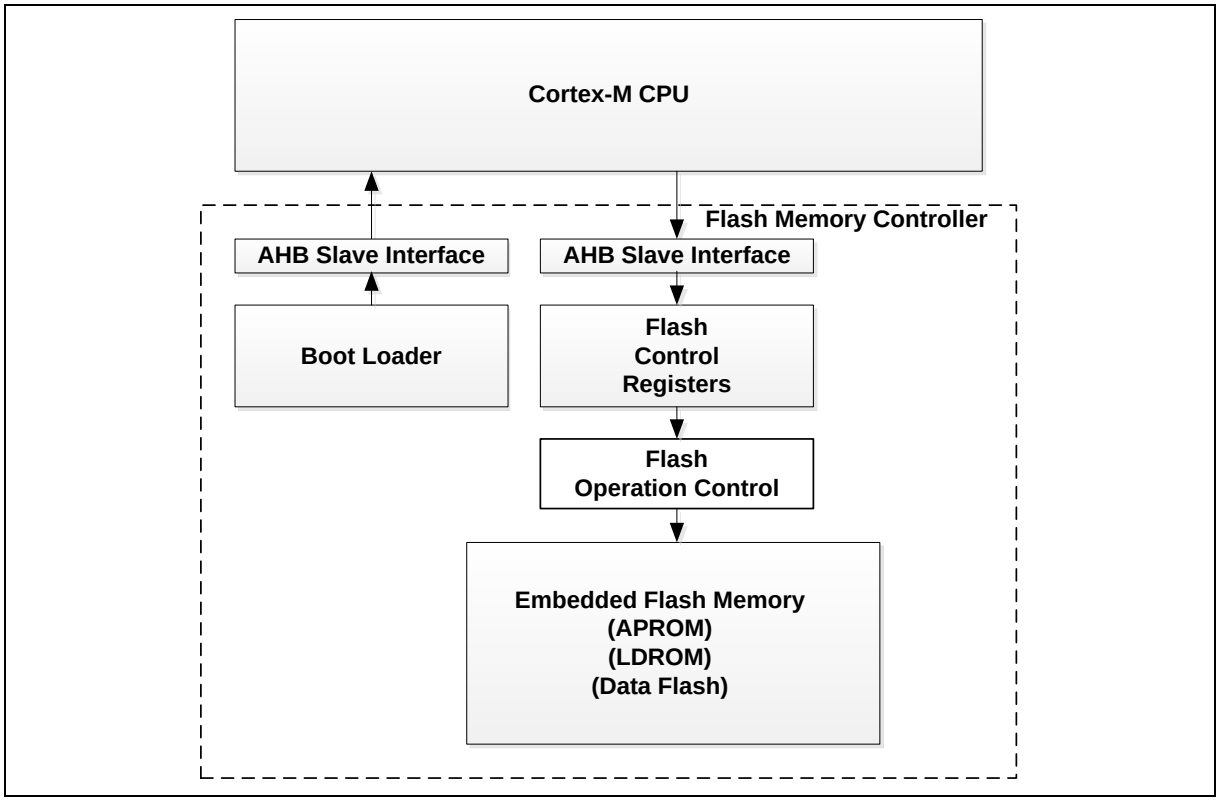


Figure 6.6-16 Firmware in Boot Loader for Multi-word Programming

The multi-word programming flow is shown in Figure 6.6-17. The starting ISP address (FMC_ISPADDR) has to be 8-byte align, FMC_ISPADDR[2:0] should be 0. ISPDAT0(FMC_MPDAT0) is the data word of the offset 0x0, ISPDAT1(FMC_MPDAT1) is the second word (offset 0x4), ISPDAT2(FMC_MPDAT2) is the third word (offset 0x8), and ISPDAT3(FMC_MPDAT3) is forth word (offset 0xC). If the starting ISP address FMC_ISPADDR [3] is 0, the 1st data word should put on ISPDAT0, and the 2nd word is ISPDAT1, the 3rd word is ISPDAT2, and the 4th word is ISPDAT3. If the starting ISP address FMC_ISPADDR [3] is 1, the 1st data word should put on ISPDAT2, and the 2nd word is ISPDAT3, the 3rd word is ISPDAT0, and the 4th word is ISPDAT1. The maximum programming size is 256 bytes and align to 256-byte address. While FMC controller performs multi-word programming operation, CPU needs to monitor the buffer status D3~D0(FMC_MPSTS[7:4]) and MPBUSY (FMC_MPSTS[0]) to wait the buffer empty ((D1,D0)=00, or (D3,D2)=00), and then CPU needs to update the next programming data (ISPDAT0, ISPDAT1, ISPDAT2 and ISPDAT3) in time. Otherwise, FMC controller will exit multi-word programming operation (MPBUSY (FMC_MPSTS[0]) = 0). If CPU cannot update the data in time (MPBUSY (FMC_MPSTS[0]) =0), CPU needs restart a new multi-word programming procedure to continue, FMC_MPADDR provides the last program address information. At the end of operation, CPU has to check ISPFF (FMC_MPSTS[2]) to confirm the multi-word operation is successfully completed.

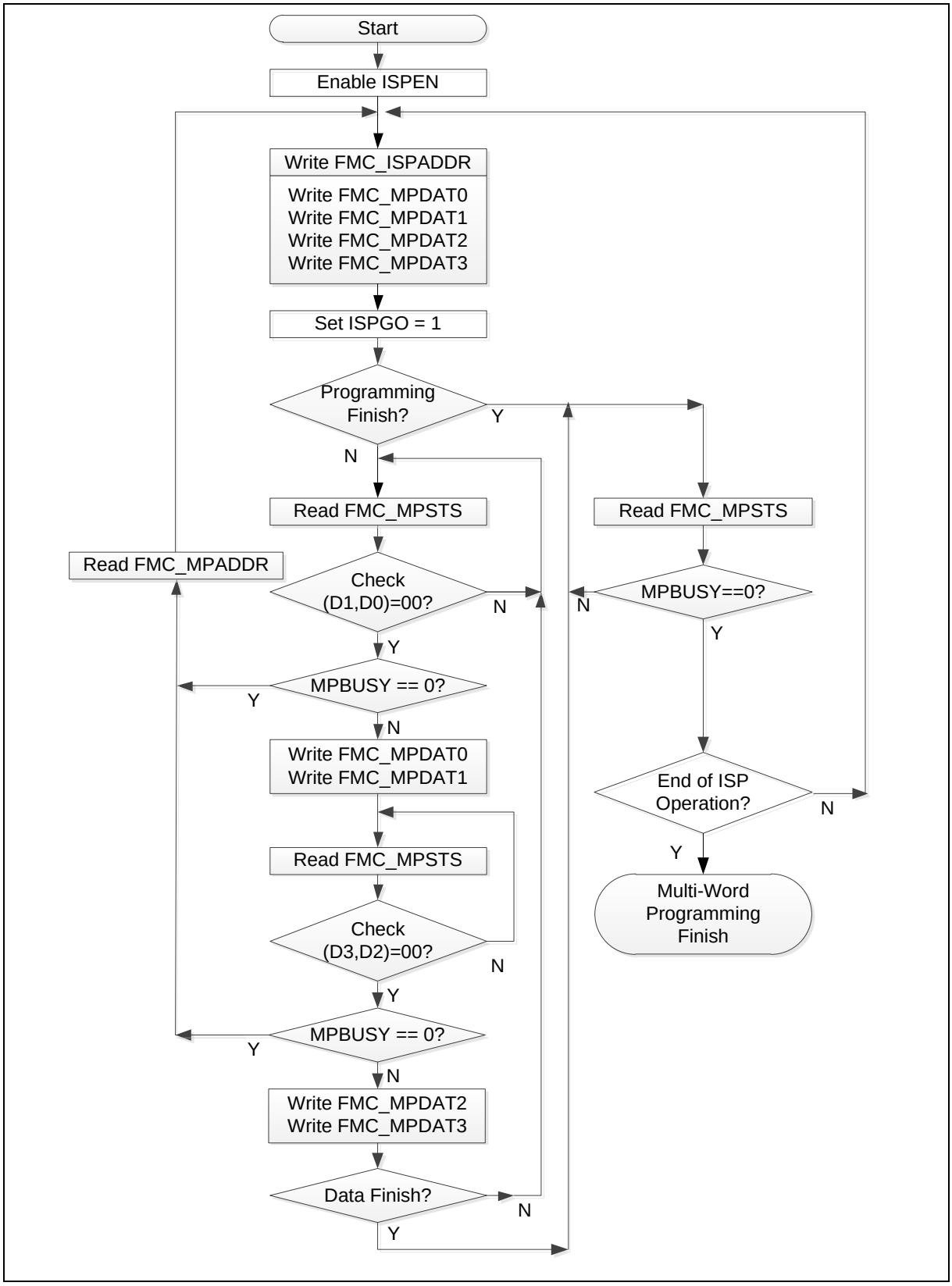


Figure 6.6-17 Multi-word Programming Flow

6.6.4.17 Fast Flash Programming Verification

In traditional Flash programming operation, the controller receives the programming trigger event then control the timing to perform the programming embedded Flash memory as show in Figure 6.6-18.

The M2354 supports the fast Flash programming verification function, which provides hardware verification for Flash programming to save time of the CPU read back and comparison. When data is programmed to the embedded Flash memory, the controller asserts the Flash read operation to read data out, and performs data comparison with data in. Finally, the comparison result is saved in PGFF (FMC_ISPSTS[5]). The PGFF is set to 1 if output data is not the same as the input programming data. The flag is kept until clear by software or a new erase operation. The fast Flash programming verification flow is shown in Figure 6.6-18.

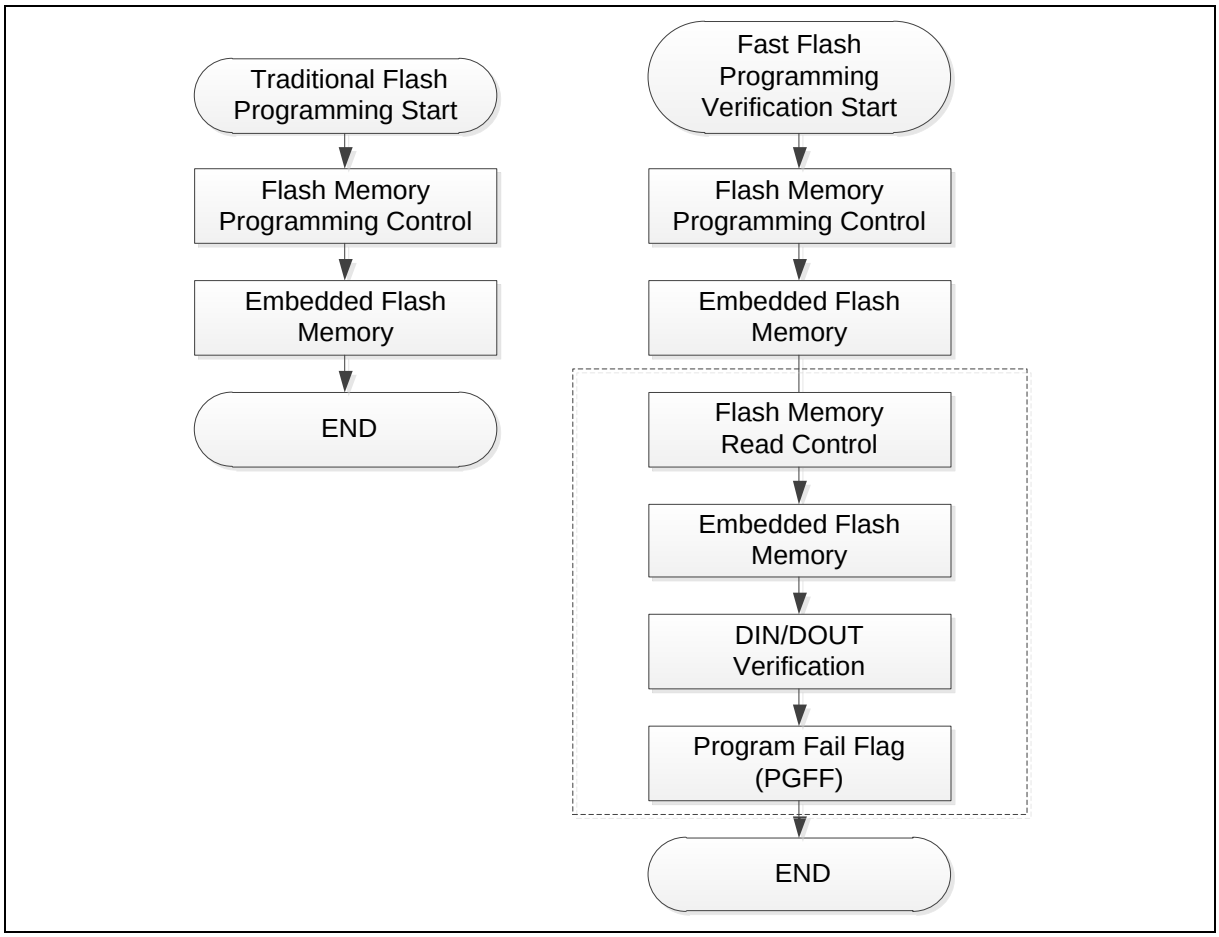


Figure 6.6-18 Fast Flash Programming Verification Flow

In traditional Flash updated operation, the Flash memory has to perform three steps to complete the Flash memory updated procedure, (1) Flash ERASE (2) Flash PROGRAM (3) Flash READ back all of data to check the correction. In the M2354 series, it only reads FMC_ISPSTS to check PGFF flag in Step (3) without reading data back to confirm. Figure 6.6-19 compares traditional programming verification flow and fast programming verification flow.

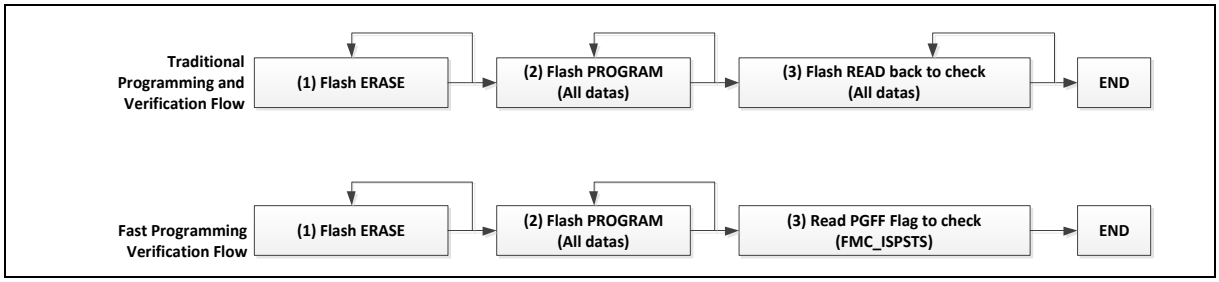


Figure 6.6-19 Verification Flow

The fast Flash programming verification function is released for 32-bit programming and 64-bit programming operation, but multi-word programming operation is not suitable due to the embedded Flash HV (High Voltage) of continue programming.

6.6.4.18 CRC32 Checksum Calculation

The M2354 series supports the CRC32 checksum calculation function to help user quickly check the memory content includes APROM, LDROM, Data Flash, and Boot Loader. The CRC32 polynomial is

CRC-32: $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$

With seed = 0xFFFF_FFFF

The CRC32 checksum calculation flow is shown in Figure 6.6-20.

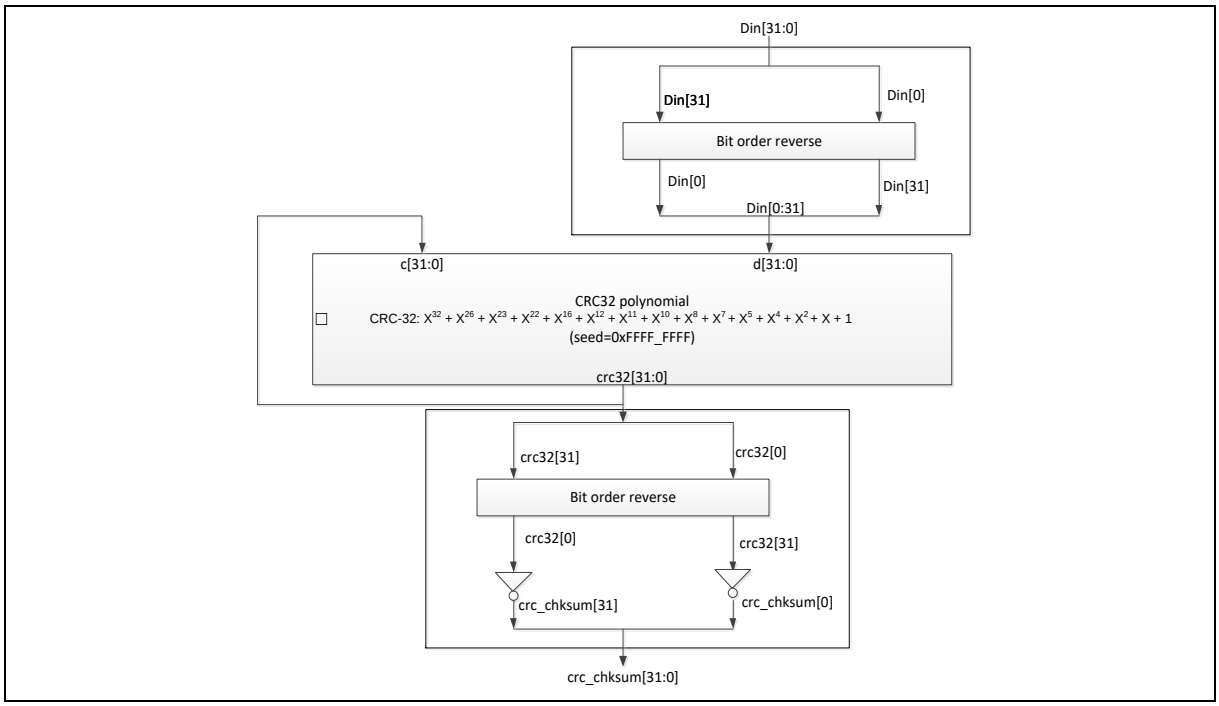


Figure 6.6-20 Flash CRC32 Checksum Calculation

Use the following three steps to complete the CRC32 checksum calculation.

1. Perform ISP “Run Memory CRC32 Checksum” operation
2. Perform ISP “Read Memory CRC32 Checksum” operation
3. Read FMC_ISPDAT to get checksum.

In Step 1, user has to set the memory starting address (FMC_ISPADDR) and size (FMC_ISPDAT) to calculate. Both address and size have to be 2 Kbytes alignment, the size should be ≥ 2 Kbytes and the starting address includes APROM, LDROM, Data Flash, and Boot Loader.

In Step 2, the FMC_ISPADDR should be kept as the same as Step 1.

In Step 3, the checksum is read from FMC_ISPDAT. If the checksum is 0x0000_0000, there is one of two conditions (1) Checksum calculation is in-progress, (2) Address and size is over device limitation

6.6.4.19 Flash All One Verification

The M2354 series supports the Flash all one verification function to help user quickly check a memory block content blanking for APROM, Data Flash, and LDROM after Flash erase operation.

Two or Three steps complete this Flash all one verification.

Two-step flow:

1. Perform ISP “Run Flash All One Verification” operation
2. Read ALLONE(FMC_ISPSTA[7])bit to get the verification result

   ALLONE: 1, all of Flash bits are 1 in verification block memory.

   ALLONE: 0, Flash bits are not all 1 in verification block memory.

Three-step flow:

1. Perform ISP “Run Flash All One Verification” operation.
2. Perform ISP “Read Flash All One Result” operation.
3. Read FMC_ISPDAT to get the verification result.

FMC_ISPDATA: 0xA11F_FFFF, all of Flash bits are 1 in verification block memory.

FMC_ISPDATA: 0xA110_0000, Flash bits are not all 1 in verification block memory

In Step 1, user has to set the memory starting address (FMC_ISPADDR) and size (FMC_ISPDAT) to verify. Both address and size have to be 2 Kbytes alignment, the size should be ≥ 2 Kbytes and the starting address includes APROM and LDROM.

In Step 2, the FMC_ISPADDR should be kept as the same as Step 1.

Note that the range of “Run All One Verification” cannot cross bank in one operation.

6.6.5 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
FMC Base Address **FMC_BA = 0x4000_C000** **FMC_NSBA = 0x5000_C000**				
FMC_ISPCTL	FMC_BA+0x00	R/W	ISP Control Register	0x0000_0008
FMC_ISPADDR	FMC_BA+0x04	R/W	ISP Address Register	0x0000_0000
FMC_ISPDAT	FMC_BA+0x08	R/W	ISP Data Register	0x0000_0000
FMC_ISPCMD	FMC_BA+0x0C	R/W	ISP Command Register	0x0000_0000
FMC_ISPTRG	FMC_BA+0x10	R/W	ISP Trigger Control Register	0x0000_0000
FMC_FTCTL	FMC_BA+0x18	R/W	Flash Access Time Control Register	0x0000_0000
FMC_ISPSTS	FMC_BA+0x40	R/W	ISP Status Register	0x0000_0000
FMC_CYCCTL	FMC_BA+0x4C	R/W	Flash Access Cycle Control Register	0x0000_0001
FMC_MPDAT0	FMC_BA+0x80	R/W	ISP Data0 Register	0x0000_0000
FMC_MPDAT1	FMC_BA+0x84	R/W	ISP Data1 Register	0x0000_0000
FMC_MPDAT2	FMC_BA+0x88	R/W	ISP Data2 Register	0x0000_0000
FMC_MPDAT3	FMC_BA+0x8C	R/W	ISP Data3 Register	0x0000_0000
FMC_MPSTS	FMC_BA+0xC0	R	ISP Multi-program Status Register	0x0000_0000
FMC_MPADDR	FMC_BA+0xC4	R	ISP Multi-program Address Register	0x0000_0000
FMC_XOMR0STS	FMC_BA+0xD0	R	XOM Region 0 Status Register	0xFFF8_00FF
FMC_XOMR1STS	FMC_BA+0xD4	R	XOM Region 1 Status Register	0xFFF8_00FF
FMC_XOMR2STS	FMC_BA+0xD8	R	XOM Region 2 Status Register	0xFFF8_00FF
FMC_XOMR3STS	FMC_BA+0xDC	R	XOM Region 3 Status Register	0xFFF8_00FF
FMC_XOMSTS	FMC_BA+0xE0	R	XOM Status Register	0x0000_0000
FMC_DFCTL	FMC_BA+0x100	R/W	Data Flash Function Control	0x0000_0000
FMC_DFSTS	FMC_BA+0x108	R/W	Data Flash Status	0x0000_0000
FMC_SCRKEY	FMC_BA+0x10C	W	Data Flash Scrambling Key	0x0000_0000
NS_FMC_ISPCTL	FMC_NSBA+0x00	R/W	Non-secure ISP Control Register	0x0000_0000
NS_FMC_ISPADDR	FMC_NSBA+0x04	R/W	Non-secure ISP Address Register	0x0000_0000
NS_FMC_ISPDAT	FMC_NSBA+0x08	R/W	Non-secure ISP Data Register	0x0000_0000

NS_FMC_ISPCMD	FMC_NSBA+0x0C	R/W	Non-secure ISP Command Register	0x0000_0000
NS_FMC_ISPTRG	FMC_NSBA+0x10	R/W	Non-secure ISP Trigger Control Register	0x0000_0000
NS_FMC_ISPSTS	FMC_NSBA+0x40	R/W	Non-secure ISP Status Register	0x0000_0000
NS_FMC_XOMR0STS	FMC_NSBA+0xD0	R	Non-secure XOM Region 0 Status Register	0xFFF8_00FF
NS_FMC_XOMR1STS	FMC_NSBA+0xD4	R	Non-secure XOM Region 1 Status Register	0xFFF8_00FF
NS_FMC_XOMR2STS	FMC_NSBA+0xD8	R	Non-secure XOM Region 2 Status Register	0xFFF8_00FF
NS_FMC_XOMR3STS	FMC_NSBA+0xDC	R	Non-secure XOM Region 3 Status Register	0xFFF8_00FF
NS_FMC_XOMSTS	FMC_NSBA+0xE0	R	Non-secure XOM Status Register	0x0000_0000

6.6.6 Register Description

ISP Control Register (FMC_ISPCTL)

Register	Offset	R/W	Description	Reset Value
FMC_ISPCTL	FMC_BA+0x00	R/W	ISP Control Register	0x0000_0008

31	30	29	28	27	26	25	24
Reserved							**INTEN**
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**ISPFF**	**LDUEN**	**CFGUEN**	**APUEN**	**Reserved**		**ISPEN**

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24]	**INTEN**	**Secure ISP INT Enable Bit (Write Protect)** 0= ISP INT Disabled. 1= ISP INT Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register. Before using INT, user needs to clear the INTFLAG(FMC_ISPSTS[24]) make sure INT happen at correct time.
[23:7]	**Reserved**	Reserved.

[6]	**ISPFF**	**ISP Fail Flag (Write Protect)** This bit is set by hardware when a triggered ISP meets any of the following conditions: This bit needs to be cleared by writing 1 to it. • APROM writes to itself if APUEN is set to 0. • LDROM writes to itself if LDUEN is set to 0. • CONFIG is erased/programmed if CFGUEN is set to 0. • Page Erase command at LOCK mode with ICE connection • Erase or Program command at brown-out detected • Destination address is illegal, such as over an available range. • Invalid ISP commands • APROM is erased/programmed if KEYLOCK is set to 1 • LDROM is erased/programmed if KEYLOCK is set to 1 • CONFIG is erased/programmed if KEYLOCK is set to 1 and KEYENROM[0] is 0 • Read any content of boot loader with ICE connection • The address of block erase and bank erase is not in APROM • ISP CMD in XOM region, except mass erase, page erase and chksum command • The wrong setting of page erase ISP CMD in XOM • Violate XOM setting one time protection • Page erase ISP CMD in Secure/Non-secure region setting page • Mass erase when MERASE (CFG0[13]) is disabled • Page erase, mass erase, multi-word program or 64-bit word program in OTP • ISP conflict error **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[5]	**LDUEN**	**LDROM Update Enable Bit (Write Protect)** 0 = LDROM cannot be updated. 1 = LDROM can be updated. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[4]	**CFGUEN**	**CONFIG Update Enable Bit (Write Protect)** 0 = CONFIG cannot be updated. 1 = CONFIG can be updated. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[3]	**APUEN**	**APROM Update Enable Bit (Write Protect)** 0 = APROM cannot be updated when the chip runs in APROM. 1 = APROM can be updated when the chip runs in APROM. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[2:1]	**Reserved**	Reserved.
[0]	**ISPEN**	**ISP Enable Bit (Read Only)** 1 = ISP function Enabled. **Note:** This bit is read only to show ISP function enable.

ISP Address (FMC_ISPADDR)

Register	Offset	R/W	Description	Reset Value
FMC_ISPADDR	FMC_BA+0x04	R/W	ISP Address Register	0x0000_0000

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

Bits	Description	
[31:0]	**ISPADDR**	**ISP Address** The M2354 series is equipped with embedded Flash. ISPADDR[1:0] must be kept 00 for ISP 32-bit operation. ISPADDR[2:0] must be kept 000 for ISP 64-bit operation. For CRC32 Checksum Calculation command, this field is the Flash starting address for checksum calculation, 2 Kbytes alignment is necessary for CRC32 checksum calculation. For Flash32-bit Program, ISP address needs word alignment (4-byte). For Flash 64-bit Program, ISP address needs double word alignment (8-byte).

ISP Data Register (FMC_ISPDAT)

Register	Offset	R/W	Description	Reset Value
FMC_ISPDAT	FMC_BA+0x08	R/W	ISP Data Register	0x0000_0000

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

Bits	Description	
[31:0]	**ISPDAT**	**ISP Data** Write data to this register before ISP program operation. Read data from this register after ISP read operation. When ISPFF (FMC_ISPCTL[6]) is 1, ISPDAT = 0xffff_ffff. For Run CRC32 Checksum Calculation command, ISPDAT is the memory size (byte) and 2 Kbytes alignment. For ISP Read CRC32 Checksum command, ISPDAT is the checksum result. If ISPDAT = 0x0000_0000, it means that (1) the checksum calculation is in progress, or (2) the memory range for checksum calculation is incorrect. For XOM page erase function, ISPDAT = 0x0055_aa03.

ISP Command Register (FMC_ISPCMD)

Register	Offset	R/W	Description	Reset Value
FMC_ISPCMD	FMC_BA+0x0C	R/W	ISP Command Register	0x0000_0000

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

Bits	Description	
[31:7]	Reserved	Reserved.
[6:0]	CMD	**ISP Command** The ISP command table is shown below: 0x00= FLASH Read. 0x04= Read Unique ID. 0x08= Read Flash All-One Result. 0x0B= Read Company ID. 0x0C= Read Device ID. 0x0D= Read Checksum. 0x21= FLASH 32-bit Program. 0x22= FLASH Page Erase. Erase any page in two banks, except for OTP. 0x23= FLASH Bank Erase. Erase all pages of APROM in BANK0 or BANK1. .0x27= FLASH Multi-Word Program. 0x28= Run Flash All-One Verification. 0x2C= Bank REMAP. 0x2D= Run Checksum Calculation. 0x2E= Vector Remap. 0x40= FLASH 64-bit Read. 0x61= FLASH 64-bit Program. The other commands are invalid.

ISP Trigger Control Register (FMC_ISPTRG)

Register	Offset	R/W	Description	Reset Value
FMC_ISPTRG	FMC_BA+0x10	R/W	ISP Trigger Control Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**ISPGO**	**ISP Start Trigger (Write Protect)** Write 1 to start ISP operation and this bit will be cleared to 0 by hardware automatically when ISP operation is finished. When ISPGO=1,the operation of writing value to address from FMC_BA+0x00 to FMC_BA+0x68 would be ignored. 0 = ISP operation is finished. 1 = ISP is progressed. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

Flash Access Time Control Register (FMC_FTCTL)

Register	Offset	R/W	Description	Reset Value
FMC_FTCTL	FMC_BA+0x18	R/W	Flash Access Time Control Register	0x0000_0000

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

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**CACHEINV**	**Flash Cache Invalidation (Write Protect)** 0 = Flash Cache Invalidation finished (default). 1 = Flash Cache Invalidation. **Note 1:** Write 1 to start cache invalidation. The value will be changed to 0 once the process is finished. **Note 2:** This bit is write-protected. Refer to the SYS_REGLCTL register.
[8:0]	**Reserved**	Reserved.

ISP Status Register (FMC_ISPSTS)

Register	Offset	R/W	Description	Reset Value
FMC_ISPSTS	FMC_BA+0x40	R/W	ISP Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	FBS	MIRBOUND	ISPCERR	Reserved			INTFLAG
23	22	21	20	19	18	17	16
VECMAP							
15	14	13	12	11	10	9	8
VECMAP							Reserved
7	6	5	4	3	2	1	0
ALLONE	ISPFF	PGFF	Reserved				ISPBUSY

Bits	Description	
[31]	**Reserved**	Reserved.
[30]	**FBS**	**Flash Bank Selection** This bit indicate which bank is selected to boot. 0 = Booting from BANK0. 1 = Booting from BANK1.
[29]	**MIRBOUND**	**Mirror Boundary** 0 = Mirror Boundary Disabled. 1 = Mirror Boundary Enabled.
[28]	**ISPCERR**	**ISP Conflict Error** This bit shows when FMC is doing ISP operation. User can not access FMC_ISP_ADDR,FMC_ISPDAT,FMC_ISPCMD,FMC_ISPTRG. It would cause ISPFF.
[27:25]	**Reserved**	Reserved.
[24]	**INTFLAG**	**ISP Interrupt Flag** 0 = ISP Not Finished. 1 = ISP done or ISPFF set. **Note:** This function needs to be enabled by FMC_ISPCTRL[24].
[23:9]	**VECMAP**	**Vector Page Mapping Address (Read Only)** All access to 0x0000_0000~0x0000_01FF is remapped to the Flash memory address {VECMAP[14:0], 9'h000} ~ {VECMAP[14:0], 9'h1FF}
[8]	**Reserved**	Reserved.
[7]	**ALLONE**	**Flash All-one Verification Flag** This bit is set by hardware if all of Flash bits are 1, and cleared if Flash bits are not all 1 after "Run Flash All-One Verification" is complete; this bit also can be cleared by writing 1 0 = Flash bits are not all 1 after "Run Flash All-One Verification" is complete. 1 = All of Flash bits are 1 after "Run Flash All-One Verification" is complete.

[6]	**ISPFF**	**ISP Fail Flag (Write Protect)** This bit is the mirror of ISPFF (FMC_ISPCTL[6]), it needs to be cleared by writing 1 to FMC_ISPCTL[6] or FMC_ISPSTS[6]. This bit is set by hardware when a triggered ISP meets any of the following conditions: • APROM writes to itself if APUEN is set to 0. • LDROM writes to itself if LDUEN is set to 0. • CONFIG is erased/programmed if CFGUEN is set to 0. • Page Erase command at LOCK mode with ICE connection • Erase or Program command at brown-out detected • Destination address is illegal, such as over an available range. • Invalid ISP commands • APROM is erased/programmed if KEYLOCK is set to 1 • LDROM is erased/programmed if KEYLOCK is set to 1 • CONFIG is erased/programmed if KEYLOCK is set to 1 and KEYENROM[0] is 0. • Read any content of boot loader with ICE connection • The address of block erase and bank erase is not in APROM • ISP CMD in XOM region, except mass erase, page erase and chksum command • The wrong setting of page erase ISP CMD in XOM • Violate XOM setting one time protection • Page erase ISP CMD in Secure/Non-secure region setting page • Mass erase when MERASE (CFG0[13]) is disabled • Page erase, mass erase, multi-word program or 64-bit word program in OTP • ISP conflict error **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[5]	**PGFF**	**Flash Program with Fast Verification Flag (Read Only)** This bit is set if data is mismatched at ISP programming verification. This bit is cleared by performing ISP Flash erase or ISP read CID operation 0 = Flash Program is success. 1 = Flash Program is failed. Program data is different with data in the Flash memory.
[4:1]	**Reserved**	Reserved.
[0]	**ISPBUSY**	**ISP Busy Flag (Read Only)** Write 1 to start ISP operation and this bit will be cleared to 0 by hardware automatically when ISP operation is finished. This bit is the mirror of ISPGO(FMC_ISPTRG[0]). 0 = ISP operation is finished. 1 = ISP is progressed.

Flash Access Cycle Control Register (FMC_CYCCTL)

Register	Offset	R/W	Description	Reset Value
FMC_CYCCTL	FMC_BA+0x4C	R/W	Flash Access Cycle Control Register	0x0000_0001

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

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3:0]	**CYCLE**	**Flash Access Cycle Control (Write Protect)** This register is updated by software.User needs to check the speed of HCLK and set the cycle >0. 0001 = CPU access with one wait cycle if cache miss; Flash access cycle is 1;. The HCLK working frequency range range is<25 MHz 0010 = CPU access with two wait cycles if cache miss; Flash access cycle is 2;. The optimized HCLK working frequency range is 26~50 MHz 0011 = CPU access with three wait cycles if cache miss; Flash access cycle is 3;. The optimized HCLK working frequency range is 51~75 MHz 0100 = CPU access with four wait cycles if cache miss; Flash access cycle is 4;. The optimized HCLK working frequency range is75~96 MHz **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

ISP Data 0 Register (FMC_MPDAT0)

Register	Offset	R/W	Description	Reset Value
FMC_MPDAT0	FMC_BA+0x80	R/W	ISP Data0 Register	0x0000_0000

31	30	29	28	27	26	25	24
ISPDAT0							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ISPDAT0							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ISPDAT0							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ISPDAT0							

Bits	Description	
[31:0]	**ISPDAT0**	**ISP Data 0** This register is the first 32-bit data for 32-bit/64-bit/multi-word programming, and it is also the mirror of FMC_ISPDAT, both registers keep the same data.

ISP Data 1 Register (FMC_MPDAT1)

Register	Offset	R/W	Description	Reset Value
FMC_MPDAT1	FMC_BA+0x84	R/W	ISP Data1 Register	0x0000_0000

31	30	29	28	27	26	25	24
ISPDAT1							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ISPDAT1							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ISPDAT1							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ISPDAT1							

Bits	Description	
[31:0]	**ISPDAT1**	**ISP Data 1** This register is the second 32-bit data for 64-bit/multi-word programming.

ISP Data 2 Register (FMC_MPDAT2)

Register	Offset	R/W	Description	Reset Value
FMC_MPDAT2	FMC_BA+0x88	R/W	ISP Data2 Register	0x0000_0000

31	30	29	28	27	26	25	24
ISPDAT2							
23	22	21	20	19	18	17	16
ISPDAT2							
15	14	13	12	11	10	9	8
ISPDAT2							
7	6	5	4	3	2	1	0
ISPDAT2							

Bits	Description	
[31:0]	**ISPDAT2**	**ISP Data 2** This register is the third 32-bit data for multi-word programming.

ISP Data 3 Register (FMC_MPDAT3)

Register	Offset	R/W	Description	Reset Value
FMC_MPDAT3	FMC_BA+0x8C	R/W	ISP Data3 Register	0x0000_0000

31	30	29	28	27	26	25	24
ISPDAT3							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ISPDAT3							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ISPDAT3							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ISPDAT3							

Bits	Description	
[31:0]	**ISPDAT3**	**ISP Data 3** This register is the fourth 32-bit data for multi-word programming.

ISP Multi-program Status Register (FMC_MPSTS)

Register	Offset	R/W	Description	Reset Value
FMC_MPSTS	FMC_BA+0xC0	R	ISP Multi-program Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
D3	D2	D1	D0	Reserved	ISPFF	PPGO	MPBUSY

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**D3**	**ISP DATA 3 Flag (Read Only)** This bit is set when FMC_MPDAT3 is written and auto-cleared to 0 when the FMC_MPDAT3 data programmed to Flash is complete. 0 = FMC_MPDAT3 register is empty, or programmed to Flash complete. 1 = FMC_MPDAT3 register has been written, and not programmed to Flash complete.
[6]	**D2**	**ISP DATA 2 Flag (Read Only)** This bit is set when FMC_MPDAT2 is written and auto-cleared to 0 when the FMC_MPDAT2 data programmed to Flash is complete. 0 = FMC_MPDAT2 register is empty, or programmed to Flash complete. 1 = FMC_MPDAT2 register has been written, and not programmed to Flash complete.
[5]	**D1**	**ISP DATA 1 Flag (Read Only)** This bit is set when FMC_MPDAT1 is written and auto-cleared to 0 when the FMC_MPDAT1 data programmed to Flash is complete. 0 = FMC_MPDAT1 register is empty, or programmed to Flash complete. 1 = FMC_MPDAT1 register has been written, and not programmed to Flash complete.
[4]	**D0**	**ISP DATA 0 Flag (Read Only)** This bit is set when FMC_MPDAT0 is written and auto-cleared to 0 when the FMC_MPDAT0 data programmed to Flash is complete. 0 = FMC_MPDAT0 register is empty, or programmed to Flash complete. 1 = FMC_MPDAT0 register has been written, and not programmed to Flash complete.
[3]	**Reserved**	Reserved.

[2]	**ISPFF**	**ISP Fail Flag (Read Only)** This bit is the mirror of ISPFF (FMC_ISPCTL[6]). It needs to be cleared by writing 1 to FMC_ISPCTL[6] or FMC_ISPSTS[6]. This bit is set by hardware when a triggered ISP meets any of the following conditions: • APROM writes to itself if APUEN is set to 0. • LDROM writes to itself if LDUEN is set to 0. • CONFIG is erased/programmed if CFGUEN is set to 0. • Page Erase command at LOCK mode with ICE connection • Erase or Program command at brown-out detected • Destination address is illegal, such as over an available range. • Invalid ISP commands. • APROM is erased/programmed if KEYLOCK is set to 1 • LDROM is erased/programmed if KEYLOCK is set to 1 • CONFIG is erased/programmed if KEYLOCK is set to 1 and KEYENROM[0] is 0. • Read any content of boot loader with ICE connection • The address of block erase and bank erase is not in APROM • ISP CMD in XOM region, except mass erase, page erase and chksum command • The wrong setting of page erase ISP CMD in XOM • Violate XOM setting one time protection • Page erase ISP CMD in Secure/Non-secure region setting page • Mass erase when MERASE (CFG0[13]) is disabled • Page erase, mass erase, multi-word program or 64-bit word program in OTP
[1]	**PPGO**	**ISP Multi-program Status (Read Only)** 0 = ISP multi-word program operation is not active. 1 = ISP multi-word program operation is in progress.
[0]	**MPBUSY**	**ISP Multi-word Program Busy Flag (Read Only)** Write 1 to start ISP Multi-Word program operation and this bit will be cleared to 0 by hardware automatically when ISP Multi-Word program operation is finished. This bit is the mirror of ISPGO(FMC_ISPTRG[0]). 0 = ISP Multi-Word program operation is finished. 1 = ISP Multi-Word program operation is progressed.

ISP Multi-word Program Address Register (FMC_MPADDR)

Register	Offset	R/W	Description	Reset Value
FMC_MPADDR	FMC_BA+0xC4	R	ISP Multi-program Address Register	0x0000_0000

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

Bits	Description	
[31:0]	**MPADDR**	**ISP Multi-word Program Address** MPADDR is the address of ISP multi-word program operation when ISPGO flag is 1. MPADDR will keep the final ISP address when ISP multi-word program is complete.

XOM Region0 Status Register (FMC_XOMR0STS)

Register	Offset	R/W	Description	Reset Value
FMC_XOMR0STS	FMC_BA+0xD0	R	XOM Region 0 Status Register	0xFFF8_00FF

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

Bits	Description	
[31:8]	**BASE**	**XOM Region 0 Base Address (Page-aligned)** BASE is the base address of XOM Region 0.
[7:0]	**SIZE**	**XOM Region 0 Size (Page-aligned)** SIZE is the page number of XOM Region 0.

XOM Region1 Status Register (FMC_XOMR1STS)

Register	Offset	R/W	Description	Reset Value
FMC_XOMR1STS	FMC_BA+0xD4	R	XOM Region 1 Status Register	0xFFF8_00FF

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

Bits	Description	
[31:8]	**BASE**	**XOM Region 1 Base Address (Page-aligned)** BASE is the base address of XOM Region 1.
[7:0]	**SIZE**	**XOM Region 1 Size (Page-aligned)** SIZE is the page number of XOM Region 1.

XOM Region2 Status Register (FMC_XOMR2STS)

Register	Offset	R/W	Description	Reset Value
FMC_XOMR2STS	FMC_BA+0xD8	R	XOM Region 2 Status Register	0xFFF8_00FF

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

Bits	Description	
[31:8]	**BASE**	**XOM Region 2 Base Address (Page-aligned)** BASE is the base address of XOM Region 2.
[7:0]	**SIZE**	**XOM Region 2 Size (Page-aligned)** SIZE is the page number of XOM Region 2.

XOM Region3 Status Register (FMC_XOMR3STS)

Register	Offset	R/W	Description	Reset Value
FMC_XOMR3STS	FMC_BA+0xDC	R	XOM Region 3 Status Register	0xFFF8_00FF

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

Bits	Description	
[31:8]	**BASE**	**XOM Region 3 Base Address (Page-aligned)** BASE is the base address of XOM Region 3.
[7:0]	**SIZE**	**XOM Region 3 Size (Page-aligned)** SIZE is the page number of XOM Region 3.

XOM Status Register (FMC_XOMSTS)

Register	Offset	R/W	Description	Reset Value
FMC_XOMSTS	FMC_BA+0xE0	R	XOM Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			XOMPEF	XOMR3ON	XOMR2ON	XOMR1ON	XOMR0ON

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**XOMPEF**	**XOM Page Erase Function Fail** XOM page erase function status. If XOMPEF is set to 1, user needs to erase XOM region again. 0 = Success. 1 = Fail.
[3]	**XOMR3ON**	**XOM Region 3 On** XOM Region 3 active status. 0 = Not active. 1 = XOM region 3 is active.
[2]	**XOMR2ON**	**XOM Region 2 On** XOM Region 2 active status. 0 = Not active. 1 = XOM region 2 is active.
[1]	**XOMR1ON**	**XOM Region 1 On** XOM Region 1 active status. 0 = Not active. 1 = XOM region 1 is active.
[0]	**XOMR0ON**	**XOM Region 0 On** XOM Region 0 active status. 0 = Not active. 1 = XOM region 0 is active.

Data Flash Function Control (FMC_DFCTL)

Register	Offset	R/W	Description	Reset Value
FMC_DFCTL	FMC_BA+0x100	R/W	Data Flash Function Control	0x0000_0000

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

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**SILENTEN**	**Silent Access Enable Bit** User can set this bit to enable Silent access protection on Data Flash. Note that the Data Flash is formed as 4 pages of 1 Kbytes when silent access protection is enabled. 0 = Silent access Disabled. 1 = Silent access Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[0]	**SCRAMEN**	**Data Scrambling Enable Bit** User can set this bit to enable Data scrambling protection on Data Flash. 0 = Data scrambling Disabled. 1 = Data scrambling Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

Data Flash Status (FMC_DFSTS)

Register	Offset	R/W	Description	Reset Value
FMC_DFSTS	FMC_BA+0x108	R/W	Data Flash Status	0x0000_0000

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

Bits	Description	
[31:2]	Reserved	Reserved.
[1]	TMPCLRBUSY	**Data Flash Temper Attack Programming Busy Status (Read Only)** 0 = Data Flash temper attack programming is not busy. 1 = Data Flash temper attack programming is busy.
[0]	TMPCLRDONE	**Data Flash Temper Attack Programming Done** 0 = Data Flash temper attack programming is not finished. 1 = Data Flash temper attack programming is done, and user can write 1 to clear this bit.

Data Flash Scramble Key (FMC_SCRKEY)

Register	Offset	R/W	Description	Reset Value
FMC_SCRKEY	FMC_BA+0x10C	W	Data Flash Scrambling Key	0x0000_0000

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

Bits	Description	
[31:0]	SCRKEY	**Data Flash Scrambling Key (Write Only)** 32-bit user defined data scrambling key for Data Flash. When data scrambling is enabled (FMC_DFCTL[0]), data in Data FLASH is scrambled when written and de-scrambled when read. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

NS_ ISP Control Register (Non-secure FMC_ISPCTL)

Register	Offset	R/W	Description	Reset Value
NS_FMC_ISPCTL	FMC_NSBA+0x00	R/W	Non-secure ISP Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							NS_INTEN
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved	NS_ISPFF	Reserved		APUEN	Reserved		ISPEN

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24]	**NS_INTEN**	**Non-secure ISP Interrupt Enable Bit** 0 = NS_ISP INT Disabled. 1 = NS_ISP INT Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register. Before using INT, user needs to clear the INTFLAG(FMC_ISPSTS[24]) make sure INT happen at correct time.
[23:7]	**Reserved**	Reserved.
[6]	**NS_ISPFF**	**Non-sec ISP Fail Flag (Write Protect)** This bit is set by hardware when a triggered ISP meets any of the following conditions: This bit needs to be cleared by writing 1 to it. • APROM writes to itself if APUEN is set to 0. • Page Erase command at LOCK mode with ICE connection • Erase or Program command at brown-out detected • Destination address is illegal, such as over an available range. • Invalid ISP commands • Read any content of boot loader with ICE connection • ISP CMD in XOM region, except xom page erase and chksum command • The wrong setting of page erase ISP CMD in XOM • Violate XOM setting one time protection • Page erase ISP CMD in Secure region setting page • Page erase, mass erase, multi-word program or 64-bit word program in OTP • NS_ISP Conflict Error **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[5:4]	**Reserved**	Reserved.

[3]	**APUEN**	**APROM Update Enable Bit (Write Protect)** 0 = APROM cannot be updated when the chip runs in APROM. 1 = APROM can be updated when the chip runs in APROM. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[2:1]	**Reserved**	Reserved.
[0]	**ISPEN**	**ISP Enable Bit (Read Only)** ISP function enable. 1 = ISP function Enabled. **Note:** This bit is read only to show ISP function enable.

NS_ISP Address (Non-secure FMC_ISPADDR)

Register	Offset	R/W	Description	Reset Value
NS_FMC_ISPADDR	FMC_NSBA+0x04	R/W	Non-secure ISP Address Register	0x0000_0000

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

Bits	Description	
[31:0]	NS_ISPADDR	**Non-sec ISP Address** The M2354 series is equipped with embedded Flash. ISPADDR[1:0] must be kept 00 for ISP 32-bit operation. For CRC32 Checksum Calculation command, this field is the Flash starting address for checksum calculation, 2 Kbytes alignment is necessary for CRC32 checksum calculation. For Flash32-bit Program, ISP address needs word alignment (4-byte). For Flash 64-bit Program, ISP address needs double word alignment (8-byte). Non-sec ISP address must be active at Non-sec region.

NS_ISP Data Register (Non-secure FMC_ISPDAT)

Register	Offset	R/W	Description	Reset Value
NS_FMC_ISPDAT	FMC_NSBA+0x08	R/W	Non-secure ISP Data Register	0x0000_0000

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

Bits	Description	
[31:0]	NS_ISPDAT	**Non-sec ISP Data** Write data to this register before ISP program operation. Read data from this register after ISP read operation. When ISPFF (FMC_ISPCTL[6]) is 1, ISPDAT = 0xffff_ffff. For Run CRC32 Checksum Calculation command, ISPDAT is the memory size (byte) and 2 Kbytes alignment. For ISP Read CRC32 Checksum command, ISPDAT is the checksum result. If ISPDAT = 0x0000_0000, it means that (1) the checksum calculation is in progress, or (2) the memory range for checksum calculation is incorrect. For XOM page erase function, SPDAT = 0x0055_aa03.

NS_ISP Command Register (Non-secure FMC_ISPCMD)

Register	Offset	R/W	Description	Reset Value
NS_FMC_ISPCMD	FMC_NSBA+0x0C	R/W	Non-secure ISP Command Register	0x0000_0000

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

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6:0]	**NS_CMD**	**Non-sec ISP Command** ISP command table is shown below: 0x00= FLASH Read. 0x04= Read Unique ID. 0x08= Read Flash All-One Result. 0x0B= Read Company ID. 0x0C= Read Device ID. 0x0D= Read Checksum. 0x21= FLASH 32-bit Program. 0x22= FLASH Page Erase. Erase only non-secure page in two banks. 0x28= Run Flash All-One Verification. 0x2D= Run Checksum Calculation. The other commands are invalid.

NS_ISP Trigger Control Register (Non-secure FMC_ISPTRG)

Register	Offset	R/W	Description	Reset Value
NS_FMC_ISPTRG	FMC_NSBA+0x10	R/W	Non-secure ISP Trigger Control Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**NS_ISPGO**	**Non-sec ISP Start Trigger (Write Protect)** Write 1 to start ISP operation and this bit will be cleared to 0 by hardware automatically when ISP operation is finished. When ISPGO=1,the operation of writing value to address from FMC_BA+0x00 to FMC_BA+0x68 would be ignored. 0 = ISP operation is finished. 1 = ISP is progressed. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

NS_ISP Status Register (FMC_ISPSTS)

Register	Offset	R/W	Description	Reset Value
NS_FMC_ISPSTS	FMC_NSBA+0x40	R/W	Non-secure ISP Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			NSISPCER	Reserved			NSINTFLAG
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
ALLONE	NS_ISPFF	Reserved					NS_ISPBUSY

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28]	**NSISPCER**	**Non-secure ISP Conflict Error** This bit shows when FMC is doing ISP operation. User can not access NS_FMC_ISP_ADDR,NS_FMC_ISPDAT,NS_FMC_ISPCMD,NS_FMC_ISPTRG. It would cause ISPFF.
[27:25]	**Reserved**	Reserved.
[24]	**NSINTFLAG**	**Non-sec ISP Interrupt Flag** 0 = ISP Not Finished. 1 = ISP done or ISPFF set. **Note:** This function needs to be enabled by Non-secure FMC_ISPCTRL[24].
[23:8]	**Reserved**	Reserved.
[7]	**ALLONE**	**Flash All-one Verification Flag** This bit is set by hardware if all of Flash bits are 1, and cleared if Flash bits are not all 1 after “Run Flash All-One Verification” is complete; this bit also can be cleared by writing 1 0 = Flash bits are not all 1 after “Run Flash All-One Verification” is complete. 1 = All of Flash bits are 1 after “Run Flash All-One Verification” is complete.

[6]	**NS_ISPFF**	**Non-sec ISP Fail Flag (Write Protect)** This bit is set by hardware when a triggered ISP meets any of the following conditions: This bit needs to be cleared by writing 1 to it. • APROM writes to itself if APUEN is set to 0. • Page Erase command at LOCK mode with ICE connection • Erase or Program command at brown-out detected • Destination address is illegal, such as over an available range. • Invalid ISP commands • Read any content of boot loader with ICE connection • ISP CMD in XOM region, except XOM page erase and chksum command • The wrong setting of page erase ISP CMD in XOM • Violate XOM setting one time protection • Page erase ISP CMD in Secure region setting page • Page erase, mass erase, multi-word program or 64-bit word program in OTP **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[5:1]	**Reserved**	Reserved.
[0]	**NS_ISPBUSY**	**Non-sec ISP Busy Flag (Read Only)** Write 1 to start ISP operation and this bit will be cleared to 0 by hardware automatically when ISP operation is finished. This bit is the mirror of ISPGO(FMC_ISPTRG[0]). 0 = ISP operation is finished. 1 = ISP is progressed.

NS_XOM Region0 Status Register (Non-secure FMC_XOMR0STS)

Register	Offset	R/W	Description	Reset Value
NS_FMC_XOMR0STS	FMC_NSBA+0xD0	R	Non-secure XOM Region 0 Status Register	0xFFF8_00FF

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

Bits	Description	
[31:8]	BASE	**XOM Region 0 Base Address (Page-aligned)** BASE is the base address of XOM Region 0.
[7:0]	SIZE	**XOM Region 0 Size (Page-aligned)** SIZE is the page number of XOM Region 0.

NS_XOM Region1 Status Register (Non-secure FMC_XOMR1STS)

Register	Offset	R/W	Description	Reset Value
NS_FMC_XOMR1STS	FMC_NSBA+0xD4	R	Non-secure XOM Region 1 Status Register	0xFFF8_00FF

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

Bits	Description	
[31:8]	BASE	**XOM Region 1 Base Address (Page-aligned)** BASE is the base address of XOM Region 1.
[7:0]	SIZE	**XOM Region 1 Size (Page-aligned)** SIZE is the page number of XOM Region 1.

NS_XOM Region2 Status Register (Non-secure FMC_XOMR2STS)

Register	Offset	R/W	Description	Reset Value
NS_FMC_XOMR2STS	FMC_NSBA+0xD8	R	Non-secure XOM Region 2 Status Register	0xFFF8_00FF

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

Bits	Description	
[31:8]	BASE	**XOM Region 2 Base Address (Page-aligned)** BASE is the base address of XOM Region 2.
[7:0]	SIZE	**XOM Region 2 Size (Page-aligned)** SIZE is the page number of XOM Region 2.

NS_XOM Region3 Status Register (Non-secure FMC_XOMR3STS)

Register	Offset	R/W	Description	Reset Value
NS_FMC_XOMR3STS	FMC_NSBA+0xDC	R	Non-secure XOM Region 3 Status Register	0xFFF8_00FF

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

Bits	Description	
[31:8]	BASE	**XOM Region 3 Base Address (Page-aligned)** BASE is the base address of XOM Region 3.
[7:0]	SIZE	**XOM Region 3 Size (Page-aligned)** SIZE is the page number of XOM Region 3.

NS_XOM Status Register (FMC_XOMSTS)

Register	Offset	R/W	Description	Reset Value
NS_FMC_XOMSTS	FMC_NSBA+0xE0	R	Non-secure XOM Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			XOMPEF	XOMR3ON	XOMR2ON	XOMR1ON	XOMR0ON

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**XOMPEF**	**XOM Page Erase Function Fail** XOM page erase function status. If XOMPEF is set to 1, user needs to erase XOM region again. 0 = Success. 1 = Fail.
[3]	**XOMR3ON**	**XOM Region 3 On** XOM Region 3 active status. 0 = Not active. 1 = XOM region 3 is active.
[2]	**XOMR2ON**	**XOM Region 2 On** XOM Region 2 active status. 0 = Not active. 1 = XOM region 2 is active.
[1]	**XOMR1ON**	**XOM Region 1 On** XOM Region 1 active status. 0 = Not active. 1 = XOM region 1 is active.
[0]	**XOMR0ON**	**XOM Region 0 On** XOM Region 0 active status. 0 = Not active. 1 = XOM region 0 is active.

6.7 General Purpose I/O (GPIO)

6.7.1 Overview

This chip has up to 106 General Purpose I/O pins to be shared with other function pins depending on the chip configuration. These 106 pins are arranged in 8 ports named as PA, PB, PC, PD, PE, PF, PG and PH. PA, PB and PE has 16 pins on port. PC and PD has 14 pins on port. PF has 12 pins on port. PG has 10 pins on port. PH has 8 pins on port. Each of the 107 pins is independent and has the corresponding register bits to control the pin mode function and data.

The I/O type of each of I/O pins can be configured by software individually as Input, Push-pull output, Open-drain output or Quasi-bidirectional mode. After the chip is reset, the I/O mode of all pins are depending on CIOINI (CONFIG0[10]). Please refer to the M2354 Datasheet for detailed pin operation voltage information about V_{DD}, V_{DDIO} and V_{BAT} electrical characteristics. PA10, PA11, PA13~15, PB0~15, PF2, PF3 are not support 5V tolerance.

6.7.2 Features

- Four I/O modes:
  - Quasi-bidirectional mode
  - Push-Pull Output mode
  - Open-Drain Output mode
  - Input only with high impendence mode
- TTL/Schmitt trigger input selectable
- I/O pin can be configured as interrupt source with edge/level setting
- Supports High Drive and High Slew Rate I/O mode
- Configurable default I/O mode of all pins after reset by CIOINI (CONFIG0[10]) setting
  - CIOINI = 0, all GPIO pins in Quasi-bidirectional mode after chip reset
  - CIOINI = 1, all GPIO pins in input mode after chip reset
- I/O pin internal pull-up resistor enabled only in Quasi-bidirectional I/O mode
- Enabling the pin interrupt function will also enable the wake-up function
- Improve access efficiency by using single cycle I/O bus

6.7.3 Block Diagram

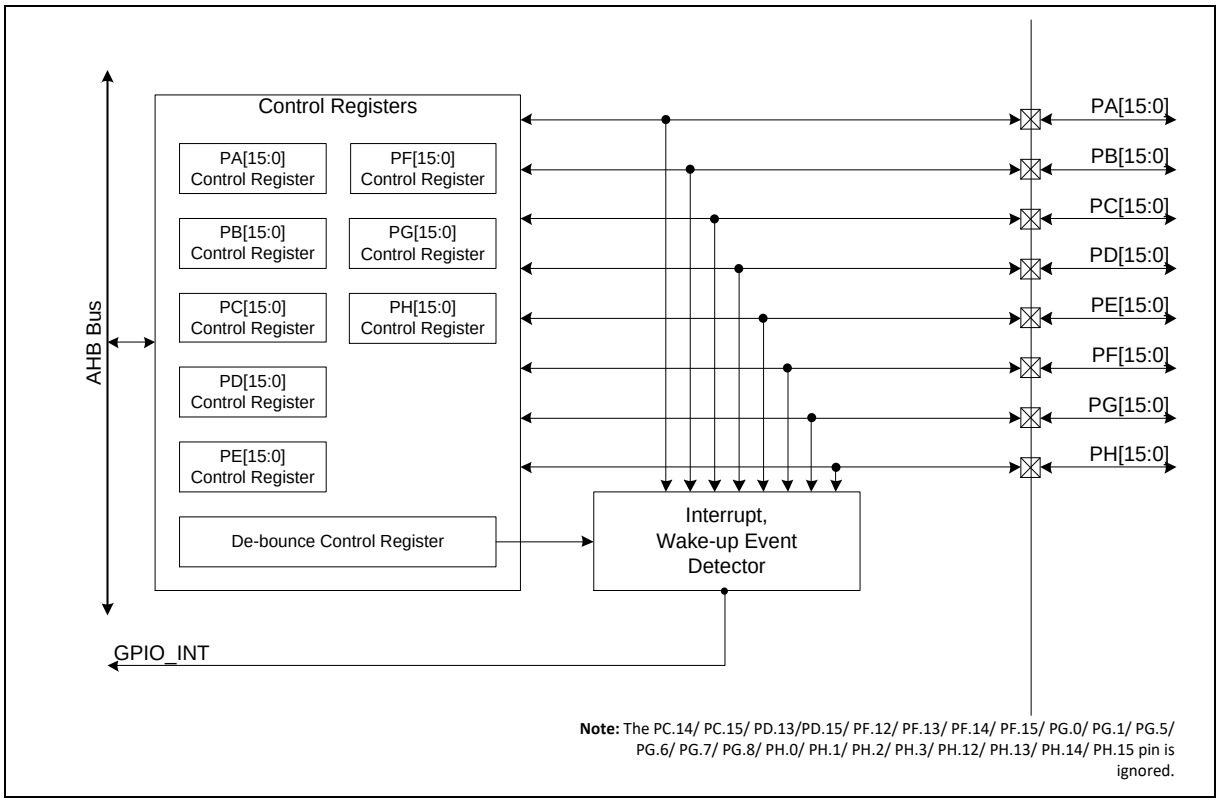


Figure 6.7-1 GPIO Controller Block Diagram

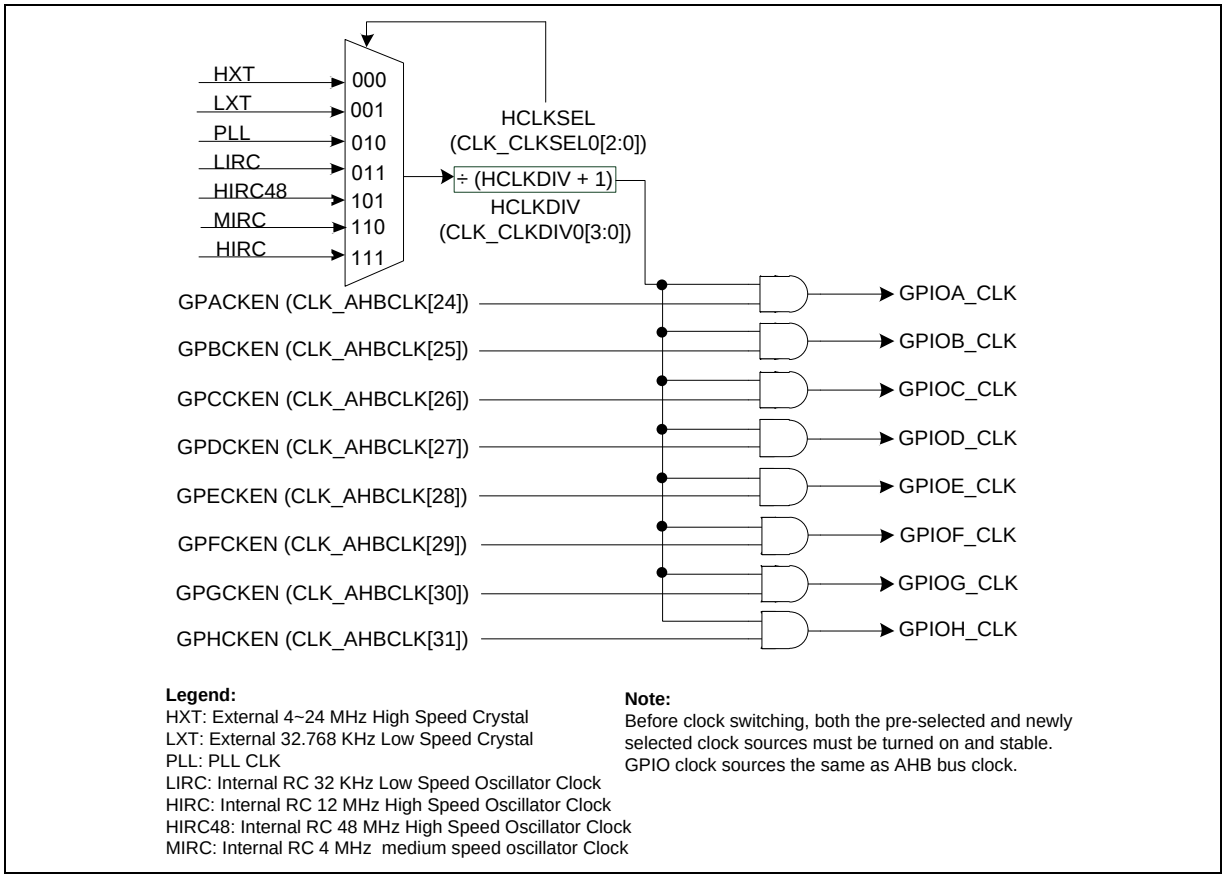


Figure 6.7-2 GPIO Clock Control Diagram

6.7.4 Basic Configuration

- Clock Source Selection
  - Enable GPIO peripheral clock in CLK_AHBCLK[31:24].
- Reset configuration
  - Reset GPIO in GPIORST (SYS_IPRST1[1])
- Pin configuration

Group	Pin Name	GPIO	MFP
INT0	INT0	PA.6, PB.5	MFP15
INT1	INT1	PA.7, PB.4	MFP15
INT2	INT2	PB.3, PC.6	MFP15
INT3	INT3	PB.2, PC.7	MFP15
INT4	INT4	PB.6	MFP13
		PA.8	MFP15
INT5	INT5	PB.7	MFP13

		PD.12	MFP15
INT6	INT6	PB.8	MFP13
		PD.11	MFP15
INT7	INT7	PB.9	MFP13
		PD.10	MFP15

6.7.5 Functional Description

6.7.5.1 Input Mode

Set MODEn (Px_MODE[2n+1:2n]) to 00 as the Px.n pin is in Input mode and the I/O pin is in tri-state (high impedance) without output drive capability. The PIN (Px_PIN[n]) value reflects the status of the corresponding port pins.

Each I/O pin includes an internal resistor. Set (Px_PUSEL[2n+1:2n]) to 01 to enable internal pull-up resistor and (Px_PUSEL[2n+1:2n]) to 10 to enable internal pull-down resistor. Setting both pull-up/down is not available.

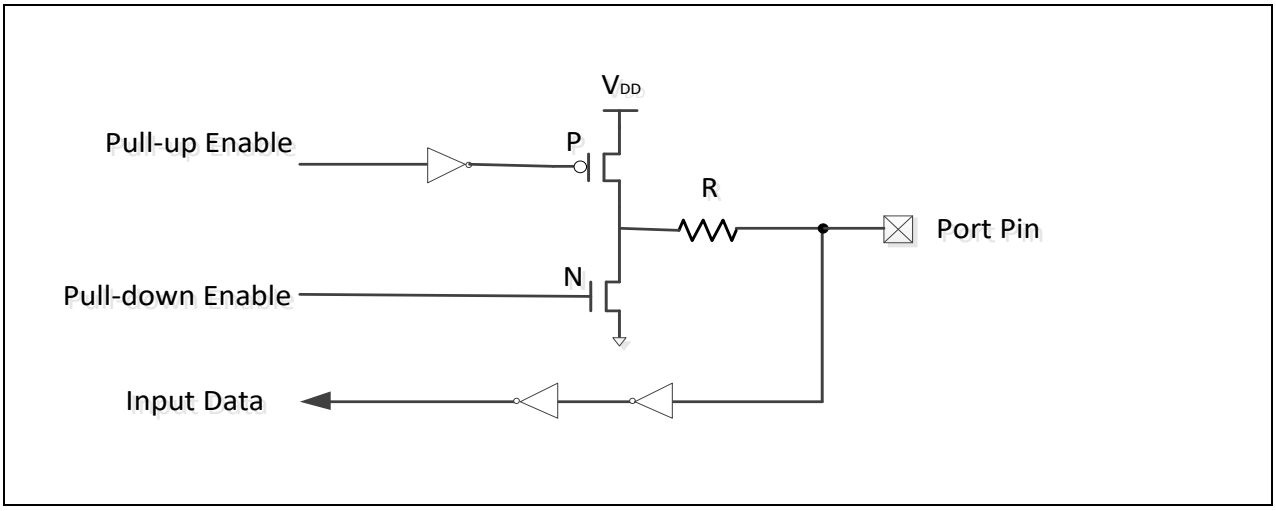


Figure 6.7-3 Input Mode

6.7.5.2 Push-pull Output Mode

Figure 6.7- shows the diagram of Push-pull Output Mode. Set MODEn (Px_MODE[2n+1:2n]) to 01 as Px.n pin is in Push-pull Output mode and the I/O pin supports digital output function with source/sink current capability. The bit value in the corresponding DOUT (Px_DOUT[n]) is driven on the pin.

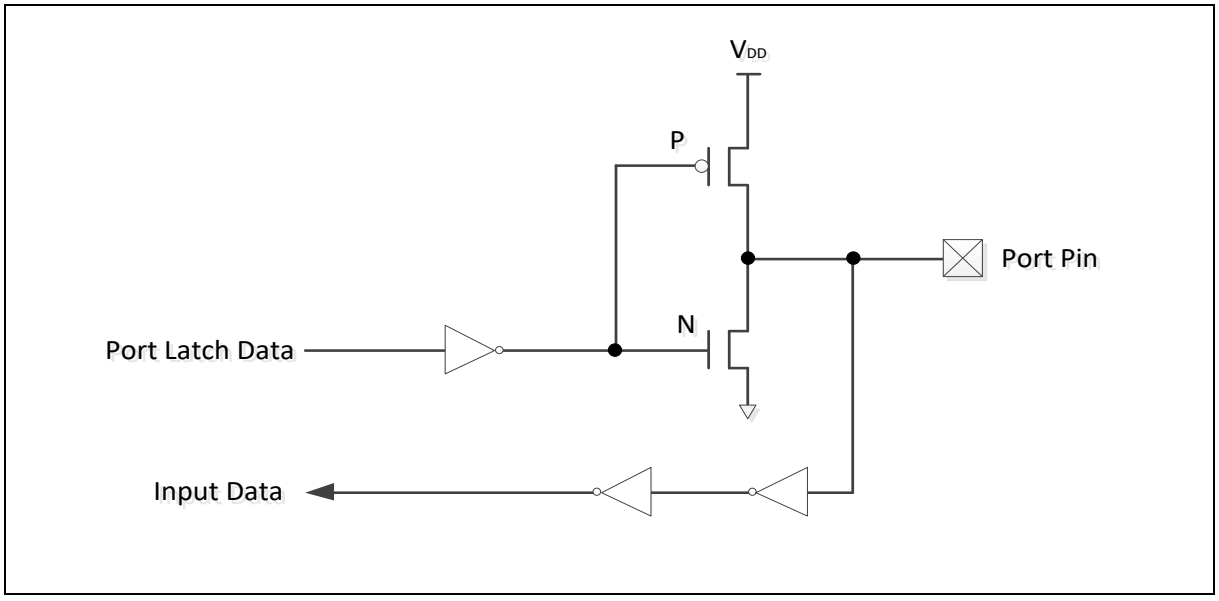


Figure 6.7-4 Push-Pull Output

6.7.5.3 Open-drain Mode

Figure 6.7- shows the diagram of Open-drain Mode. Set MODEn (Px_MODE[2n+1:2n]) to 10 the Px.n pin is in Open-drain mode and the digital output function of I/O pin supports only sink current capability, an external pull-up register is needed for driving high state. If the bit value in the corresponding DOUT (Px_DOUT[n]) bit is 0, the pin drive a low output on the pin. If the bit value in the corresponding DOUT (Px_DOUT[n]) bit is 1, the pin output drives high that is controlled by external pull high resistor.

Each I/O pin includes an internal resistor. Set (Px_PUSEL[2n+1:2n]) to 01 to enable internal pull-up resistor. Only pull-up is available in Open-drain mode.

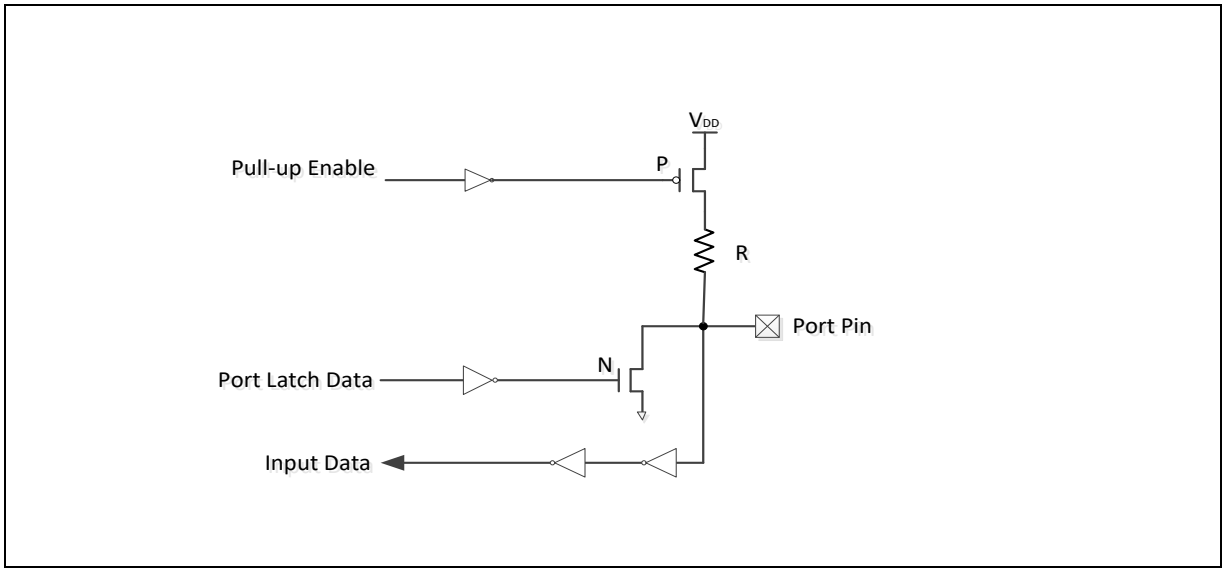


Figure 6.7-5 Open-Drain Output

6.7.5.4 Quasi-bidirectional Mode

Figure 6.7- shows the diagram of Quasi-bidirectional Mode. Set MODEn (Px_MODE[2n+1:2n]) to 11

as the Px.n pin is in Quasi-bidirectional mode and the I/O pin supports digital output and input function at the same time but the source current is only up to hundreds uA. Before the digital input function is performed the corresponding DOUT (Px_DOUT[n]) bit must be set to 1. The quasi-bidirectional output is common on the 80C51 and most of its derivatives. If the bit value in the corresponding DOUT (Px_DOUT[n]) bit is 0, the pin drive a low output on the pin. If the bit value in the corresponding DOUT (Px_DOUT[n]) bit is 1, the pin will check the pin value. If pin value is high, no action takes. If pin state is low, the pin will drive strong high with 2 clock cycles on the pin and then disable the strong output drive. Meanwhile, the pin status is controlled by internal pull-up resistor. Note that the source current capability in quasi-bidirectional mode, please refer to the M2354 Datasheet for detailed information.

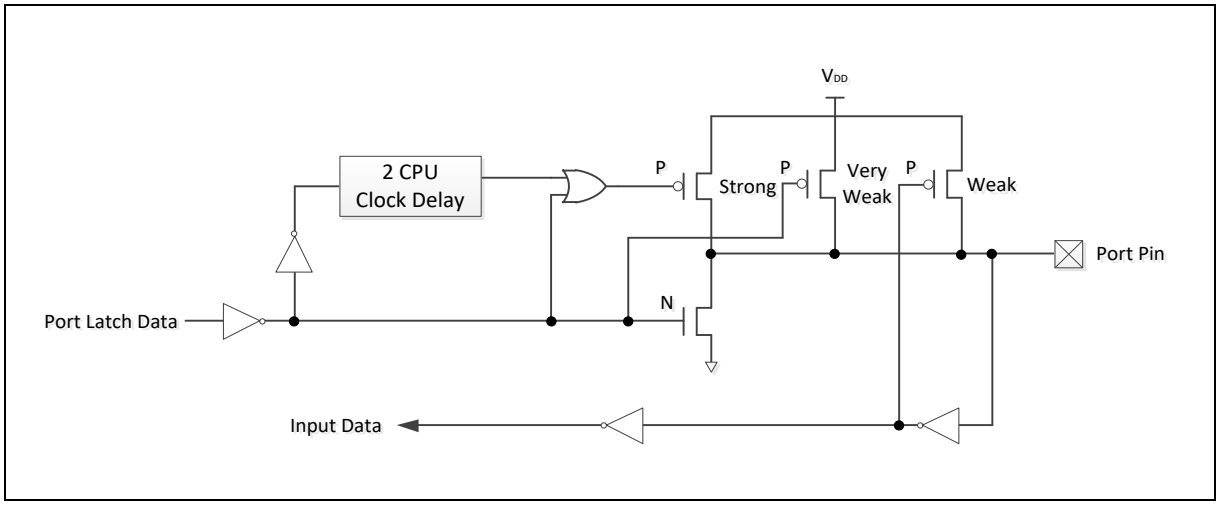


Figure 6.7-6 Quasi-Bidirectional I/O Mode

6.7.5.5 GPIO Interrupt and Wake-up Function

Each GPIO pin can be set as chip interrupt source by setting correlative RHIEN (Px_INTEN[n+16])/ FLIEN (Px_INTEN[n]) bit and TYPE (Px_INTTYPE[n]). There are five types of interrupt condition can be selected: low level trigger, high level trigger, falling edge trigger, rising edge trigger and both rising and falling edge trigger. The GPIO can also be the chip wake-up source when chip enters Idle/Power-down mode. Level trigger source needs to keep its state until entering interrupt handler. The setting of wake-up trigger condition is the same as GPIO interrupt trigger.

6.7.5.6 GPIO De-bounce Function

GPIO de-bounce function can be used to sample interrupt input for each GPIO pin and prevent unexpected interrupt happened which caused by noise. GPIO de-bounce function only support edge detection trigger type For edge trigger condition, there are three types of interrupt conditions that can be selected for de-bounce function: falling edge trigger, rising edge trigger and both rising and falling edge trigger If user wants to use de-bounce function, de-bounce enable control register Px_DBEN must be set for corresponding GPIO pin. The de-bounce clock source can be HCLK or LIRC (32 kHz) by setting DBCLKSRC (Px_DBCTL[4]) register. And DBCLKSEL (Px_DBCTL[3:0]) register can control sampling cycle period.

Figure 6.7- shows GPIO rising edge trigger interrupt. The interval of time between the two valid sample signal is determined by DBCLKSRC (Px_DBCTL[4]) and DBCLKSEL (Px_DBCTL[3:0]). Each valid data from GPIO pin need to be sample twice. For rising edge setting, if pin status is low before setting DBEN (Px_DBEN), interrupt will happen when generating a pin high valid data. But, if pin status is high before setting DBEN (Px_DBEN), interrupt will happen when generating a pin low valid data first, and then generating a pin high valid data. For falling edge trigger, Figure 6.7- shows the situation is opposite to rising edge trigger.

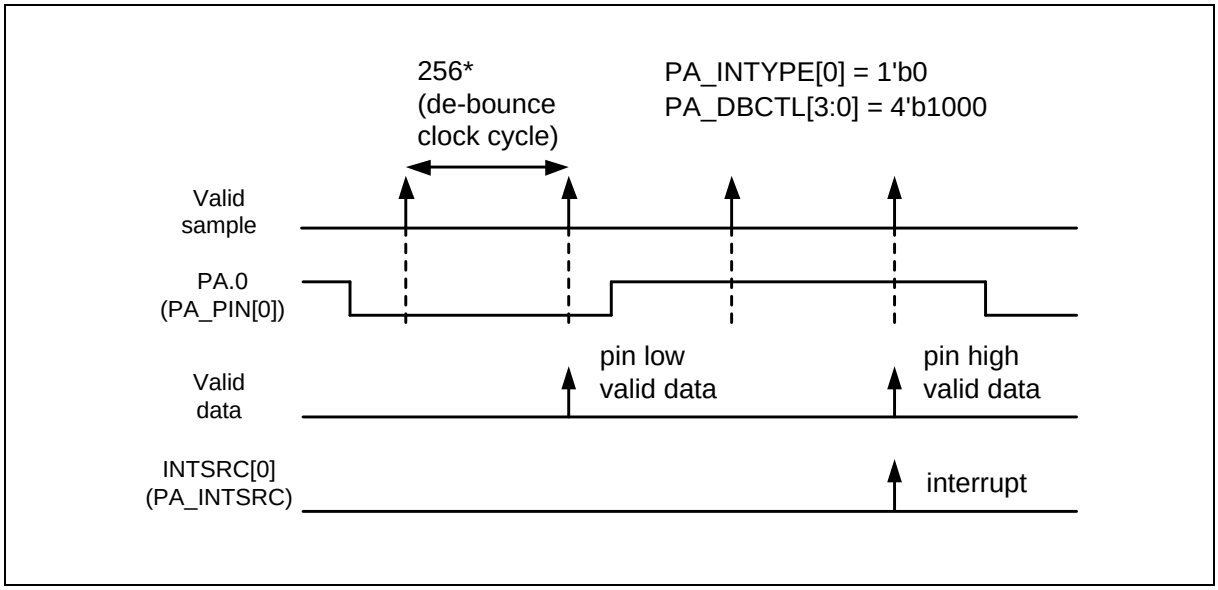


Figure 6.7-7 GPIO Rising Edge Trigger Interrupt

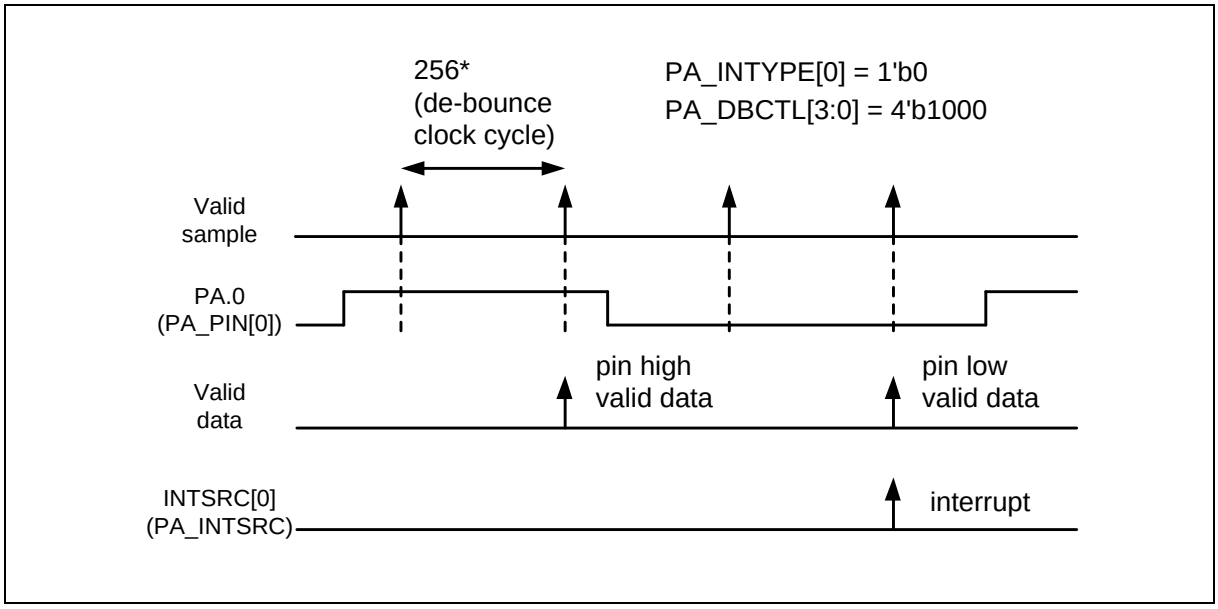


Figure 6.7-8 GPIO Falling Edge Trigger Interrupt

6.7.5.7 GPIO Digital Input Path Disable Control

User can disable GPIO digital input path by setting DINOFF (Px_DINOFF[n+16]). When GPIO digital input path is disabled, the digital input pin value PIN (Px_PIN[n]) is tied to low. By the way, the GPIO digital input path is force disabled by hardware and DINOFF control is useless when I/O function configure as ADC/ACMP/ext. XTL.

6.7.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
GPIO Base Address: **GPIO_BA = 0x4000_4000** **GPIO non-secure base address is GPIO_BA + 0x1000_0000**				
PA_MODE	GPIO_BA+0x000	R/W	PA I/O Mode Control	0xXXXX_XXXX
PA_DINOFF	GPIO_BA+0x004	R/W	PA Digital Input Path Disable Control	0x0000_0000
PA_DOUT	GPIO_BA+0x008	R/W	PA Data Output Value	0x0000_FFFF
PA_DATMSK	GPIO_BA+0x00C	R/W	PA Data Output Write Mask	0x0000_0000
PA_PIN	GPIO_BA+0x010	R	PA Pin Value	0x0000_XXXX
PA_DBEN	GPIO_BA+0x014	R/W	PA De-bounce Enable Control Register	0x0000_0000
PA_INTTYPE	GPIO_BA+0x018	R/W	PA Interrupt Trigger Type Control	0x0000_0000
PA_INTEN	GPIO_BA+0x01C	R/W	PA Interrupt Enable Control Register	0x0000_0000
PA_INTSRC	GPIO_BA+0x020	R/W	PA Interrupt Source Flag	0x0000_XXXX
PA_SMTEN	GPIO_BA+0x024	R/W	PA Input Schmitt Trigger Enable Register	0x0000_0000
PA_SLEWCTL	GPIO_BA+0x028	R/W	PA High Slew Rate Control Register	0x0000_0000
PA_PUSEL	GPIO_BA+0x030	R/W	PA Pull-up and Pull-down Selection Register	0x0000_0000
PA_DBCTL	GPIO_BA+0x034	R/W	PA Interrupt De-bounce Control Register	0x0000_0020
PB_MODE	GPIO_BA+0x040	R/W	PB I/O Mode Control	0xXXXX_XXXX
PB_DINOFF	GPIO_BA+0x044	R/W	PB Digital Input Path Disable Control	0x0000_0000
PB_DOUT	GPIO_BA+0x048	R/W	PB Data Output Value	0x0000_FFFF
PB_DATMSK	GPIO_BA+0x04C	R/W	PB Data Output Write Mask	0x0000_0000
PB_PIN	GPIO_BA+0x050	R	PB Pin Value	0x0000_XXXX
PB_DBEN	GPIO_BA+0x054	R/W	PB De-bounce Enable Control Register	0x0000_0000
PB_INTTYPE	GPIO_BA+0x058	R/W	PB Interrupt Trigger Type Control	0x0000_0000
PB_INTEN	GPIO_BA+0x05C	R/W	PB Interrupt Enable Control Register	0x0000_0000
PB_INTSRC	GPIO_BA+0x060	R/W	PB Interrupt Source Flag	0x0000_XXXX
PB_SMTEN	GPIO_BA+0x064	R/W	PB Input Schmitt Trigger Enable Register	0x0000_0000
PB_SLEWCTL	GPIO_BA+0x068	R/W	PB High Slew Rate Control Register	0x0000_0000
PB_PUSEL	GPIO_BA+0x070	R/W	PB Pull-up and Pull-down Selection Register	0x0000_0000

PB_DBCTL	GPIO_BA+0x074	R/W	PB Interrupt De-bounce Control Register	0x0000_0020
PC_MODE	GPIO_BA+0x080	R/W	PC I/O Mode Control	0xXXXX_XXXX
PC_DINOFF	GPIO_BA+0x084	R/W	PC Digital Input Path Disable Control	0x0000_0000
PC_DOUT	GPIO_BA+0x088	R/W	PC Data Output Value	0x0000_3FFF
PC_DATMSK	GPIO_BA+0x08C	R/W	PC Data Output Write Mask	0x0000_0000
PC_PIN	GPIO_BA+0x090	R	PC Pin Value	0x0000_XXXX
PC_DBEN	GPIO_BA+0x094	R/W	PC De-bounce Enable Control Register	0x0000_0000
PC_INTTYPE	GPIO_BA+0x098	R/W	PC Interrupt Trigger Type Control	0x0000_0000
PC_INTEN	GPIO_BA+0x09C	R/W	PC Interrupt Enable Control Register	0x0000_0000
PC_INTSRC	GPIO_BA+0x0A0	R/W	PC Interrupt Source Flag	0x0000_XXXX
PC_SMTEN	GPIO_BA+0x0A4	R/W	PC Input Schmitt Trigger Enable Register	0x0000_0000
PC_SLEWCTL	GPIO_BA+0x0A8	R/W	PC High Slew Rate Control Register	0x0000_0000
PC_PUSEL	GPIO_BA+0x0B0	R/W	PC Pull-up and Pull-down Selection Register	0x0000_0000
PC_DBCTL	GPIO_BA+0x0B4	R/W	PC Interrupt De-bounce Control Register	0x0000_0020
PD_MODE	GPIO_BA+0x0C0	R/W	PD I/O Mode Control	0xXXXX_XXXX
PD_DINOFF	GPIO_BA+0x0C4	R/W	PD Digital Input Path Disable Control	0x0000_0000
PD_DOUT	GPIO_BA+0x0C8	R/W	PD Data Output Value	0x0000_7FFF
PD_DATMSK	GPIO_BA+0x0CC	R/W	PD Data Output Write Mask	0x0000_0000
PD_PIN	GPIO_BA+0x0D0	R	PD Pin Value	0x0000_XXXX
PD_DBEN	GPIO_BA+0x0D4	R/W	PD De-bounce Enable Control Register	0x0000_0000
PD_INTTYPE	GPIO_BA+0x0D8	R/W	PD Interrupt Trigger Type Control	0x0000_0000
PD_INTEN	GPIO_BA+0x0DC	R/W	PD Interrupt Enable Control Register	0x0000_0000
PD_INTSRC	GPIO_BA+0x0E0	R/W	PD Interrupt Source Flag	0x0000_XXXX
PD_SMTEN	GPIO_BA+0x0E4	R/W	PD Input Schmitt Trigger Enable Register	0x0000_0000
PD_SLEWCTL	GPIO_BA+0x0E8	R/W	PD High Slew Rate Control Register	0x0000_0000
PD_PUSEL	GPIO_BA+0x0F0	R/W	PD Pull-up and Pull-down Selection Register	0x0000_0000
PD_DBCTL	GPIO_BA+0x0F4	R/W	PD Interrupt De-bounce Control Register	0x0000_0020
PE_MODE	GPIO_BA+0x100	R/W	PE I/O Mode Control	0xXXXX_XXXX
PE_DINOFF	GPIO_BA+0x104	R/W	PE Digital Input Path Disable Control	0x0000_0000
PE_DOUT	GPIO_BA+0x108	R/W	PE Data Output Value	0x0000_FFFF

PE_DATMSK	GPIO_BA+0x10C	R/W	PE Data Output Write Mask	0x0000_0000
PE_PIN	GPIO_BA+0x110	R	PE Pin Value	0x0000_XXXX
PE_DBEN	GPIO_BA+0x114	R/W	PE De-bounce Enable Control Register	0x0000_0000
PE_INTTYPE	GPIO_BA+0x118	R/W	PE Interrupt Trigger Type Control	0x0000_0000
PE_INTEN	GPIO_BA+0x11C	R/W	PE Interrupt Enable Control Register	0x0000_0000
PE_INTSRC	GPIO_BA+0x120	R/W	PE Interrupt Source Flag	0x0000_XXXX
PE_SMTEN	GPIO_BA+0x124	R/W	PE Input Schmitt Trigger Enable Register	0x0000_0000
PE_SLEWCTL	GPIO_BA+0x128	R/W	PE High Slew Rate Control Register	0x0000_0000
PE_PUSEL	GPIO_BA+0x130	R/W	PE Pull-up and Pull-down Selection Register	0x0000_0000
PE_DBCTL	GPIO_BA+0x134	R/W	PE Interrupt De-bounce Control Register	0x0000_0020
PF_MODE	GPIO_BA+0x140	R/W	PF I/O Mode Control	0xXXXX_XXXX
PF_DINOFF	GPIO_BA+0x144	R/W	PF Digital Input Path Disable Control	0x0000_0000
PF_DOUT	GPIO_BA+0x148	R/W	PF Data Output Value	0x0000_0FFF
PF_DATMSK	GPIO_BA+0x14C	R/W	PF Data Output Write Mask	0x0000_0000
PF_PIN	GPIO_BA+0x150	R	PF Pin Value	0x0000_XXXX
PF_DBEN	GPIO_BA+0x154	R/W	PF De-bounce Enable Control Register	0x0000_0000
PF_INTTYPE	GPIO_BA+0x158	R/W	PF Interrupt Trigger Type Control	0x0000_0000
PF_INTEN	GPIO_BA+0x15C	R/W	PF Interrupt Enable Control Register	0x0000_0000
PF_INTSRC	GPIO_BA+0x160	R/W	PF Interrupt Source Flag	0x0000_XXXX
PF_SMTEN	GPIO_BA+0x164	R/W	PF Input Schmitt Trigger Enable Register	0x0000_0000
PF_SLEWCTL	GPIO_BA+0x168	R/W	PF High Slew Rate Control Register	0x0000_0000
PF_PUSEL	GPIO_BA+0x170	R/W	PF Pull-up and Pull-down Selection Register	0x0000_0000
PF_DBCTL	GPIO_BA+0x174	R/W	PF Interrupt De-bounce Control Register	0x0000_0020
PG_MODE	GPIO_BA+0x180	R/W	PG I/O Mode Control	0xXXXX_XXXX
PG_DINOFF	GPIO_BA+0x184	R/W	PG Digital Input Path Disable Control	0x0000_0000
PG_DOUT	GPIO_BA+0x188	R/W	PG Data Output Value	0x0000_FE1C
PG_DATMSK	GPIO_BA+0x18C	R/W	PG Data Output Write Mask	0x0000_0000
PG_PIN	GPIO_BA+0x190	R	PG Pin Value	0x0000_XXXX
PG_DBEN	GPIO_BA+0x194	R/W	PG De-bounce Enable Control Register	0x0000_0000
PG_INTTYPE	GPIO_BA+0x198	R/W	PG Interrupt Trigger Type Control	0x0000_0000

PG_INTEN	GPIO_BA+0x19C	R/W	PG Interrupt Enable Control Register	0x0000_0000
PG_INTSRC	GPIO_BA+0x1A0	R/W	PG Interrupt Source Flag	0x0000_XXXX
PG_SMTEN	GPIO_BA+0x1A4	R/W	PG Input Schmitt Trigger Enable Register	0x0000_0000
PG_SLEWCTL	GPIO_BA+0x1A8	R/W	PG High Slew Rate Control Register	0x0000_0000
PG_PUSEL	GPIO_BA+0x1B0	R/W	PG Pull-up and Pull-down Selection Register	0x0000_0000
PG_DBCTL	GPIO_BA+0x1B4	R/W	PG Interrupt De-bounce Control Register	0x0000_0020
PH_MODE	GPIO_BA+0x1C0	R/W	PH I/O Mode Control	0xXXXX_XXXX
PH_DINOFF	GPIO_BA+0x1C4	R/W	PH Digital Input Path Disable Control	0x0000_0000
PH_DOUT	GPIO_BA+0x1C8	R/W	PH Data Output Value	0x0000_0FF0
PH_DATMSK	GPIO_BA+0x1CC	R/W	PH Data Output Write Mask	0x0000_0000
PH_PIN	GPIO_BA+0x1D0	R	PH Pin Value	0x0000_XXXX
PH_DBEN	GPIO_BA+0x1D4	R/W	PH De-bounce Enable Control Register	0x0000_0000
PH_INTTYPE	GPIO_BA+0x1D8	R/W	PH Interrupt Trigger Type Control	0x0000_0000
PH_INTEN	GPIO_BA+0x1DC	R/W	PH Interrupt Enable Control Register	0x0000_0000
PH_INTSRC	GPIO_BA+0x1E0	R/W	PH Interrupt Source Flag	0x0000_XXXX
PH_SMTEN	GPIO_BA+0x1E4	R/W	PH Input Schmitt Trigger Enable Register	0x0000_0000
PH_SLEWCTL	GPIO_BA+0x1E8	R/W	PH High Slew Rate Control Register	0x0000_0000
PH_PUSEL	GPIO_BA+0x1F0	R/W	PH Pull-up and Pull-down Selection Register	0x0000_0000
PH_DBCTL	GPIO_BA+0x1F4	R/W	PH Interrupt De-bounce Control Register	0x0000_0020
PAn_PDIO n=0,1..15	GPIO_BA+0x800+(0x04 * n)	R/W	GPIO PA.n Pin Data Input/Output Register	0x0000_000X
PBn_PDIO n=0,1..15	GPIO_BA+0x840+(0x04 * n)	R/W	GPIO PB.n Pin Data Input/Output Register	0x0000_000X
PCn_PDIO n=0,1..15	GPIO_BA+0x880+(0x04 * n)	R/W	GPIO PC.n Pin Data Input/Output Register	0x0000_000X
PDn_PDIO n=0,1..15	GPIO_BA+0x8C0+(0x04 * n)	R/W	GPIO PD.n Pin Data Input/Output Register	0x0000_000X
PEn_PDIO n=0,1..15	GPIO_BA+0x900+(0x04 * n)	R/W	GPIO PE.n Pin Data Input/Output Register	0x0000_000X
PFn_PDIO n=0,1..15	GPIO_BA+0x940+(0x04 * n)	R/W	GPIO PF.n Pin Data Input/Output Register	0x0000_000X
PGn_PDIO n=0,1..15	GPIO_BA+0x980+(0x04 * n)	R/W	GPIO PG.n Pin Data Input/Output Register	0x0000_000X
PHn_PDIO	GPIO_BA+0x9C0+(0x04 * n)	R/W	GPIO PH.n Pin Data Input/Output Register	0x0000_000X

n=0,1..15				

6.7.7 Register Description

Port A-H I/O Mode Control (Px_MODE)

Register	Offset	R/W	Description	Reset Value
PA_MODE	GPIO_BA+0x000	R/W	PA I/O Mode Control	0xXXXX_XXXX
PB_MODE	GPIO_BA+0x040	R/W	PB I/O Mode Control	0xXXXX_XXXX
PC_MODE	GPIO_BA+0x080	R/W	PC I/O Mode Control	0xXXXX_XXXX
PD_MODE	GPIO_BA+0x0C0	R/W	PD I/O Mode Control	0xXXXX_XXXX
PE_MODE	GPIO_BA+0x100	R/W	PE I/O Mode Control	0xXXXX_XXXX
PF_MODE	GPIO_BA+0x140	R/W	PF I/O Mode Control	0xXXXX_XXXX
PG_MODE	GPIO_BA+0x180	R/W	PG I/O Mode Control	0xXXXX_XXXX
PH_MODE	GPIO_BA+0x1C0	R/W	PH I/O Mode Control	0xXXXX_XXXX

31	30	29	28	27	26	25	24
MODE15		MODE14		MODE13		MODE12	
23	22	21	20	19	18	17	16
MODE11		MODE10		MODE9		MODE8	
15	14	13	12	11	10	9	8
MODE7		MODE6		MODE5		MODE4	
7	6	5	4	3	2	1	0
MODE3		MODE2		MODE1		MODE0	

Bits	Description	
[2n+1:2n] n=0,1..15	**MODEn**	**Port A-H I/O Pin[n] Mode Control** Determine each I/O mode of Px.n pins. 00 = Px.n is in Input mode (tri-state). 01 = Px.n is in Push-pull Output mode. 10 = Px.n is in Open-drain Output mode. 11 = Px.n is in Quasi-bidirectional mode. **Note 1:** The initial value of this field is defined by CIOINI (CONFIG0 [10]). If CIOINI is set to 0, the default value is 0xFFFF_FFFF and all pins will be quasi-bidirectional mode after chip powered on. If CIOINI is set to 1, the default value is 0x0000_0000 and all pins will be input mode after chip powered on. **Note 2:** The PC.14/PC.15/ PD.13/ PD.15/ PF.12~15/ PG.0/ PG.1/ PG.5~8/ PH.0~3/ PH.12~15 pin is ignored. **Note 3:** If MFOS is enabled then GPIO mode setting is ignored.

Port A-H Digital Input Path Disable Control (Px_DINOFF)

Register	Offset	R/W	Description	Reset Value
PA_DINOFF	GPIO_BA+0x004	R/W	PA Digital Input Path Disable Control	0x0000_0000
PB_DINOFF	GPIO_BA+0x044	R/W	PB Digital Input Path Disable Control	0x0000_0000
PC_DINOFF	GPIO_BA+0x084	R/W	PC Digital Input Path Disable Control	0x0000_0000
PD_DINOFF	GPIO_BA+0x0C4	R/W	PD Digital Input Path Disable Control	0x0000_0000
PE_DINOFF	GPIO_BA+0x104	R/W	PE Digital Input Path Disable Control	0x0000_0000
PF_DINOFF	GPIO_BA+0x144	R/W	PF Digital Input Path Disable Control	0x0000_0000
PG_DINOFF	GPIO_BA+0x184	R/W	PG Digital Input Path Disable Control	0x0000_0000
PH_DINOFF	GPIO_BA+0x1C4	R/W	PH Digital Input Path Disable Control	0x0000_0000

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

Bits	Description	
[n+16] n=0,1..15	**DINOFF[n]**	**Port A-H Pin[n] Digital Input Path Disable Bit** Each of these bits is used to control if the digital input path of corresponding Px.n pin is disabled. If input is analog signal, users can disable Px.n digital input path to avoid input current leakage. 0 = Px.n digital input path Enabled. 1 = Px.n digital input path Disabled (digital input tied to low). **Note:** The PC.14/PC.15/ PD.13/ PD.15/ PF.12~15/ PG.0/ PG.1/ PG.5~8/ PH.0~3/ PH.12~15 pin is ignored.
[15:0]	**Reserved**	Reserved.

Port A-H Data Output Value (Px_DOUT)

Register	Offset	R/W	Description	Reset Value
PA_DOUT	GPIO_BA+0x008	R/W	PA Data Output Value	0x0000_FFFF
PB_DOUT	GPIO_BA+0x048	R/W	PB Data Output Value	0x0000_FFFF
PC_DOUT	GPIO_BA+0x088	R/W	PC Data Output Value	0x0000_3FFF
PD_DOUT	GPIO_BA+0x0C8	R/W	PD Data Output Value	0x0000_7FFF
PE_DOUT	GPIO_BA+0x108	R/W	PE Data Output Value	0x0000_FFFF
PF_DOUT	GPIO_BA+0x148	R/W	PF Data Output Value	0x0000_0FFF
PG_DOUT	GPIO_BA+0x188	R/W	PG Data Output Value	0x0000_FE1C
PH_DOUT	GPIO_BA+0x1C8	R/W	PH Data Output Value	0x0000_0FF0

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**DOUT[n]**	**Port A-H Pin[n] Output Value** Each of these bits controls the status of a Px.n pin when the Px.n is configured as Push-pull output, Open-drain output or Quasi-bidirectional mode. 0 = Px.n will drive Low if the Px.n pin is configured as Push-pull output, Open-drain output or Quasi-bidirectional mode. 1 = Px.n will drive High if the Px.n pin is configured as Push-pull output or Quasi-bidirectional mode. **Note:** The PC.14/PC.15/ PD.13/ PD.15/ PF.12~15/ PG.0/ PG.1/ PG.5~8/ PH.0~3/ PH.12~15 pin is ignored.

Port A-H Data Output Write Mask (Px_DATMSK)

Register	Offset	R/W	Description	Reset Value
PA_DATMSK	GPIO_BA+0x00C	R/W	PA Data Output Write Mask	0x0000_0000
PB_DATMSK	GPIO_BA+0x04C	R/W	PB Data Output Write Mask	0x0000_0000
PC_DATMSK	GPIO_BA+0x08C	R/W	PC Data Output Write Mask	0x0000_0000
PD_DATMSK	GPIO_BA+0x0CC	R/W	PD Data Output Write Mask	0x0000_0000
PE_DATMSK	GPIO_BA+0x10C	R/W	PE Data Output Write Mask	0x0000_0000
PF_DATMSK	GPIO_BA+0x14C	R/W	PF Data Output Write Mask	0x0000_0000
PG_DATMSK	GPIO_BA+0x18C	R/W	PG Data Output Write Mask	0x0000_0000
PH_DATMSK	GPIO_BA+0x1CC	R/W	PH Data Output Write Mask	0x0000_0000

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

Bits	Description	
[31:8]	Reserved	Reserved.
[n] n=0,1..15	DATMSK[n]	**Port A-H Pin[n] Data Output Write Mask** These bits are used to protect the corresponding DOUT (Px_DOUT[n]) bit. When the DATMSK (Px_DATMSK[n]) bit is set to 1, the corresponding DOUT (Px_DOUT[n]) bit is protected. If the write signal is masked, writing data to the protect bit is ineffective. 0 = Corresponding DOUT (Px_DOUT[n]) bit can be updated. 1 = Corresponding DOUT (Px_DOUT[n]) bit protected. **Note 1:** This function only protects the corresponding DOUT (Px_DOUT[n]) bit, and will not protect the corresponding PDIO (Pxn_PDIO[0]) bit. **Note 2:** The PC.14/PC.15/ PD.13/ PD.15/ PF.12~15/ PG.0/ PG.1/ PG.5~8/ PH.0~3/ PH.12~15 pin is ignored.

Port A-H Pin Value (Px_PIN)

Register	Offset	R/W	Description	Reset Value
PA_PIN	GPIO_BA+0x010	R	PA Pin Value	0x0000_XXXX
PB_PIN	GPIO_BA+0x050	R	PB Pin Value	0x0000_XXXX
PC_PIN	GPIO_BA+0x090	R	PC Pin Value	0x0000_XXXX
PD_PIN	GPIO_BA+0x0D0	R	PD Pin Value	0x0000_XXXX
PE_PIN	GPIO_BA+0x110	R	PE Pin Value	0x0000_XXXX
PF_PIN	GPIO_BA+0x150	R	PF Pin Value	0x0000_XXXX
PG_PIN	GPIO_BA+0x190	R	PG Pin Value	0x0000_XXXX
PH_PIN	GPIO_BA+0x1D0	R	PH Pin Value	0x0000_XXXX

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**PIN[n]**	**Port A-H Pin[n] Pin Value** Each bit of the register reflects the actual status of the respective Px.n pin. 0 = The corresponding pin status is low. 1 = The corresponding pin status is high. **Note:** The PC.14/PC.15/ PD.13/ PD.15/ PF.12~15/ PG.0/ PG.1/ PG.5~8/ PH.0~3/ PH.12~15 pin is ignored.

Port A-H De-bounce Enable Control Register (Px_DBEN)

Register	Offset	R/W	Description	Reset Value
PA_DBEN	GPIO_BA+0x014	R/W	PA De-bounce Enable Control Register	0x0000_0000
PB_DBEN	GPIO_BA+0x054	R/W	PB De-bounce Enable Control Register	0x0000_0000
PC_DBEN	GPIO_BA+0x094	R/W	PC De-bounce Enable Control Register	0x0000_0000
PD_DBEN	GPIO_BA+0x0D4	R/W	PD De-bounce Enable Control Register	0x0000_0000
PE_DBEN	GPIO_BA+0x114	R/W	PE De-bounce Enable Control Register	0x0000_0000
PF_DBEN	GPIO_BA+0x154	R/W	PF De-bounce Enable Control Register	0x0000_0000
PG_DBEN	GPIO_BA+0x194	R/W	PG De-bounce Enable Control Register	0x0000_0000
PH_DBEN	GPIO_BA+0x1D4	R/W	PH De-bounce Enable Control Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**DBEN[n]**	**Port A-H Pin[n] Input Signal De-bounce Enable Bit** The DBEN[n] bit is used to enable the de-bounce function for each corresponding bit. If the input signal pulse width cannot be sampled by continuous two de-bounce sample cycle, the input signal transition is seen as the signal bounce and will not trigger the interrupt. The de-bounce clock source is controlled by DBCLKSRC (Px_DBCTL [4]), one de-bounce sample cycle period is controlled by DBCLKSEL (Px_DBCTL [3:0]). 0 = Px.n de-bounce function Disabled. 1 = Px.n de-bounce function Enabled. The de-bounce function is valid only for edge triggered interrupt. If the interrupt mode is level triggered, the de-bounce enable bit is ineffective. **Note:** The PC.14/PC.15/ PD.13/ PD.15/ PF.12~15/ PG.0/ PG.1/ PG.5~8/ PH.0~3/ PH.12~15 pin is ignored.

Port A-H Interrupt Trigger Type Control (Px_INTTYPE)

Register	Offset	R/W	Description	Reset Value
PA_INTTYPE	GPIO_BA+0x018	R/W	PA Interrupt Trigger Type Control	0x0000_0000
PB_INTTYPE	GPIO_BA+0x058	R/W	PB Interrupt Trigger Type Control	0x0000_0000
PC_INTTYPE	GPIO_BA+0x098	R/W	PC Interrupt Trigger Type Control	0x0000_0000
PD_INTTYPE	GPIO_BA+0x0D8	R/W	PD Interrupt Trigger Type Control	0x0000_0000
PE_INTTYPE	GPIO_BA+0x118	R/W	PE Interrupt Trigger Type Control	0x0000_0000
PF_INTTYPE	GPIO_BA+0x158	R/W	PF Interrupt Trigger Type Control	0x0000_0000
PG_INTTYPE	GPIO_BA+0x198	R/W	PG Interrupt Trigger Type Control	0x0000_0000
PH_INTTYPE	GPIO_BA+0x1D8	R/W	PH Interrupt Trigger Type Control	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**TYPE[n]**	**Port A-H Pin[n] Edge or Level Detection Interrupt Trigger Type Control** TYPE (Px_INTTYPE[n]) bit is used to control the triggered interrupt is by level trigger or by edge trigger. If the interrupt is by edge trigger, the trigger source can be controlled by de-bounce. If the interrupt is by level trigger, the input source is sampled by one HCLK clock and generates the interrupt. 0 = Edge trigger interrupt. 1 = Level trigger interrupt. If the pin is set as the level trigger interrupt, only one level can be set on the registers RHIEN (Px_INTEN[n+16])/FLIEN (Px_INTEN[n]). If both levels to trigger interrupt are set, the setting has no effect and no interrupt will occur. The de-bounce function is valid only for edge triggered interrupt. If the interrupt mode is level triggered, the de-bounce enable bit is ineffective. **Note:** The PC.14/PC.15/ PD.13/ PD.15/ PF.12~15/ PG.0/ PG.1/ PG.5~8/ PH.0~3/ PH.12~15 pin is ignored.

Port A-H Interrupt Enable Control Register (Px_INTEN)

Register	Offset	R/W	Description	Reset Value
PA_INTEN	GPIO_BA+0x01C	R/W	PA Interrupt Enable Control Register	0x0000_0000
PB_INTEN	GPIO_BA+0x05C	R/W	PB Interrupt Enable Control Register	0x0000_0000
PC_INTEN	GPIO_BA+0x09C	R/W	PC Interrupt Enable Control Register	0x0000_0000
PD_INTEN	GPIO_BA+0x0DC	R/W	PD Interrupt Enable Control Register	0x0000_0000
PE_INTEN	GPIO_BA+0x11C	R/W	PE Interrupt Enable Control Register	0x0000_0000
PF_INTEN	GPIO_BA+0x15C	R/W	PF Interrupt Enable Control Register	0x0000_0000
PG_INTEN	GPIO_BA+0x19C	R/W	PG Interrupt Enable Control Register	0x0000_0000
PH_INTEN	GPIO_BA+0x1DC	R/W	PH Interrupt Enable Control Register	0x0000_0000

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

Bits	Description	
[n+16] n=0,1..15	**RHIEN[n]**	**Port A-H Pin[n] Rising Edge or High Level Interrupt Trigger Type Enable Bit** The RHIEN (Px_INTEN[n+16]) bit is used to enable the interrupt for each of the corresponding input Px.n pin. Set bit to 1 also enable the pin wake-up function. When setting the RHIEN (Px_INTEN[n+16]) bit to 1 : If the interrupt is level trigger (TYPE (Px_INTTYPE[n]) bit is set to 1), the input Px.n pin will generate the interrupt while this pin state is at high level. If the interrupt is edge trigger (TYPE (Px_INTTYPE[n]) bit is set to 0), the input Px.n pin will generate the interrupt while this pin state changed from low to high. 0 = Px.n level high or low to high interrupt Disabled. 1 = Px.n level high or low to high interrupt Enabled. **Note:** The PC.14/PC.15/ PD.13/ PD.15/ PF.12~15/ PG.0/ PG.1/ PG.5~8/ PH.0~3/ PH.12~15 pin is ignored.
[n] n=0,1..15	**FLIEN[n]**	**Port A-H Pin[n] Falling Edge or Low Level Interrupt Trigger Type Enable Bit** The FLIEN (Px_INTEN[n]) bit is used to enable the interrupt for each of the corresponding input Px.n pin. Set bit to 1 also enable the pin wake-up function. When setting the FLIEN (Px_INTEN[n]) bit to 1 : If the interrupt is level trigger (TYPE (Px_INTTYPE[n]) bit is set to 1), the input Px.n pin will generate the interrupt while this pin state is at low level.

		If the interrupt is edge trigger(TYPE (Px_INTTYPE[n]) bit is set to 0), the input Px.n pin will generate the interrupt while this pin state changed from high to low. 0 = Px.n level low or high to low interrupt Disabled. 1 = Px.n level low or high to low interrupt Enabled. **Note:** The PC.14/PC.15/ PD.13/ PD.15/ PF.12~15/ PG.0/ PG.1/ PG.5~8/ PH.0~3/ PH.12~15 pin is ignored.

Port A-H Interrupt Source Flag (Px_INTSRC)

Register	Offset	R/W	Description	Reset Value
PA_INTSRC	GPIO_BA+0x020	R/W	PA Interrupt Source Flag	0x0000_XXXX
PB_INTSRC	GPIO_BA+0x060	R/W	PB Interrupt Source Flag	0x0000_XXXX
PC_INTSRC	GPIO_BA+0x0A0	R/W	PC Interrupt Source Flag	0x0000_XXXX
PD_INTSRC	GPIO_BA+0x0E0	R/W	PD Interrupt Source Flag	0x0000_XXXX
PE_INTSRC	GPIO_BA+0x120	R/W	PE Interrupt Source Flag	0x0000_XXXX
PF_INTSRC	GPIO_BA+0x160	R/W	PF Interrupt Source Flag	0x0000_XXXX
PG_INTSRC	GPIO_BA+0x1A0	R/W	PG Interrupt Source Flag	0x0000_XXXX
PH_INTSRC	GPIO_BA+0x1E0	R/W	PH Interrupt Source Flag	0x0000_XXXX

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**INTSRC[n]**	**Port A-H Pin[n] Interrupt Source Flag** Write Operation: 0 = No action. 1 = Clear the corresponding pending interrupt. Read Operation: 0 = No interrupt at Px.n. 1 = Px.n generates an interrupt. **Note:** The PC.14/PC.15/ PD.13/ PD.15/ PF.12~15/ PG.0/ PG.1/ PG.5~8/ PH.0~3/ PH.12~15 pin is ignored.

Port A-H Input Schmitt Trigger Enable Register (Px_SMTEN)

Register	Offset	R/W	Description	Reset Value
PA_SMTEN	GPIO_BA+0x024	R/W	PA Input Schmitt Trigger Enable Register	0x0000_0000
PB_SMTEN	GPIO_BA+0x064	R/W	PB Input Schmitt Trigger Enable Register	0x0000_0000
PC_SMTEN	GPIO_BA+0x0A4	R/W	PC Input Schmitt Trigger Enable Register	0x0000_0000
PD_SMTEN	GPIO_BA+0x0E4	R/W	PD Input Schmitt Trigger Enable Register	0x0000_0000
PE_SMTEN	GPIO_BA+0x124	R/W	PE Input Schmitt Trigger Enable Register	0x0000_0000
PF_SMTEN	GPIO_BA+0x164	R/W	PF Input Schmitt Trigger Enable Register	0x0000_0000
PG_SMTEN	GPIO_BA+0x1A4	R/W	PG Input Schmitt Trigger Enable Register	0x0000_0000
PH_SMTEN	GPIO_BA+0x1E4	R/W	PH Input Schmitt Trigger Enable Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**SMTEN[n]**	**Port A-H Pin[n] Input Schmitt Trigger Enable Bit** 0 = Px.n input schmitt trigger function Disabled. 1 = Px.n input schmitt trigger function Enabled. **Note:** The PC.14/PC.15/ PD.13/ PD.15/ PF.12~15/ PG.0/ PG.1/ PG.5~8/ PH.0~3/ PH.12~15 pin is ignored.

Port A-H High Slew Rate Control Register (Px_SLEWCTL)

Register	Offset	R/W	Description	Reset Value
PA_SLEWCTL	GPIO_BA+0x028	R/W	PA High Slew Rate Control Register	0x0000_0000
PB_SLEWCTL	GPIO_BA+0x068	R/W	PB High Slew Rate Control Register	0x0000_0000
PC_SLEWCTL	GPIO_BA+0x0A8	R/W	PC High Slew Rate Control Register	0x0000_0000
PD_SLEWCTL	GPIO_BA+0x0E8	R/W	PD High Slew Rate Control Register	0x0000_0000
PE_SLEWCTL	GPIO_BA+0x128	R/W	PE High Slew Rate Control Register	0x0000_0000
PF_SLEWCTL	GPIO_BA+0x168	R/W	PF High Slew Rate Control Register	0x0000_0000
PG_SLEWCTL	GPIO_BA+0x1A8	R/W	PG High Slew Rate Control Register	0x0000_0000
PH_SLEWCTL	GPIO_BA+0x1E8	R/W	PH High Slew Rate Control Register	0x0000_0000

31	30	29	28	27	26	25	24
HSREN15		HSREN14		HSREN13		HSREN12	
23	22	21	20	19	18	17	16
HSREN11		HSREN10		HSREN9		HSREN8	
15	14	13	12	11	10	9	8
HSREN07		HSREN6		HSREN5		HSREN4	
7	6	5	4	3	2	1	0
HSREN03		HSREN2		HSREN1		HSREN0	

Bits	Description	
[2n+1:2n] n=0,1..15	**HSRENn**	**Port A-H Pin[n] High Slew Rate Control** 00 = Px.n output with normal slew rate mode. 01 = Px.n output with high slew rate mode. 10 = Px.n output with fast slew rate mode. 11 = Reserved. **Note 1:** The PC.14/PC.15/ PD.13/ PD.15/ PF.12~15/ PG.0/ PG.1/ PG.5~8/ PH.0~3/ PH.12~15 pin is ignored. **Note 2:** Please refer to the M2354 Datasheet for detailed pin operation voltage information about V_{DD}, V_{DDIO} and V_{BAT} electrical characteristics.

Port A-H Pull-up and Pull-down Selection Register (Px_PUSEL)

Register	Offset	R/W	Description	Reset Value
PA_PUSEL	GPIO_BA+0x030	R/W	PA Pull-up and Pull-down Selection Register	0x0000_0000
PB_PUSEL	GPIO_BA+0x070	R/W	PB Pull-up and Pull-down Selection Register	0x0000_0000
PC_PUSEL	GPIO_BA+0x0B0	R/W	PC Pull-up and Pull-down Selection Register	0x0000_0000
PD_PUSEL	GPIO_BA+0x0F0	R/W	PD Pull-up and Pull-down Selection Register	0x0000_0000
PE_PUSEL	GPIO_BA+0x130	R/W	PE Pull-up and Pull-down Selection Register	0x0000_0000
PF_PUSEL	GPIO_BA+0x170	R/W	PF Pull-up and Pull-down Selection Register	0x0000_0000
PG_PUSEL	GPIO_BA+0x1B0	R/W	PG Pull-up and Pull-down Selection Register	0x0000_0000
PH_PUSEL	GPIO_BA+0x1F0	R/W	PH Pull-up and Pull-down Selection Register	0x0000_0000

31	30	29	28	27	26	25	24
PUSEL15		PUSEL14		PUSEL13		PUSEL12	
23	22	21	20	19	18	17	16
PUSEL11		PUSEL10		PUSEL9		PUSEL8	
15	14	13	12	11	10	9	8
PUSEL7		PUSEL6		PUSEL5		PUSEL4	
7	6	5	4	3	2	1	0
PUSEL3		PUSEL2		PUSEL1		PUSEL0	

Bits	Description	
[2n+1:2n] n=0,1..15	PUSELn	**Port A-H Pin[n] Pull-up and Pull-down Enable Register** Determine each I/O Pull-up/pull-down of Px.n pins. 00 = Px.n pull-up and pull-down disabled. 01 = Px.n pull-up enabled. 10 = Px.n pull-down enabled. 11 = Px.n pull-up and pull- down disabled. **Note 1:** Basically, the pull-up control and pull-down control has following behavior limitation. The independent pull-up control register only valid when MODEn set as input and open-drain mode even if I/O function is switched to multi-function pin. Ex: UARTx_RXD. The independent pull-down control register only valid when MODEn set as tri-state mode. When both pull-up pull-down is set as 1 at “tri-state” mode, keep I/O in tri-state mode. **Note 2:** The PC.14/PC.15/ PD.13/ PD.15/ PF.12~15/ PG.0/ PG.1/ PG.5~8/ PH.0~3/ PH.12~15 pin is ignored.

Port A-H Interrupt De-bounce Control Register (Px_DBCTL)

Register	Offset	R/W	Description	Reset Value
PA_DBCTL	GPIO_BA+0x034	R/W	PA Interrupt De-bounce Control Register	0x0000_0020
PB_DBCTL	GPIO_BA+0x074	R/W	PB Interrupt De-bounce Control Register	0x0000_0020
PC_DBCTL	GPIO_BA+0x0B4	R/W	PC Interrupt De-bounce Control Register	0x0000_0020
PD_DBCTL	GPIO_BA+0x0F4	R/W	PD Interrupt De-bounce Control Register	0x0000_0020
PE_DBCTL	GPIO_BA+0x134	R/W	PE Interrupt De-bounce Control Register	0x0000_0020
PF_DBCTL	GPIO_BA+0x174	R/W	PF Interrupt De-bounce Control Register	0x0000_0020
PG_DBCTL	GPIO_BA+0x1B4	R/W	PG Interrupt De-bounce Control Register	0x0000_0020
PH_DBCTL	GPIO_BA+0x1F4	R/W	PH Interrupt De-bounce Control Register	0x0000_0020

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**ICLKON**	**DBCLKSRC**	**DBCLKSEL**			

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**ICLKON**	**Interrupt Clock on Mode (Secure Only)** 0 = Edge detection circuit is active only if I/O pin corresponding RHIEN (Px_INTEN[n+16])/FLIEN (Px_INTEN[n]) bit is set to 1. 1 = All I/O pins edge detection circuit is always active after reset. **Note 1:** It is recommended to disable this bit to save system power if no special application concern. **Note 2:** Max. n=15 for port A/B/E/G. Max. n=13 for port C. Max. n=14 for port D. Max. n=11 for port F/H. **Note 3:** This bit is only accessible from the Secure state.

Bits	Description	
[4]	**DBCLKSRC**	**De-bounce Counter Clock Source Selection (Secure only)** 0 = De-bounce counter clock source is the HCLK. 1 = De-bounce counter clock source is the 32 kHz internal low speed RC oscillator (LIRC). **Note:** This bit is only accessible from the Secure state.
[3:0]	**DBCLKSEL**	**De-bounce Sampling Cycle Selection (Secure only)** 0000 = Sample interrupt input once per 1 clocks. 0001 = Sample interrupt input once per 2 clocks. 0010 = Sample interrupt input once per 4 clocks. 0011 = Sample interrupt input once per 8 clocks. 0100 = Sample interrupt input once per 16 clocks. 0101 = Sample interrupt input once per 32 clocks. 0110 = Sample interrupt input once per 64 clocks. 0111 = Sample interrupt input once per 128 clocks. 1000 = Sample interrupt input once per 256 clocks. 1001 = Sample interrupt input once per 2*256 clocks. 1010 = Sample interrupt input once per 4*256 clocks. 1011 = Sample interrupt input once per 8*256 clocks. 1100 = Sample interrupt input once per 16*256 clocks. 1101 = Sample interrupt input once per 32*256 clocks. 1110 = Sample interrupt input once per 64*256 clocks. 1111 = Sample interrupt input once per 128*256 clocks. **Note:** These bits are only accessible from the Secure state.

GPIO Px.n Pin Data Input/Output Register (Pxn_PDIO)

Register	Offset	R/W	Description	Reset Value
PAn_PDIO n=0,1..15	GPIO_BA+0x800+(0x04 * n)	R/W	GPIO PA.n Pin Data Input/Output Register	0x0000_000X
PBn_PDIO n=0,1..15	GPIO_BA+0x840+(0x04 * n)	R/W	GPIO PB.n Pin Data Input/Output Register	0x0000_000X
PCn_PDIO n=0,1..15	GPIO_BA+0x880+(0x04 * n)	R/W	GPIO PC.n Pin Data Input/Output Register	0x0000_000X
PDn_PDIO n=0,1..15	GPIO_BA+0x8C0+(0x04 * n)	R/W	GPIO PD.n Pin Data Input/Output Register	0x0000_000X
PEn_PDIO n=0,1..15	GPIO_BA+0x900+(0x04 * n)	R/W	GPIO PE.n Pin Data Input/Output Register	0x0000_000X
PFn_PDIO n=0,1..15	GPIO_BA+0x940+(0x04 * n)	R/W	GPIO PF.n Pin Data Input/Output Register	0x0000_000X
PGn_PDIO n=0,1..15	GPIO_BA+0x980+(0x04 * n)	R/W	GPIO PG.n Pin Data Input/Output Register	0x0000_000X
PHn_PDIO n=0,1..15	GPIO_BA+0x9C0+(0x04 * n)	R/W	GPIO PH.n Pin Data Input/Output Register	0x0000_000X

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**PDIO**	**GPIO Px.n Pin Data Input/Output** Writing this bit can control one GPIO pin output value. 0 = Corresponding GPIO pin set to low. 1 = Corresponding GPIO pin set to high. Read this register to get GPIO pin status. For example, writing PA0_PDIO will reflect the written value to bit DOUT (PA_DOUT[0]), reading PA0_PDIO will return the value of PIN (PA_PIN[0]). **Note 1:** The writing operation will not be affected by register DATMSK (Px_DATMSK[n]). **Note 2:** The PC.14/PC.15/ PD.13/ PD.15/ PF.12~15/ PG.0/ PG.1/ PG.5~8/ PH.0~3/ PH.12~15 pin is ignored.

6.8 PDMA Controller (PDMA)

6.8.1 Overview

The peripheral direct memory access (PDMA) controller is used to provide high-speed data transfer. The PDMA controller can transfer data from one address to another without CPU intervention. This has the benefit of reducing the workload of CPU and keeps CPU resources free for other applications. There are two PDMA controller PDMA0 and PDMA1. PDMA0 is secure PDMA, PDMA1 can be configured as secure or non-secure PDMA. Each PDMA controller has a total of 8 channels and each channel can perform transfer between memory and peripherals or between memory and memory.

6.8.2 Features

- Supports 8 independently configurable channels
- Supports selectable 2 level of priority (fixed priority or round-robin priority)
- Supports 2 PDMA controller PDMA0 and PDMA1, PDMA0 is secure PDMA, PDMA1 can be configured as secure or non-secure PDMA
- Supports transfer data width of 8, 16, and 32 bits
- Supports source and destination address increment size can be byte, half-word, word or no increment
- Supports software and USB, UART, USCI, SPI, EPWM, I^2C, I^2S, Timer, ADC, and DAC request
- Supports Scatter-gather mode to perform sophisticated transfer through the use of the descriptor link list table
- Supports single and burst transfer type
- Supports time-out function on channel 0 and channel1
- Supports stride function from channel 0 to channel 5
- Supports enhanced stride function on channel 0 and channel1

6.8.3 Block Diagram

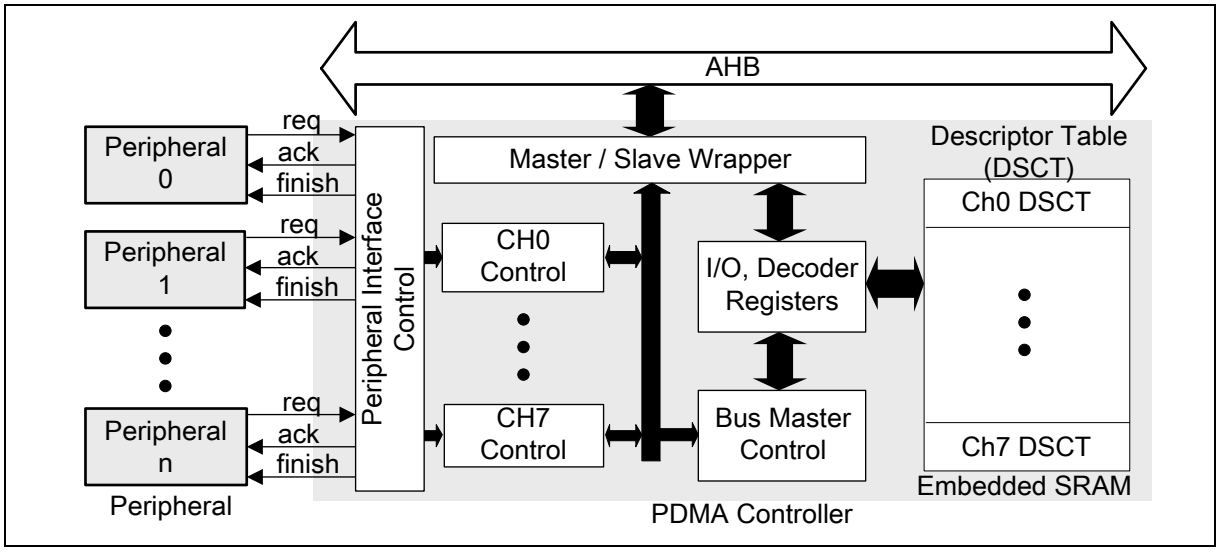


Figure 6.8-1 PDMA Controller Block Diagram

6.8.4 Basic Configuration

6.8.4.1 Basic Configuration of PDMA0

- Clock Source Configuration
  - Enable PDMA0 controller clock in PDMA0CKEN(CLK_AHBCLK [0]).
- Reset Configuration
  - Reset PDMA0 controller in PDMA0RST (SYS_IPRST0[2]).

6.8.4.2 Basic Configuration of PDMA1

- Clock Source Configuration
  - Enable PDMA1 controller clock in PDMA1CKEN (CLK_AHBCLK [1]).
- Reset Configuration
  - Reset PDMA1 controller in PDMA1RST (SYS_IPRST0[29]).

6.8.5 Functional Description

The PDMA controller transfers data from one address to another without CPU intervention. The PDMA controller supports 8 independent channels and serves only one channel at one time, as the result, PDMA controller supports two level channel priorities: fixed and round-robin priority, PDMA controller serves channel in order from highest to lowest priority channel. The PDMA controller supports two operation modes: Basic mode and Scatter-gather mode. Basic mode is used to perform one descriptor table transfer. Scatter-gather mode has more entries for each PDMA channel, and thus the PDMA controller supports sophisticated transfer through the entries. The descriptor table entry data structure contains many transfer information including the transfer source address, transfer destination address, transfer count, burst size, transfer type and operation mode. Figure 6.8-2 shows the diagram of descriptor table (DSCT) data structure.

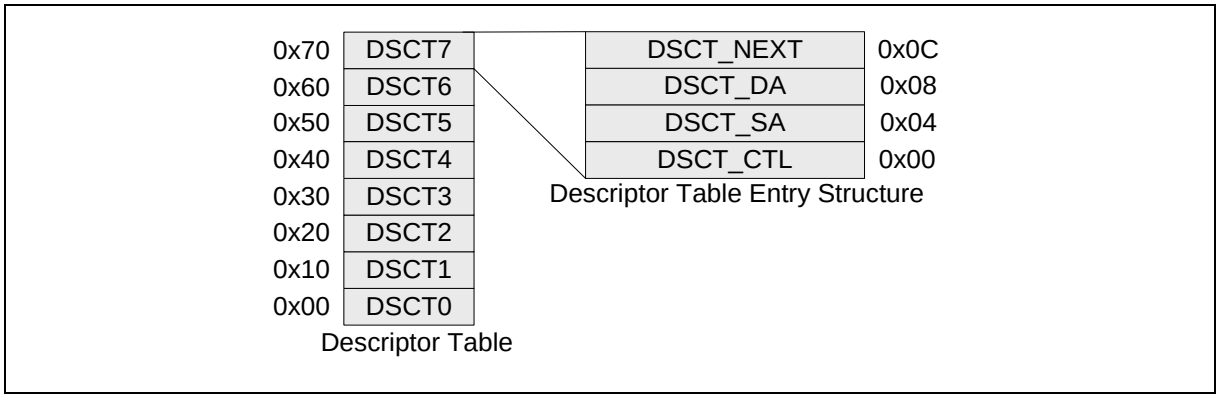


Figure 6.8-2 Descriptor Table Entry Structure

The PDMA controller also supports single and burst transfer type and the request source can be from software or peripheral request, transfer between memory to memory using software request. A single transfer means that software or peripheral is ready to transfer one data (every data needs one request), and the burst transfer means that software or peripherals will transfer multiple data (multiple data only need one request).

6.8.5.1 Channel Priority

The PDMA controller supports two level channel priorities including fixed and round-robin priority. The fixed priority channel has higher priority than round-robin priority channel. If multiple channels are set as fixed or round-robin priority, the higher channel will have higher priority. The priority order is listed in

Table 6.8-1.

PDMA_PRISET	Channel Number	Priority Setting	Arbitration Priority In Descending Order
1	7	Channel7, Fixed Priority	Highest
1	6	Channel6, Fixed Priority	---
---	---	---	---
1	0	Channel0, Fixed Priority	---
0	7	Channel7, Round-Robin Priority	---
0	6	Channel6, Round-Robin Priority	---
---	---	---	---
0	0	Channel0, Round-Robin Priority	Lowest

Table 6.8-1 Channel Priority Table

6.8.5.2 PDMA Operation Mode

The PDMA controller supports two operation modes including Basic mode and Scatter-gather mode.

Basic Mode

Basic mode is used to perform one descriptor table transfer mode. This mode can be used to transfer data between memory and memory, peripherals and memory or peripherals and peripherals, but if user want to transfer data between peripherals and peripherals, one thing must be sured is that the request from peripherals knows that the data is ready for transfer or not. PDMA controller operation mode can be set from OPMODE (PDMA_DSCTn_CTL[1:0], n denotes PDMA channel), the default setting is in idle state (OPMODE (PDMA_DSCTn_CTL[1:0]) = 0x0) and recommend user configure the descriptor table in idle state. If operation mode is not in idle state, user re-configure channel setting may make some operation error.

User must fill the transfer count TXCNT (PDMA_DSCTn_CTL[31:16]) register and select transfer width TXWIDTH (PDMA_DSCTn_CTL[13:12]), destination address increment size DAINC (PDMA_DSCTn_CTL[11:10]), source address increment size SAINC (PDMA_DSCTn_CTL[9:8]), burst size BURSIZE (PDMA_DSCTn_CTL[6:4]) and transfer type TXTYPE (PDMA_DSCTn_CTL[2]), then the PDMA controller will perform transfer operation in transfer state after receiving request signal. Finishing this task will generate an interrupt to CPU if corresponding PDMA interrupt bit INTENn (PDMA_INTEN[7:0]) is enabled and the operation mode will be updated to idle state as shown in Figure 6.8-3. If software configures the operation mode to idle state, the PDMA controller will not perform any transfer and then clear this operation request. Finishing this task will also generate an interrupt to CPU if corresponding PDMA interrupt bit is enabled.

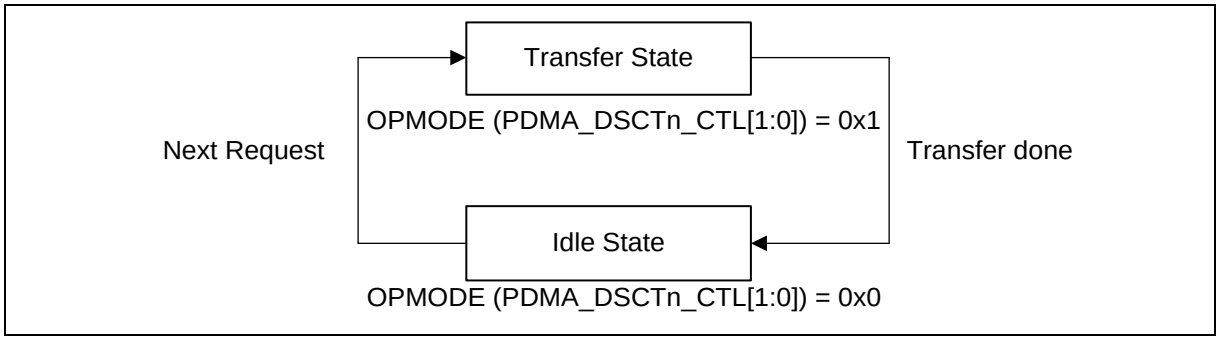


Figure 6.8-3 Basic Mode Finite State Machine

Scatter-gather Mode

Scatter-gather mode is a complex mode and can perform sophisticated transfer through the use of the description link list table as shown in Figure 6.8-4. Through operation mode user can perform peripheral wrapper-around, and multiple PDMA task can be used for data transfer between varied locations in system memory instead of a set of contiguous locations. Scatter-gather mode only needs a request to finish all table entries task till the last task with OPMODE (PDMA_DSCTn_CTL[1:0]) is idle state without ack. It also means Scatter-gather mode can only be used to transfer data between memory to memory without handshaking.

In Scatter-gather mode, the table is just used for jumping to the next table entry. The first task will not perform any operation transfer. Finishing each task will generate an interrupt to CPU if corresponding PDMA interrupt bit is enabled and TBINTDIS (PDMA_DSCTn_CTL[7]) bit is “0” (when finishing task and TBINTDIS bit is “0”, corresponding TDIFn (PDMA_TDSTS[7:0]) flag will be asserted and if this bit is “1” TDIFn will not be active).

If channel 7 has been triggered, and the operation mode is in Scatter-gather mode (OPMODE (PDMA_DSCTn_CTL[1:0]) = 0x2), the hardware will load the real PDMA information task from the address generated by adding PDMA_DSCTn_NEXT (link address) and PDMA_SCATBA (base address) registers. For example, base address is 0x2000_0000 (only MSB 16 bits valid in PDMA_SCATBA), the current link address is 0x0000_0100 (only LSB 16bits without last two bits [1:0] valid in PDMA_DSCTn_NEXT), and then the next DSCT entry start address is 0x2000_0100.

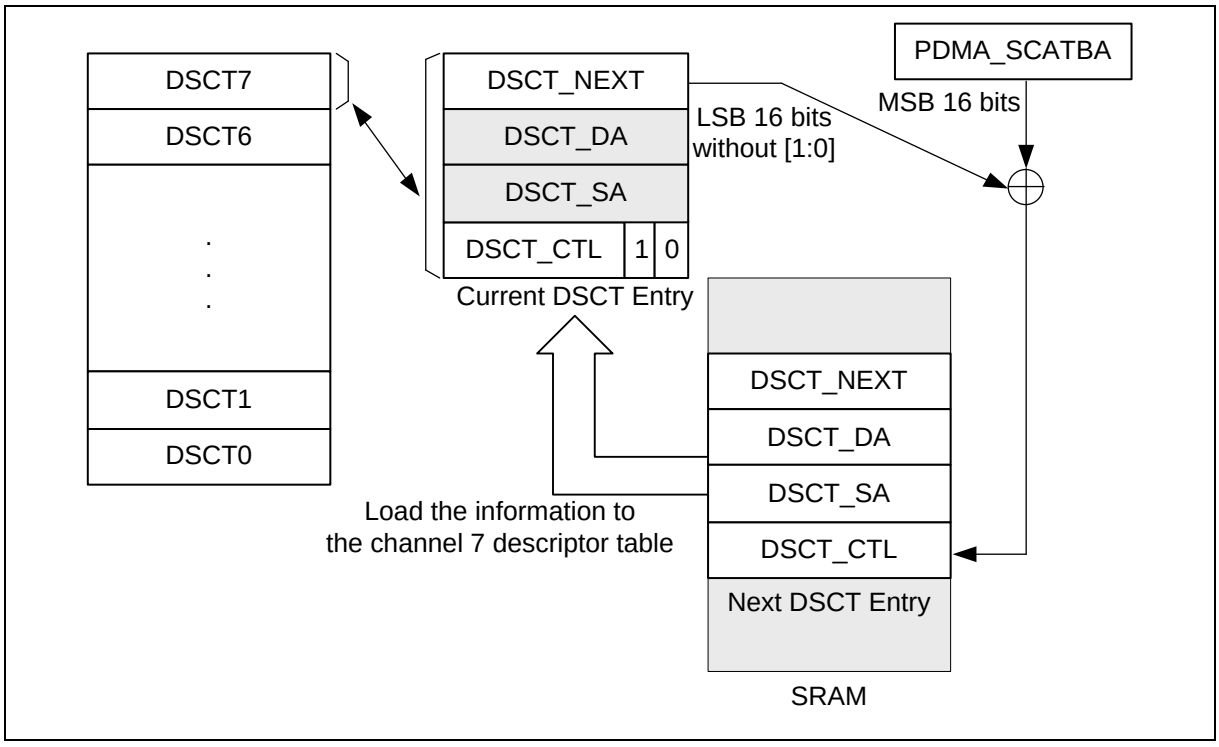


Figure 6.8-4 Descriptor Table Link List Structure

The above link list table operation is DSCT state in Scatter-gather Mode as shown in Figure 6.8-5. When loading the information is finished, it will go to transfer state and start transfer by this information automatically. However, if the next PDMA information is also set to Scatter-gather mode, the hardware will catch the next PDMA information block when the current task is finished. The Scatter-gather mode switches to basic mode when doing the next task. Then, the basic mode switches to Idle state when

the last task is finished.

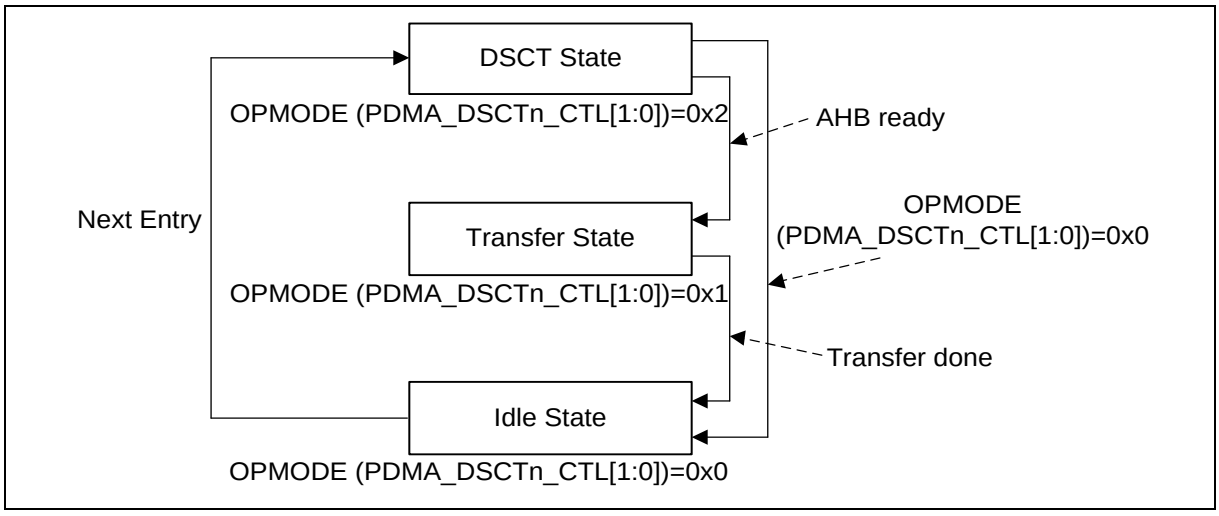


Figure 6.8-5 Scatter-gather Mode Finite State Machine

6.8.5.3 Transfer Type

The PDMA controller supports two transfer types: single transfer type and burst transfer type, configure by setting TXTYPE (PDMA_DSCTn_CTL[2]).

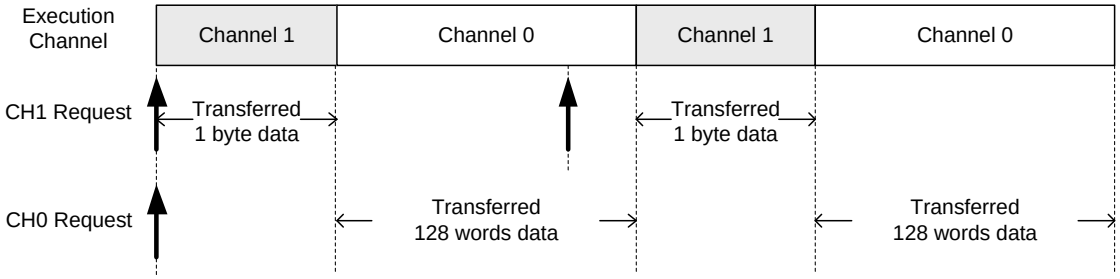
When the PDMA controller is operated in single transfer type, each transfer data needs one request signal for one transfer, after transferred data, TXCNT (PDMA_DSCTn_CTL[31:16]) will decrease 1. Transfer will be finished after the TXCNT (PDMA_DSCTn_CTL[31:16]) decreases to 0. In this mode, the BURSIZE (PDMA_DSCTn_CTL[6:4]) is not useful to control the transfer size. The BURSIZE (PDMA_DSCTn_CTL[6:4]) will be fixed as one.

For the burst transfer type, the PDMA controller transfers TXCNT (PDMA_DSCTn_CTL[31:16]) of data and need only one request signal. After transferred BURSIZE (PDMA_DSCTn_CTL[6:4]) of data, TXCNT (PDMA_DSCTn_CTL[31:16]) will decrease BURSIZE number. Transfer will be done after the transfer count TXCNT (PDMA_DSCTn_CTL[31:16]) decreases to 0. Note that burst transfer type can only be used for PDMA controller to do burst transfer between memory and memory. User must use single request type for memory-to-peripheral and peripheral-to-memory transfers. Please note that, PDMA transfer data between Flash and memory should finish before MCU enter idle mode or power done mode to prevent access wrong data.

Figure 6.8-6 shows an example about single and burst transfer type in basic mode. In this example, channel 1 uses single transfer type and TXCNT (PDMA_DSCTn_CTL[31:16]) = 127. Channel 0 uses burst transfer type, BURSIZE (PDMA_DSCTn_CTL[6:4]) = 128 and TXCNT (PDMA_DSCTn_CTL[31:16]) = 255. The operation sequence is described below:

1. Channel 0 and channel 1 get the trigger signal at the same time.
2. Channel 1 has higher priority than channel 0 by default; the PDMA controller will load the channel 1 descriptor table first and executing. But channel 1 is single transfer type, and thus the PDMA controller will only transfer one transfer data.
3. Then, the PDMA controller turns to the channel 0 and loads channel 0's descriptor table. The channel 0 is burst transfer type and the burst size selected to 128. Therefore, the PDMA controller will transfer 128 transfer data.
4. When channel 0 transfers 128 data, channel 1 gets another request signal, then after channel 0 finishes 128 transfer data, the PDMA controller will turn to channel 1 and transfer next one data.

5. After channel 1 transfers data, the PDMA controller switches to low priority channel 0 to continuous next 128 data transfer. If no channel 1 request receives, PDMA will start next channel 0, 128 data transfer.
6. The PDMA controller will complete transfer when channel 0 finishes data transfer 256 times, and channel 1 finishes transferring 128 times.



	CH1	CH0
TXCNT (PDMA_DSCTn_CTL[31:16])	127	255
BURSIZE (PDMA_DSCTn_CTL[6:4])	0 (1 byte)	2 (1 word)
BURSIZE (PDMA_DSCTn_CTL[6:4])	Non-useful	0 (128 transfers)
TXTYPE (PDMA_DSCTn_CTL[2])	1 (Single Transfer)	0 (Burst Transfer)

Figure 6.8-6 Example of Single Transfer Type and Burst Transfer Type in Basic Mode

6.8.5.4 Channel Time-out

Only PDMA channel 0 and channel 1 support time-out function. When the transfer channel is enabled and selected to the peripheral, corresponding channel time-out TOUTENn (PDMA_TOUTEN [n], n=0,1) is enabled, then channel's corresponding time-out counter will start count up from 0 while the channel has received trigger signal from the peripheral.

The time-out counter is based on output of HCLK prescaler, which is setting by corresponding channel's TOUTPSCn (PDMA_TOUTPSC [2+4n:4n], n=0,1). If time-out counter counts up from 0 to corresponding channel's TOCn (PDMA_TOC0_1 [16(n+1)-1):16n], n=0,1), the PDMA controller will generate interrupt signal when corresponding TOUTIENn (PDMA_TOUTIEN [n], n=0,1) is enabled. When time-out occurred, corresponding channel's REQTOFn (PDMA_INTSTS [n+8], n=0,1) will be set to indicate channel time-out is happened.

Time-out counter will restart from 0 while counter count to TOCn (PDMA_TOC0_1 [16(n+1)-1:16n], n=0,1), received trigger signal, time-out function is disabled or chip enters Power-down mode. The time-out counter will keep counting until time-out function is disabled.

Figure 6.8-7 shows an example about time-out counter operation. The operation sequence is described below:

1. The channel 0 time-out counter is not counting when time-out function is enabled by setting TOUTEN0(PDMA_TOUTEN[0]) bit to 1.
2. Time-out counter starts counting from 0 to the value of TOC0(PDMA_TOC0_1[15:0]) bits when receiving the first peripheral request.
3. Time-out counter is reset to 0 by received second peripheral request.

4. Channel 0 request time-out flag(REQTOF0(PDMA_INTSTS[8])) is set to high when time-out counter counts to 5. The counter will keep counting.
5. Time-out counter is reset to 0 when time-out function is disabled.

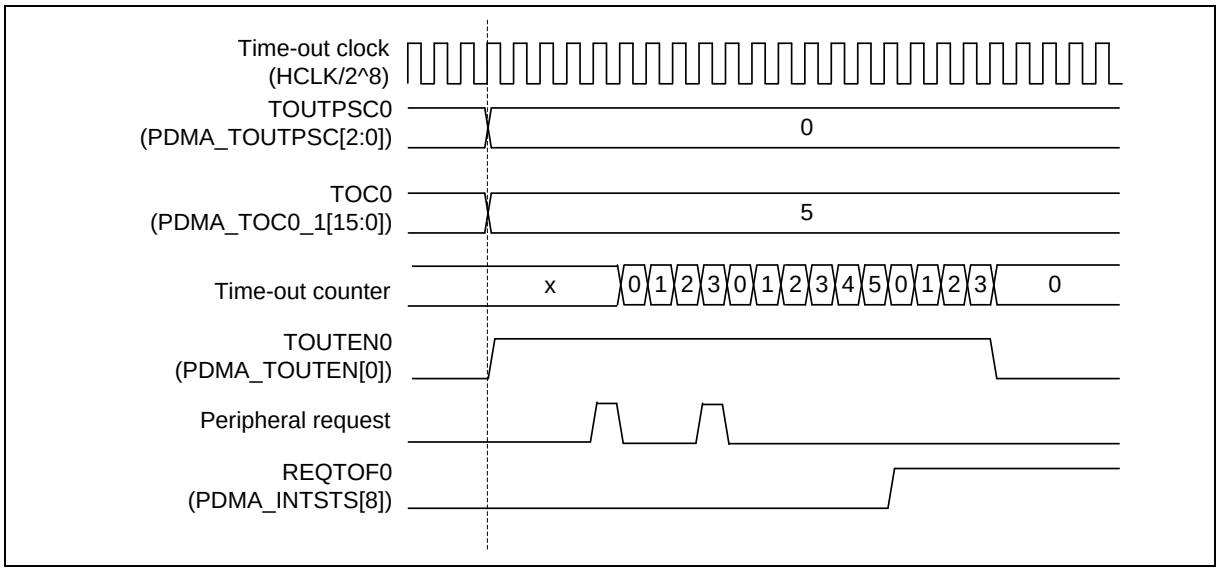


Figure 6.8-7 Example of PDMA Channel 0 Time-out Counter Operation

6.8.5.5 Stride Function

The PDMA supports six channels channel 0 to channel 5 with stride function. The stride function can transfer data from one address to another address and support block transfer with stride. When operating with stride function, the transfer address can be fixed or incremented successively.

Set STRIDEEN (PDMA_DSCTn_CTL[15]) to enable the stride function, and then write a valid source address to the PDMA_DSCTn_SA register and a source address offset count to SASOL (PDMA_ASOCRn[15:0]) register, a destination address to the PDMA_DSCTn_DA register and a destination address offset count to DASOL (PDMA_ASOCRn[31:16]), and a transfer count to the TXCNT (PDMA_DSCTn_CTL[31:16]) and a stride transfer count to STC (PDMA_STCRn[15:0]). Next, trigger the SWREQn (PDMA_SWREQ[5:0]). The PDMA will start and then stop the transfer after TXCNT (PDMA_DSCTn_CTL[31:16]) counts down to 0. Figure 6.8-8 shows the block transfer relationship between source memory and destination memory. The stride function also supports peripheral to memory or memory to peripheral transfer.

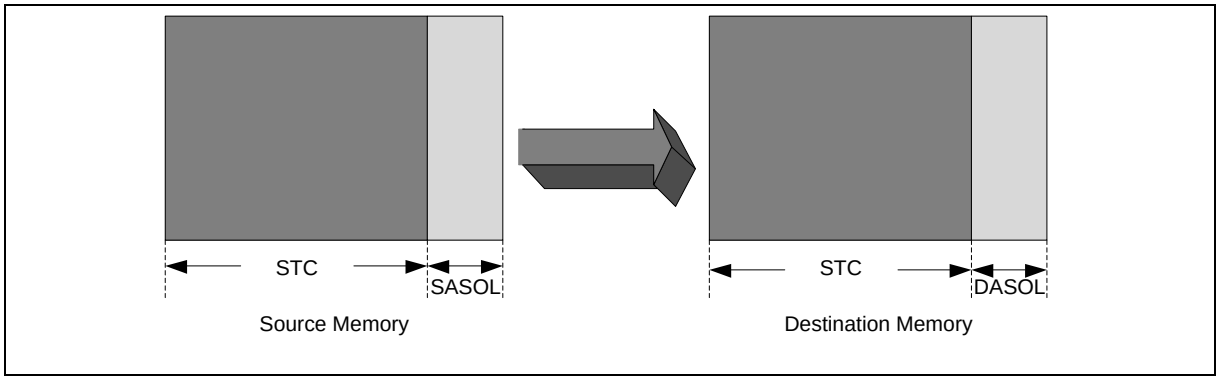


Figure 6.8-8 Stride Function Block Transfer

6.8.5.6 Enhanced Stride Function

To improve the stride function for DSP, image processing and other applications, channel 0 and channel 1 support interval count and repeat count. Interval count is used to decide the interval between each block transfer. SAICNT (PDMA_AICTLn[15:0]) is for source address interval count, and DAICNT (PDMA_AICTLn[31:16]) is for destination address interval count. Repeat count is used to decide repeat times of block transfer with only one trigger. If block transfer repeat RCNT (PDMA_RCNTn[15:0]) times, transfer is stopped. Figure 6.8-9 shows block transfer with interval count = 3 and repeat count = 2.

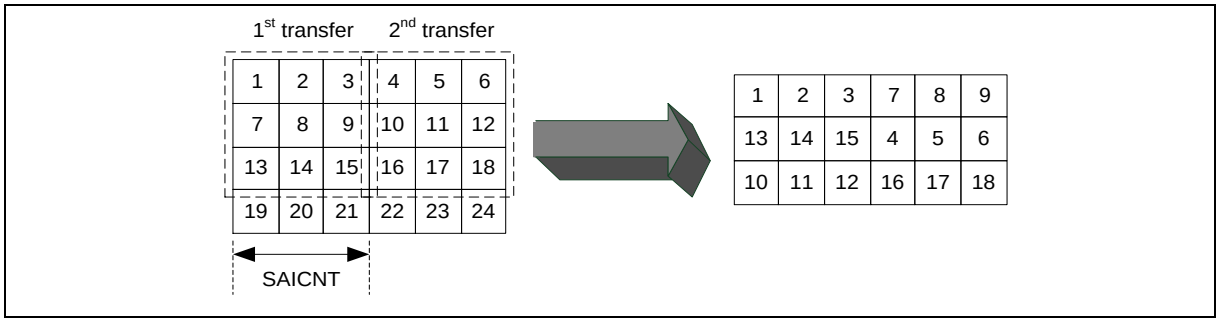


Figure 6.8-9 Block Transfer with interval =3 and repeat count =2

6.8.5.7 Security Policy

There are two PDMA controllers, PDMA0 is secure PDMA and PDMA1 can be configured as secure or non-secure PDMA. Secure PDMA is only served as secure master to access secure or non-secure region. If non-secure master requests secure PDMA to access secure region, the PDMA will return fault signal. Non-secure PDMA can be served as both secure and non-secure master, but it can only access non-secure region. For memory access policy, please refer to Secure Configuration Unit (SCU) chapter.

6.8.6 Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
PDMA Base Address: **PDMAx_BA = 0x4000_8000+(0x1_0000*x)** **x=0,1** **PDMA1 non-secure base address is PDMA1_BA+0x1000_0000**				
PDMAx_DSCTn_CTL **n = 0,1..7**	PDMAx_BA+0x10*n	R/W	Descriptor Table Control Register of PDMA Channel n	0xXXXX_XXXX
PDMAx_DSCTn_SA **n = 0,1..7**	PDMAx_BA+0x0004+0x10*n	R/W	Source Address Register of PDMA Channel n	0xXXXX_XXXX
PDMAx_DSCTn_DA **n = 0,1..7**	PDMAx_BA+0x0008+0x10*n	R/W	Destination Address Register of PDMA Channel n	0xXXXX_XXXX
PDMAx_DSCTn_NEXT **n = 0,1..7**	PDMAx_BA+0x000c+0x10*n	R/W	Next Scatter-gather Descriptor Table Offset Address of PDMA Channel n	0xXXXX_XXXX
PDMAx_CURSCATn **n = 0,1..7**	PDMAx_BA+0x0080+0x004*n	R	Current Scatter-gather Descriptor Table Address of PDMA Channel n	0xXXXX_XXXX
PDMAx_CHCTL	PDMAx_BA+0x400	R/W	PDMA Channel Control Register	0x0000_0000
PDMAx_PAUSE	PDMAx_BA+0x404	W	PDMA Transfer Pause Control Register	0x0000_0000
PDMAx_SWREQ	PDMAx_BA+0x408	W	PDMA Software Request Register	0x0000_0000
PDMAx_TRGSTS	PDMAx_BA+0x40C	R	PDMA Channel Request Status Register	0x0000_0000
PDMAx_PRISET	PDMAx_BA+0x410	R/W	PDMA Fixed Priority Setting Register	0x0000_0000
PDMAx_PRICLR	PDMAx_BA+0x414	W	PDMA Fixed Priority Clear Register	0x0000_0000
PDMAx_INTEN	PDMAx_BA+0x418	R/W	PDMA Interrupt Enable Register	0x0000_0000
PDMAx_INTSTS	PDMAx_BA+0x41C	R/W	PDMA Interrupt Status Register	0x0000_0000
PDMAx_ABTSTS	PDMAx_BA+0x420	R/W	PDMA Channel Read/Write Target Abort Flag Register	0x0000_0000
PDMAx_TDSTS	PDMAx_BA+0x424	R/W	PDMA Channel Transfer Done Flag Register	0x0000_0000
PDMAx_ALIGN	PDMAx_BA+0x428	R/W	PDMA Transfer Alignment Status Register	0x0000_0000
PDMAx_TACTSTS	PDMAx_BA+0x42C	R	PDMA Transfer Active Flag Register	0x0000_0000
PDMAx_TOUTPSC	PDMAx_BA+0x430	R/W	PDMA Time-out Prescaler Register	0x0000_0000
PDMAx_TOUTEN	PDMAx_BA+0x434	R/W	PDMA Time-out Enable Register	0x0000_0000
PDMAx_TOUTIEN	PDMAx_BA+0x438	R/W	PDMA Time-out Interrupt Enable Register	0x0000_0000
PDMAx_SCATBA	PDMAx_BA+0x43C	R/W	PDMA Scatter-gather Descriptor Table Base Address Register	0x2000_0000
PDMAx_TOC0_1	PDMAx_BA+0x440	R/W	PDMA Time-out Counter Ch0 and Ch1 Register	0x0000_0000
PDMAx_CHRST	PDMAx_BA+0x460	R/W	PDMA Channel Reset Register	0x0000_0000

PDMAx_REQSEL0_3	PDMAx_BA+0x480	R/W	PDMA Request Source Select Register 0	0x0000_0000
PDMAx_REQSEL4_7	PDMAx_BA+0x484	R/W	PDMA Request Source Select Register 1	0x0000_0000
PDMAx_STCR0	PDMAx_BA+0x500	R/W	Stride Transfer Count Register of PDMA Channel 0	0x0000_0000
PDMAx_ASOCR0	PDMAx_BA+0x504	R/W	Address Stride Offset Register of PDMA Channel 0	0x0000_0000
PDMAx_STCR1	PDMAx_BA+0x508	R/W	Stride Transfer Count Register of PDMA Channel 1	0x0000_0000
PDMAx_ASOCR1	PDMAx_BA+0x50C	R/W	Address Stride Offset Register of PDMA Channel 1	0x0000_0000
PDMAx_STCR2	PDMAx_BA+0x510	R/W	Stride Transfer Count Register of PDMA Channel 2	0x0000_0000
PDMAx_ASOCR2	PDMAx_BA+0x514	R/W	Address Stride Offset Register of PDMA Channel 2	0x0000_0000
PDMAx_STCR3	PDMAx_BA+0x518	R/W	Stride Transfer Count Register of PDMA Channel 3	0x0000_0000
PDMAx_ASOCR3	PDMAx_BA+0x51C	R/W	Address Stride Offset Register of PDMA Channel 3	0x0000_0000
PDMAx_STCR4	PDMAx_BA+0x520	R/W	Stride Transfer Count Register of PDMA Channel 4	0x0000_0000
PDMAx_ASOCR4	PDMAx_BA+0x524	R/W	Address Stride Offset Register of PDMA Channel 4	0x0000_0000
PDMAx_STCR5	PDMAx_BA+0x528	R/W	Stride Transfer Count Register of PDMA Channel 5	0x0000_0000
PDMAx_ASOCR5	PDMAx_BA+0x52C	R/W	Address Stride Offset Register of PDMA Channel 5	0x0000_0000
PDMAx_AICTL0	PDMAx_BA + 0x600	R/W	Address Interval Control Register of PDMA Channel 0	0x0000_0000
PDMAx_RCNT0	PDMAx_BA + 0x604	R/W	Repeat Count Register of PDMA Channel 0	0x0000_0000
PDMAx_AICTL1	PDMAx_BA + 0x608	R/W	Address Interval Control Register of PDMA Channel 1	0x0000_0000
PDMAx_RCNT1	PDMAx_BA + 0x60C	R/W	Repeat Count Register of PDMA Channel 1	0x0000_0000

6.8.7 Register Description

Descriptor Table Control Register (PDMAx_DSCTn_CTL)

Register	Offset	R/W	Description	Reset Value
PDMAx_DSCTn_CTL	PDMAx_BA+0x10*n	R/W	Descriptor Table Control Register of PDMA Channel n	0xXXXX_XXXX

31	30	29	28	27	26	25	24
TXCNT							
23	22	21	20	19	18	17	16
TXCNT							
15	14	13	12	11	10	9	8
STRIDEEN	Reserved	TXWIDTH		DAINC		SAINC	
7	6	5	4	3	2	1	0
TBINTDIS	BURSIZE			Reserved	TXTYPE	OPMODE	

Bits	Description	
[31:16]	**TXCNT**	**Transfer Count** The TXCNT represents the required number of PDMA transfer, the real transfer count is (TXCNT + 1); The maximum transfer count is 65536, every transfer may be byte, half-word or word that is dependent on TXWIDTH field. **Note:** When PDMA finishes each transfer data, this field will be decreased immediately.
[15]	**STRIDEEN**	**Stride Mode Enable Bit** 0 = Stride transfer mode Disabled. 1 = Stride transfer mode Enabled.
[14]	**Reserved**	Reserved.
[13:12]	**TXWIDTH**	**Transfer Width Selection** This field is used for transfer width. 00 = One byte (8 bit) is transferred for every operation. 01= One half-word (16 bit) is transferred for every operation. 10 = One word (32-bit) is transferred for every operation. 11 = Reserved. **Note:** The PDMA transfer source address (PDMA_DSCTn_SA) and PDMA transfer destination address (PDMA_DSCTn_DA) should be alignment under the TXWIDTH selection
[11:10]	**DAINC**	**Destination Address Increment** This field is used to set the destination address increment size. 11 = No increment (fixed address). Others = Increment and size is depended on TXWIDTH selection. **Note:** The fixed address function does not support in memory to memory transfer type.
[9:8]	**SAINC**	**Source Address Increment** This field is used to set the source address increment size. 11 = No increment (fixed address).

Bits	Description	
		Others = Increment and size is depended on TXWIDTH selection. **Note:** The fixed address function does not support in memory to memory transfer type.
[7]	**TBINTDIS**	**Table Interrupt Disable Bit** This field can be used to decide whether to enable table interrupt or not. If the TBINTDIS bit is enabled it will not generates TDIFn(PDMA_TDSTS[7:0]) when PDMA controller finishes transfer task. 0 = Table interrupt Enabled. 1 = Table interrupt Disabled. **Note:** This function only for Scatter-gather mode.
[6:4]	**BURSIZE**	**Burst Size** This field is used for peripheral to determine the burst size or used for determine the re-arbitration size. 000 = 128 Transfers. 001 = 64 Transfers. 010 = 32 Transfers. 011 = 16 Transfers. 100 = 8 Transfers. 101 = 4 Transfers. 110 = 2 Transfers. 111 = 1 Transfers. **Note:** This field is only useful in burst transfer type.
[3]	**Reserved**	Reserved.
[2]	**TXTYPE**	**Transfer Type** 0 = Burst transfer type. 1 = Single transfer type.
[1:0]	**OPMODE**	**PDMA Operation Mode Selection** 00 = Idle state: Channel is stopped or this table is complete, when PDMA finish channel table task, OPMODE will be cleared to idle state automatically. 01 = Basic mode: The descriptor table only has one task. When this task is finished, the PDMA_INTSTS[1] will be asserted. 10 = Scatter-gather mode: When operating in this mode, user must give the next descriptor table address in PDMA_DSCTn_NEXT register; PDMA controller will ignore this task, then load the next task to execute. 11 = Reserved. **Note:** Before filling new transfer task in the Descriptor Table, user must check the PDMA_INTSTS[1] to make sure the current task is complete.

Start Source Address Register (PDMAx_DSCTn_SA)

Register	Offset	R/W	Description	Reset Value
PDMAx_DSCTn_SA	PDMAx_BA+0x0004+0x10*n	R/W	Source Address Register of PDMA Channel n	0xXXXX_XXXX

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

Bits	Description	
[31:0]	**SA**	**PDMA Transfer Source Address** This field indicates a 32-bit source address of PDMA controller.

Destination Address Register (PDMAx_DSCTn_DA)

Register	Offset	R/W	Description	Reset Value
PDMAx_DSCTn_DA	PDMAx_BA+0x0008+0x10*n	R/W	Destination Address Register of PDMA Channel n	0xXXXX_XXXX

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

Bits	Description	
[31:0]	DA	**PDMA Transfer Destination Address** This field indicates a 32-bit destination address of PDMA controller.

Next Scatter-gather Descriptor Table Offset Address (PDMAx_DSCTn_NEXT)

Register	Offset	R/W	Description	Reset Value
PDMAx_DSCTn_NEXT	PDMAx_BA+0x000c+0x10*n	R/W	Next Scatter-gather Descriptor Table Offset Address of PDMA Channel n	0xXXXX_XXXX

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

Bits	Description	
[31:16]	**EXENEXT**	**PDMA Execution Next Descriptor Table Offset** This field indicates the offset of next descriptor table address of current execution descriptor table in system memory. **Note:** Write operation is useless in this field.
[15:0]	**NEXT**	**PDMA Next Descriptor Table Offset** This field indicates the offset of the next descriptor table address in system memory. **Write Operation:** If the system memory based address is 0x2000_0000 (PDMA_SCATBA), and the next descriptor table is start from 0x2000_0100, then this field must fill in 0x0100. **Read Operation:** When operating in Scatter-gather mode, the last two bits NEXT[1:0] will become reserved, and indicate the first next address of system memory. **Note 1:** The descriptor table address must be word boundary. **Note 2:** Before filling transfer task in the descriptor table, user must check if the descriptor table is complete.

Current Scatter-gather Descriptor Table Address (PDMAx_CURSCATn)

Register	Offset	R/W	Description	Reset Value
PDMAx_CURSCATn	PDMAx_BA+0x0080+0x004*n	R	Current Scatter-gather Descriptor Table Address of PDMA Channel n	0xXXXX_XXXX

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

Bits	Description	
[31:0]	**CURADDR**	**PDMA Current Description Address (Read Only)** This field indicates a 32-bit current external description address of PDMA controller. **Note:** This field is read only and used for Scatter-gather mode only to indicate the current external description address.

Channel Control Register (PDMAx_CHCTL)

Register	Offset	R/W	Description	Reset Value
PDMAx_CHCTL	PDMAx_BA+0x400	R/W	PDMA Channel Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CHEN7	**CHEN6**	**CHEN5**	**CHEN4**	**CHEN3**	**CHEN2**	**CHEN1**	**CHEN0**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[n] n=0,1..7	**CHENn**	**PDMA Channel Enable Bits** Set this bit to 1 to enable PDMAn operation. Channel cannot be active if it is not set as enabled. 0 = PDMA channel [n] Disabled. 1 = PDMA channel [n] Enabled. **Note:** Setting the corresponding bit of PDMA_PAUSE or PDMA_CHRST register will also clear this bit.

PDMA Transfer Pause Control Register (PDMAx_PAUSE)

Register	Offset	R/W	Description	Reset Value
PDMAx_PAUSE	PDMAx_BA+0x404	W	PDMA Transfer Pause Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PAUSE7	**PAUSE6**	**PAUSE5**	**PAUSE4**	**PAUSE3**	**PAUSE2**	**PAUSE1**	**PAUSE0**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[n] n=0,1..7	**PAUSEn**	**PDMA Channel n Transfer Pause Control (Write Only)** User can set PAUSEn bit field to pause the PDMA transfer. When user sets PAUSEn bit, the PDMA controller will pause the on-going transfer, then clear the channel enable bit CHENn(PDMA_CHCTL [n], n=0,1.. 7) and clear request active flag(PDMA_TRGSTS[n:0], n=0,1.. 7). If the paused channel is re-enabled again, the remaining transfers will be processed. 0 = No effect. 1 = Pause PDMA channel n transfer.

PDMA Software Request Register (PDMAx_SWREQ)

Register	Offset	R/W	Description	Reset Value
PDMAx_SWREQ	PDMAx_BA+0x408	W	PDMA Software Request Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SWREQ7	**SWREQ6**	**SWREQ5**	**SWREQ4**	**SWREQ3**	**SWREQ2**	**SWREQ1**	**SWREQ0**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[n] n=0,1..7	**SWREQn**	**PDMA Software Request (Write Only)** Set this bit to 1 to generate a software request to PDMA [n]. 0 = No effect. 1 = Generate a software request. **Note 1:** User can read PDMA_TRGSTS register to know which channel is on active. Active flag may be triggered by software request or peripheral request. **Note 2:** If user does not enable corresponding PDMA channel, the software request will be ignored.

PDMA Channel Request Status Register (PDMAx_TRGSTS)

Register	Offset	R/W	Description	Reset Value
PDMAx_TRGSTS	PDMAx_BA+0x40C	R	PDMA Channel Request Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
REQSTS7	**REQSTS6**	**REQSTS5**	**REQSTS4**	**REQSTS3**	**REQSTS2**	**REQSTS1**	**REQSTS0**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[n] n=0,1..7	**REQSTSn**	**PDMA Channel Request Status (Read Only)** This flag indicates whether channel[n] have a request or not, no matter request from software or peripheral. When PDMA controller finishes channel transfer, this bit will be cleared automatically. 0 = PDMA Channel n has no request. 1 = PDMA Channel n has a request. **Note:** If user pauses or resets each PDMA transfer by setting PDMA_PAUSE or PDMA_CHRST register respectively, this bit will be cleared automatically after finishing the current transfer.

PDMA Fixed Priority Setting Register (PDMAx_PRISET)

Register	Offset	R/W	Description	Reset Value
PDMAx_PRISET	PDMAx_BA+0x410	R/W	PDMA Fixed Priority Setting Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FPRISET7	**FPRISET6**	**FPRISET5**	**FPRISET4**	**FPRISET3**	**FPRISET2**	**FPRISET1**	**FPRISET0**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[n] n=0,1..7	**FPRISETn**	**PDMA Fixed Priority Setting** Set this bit to 1 to enable fixed priority level. **Write Operation:** 0 = No effect. 1 = Set PDMA channel [n] to fixed priority channel. **Read Operation:** 0 = Corresponding PDMA channel is round-robin priority. 1 = Corresponding PDMA channel is fixed priority. **Note:** This field is only set to fixed priority. To clear fixed priority, use PDMA_PRICLR register.

PDMA Fix Priority Clear Register (PDMAx_PRICLR)

Register	Offset	R/W	Description	Reset Value
PDMAx_PRICLR	PDMAx_BA+0x414	W	PDMA Fixed Priority Clear Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FPRICLR7	**FPRICLR6**	**FPRICLR5**	**FPRICLR4**	**FPRICLR3**	**FPRICLR2**	**FPRICLR1**	**FPRICLR0**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[n] n=0,1..7	**FPRICLRn**	**PDMA Fixed Priority Clear Bits (Write Only)** Set this bit to 1 to clear fixed priority level. 0 = No effect. 1 = Clear PDMA channel [n] fixed priority setting. **Note:** User can read PDMA_PRISET register to know the channel priority.

PDMA Interrupt Enable Register (PDMAx_INTEN)

Register	Offset	R/W	Description	Reset Value
PDMAx_INTEN	PDMAx_BA+0x418	R/W	PDMA Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INTEN7	**INTEN6**	**INTEN5**	**INTEN4**	**INTEN3**	**INTEN2**	**INTEN1**	**INTEN0**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[n] n=0,1..7	**INTENn**	**PDMA Interrupt Enable Bits** This field is used to enable PDMA channel[n] interrupt. 0 = PDMA channel n interrupt Disabled. 1 = PDMA channel n interrupt Enabled. **Note:** The interrupt flag is time-out, abort, transfer done and align.

PDMA Interrupt Status Register (PDMAx_INTSTS)

Register	Offset	R/W	Description	Reset Value
PDMAx_INTSTS	PDMAx_BA+0x41C	R/W	PDMA Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved						**REQTOF1**	**REQTOF0**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**ALIGNF**	**TDIF**	**ABTIF**

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**REQTOF1**	**Request Time-out Flag for Channel 1** This flag indicates that PDMA controller has waited peripheral request for a period defined by TOC1(PDMA_TOC0_1[31:16]). 0 = No request time-out. 1 = Peripheral request time-out. **Note 1:** Please disable time-out function before clearing this bit. **Note 2:** User can write 1 to clear this bit.
[8]	**REQTOF0**	**Request Time-out Flag for Channel 0** This flag indicates that PDMA controller has waited peripheral request for a period defined by TOC0(PDMA_TOC0_1[15:0]. 0 = No request time-out. 1 = Peripheral request time-out. **Note 1:** Please disable time-out function before clearing this bit. **Note 2:** User can write 1 to clear this bit.
[7:3]	**Reserved**	Reserved.
[2]	**ALIGNF**	**Transfer Alignment Interrupt Flag (Read Only)** 0 = PDMA channel source address and destination address both follow transfer width setting. 1 = PDMA channel source address or destination address is not follow transfer width setting.
[1]	**TDIF**	**Transfer Done Interrupt Flag (Read Only)** This bit indicates that PDMA controller has finished transmission; User can read PDMA_TDSTS register to indicate which channel finished transfer. 0 = Not finished yet. 1 = PDMA channel has finished transmission.
[0]	**ABTIF**	**PDMA Read/Write Target Abort Interrupt Flag (Read Only)** This bit indicates that PDMA has target abort error; Software can read PDMA_ABTSTS register to find

Bits	Description	
		which channel has target abort error. 0 = No AHB bus ERROR response received. 1 = AHB bus ERROR response received.

PDMA Channel Read/Write Target Abort Flag Register (PDMAx_ABTSTS)

Register	Offset	R/W	Description	Reset Value
PDMAx_ABTSTS	PDMAx_BA+0x420	R/W	PDMA Channel Read/Write Target Abort Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ABTIF7	**ABTIF6**	**ABTIF5**	**ABTIF4**	**ABTIF3**	**ABTIF2**	**ABTIF1**	**ABTIF0**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[n] n=0,1..7	**ABTIFn**	**PDMA Read/Write Target Abort Interrupt Status Flag** This bit indicates which PDMA controller has target abort error. 0 = No AHB bus ERROR response received when channel n transfer. 1 = AHB bus ERROR response received when channel n transfer. **Note 1:** If channel n target abort, REQSRCn should set 0 to disable peripheral request. **Note 2:** User can write 1 to clear this bit.

PDMA Channel Transfer Done Flag Register (PDMAx_TDSTS)

Register	Offset	R/W	Description	Reset Value
PDMAx_TDSTS	PDMAx_BA+0x424	R/W	PDMA Channel Transfer Done Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TDIF7	**TDIF6**	**TDIF5**	**TDIF4**	**TDIF3**	**TDIF2**	**TDIF1**	**TDIF0**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[n] n=0,1..7	**TDIFn**	**Transfer Done Flag** This bit indicates whether PDMA controller channel transfer has been finished or not. 0 = PDMA channel transfer has not finished. 1 = PDMA channel has finished transmission. **Note:** User can write 1 to clear these bits.

PDMA Transfer Alignment Status Register (PDMAx_ALIGN)

Register	Offset	R/W	Description	Reset Value
PDMAx_ALIGN	PDMAx_BA+0x428	R/W	PDMA Transfer Alignment Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ALIGN7	**ALIGN6**	**ALIGN5**	**ALIGN4**	**ALIGN3**	**ALIGN2**	**ALIGN1**	**ALIGN0**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[n] n=0,1..7	**ALIGNn**	**Transfer Alignment Flag** This bit indicates whether source and destination address both follow transfer width setting. 0 = PDMA channel source address and destination address both follow transfer width setting. 1 = PDMA channel source address or destination address is not follow transfer width setting. **Note:** User can write 1 to clear these bits.

PDMA Transfer Active Flag Register (PDMAx_TACTSTS)

Register	Offset	R/W	Description	Reset Value
PDMAx_TACTSTS	PDMAx_BA+0x42C	R	PDMA Transfer Active Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXACTF7	**TXACTF6**	**TXACTF5**	**TXACTF4**	**TXACTF3**	**TXACTF2**	**TXACTF1**	**TXACTF0**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[n] n=0,1..7	**TXACTFn**	**Transfer on Active Flag (Read Only)** This bit indicates which PDMA channel is in active. 0 = PDMA channel is finished. 1 = PDMA channel is active.

PDMA Time-out Prescaler Register (PDMAx_TOUTPSC)

Register	Offset	R/W	Description	Reset Value
PDMAx_TOUTPSC	PDMAx_BA+0x430	R/W	PDMA Time-out Prescaler Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**TOUTPSC1**			**Reserved**	**TOUTPSC0**		

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6:4]	**TOUTPSC1**	**PDMA Channel 1 Time-out Clock Source Prescaler Bits** 000 = PDMA channel 1 time-out clock source is HCLK/2^8. 001 = PDMA channel 1 time-out clock source is HCLK/2^9. 010 = PDMA channel 1 time-out clock source is HCLK/2^{10}. 011 = PDMA channel 1 time-out clock source is HCLK/2^{11}. 100 = PDMA channel 1 time-out clock source is HCLK/2^{12}. 101 = PDMA channel 1 time-out clock source is HCLK/2^{13}. 110 = PDMA channel 1 time-out clock source is HCLK/2^{14}. 111 = PDMA channel 1 time-out clock source is HCLK/2^{15}.
[3]	**Reserved**	Reserved.
[2:0]	**TOUTPSC0**	**PDMA Channel 0 Time-out Clock Source Prescaler Bits** 000 = PDMA channel 0 time-out clock source is HCLK/2^8. 001 = PDMA channel 0 time-out clock source is HCLK/2^9. 010 = PDMA channel 0 time-out clock source is HCLK/2^{10}. 011 = PDMA channel 0 time-out clock source is HCLK/2^{11}. 100 = PDMA channel 0 time-out clock source is HCLK/2^{12}. 101 = PDMA channel 0 time-out clock source is HCLK/2^{13}. 110 = PDMA channel 0 time-out clock source is HCLK/2^{14}. 111 = PDMA channel 0 time-out clock source is HCLK/2^{15}.

PDMA Time-out Enable Register (PDMAx_TOUTEN)

Register	Offset	R/W	Description	Reset Value
PDMAx_TOUTEN	PDMAx_BA+0x434	R/W	PDMA Time-out Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**TOUTEN1**	**TOUTEN0**

Bits	Description	
[31:2]	**Reserved**	Reserved.
[n] n=0,1	**TOUTENn**	**PDMA Time-out Enable Bits** 0 = PDMA Channel n time-out function Disabled. 1 = PDMA Channel n time-out function Enabled.

PDMA Time-out Interrupt Enable Register (PDMAx_TOUTIEN)

Register	Offset	R/W	Description	Reset Value
PDMAx_TOUTIEN	PDMAx_BA+0x438	R/W	PDMA Time-out Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**TOUTIEN1**	**TOUTIEN0**

Bits	Description	
[31:2]	**Reserved**	Reserved.
[n] n=0,1	**TOUTIENn**	**PDMA Time-out Interrupt Enable Bits** 0 = PDMA Channel n time-out interrupt Disabled. 1 = PDMA Channel n time-out interrupt Enabled.

PDMA Scatter-gather Descriptor Table Base Address Register (PDMAx_SCATBA)

Register	Offset	R/W	Description	Reset Value
PDMAx_SCATBA	PDMAx_BA+0x43C	R/W	PDMA Scatter-gather Descriptor Table Base Address Register	0x2000_0000

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

Bits	Description	
[31:16]	**SCATBA**	**PDMA Scatter-gather Descriptor Table Address** In Scatter-gather mode, this is the base address for calculating the next link - list address. The next link address equation is Next Link Address = PDMA_SCATBA + PDMA_DSCTn_NEXT. **Note:** Only useful in Scatter-gather mode.
[15:0]	**Reserved**	Reserved.

PDMA Time-out Period Counter Register 0 (PDMAx_TOC0_1)

Register	Offset	R/W	Description	Reset Value
PDMAx_TOC0_1	PDMAx_BA+0x440	R/W	PDMA Time-out Counter Ch0 and Ch1 Register	0x0000_0000

31	30	29	28	27	26	25	24
TOC1							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TOC1							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TOC0							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TOC0							

Bits	Description	
[31:16]	**TOC1**	**Time-out Counter for Channel 1** This controls the period of time-out function for channel 1. The calculation unit is based on TOUTPSC1 (PDMA_TOUTPSC[6:4]) clock. The example of time-out period can refer TOC0 bit description.
[15:0]	**TOC0**	**Time-out Counter for Channel 0** This controls the period of time-out function for channel 0. The calculation unit is based on TOUTPSC0 (PDMA_TOUTPSC[2:0]) clock. Time-out period = (Period of time-out clock) * (16-bit TOCn), n = 0,1.

PDMA Channel Reset Register (PDMAx_CHRST)

Register	Offset	R/W	Description	Reset Value
PDMAx_CHRST	PDMAx_BA+0x460	R/W	PDMA Channel Reset Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CH7RST	**CH6RST**	**CH5RST**	**CH4RST**	**CH3RST**	**CH2RST**	**CH1RST**	**CH0RST**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[n] n=0,1..7	**CHnRST**	**Channel n Reset** 0 = Corresponding channel n is not reset. 1 = Corresponding channel n is reset.

PDMA Request Source Select Register 0 (PDMAx_REQSEL0_3)

Register	Offset	R/W	Description	Reset Value
PDMAx_REQSEL0_3	PDMAx_BA+0x480	R/W	PDMA Request Source Select Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	**REQSRC3**						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved	**REQSRC2**						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved	**REQSRC1**						
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**REQSRC0**						

Bits	Description	
[31]	**Reserved**	Reserved.
[30:24]	**REQSRC3**	**Channel 3 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 3. User can configure the peripheral setting by REQSRC3. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[23]	**Reserved**	Reserved.
[22:16]	**REQSRC2**	**Channel 2 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 2. User can configure the peripheral setting by REQSRC2. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[15]	**Reserved**	Reserved.
[14:8]	**REQSRC1**	**Channel 1 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 1. User can configure the peripheral setting by REQSRC1. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[7]	**Reserved**	Reserved.
[6:0]	**REQSRC0**	**Channel 0 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 0. User can configure the peripheral by setting REQSRC0. 0 = Disable PDMA peripheral request. 1 = Reserved. 2 = Channel connects to USB_TX. 3 = Channel connects to USB_RX.

Bits	Description	
		4 = Channel connects to UART0_TX. 5 = Channel connects to UART0_RX. 6 = Channel connects to UART1_TX. 7 = Channel connects to UART1_RX. 8 = Channel connects to UART2_TX. 9 = Channel connects to UART2_RX. 10=Channel connects to UART3_TX. 11 = Channel connects to UART3_RX. 12 = Channel connects to UART4_TX. 13 = Channel connects to UART4_RX. 14 = Channel connects to UART5_TX. 15 = Channel connects to UART5_RX. 16 = Channel connects to USCI0_TX. 17 = Channel connects to USCI0_RX. 18 = Channel connects to USCI1_TX. 19 = Channel connects to USCI1_RX. 20 = Channel connects to QSPI0_TX. 21 = Channel connects to QSPI0_RX. 22 = Channel connects to SPI0_TX. 23 = Channel connects to SPI0_RX. 24 = Channel connects to SPI1_TX. 25 = Channel connects to SPI1_RX. 26 = Channel connects to SPI2_TX. 27 = Channel connects to SPI2_RX. 28 = Channel connects to SPI3_TX. 29 = Channel connects to SPI3_RX. 30 = Channel connects to ADC_RX. 31 = Reserved. 32 = Channel connects to EPWM0_P1_RX. 33 = Channel connects to EPWM0_P2_RX. 34 = Channel connects to EPWM0_P3_RX. 35 = Channel connects to EPWM1_P1_RX. 36 = Channel connects to EPWM1_P2_RX. 37 = Channel connects to EPWM1_P3_RX. 38 = Channel connects to I2C0_TX. 39 = Channel connects to I2C0_RX. 40 = Channel connects to I2C1_TX. 41 = Channel connects to I2C1_RX. 42 = Channel connects to I2C2_TX. 43 = Channel connects to I2C2_RX. 44 = Channel connects to I2S0_TX. 45 = Channel connects to I2S0_RX. 46 = Channel connects to TMR0. 47 = Channel connects to TMR1. 48 = Channel connects to TMR2. 49 = Channel connects to TMR3.

Bits	Description	
		50 = Channel connects to TMR4. 51 = Channel connects to TMR5. 52 = Channel connects to DAC0_TX. 53 = Channel connects to DAC1_TX. 54 = Channel connects to EPWM0_CH0_TX. 55 = Channel connects to EPWM0_CH1_TX. 56 = Channel connects to EPWM0_CH2_TX. 57 = Channel connects to EPWM0_CH3_TX. 58 = Channel connects to EPWM0_CH4_TX. 59 = Channel connects to EPWM0_CH5_TX. 60 = Channel connects to EPWM1_CH0_TX. 61 = Channel connects to EPWM1_CH1_TX. 62 = Channel connects to EPWM1_CH2_TX. 63 = Channel connects to EPWM1_CH3_TX. 64 = Channel connects to EPWM1_CH4_TX. 65 = Channel connects to EPWM1_CH5_TX. Others = Reserved. **Note 1:** A peripheral cannot be assigned to two channels at the same time. **Note 2:** This field is useless when transfer between memory and memory.

PDMA Request Source Select Register 1 (PDMAx_REQSEL4_7)

Register	Offset	R/W	Description	Reset Value
PDMAx_REQSEL4_7	PDMAx_BA+0x484	R/W	PDMA Request Source Select Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	**REQSRC7**						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved	**REQSRC6**						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved	**REQSRC5**						
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**REQSRC4**						

Bits	Description	
[31]	**Reserved**	Reserved.
[30:24]	**REQSRC7**	**Channel 7 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 7. User can configure the peripheral setting by REQSRC7. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[23]	**Reserved**	Reserved.
[22:16]	**REQSRC6**	**Channel 6 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 6. User can configure the peripheral setting by REQSRC6. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[15]	**Reserved**	Reserved.
[14:8]	**REQSRC5**	**Channel 5 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 5. User can configure the peripheral setting by REQSRC5. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[7]	**Reserved**	Reserved.
[6:0]	**REQSRC4**	**Channel 4 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 4. User can configure the peripheral setting by REQSRC4. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.

PDMA Stride Transfer Count Register n (PDMAx_STCRn)

Register	Offset	R/W	Description	Reset Value
PDMAx_STCR0	PDMAx_BA+0x500	R/W	Stride Transfer Count Register of PDMA Channel 0	0x0000_0000
PDMAx_STCR1	PDMAx_BA+0x508	R/W	Stride Transfer Count Register of PDMA Channel 1	0x0000_0000
PDMAx_STCR2	PDMAx_BA+0x510	R/W	Stride Transfer Count Register of PDMA Channel 2	0x0000_0000
PDMAx_STCR3	PDMAx_BA+0x518	R/W	Stride Transfer Count Register of PDMA Channel 3	0x0000_0000
PDMAx_STCR4	PDMAx_BA+0x520	R/W	Stride Transfer Count Register of PDMA Channel 4	0x0000_0000
PDMAx_STCR5	PDMAx_BA+0x528	R/W	Stride Transfer Count Register of PDMA Channel 5	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**STC**	**PDMA Stride Transfer Count** The 16-bit register defines the stride transfer count of each row.

PDMA Address Stride Offset Control Register n (PDMAx_ASOCRn)

Register	Offset	R/W	Description	Reset Value
PDMAx_ASOCR0	PDMAx_BA+0x504	R/W	Address Stride Offset Register of PDMA Channel 0	0x0000_0000
PDMAx_ASOCR1	PDMAx_BA+0x50C	R/W	Address Stride Offset Register of PDMA Channel 1	0x0000_0000
PDMAx_ASOCR2	PDMAx_BA+0x514	R/W	Address Stride Offset Register of PDMA Channel 2	0x0000_0000
PDMAx_ASOCR3	PDMAx_BA+0x51C	R/W	Address Stride Offset Register of PDMA Channel 3	0x0000_0000
PDMAx_ASOCR4	PDMAx_BA+0x524	R/W	Address Stride Offset Register of PDMA Channel 4	0x0000_0000
PDMAx_ASOCR5	PDMAx_BA+0x52C	R/W	Address Stride Offset Register of PDMA Channel 5	0x0000_0000

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

Bits	Description	
[31:16]	**DASOL**	**PDMA Destination Address Stride Offset Length** The 16-bit register defines the destination address stride transfer offset count of each row.
[15:0]	**SASOL**	**PDMA Source Address Stride Offset Length** The 16-bit register defines the source address stride transfer offset count of each row.

PDMA Address Interval Control Register n (PDMAx_AICTLn)

Register	Offset	R/W	Description	Reset Value
PDMAx_AICTL0	PDMAx_BA + 0x600	R/W	Address Interval Control Register of PDMA Channel 0	0x0000_0000
PDMAx_AICTL1	PDMAx_BA + 0x608	R/W	Address Interval Control Register of PDMA Channel 1	0x0000_0000

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

Bits	Description	
[31:16]	**DAICNT**	**PDMA Destination Address Interval Count** The 16-bit register defines the destination address interval count of each row.
[15:0]	**SAICNT**	**PDMA Source Address Interval Count** The 16-bit register defines the source address interval count of each row.

PDMA Repeat Count Register n (PDMAx_RCNTn)

Register	Offset	R/W	Description	Reset Value
PDMAx_RCNT0	PDMAx_BA + 0x604	R/W	Repeat Count Register of PDMA Channel 0	0x0000_0000
PDMAx_RCNT1	PDMAx_BA + 0x60C	R/W	Repeat Count Register of PDMA Channel 1	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**RCNT**	**PDMA Repeat Count** The 16-bit register defines the repeat times of block transfer.

6.9 Timer Controller (TMR)

6.9.1 Overview

The timer controller includes six 32-bit timers, Timer0 ~ Timer5, allowing user to easily implement a timer control for applications. The timer can perform functions, such as frequency measurement, delay timing, clock generation, and event counting by external input pins, and interval measurement by external capture pins.

The timer controller also provides the PWM generator function.
In Timer0 ~ Timer3, each PWM generator supports two PWM output channels in independent mode and complementary mode. The output state of PWM output pin can be controlled by pin mask, polarity and break control, and dead-time generator.
In Timer4 and Timer5, each PWM generator supports only one PWM output channel. The output state of PWM output pin can be controlled by polarity control, output enable control and output channel select.

6.9.2 Features

6.9.2.1 Timer Function Features

- Six sets of 32-bit timers, Timer0 ~ Timer5, each timer having one 24-bit up counter and one 8-bit prescale counter
- Independent clock source for each timer
- Provides one-shot, periodic, toggle-output and continuous counting operation modes
- 24-bit up counter value is readable through CNT (TIMERx_CNT[23:0])
- Supports event counting function
- 24-bit capture value is readable through CAPDAT (TIMERx_CAP[23:0])
- Supports external capture event for interval measurement
- Supports capture event to reset 24-bit up counter
- Supports internal clock (HIRC, LIRC) and external clock (HXT, LXT) for capture event in Timer0 ~ Timer3
- Supports internal clock (HIRC, LIRC, MIRC) and external clock (HXT, LXT) for capture event in Timer4 and Timer5
- Supports chip wake-up from Idle/Power-down mode if a timer interrupt signal is generated
- Supports Timer0 ~ Timer3 time-out interrupt signal or capture interrupt signal to trigger EPWM, BPWM, EADC, DAC and PDMA function
- Supports Timer4 and Timer5 time-out interrupt signal or capture interrupt signal to trigger EADC and PDMA function
- Supports internal capture triggered while internal ACMP output signal transition
- Supports Inter-Timer trigger mode
- Supports event counting source from internal USB SOF signal

6.9.2.2 PWM Function Features

In the Timer0 ~ Timer3 PWM,

- Supports maximum clock frequency up to maximum PCLK
- Supports independent mode for PWM generator with two output channels

- Supports complementary mode for PWM generator with paired PWM output channel
  - 12-bit dead-time insertion with 12-bit prescale
- Supports 12-bit prescale from 1 to 4096
- Supports 16-bit PWM counter
  - Up, down and up-down count operation type
  - One-shot or auto-reload counter operation mode
- Supports 16-bit compare register and period register and double buffer for period register and compare register
- Supports mask function and tri-state enable for each PWM output pin
- Supports brake function
  - Brake source from pin, analog comparator and system safety events (clock failed, Brown-out detection, SRAM parity error and CPU lockup)
  - Brake pin noise filter control for brake source
  - Edge detect brake source to control brake state until brake status cleared
  - Level detect brake source to auto recover function after brake condition removed
- Supports interrupt on the following events:
  - PWM zero point, period point, up-count compared or down-count compared point events
  - Brake condition happened
- Supports trigger EADC on the following events:
  - PWM zero point, period, zero or period point, up-count compared or down-count compared point events

In the Timer4 and Timer5 PWM,

- Supports independent mode for PWM generator with one output channel
- Supports 16-bit PWM counter
  - Up count operation type
  - One-shot or auto-reload counter operation mode
- Supports 8-bit prescale from 1 to 256
- Supports 16-bit compare register and period register and double buffer for period register and compare register
- Supports tri-state enable and polarity control for each PWM selectable output channel
- Supports interrupt on the following events:
  - PWM period point, up-count compared point events
- Supports wake-up when interrupt occurs when clock source is LXT, LIRC or MIRC
- PWM can generator output in Power-down mode
- Supports trigger EADC and PDMA on the following events:
  - PWM period point and up-count compared point events

6.9.3 Block Diagram

The timer controller block diagram and clock control are shown as follows.

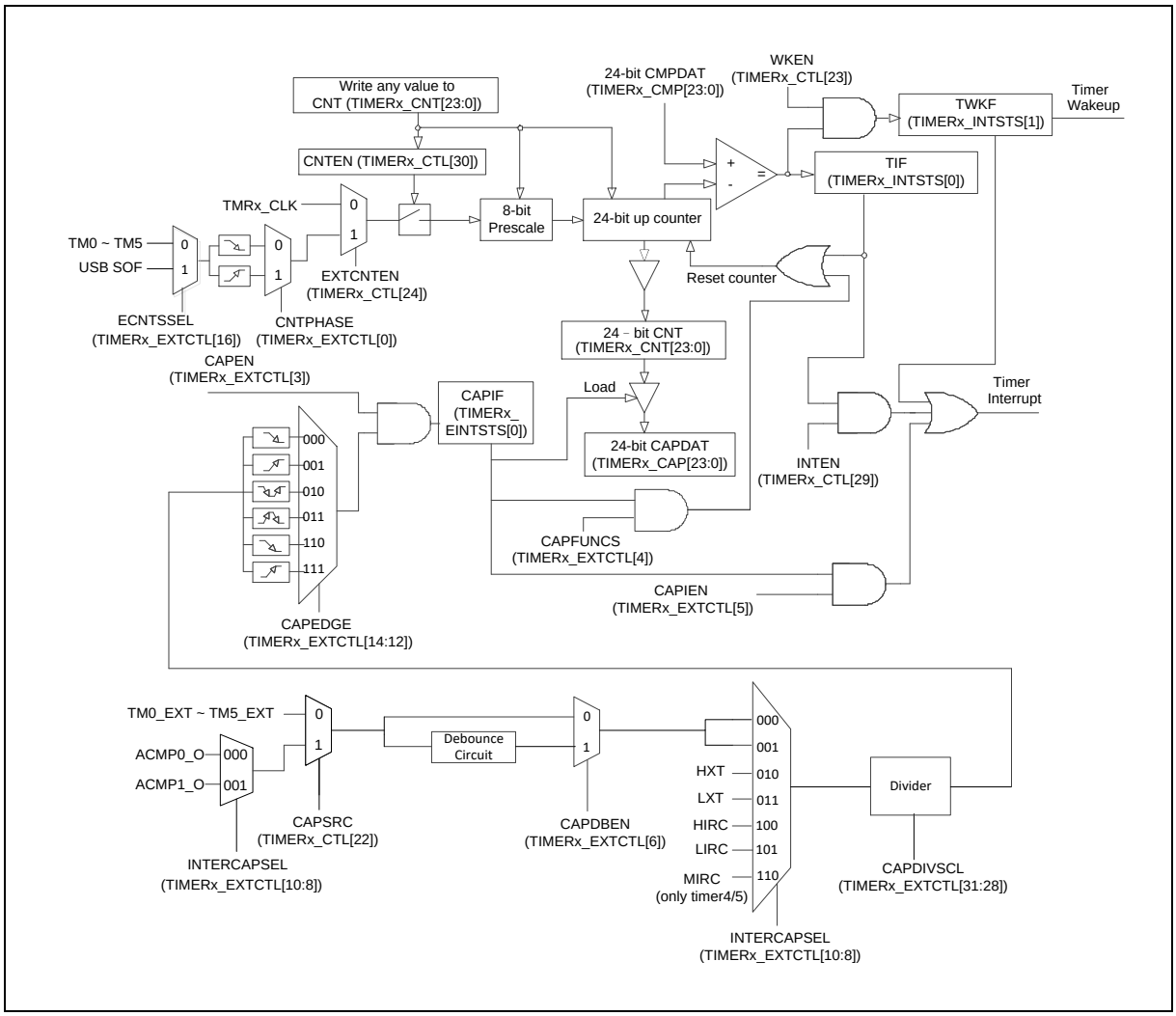


Figure 6.9-1 Timer Controller Block Diagram

Set FUNCSEL (TIMERx_ALTCTL[0]) 0 to enable timer mode in Timer0 ~ Timer3. The clock source of Timer0 ~ Timer3 in timer mode can be enabled in TMRxCKEN (CLK_APBCLK0[5:2]) and selected as different frequency in TMR0SEL (CLK_CLKSEL1[10:8]) for Timer0, TMR1SEL (CLK_CLKSEL1[14:12]) for Timer1, TMR2SEL (CLK_CLKSEL1[18:16]) for Timer2 and TMR3SEL (CLK_CLKSEL1[22:20]) for Timer3 as Figure 6.9-2.

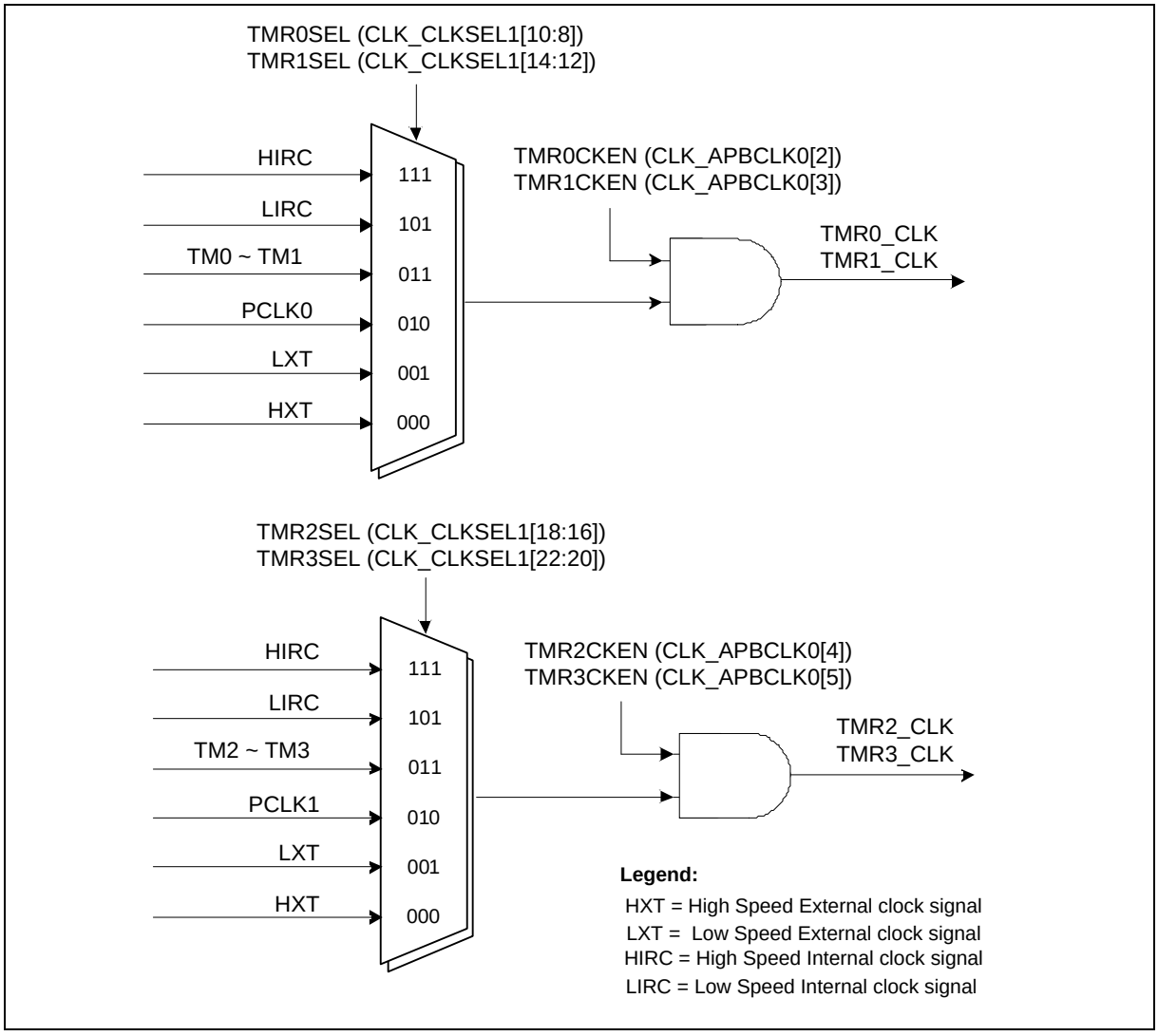


Figure 6.9-2 Clock Source of Timer0 ~ Timer 3 Controller

Set FUNCSEL (TIMERx_CTL[15]) 0 to enable timer mode in Timer4 and Timer5. The clock source of Timer4 and Timer5 in timer mode can be enabled in TMRxCKEN (CLK_APBCLK1[5:4]) and selected as different frequency in TMR4SEL (CLK_CLKSEL3[10:8]) for Timer4, TMR5SEL (CLK_CLKSEL3[14:12]) for Timer5 as Figure 6.9-3.

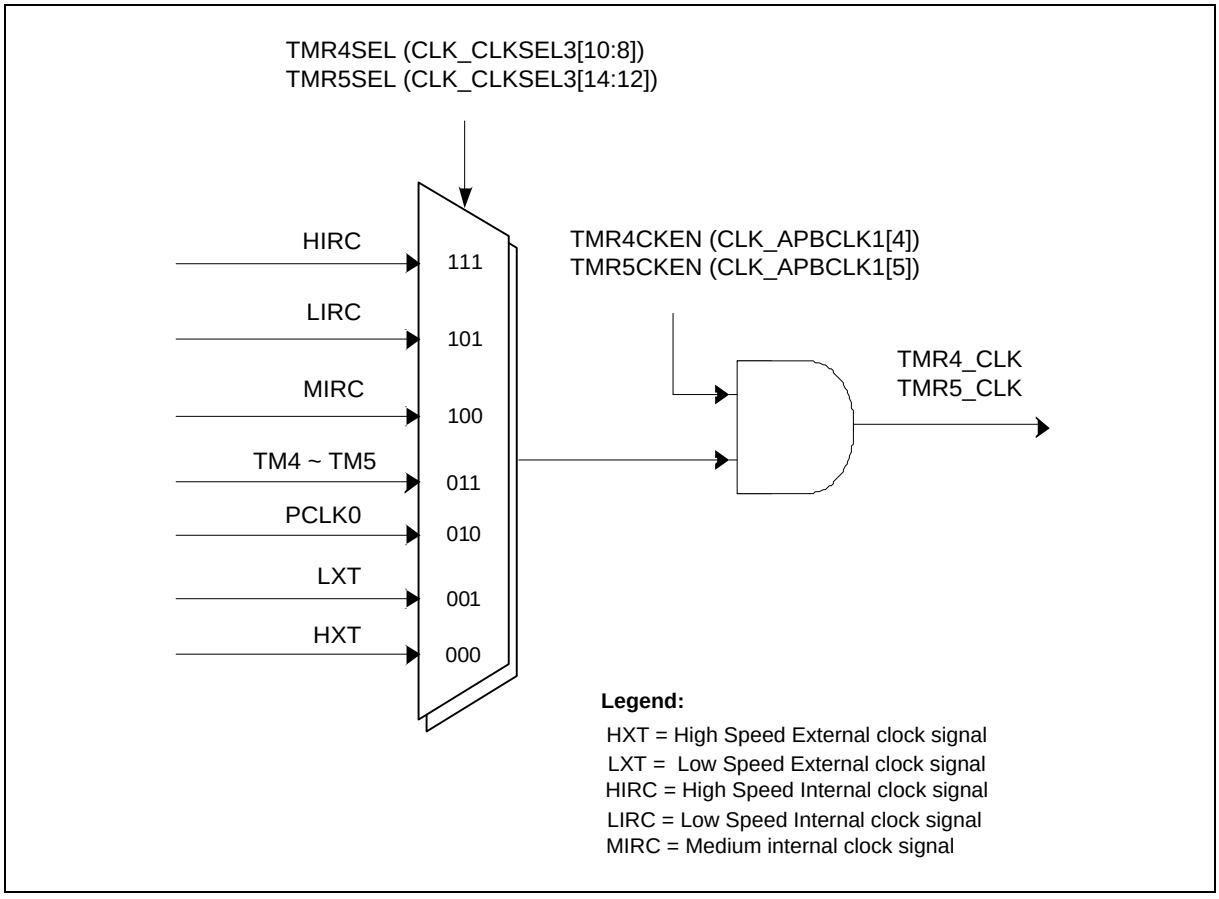


Figure 6.9-3 Clock Source of Timer4 and Timer5 Controller

The Timer0 ~ Timer3 PWM generator block diagram is shown as Figure 6.9-4. Each PWM generator has three clock source inputs. Each clock source can be selected from PCLK or four TIMER trigger PWM outputs as Figure 6.9-6 by CLKSRC (TIMERx_PWMCLKSRC[2:0]) for TMRx_PWMCLK.

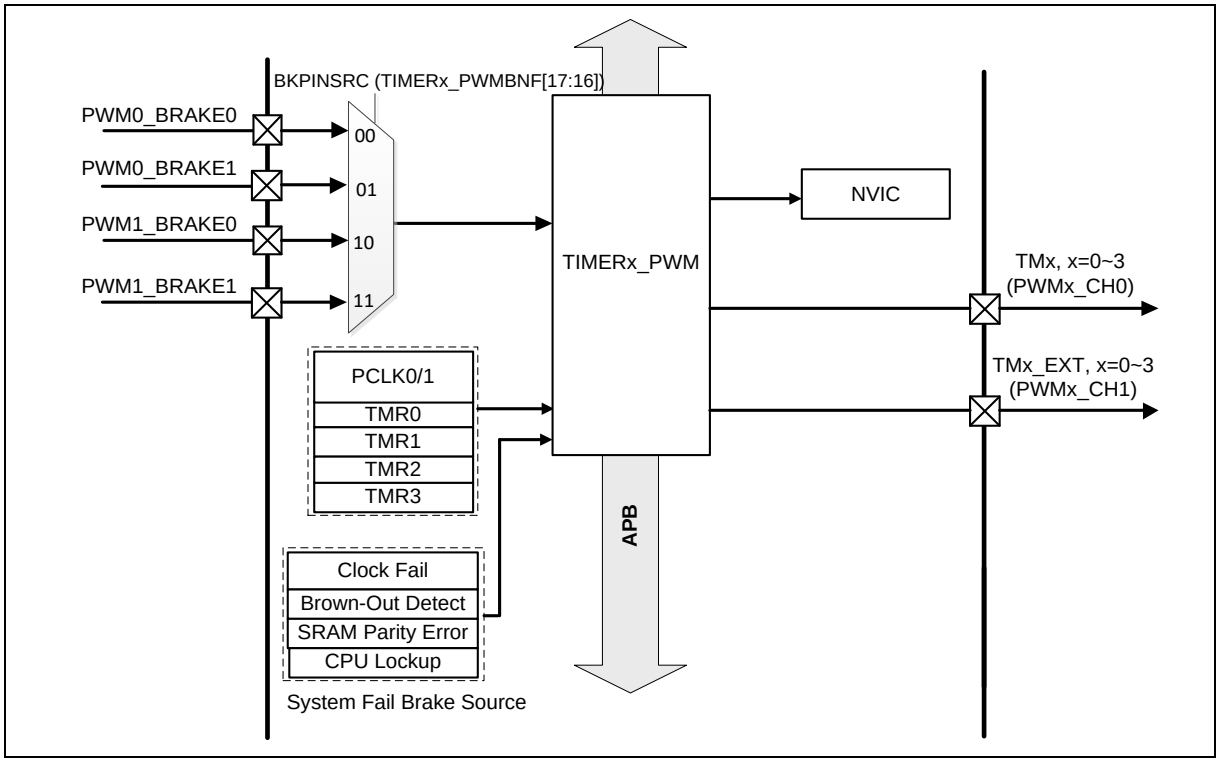


Figure 6.9-4 Timer0 ~ Timer3 PWM Generator Overview Block Diagram

Set FUNCSEL (TIMERx_ALTCTL[0]) 1 to enable PWM mode in Timer0 ~ Timer3. The clock source of Timer0 ~ Timer3 in PWM mode can be enabled in TMRxCKEN (CLK_APBCLK0[5:2]). TMR0_CLK and TMR1_CLK clock sources are fixed to PCLK0. TMR2_CLK and TMR3_CLK clock sources are fixed to PCLK1. PWM system clock frequency will be PCLKx frequency as Figure 6.9-5 .

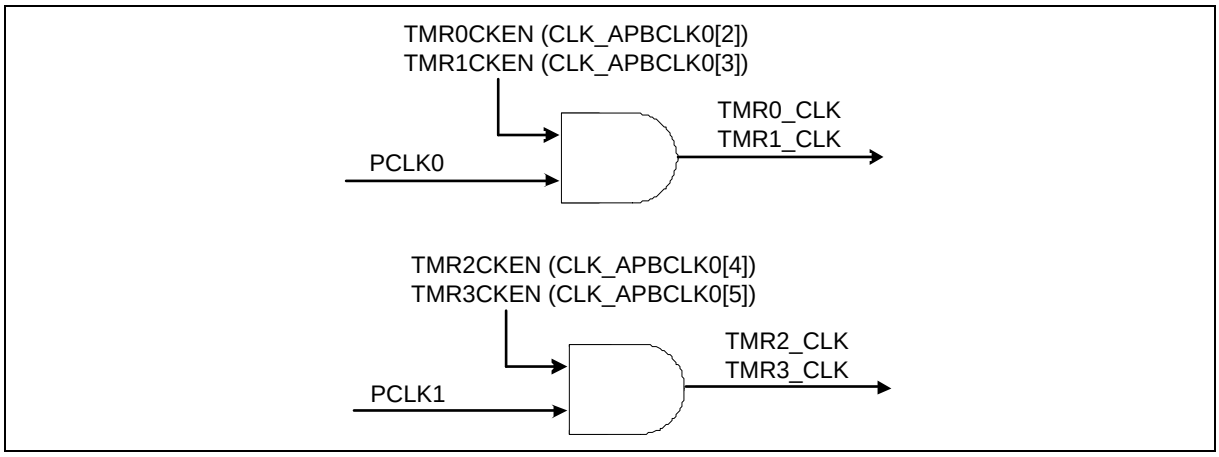


Figure 6.9-5 Timer0 ~ Timer3 PWM System Clock Source Control

The clock source of PWM counter (TIMERx_PWMCLK) can be selected from PWM system clock (TMRx_CLK) or Timer interrupt events (TMRx_INT) as Figure 6.9-6.

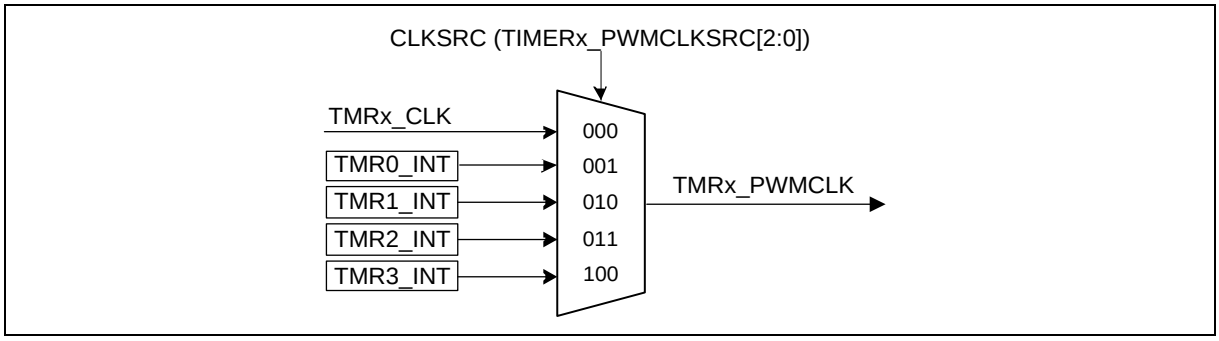


Figure 6.9-6 Timer0 ~ Timer3 PWM Counter Clock Source Control

Figure 6.9-7 and Figure 6.9-8 illustrate the architecture of PWM independent mode and complementary mode. Both independent mode and complementary mode supports PWMx_CH0 and PWMx_CH1 output channels in each PWM generator.

When counter counts to 0, PERIOD (TIMERx_PWMPERIOD[15:0]) or equal to CMP (TIMERx_PWMCMPDAT[15:0]), relative events will be generated. These events are passed to corresponding generators to generate PWM pulse (Pulse Generator), interrupt signal (Interrupt Generator) and trigger signal (Trigger Generator) for EADC to start conversion. Output Control block is used to decide PWM pulse output; brake function in Output Control block also generates interrupt events. And Dead-Time Control is available only in PWM complementary mode.

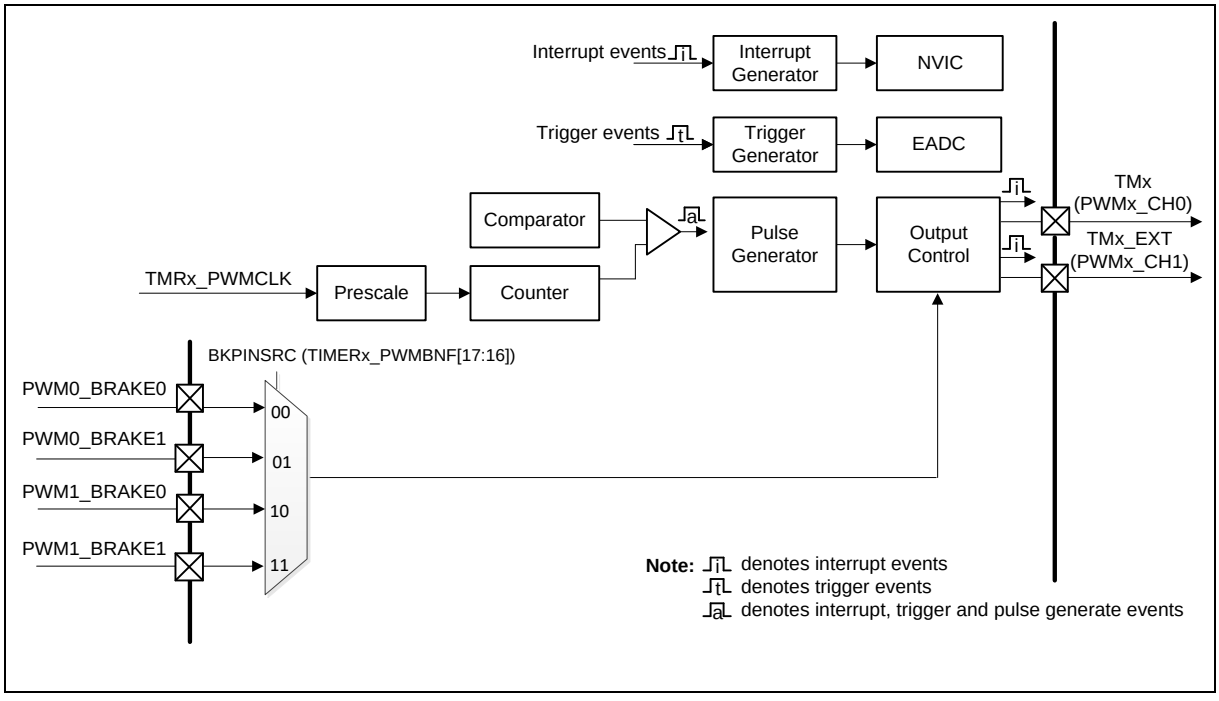


Figure 6.9-7 Timer0 ~ Timer3 PWM Independent Mode Architecture Diagram

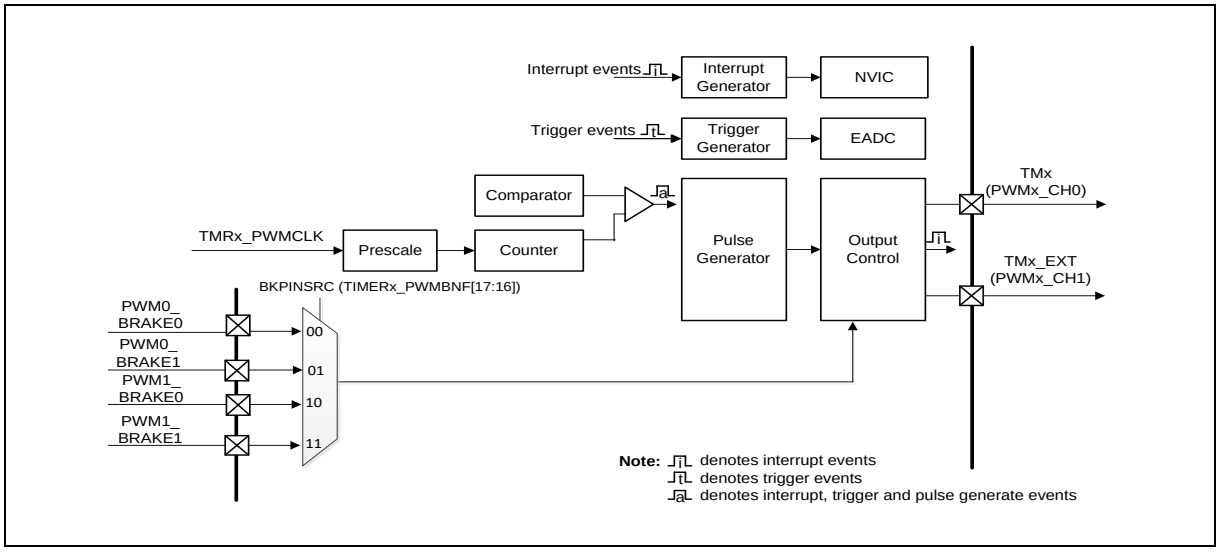


Figure 6.9-8 Timer0 ~ Timer3 PWM Complementary Mode Architecture Diagram

The Timer4 ~ Timer5 PWM generator block diagram is shown as Figure 6.9-9.

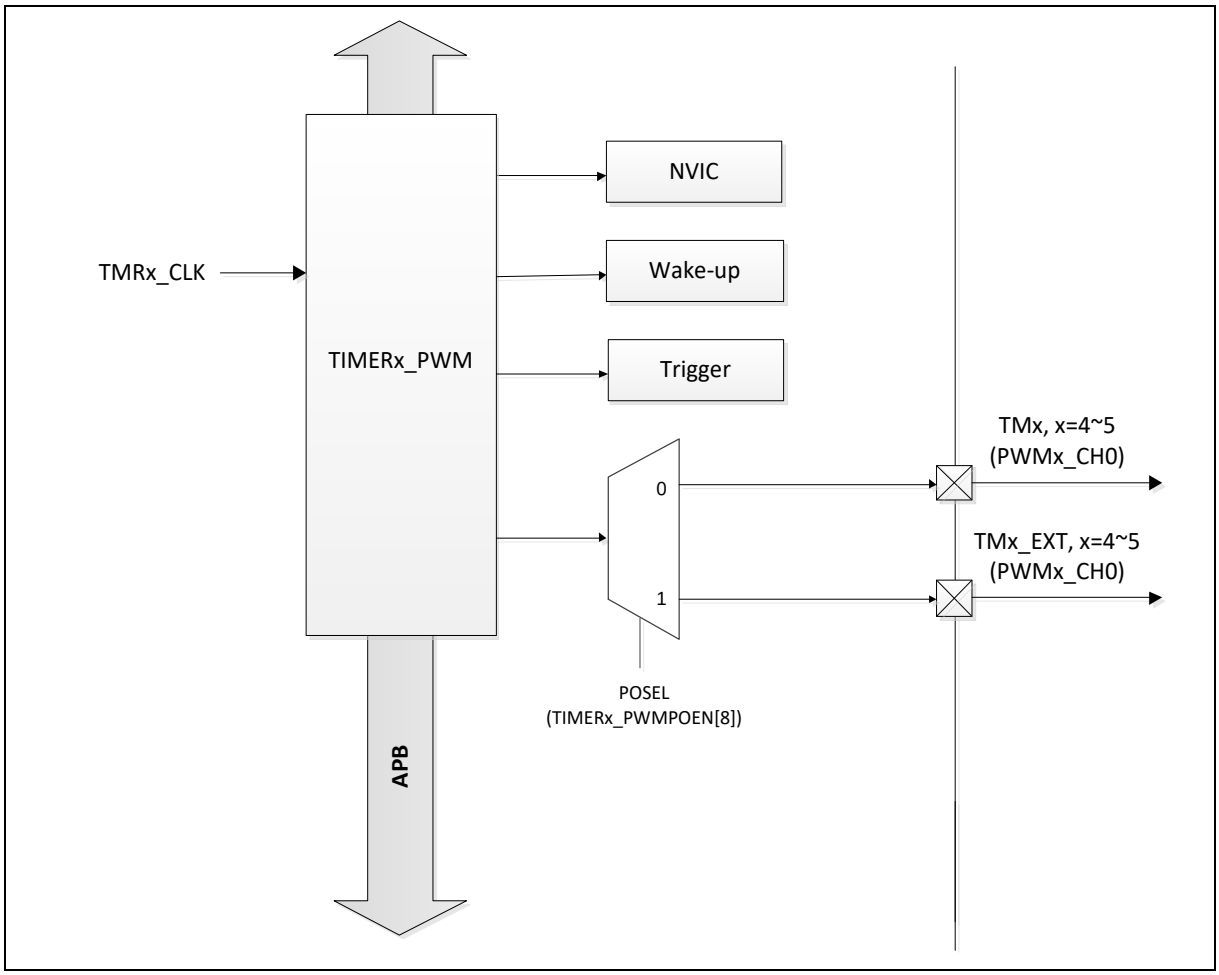


Figure 6.9-9 Timer4 and Timer5 PWM Generator Overview Block Diagram

Set FUNCSEL (TIMERx_CTL[15]) 1 to enable PWM mode in Timer4 and Timer5. The clock source of Timer4 and Timer5 in PWM mode can be enabled in TMRxCKEN (CLK_APBCLK1[5:4]). PWM system clock and counter clock source are from TMRx_CLK.

Figure 6.9-10 illustrates the PWM architecture that supports one PWM output and two selectable TMx or TMx_EXT output channels in each PWM generator.

When counter counts to 0, PERIOD (TIMERx_PWMPERIOD[15:0]) or equal to CMP (TIMERx_PWMCMPDAT[15:0]), relative events will be generated. These events are passed to corresponding generators to generate PWM pulse (Pulse Generator), interrupt signal (Interrupt Generator), wake-up (Wake-up Generator) and trigger signal (Trigger Generator) for EADC and PDMA to start conversion. Output Control block is used to decide PWM pulse output.

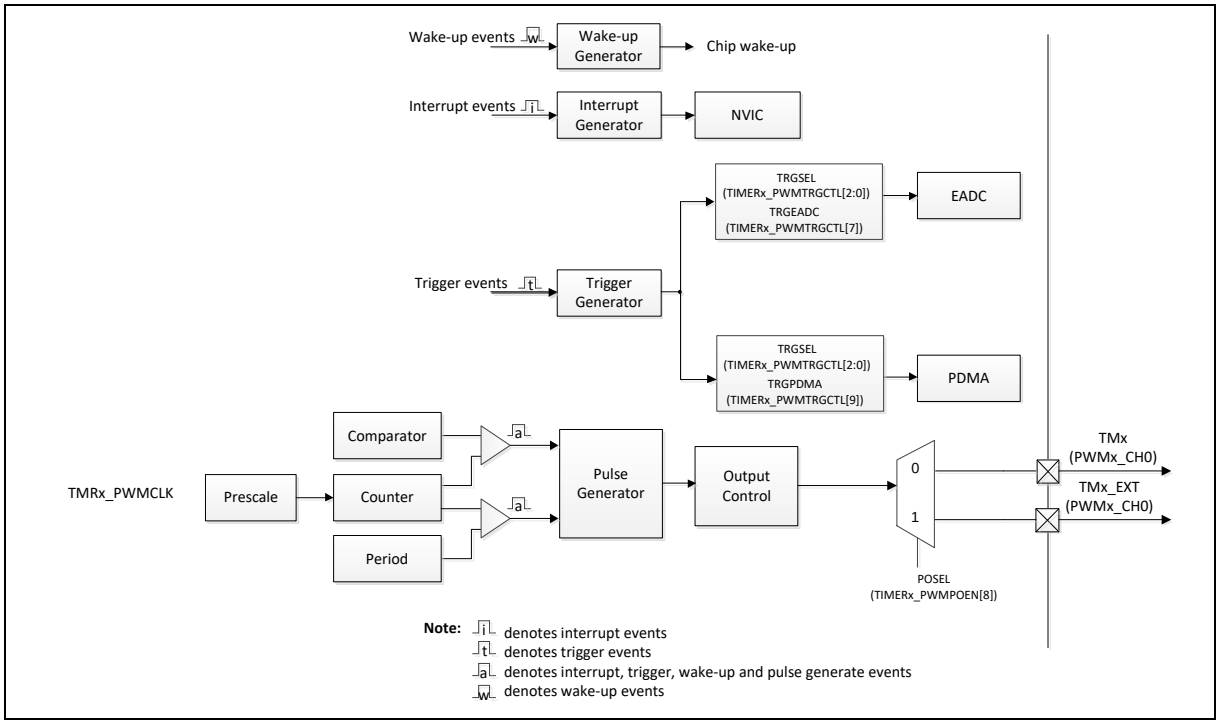


Figure 6.9-10 Timer4 and Timer5 PWM Architecture Diagram

6.9.4 Basic Configuration

6.9.4.1 TIMER01 Basic Configurations

Set FUNCSEL (TIMERx_ALTCTL[0]) 0 to enable timer mode. The clock source of Timer0 and Timer1 in timer mode can be enabled in TMRxCKEN (CLK_APBCLK0[3:2]) and selected as different frequency in TMR0SEL (CLK_CLKSEL1[10:8]) for Timer0 and TMR1SEL (CLK_CLKSEL1[14:12]) for Timer1.

Set FUNCSEL (TIMERx_ALTCTL[0]) 1 to enable PWM mode. The clock source of Timer0 and Timer1 in PWM mode can be enabled in TMRxCKEN (CLK_APBCLK0[3:2]). TMR0_CLK and TMR1_CLK clock sources are fixed to PCLK0.

- Clock source Configuration
  - Enable TIMER0 peripheral clock in TMR0CKEN (CLK_APBCLK0[2]).
  - Enable TIMER1 peripheral clock in TMR1CKEN (CLK_APBCLK0[3]).

- Reset Configuration
  - Reset TIMER0 controller in TMR0RST (SYS_IPRST2[2]).
  - Reset TIMER0 controller in TMR1RST (SYS_IPRST2[3]).
- Pin configuration

Group	Pin Name	GPIO	MFP
TM0	TM0	PG.2	MFP13
		PB.5, PC.7	MFP14
	TM0_EXT	PA.11, PB.15, PH.0	MFP13
TM1	TM1	PC.14, PG.3	MFP13
		PB.4, PC.6	MFP14
	TM1_EXT	PA.10, PB.14, PH.1	MFP13

Table 6.9-1 Timer0 and Timer1 Pin Configuration

6.9.4.2 TIMER23 Basic Configurations

Set FUNCSEL (TIMERx_ALTCTL[0]) 0 to enable timer mode. The clock source of Timer2 and Timer3 in timer mode can be enabled in TMRxCKEN (CLK_APBCLK0[5:4]) and selected as different frequency in TMR2SEL (CLK_CLKSEL1[18:16]) for Timer2 and TMR3SEL (CLK_CLKSEL1[22:20]) for Timer3.

Set FUNCSEL (TIMERx_ALTCTL[0]) 1 to enable PWM mode. The clock source of Timer2 and Timer3 in PWM mode can be enabled in TMRxCKEN (CLK_APBCLK0[5:4]). TMR2_CLK and TMR3_CLK clock sources are fixed to PCLK1.

- Clock source Configuration
  - Enable TIMER2 peripheral clock in TMR2CKEN (CLK_APBCLK0[4]).
  - Enable TIMER3 peripheral clock in TMR3CKEN (CLK_APBCLK0[5]).
- Reset Configuration
  - Reset TIMER2 controller in TMR2RST (SYS_IPRST2[4]).
  - Reset TIMER3 controller in TMR3RST (SYS_IPRST2[5]).
- Pin configuration

Group	Pin Name	GPIO	MFP
TM2	TM2	PG.4	MFP13
		PA.7, PB.3, PD.0	MFP14
	TM2_EXT	PA.9, PB.13, PH.2	MFP13
TM3	TM3	PF.11	MFP13
		PA.6, PB.2	MFP14
	TM3_EXT	PA.8, PB.12, PH.3	MFP13

Table 6.9-2 Timer2 and Timer3 Pin Configuration

6.9.4.3 TIMER45 Basic Configurations

Set FUNCSEL (TIMERx_CTL[15]) 0 to enable timer mode. The clock source of Timer4 and Timer5 in timer mode can be enabled in TMRxCKEN (CLK_APBCLK1[5:4]) and selected as different frequency in TMR4SEL (CLK_CLKSEL3[10:8]) for Timer4, TMR5SEL (CLK_CLKSEL3[14:12]) for Timer5.

Set FUNCSEL (TIMERx_CTL[15]) 1 to enable PWM mode. The clock source of Timer4 and Timer5 in PWM mode can be enabled in TMRxCKEN (CLK_APBCLK1[5:4]). PWM system clock and counter clock source are from TMRx_CLK.
User must poll FUNCEL to ensure that the operation mode change is completed, and then make subsequent settings.

- Clock source Configuration
  - Enable TIMER4 peripheral clock in TMR4CKEN (CLK_APBCLK1[4]).
  - Enable TIMER5 peripheral clock in TMR5CKEN (CLK_APBCLK1[5]).
- Reset Configuration
  - Reset TIMER4 controller in TMR4RST (SYS_IPRST2[20]).
  - Reset TIMER5 controller in TMR5RST (SYS_IPRST2[21]).
- Pin configuration

Group	Pin Name	GPIO	MFP
TM4	TM4	PA.7	MFP10
		PG.4	MFP12
		PB.3	MFP13
	TM4_EXT	PA.9	MFP12
		PB.13	MFP14
TM5	TM5	PA.6	MFP10
		PF.11	MFP12
		PB.2	MFP13
	TM5_EXT	PA.8	MFP12
		PB.12	MFP14

Table 6.9-3 Timer4 and Timer5 Pin Configuration

6.9.5 Timer Functional Description

6.9.5.1 Timer Interrupt Flag

The timer controller supports the following interrupt flags: TIF (TIMERx_INTSTS[0]) which is set while the timer counter value CNT (TIMERx_CNT[23:0]) matches the timer compared value CMPDAT (TIMERx_CMP[23:0]), and CAPIF (TIMERx_EINTSTS[0]) which is set when the transition on the TMx_EXT pin, ACMP, internal clock (HIRC, LIRC, MIRC) or external clock (HXT, LXT) associated CAPEDGE (TIMERx_EXTCTL[14:12]) setting. User can set CAPSRC (TIMERx_CTL[22]) and INTERCAPSEL (TIMERx_EXTCTL[10:8]) to select capture source. The TWKF (TIMERx_INTSTS[1]) bit indicates the interrupt wake-up flag status of timer. Set WKEN (TIMERx_CTL[23]) to 1 can use wake-up function.

6.9.5.2 Timer Counting Mode

The timer controller provides four timer counting modes: one-shot, periodic, toggle-output and continuous counting operation modes:

6.9.5.3 One–shot Mode

If the timer controller is configured at one-shot mode (TIMERx_CTL[28:27] is 00) and CNTEN (TIMERx_CTL[30]) is set, the timer counter starts up counting. Once the CNT (TIMERx_CNT[23:0]) value reaches CMPDAT (TIMERx_CMP[23:0]) value, the TIF (TIMERx_INTSTS[0]) will be set to 1, CNT value and CNTEN bit is cleared automatically by timer controller then timer counting operation stops. In the meantime, if the INTEN (TIMERx_CTL[29]) is enabled, the timer interrupt signal is generated and sent to NVIC to inform CPU also.

User can monitor the RSTACT (TIMERx_CNT[31]) to ensure counter reset operation active and disable ICE debug mode acknowledgement effects TIMER counting by setting ICEDEBUG (TIMERx_CTL[31]) to 1.

6.9.5.4 Periodic Mode

If the timer controller is configured at periodic mode (TIMERx_CTL[28:27] is 01) and CNTEN (TIMERx_CTL[30]) is set, the timer counter starts up counting. Once the CNT (TIMERx_CNT[23:0]) value reaches CMPDAT (TIMERx_CMP[23:0]) value, the TIF (TIMERx_INTSTS[0]) will be set to 1, CNT value will be cleared automatically by timer controller and timer counter operates counting again. In the meantime, if the INTEN (TIMERx_CTL[29]) bit is enabled, the timer interrupt signal is generated and sent to NVIC to inform CPU also. In this mode, the timer controller operates counting and compares with CMPDAT value periodically until the CNTEN bit is cleared by user.

User can set PERIOSEL (TIMERx_CTL[20]) to select Timer behavior at periodic mode.

6.9.5.5 Toggle-Output Mode

If the timer controller is configured at toggle-output mode (TIMERx_CTL[28:27] is 10) and CNTEN (TIMERx_CTL[30]) is set, the timer counter starts up counting. The counting operation of toggle-output mode is almost the same as periodic mode, except toggle-output mode has associated TM0 ~ TM5 or TM0_EXT ~ TM5_EXT pin to output signal while specify TIF (TIMERx_INTSTS[0]) is set. Use can set TGLPINSEL (TIMERx_CTL[21]) to choose TMx or TMx_EXT as toggle-output pin. Thus, the toggle-output signal on toggle-output pin is high and changing back and forth with 50% duty cycle.

6.9.5.6 Continuous Counting Mode

If the timer controller is configured at continuous counting mode (TIMERx_CTL[28:27] is 11) and CNTEN (TIMERx_CTL[30]) is set, the timer counter starts up counting. Once the CNT (TIMERx_CNT[23:0]) value reaches CMPDAT (TIMERx_CMP[23:0]) value, the TIF (TIMERx_INTSTS[0]) will be set to 1 and CNT value keeps up counting. In the meantime, if the INTEN (TIMERx_CTL[29]) is enabled, the timer interrupt signal is generated and sent to NVIC to inform CPU also. User can change different CMPDAT value immediately without disabling timer counting and restarting timer counting in this mode.

For example, CMPDAT value is set as 80, first. The TIF will be set to 1 when CNT value is equal to 80, timer counter is kept counting and CNT value will not goes back to 0, it continues to count 81, 82, 83, ⋯ to 224 -1, 0, 1, 2, 3, ⋯ to 224 -1 again and again. Next, if user programs CMPDAT value as 200 and clears TIF, the TIF will be set to 1 again when CNT value reaches to 200. At last, user programs CMPDAT as 500 and clears TIF, the TIF will be set to 1 again when CNT value reaches to 500.

In this mode, the timer counting is continuous. So, this operation mode is called as continuous counting mode.

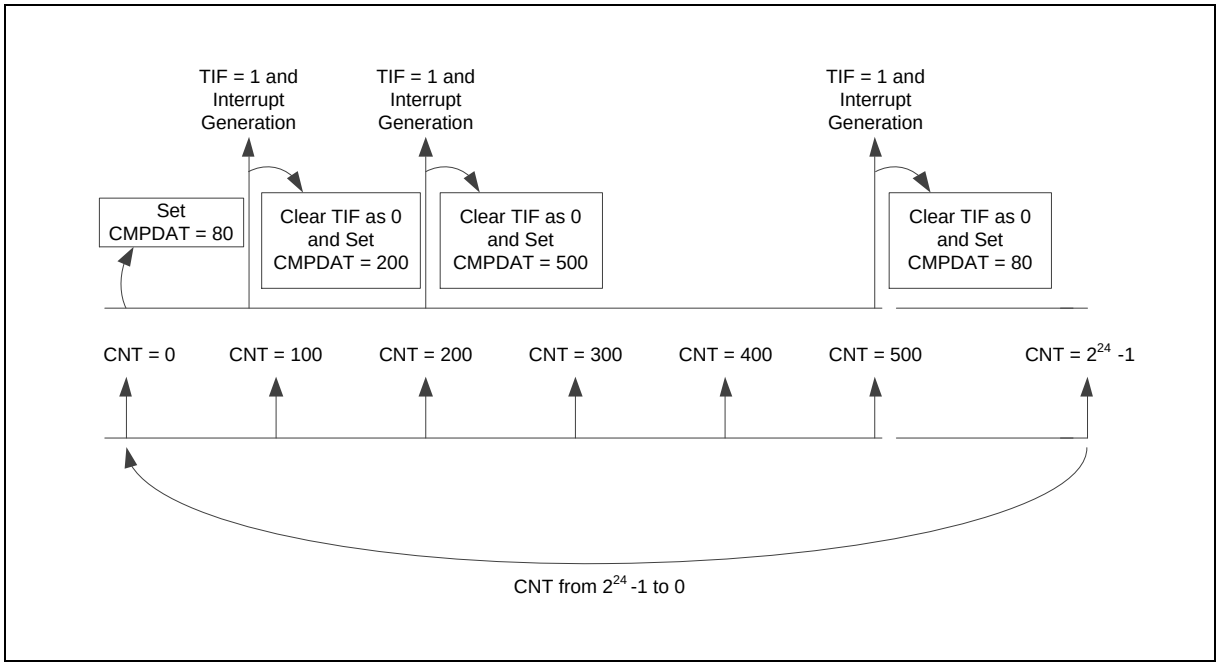


Figure 6.9-11 Continuous Counting Mode

6.9.5.7 Event Counting Mode

The timer controller also provides an application which can count the input event from TMx (x= 0~5) pin and the number of event will reflect to CNT (TIMERx_CNT[23:0]) value. It is also called as event counting function. In this function, EXTCNTEN (TIMERx_CTL[24]) should be set and the timer peripheral clock source should be set as PCLK.

If ECNTSSEL (TIMERx_EXTCTL[16]) is 0, the event counter source is from external TMx pin. User can enable or disable TMx pin de-bounce circuit by setting CNTDBEN (TIMERx_EXTCTL[7]). The input event frequency should be less than 1/3 PCLK if TMx pin de-bounce disabled or less than 1/8 PCLK if TMx pin de-bounce enabled to assure the returned CNT value is correct, and user can also select edge detection phase of TMx pin by setting CNTPHASE (TIMERx_EXTCTL[0]) bit.

In event counting mode, the timer counting operation mode can be selected as one-shot, periodic and continuous counting mode to counts the counter value CNT (TIMERx_CNT[23:0]) for TMx pin.

If ECNTSSEL (TIMERx_EXTCTL[16]) is 1, the event counter source will generate by USB device detect the start-of-frame (SOF) packet. Please refer USB device specifications.

6.9.5.8 Capture Mode

The event capture function is used to load CNT (TIMERx_CNT[23:0]) value to CAPDAT (TIMERx_CAP[23:0]) value while edge transition detected on capture source, TMx_EXT (x= 0~5) pin, ACMP, internal clock and external clock. In this mode, CAPFUNCS (TIMERx_EXTCTL[4]) should be as 0 to trigger event capture function and the timer peripheral clock source should be set as PCLK.

If CAPSRC (TIMERx_CTL[22]) is 0, the capture event is triggered by TMx_EXT pin transition. User can enable or disable TMx_EXT pin de-bounce circuit by setting CAPDBEN (TIMERx_EXTCTL[6]). The transition frequency of TMx_EXT pin should be less than 1/3 PCLK if TMx_EXT pin de-bounce disabled or less than 1/8 PCLK if TMx_EXT pin de-bounce enabled to assure the capture function can be work normally, and user can also select edge transition detection of TMx_EXT pin by setting CAPEDGE (TIMERx_EXTCTL[14:12]).

In external pin capture mode, user does not consider what timer counting operation mode is selected, and the capture event occurred only if edge transition on TMx_EXT pin is detected.

User can enable CAPIEN (TIMERx_EXTCTL[5]) to use capture interrupt function. When the TMx_EXT edge transition meets setting, CAPIF is high.

User must consider the Timer will keep register TIMERx_CAP unchanged and drop the new capture value if the CPU does not clear the CAPIF status.

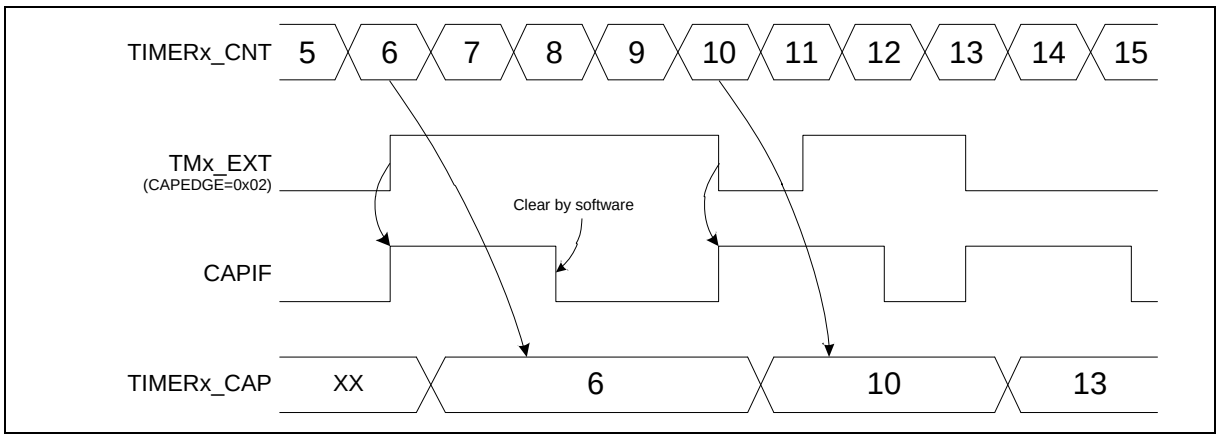


Figure 6.9-12 Capture Mode

If CAPSRC (TIMERx_CTL[22]) is 1, set INTERCAPSEL (TIMERx_EXTCTL[10:8]) to choose different capture source. The capture event can be triggered by internal output signal transition on ACMP0 if INTERCAPSEL (TIMERx_EXTCTL[10:8]) is 000, or ACMP1 if INTERCAPSEL is 001; Other capture sources are HXT, LXT, HIRC, LIRC, MIRC if INTERCAPSEL is 010, 011, 100, 101, 110 respectively. User can switch capture source only when original source and target source is all off. For example, if user wants to switch LXT or LIRC, both capture sources should be set off first.

6.9.5.9 Capture And Reset Counter Mode

The timer controller also provides reset counter function to reset CNT (TIMERx_CNT[23:0]) value while capture event is generated. In this mode, CAPFUNCS (TIMERx_EXTCTL[4]) should be as 1, and user must set CAPSRC and INTERCAPSEL to select capture source to obtain the capture data and reset the counter value.

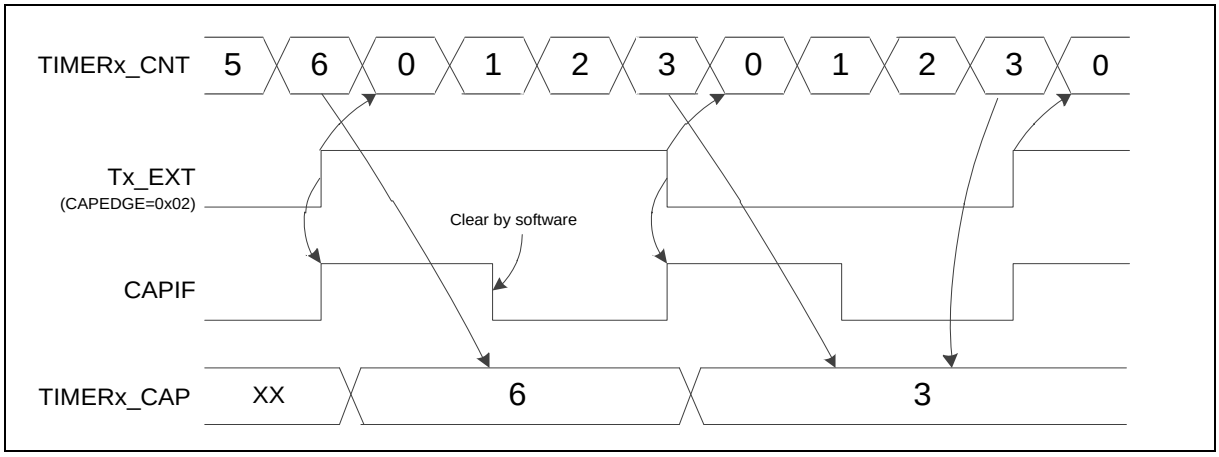


Figure 6.9-13 Reset Counter Mode

6.9.5.10 Timer Trigger Function

The timer controller provides timer time-out interrupt signal or capture interrupt signal to trigger EPWM/BPWM, EADC, DAC and PDMA.

In Timer0 ~ Timer3, if TRGSSEL (TIMERx_TRGCTL[0]) is 0, the time-out interrupt signal is used to trigger EPWM/BPWM, EADC, DAC and PDMA; if TRGSSEL (TIMERx_TRGCTL[0]) is 1, the capture interrupt signal is used to trigger EPWM/BPWM, EADC, DAC and PDMA.

In Timer4 and Timer5, if TRGSSEL (TIMERx_TRGCTL[0]) is 0, the time-out interrupt signal is used to trigger EADC and PDMA; if TRGSSEL (TIMERx_TRGCTL[0]) is 1, the capture interrupt signal is used to trigger EADC and PDMA.

When the TRGPWM (TIMERx_TRGCTL[1]) is set, if the timer interrupt signal is generated, the timer controller will generate a trigger pulse as EPWM and BPWM counter clock source.

When the TRGEADC (TIMERx_TRGCTL[2]) is set, if the timer interrupt signal is generated, the timer controller will trigger EADC to start conversion.

When the TRGDAC (TIMERx_TRGCTL[3]) is set, if the timer interrupt signal is generated, the timer controller will trigger DAC to start conversion.

When the TRGPDMA (TIMERx_TRGCTL[4]) is set, if the timer interrupt signal is generated, the timer controller will trigger PDMA.

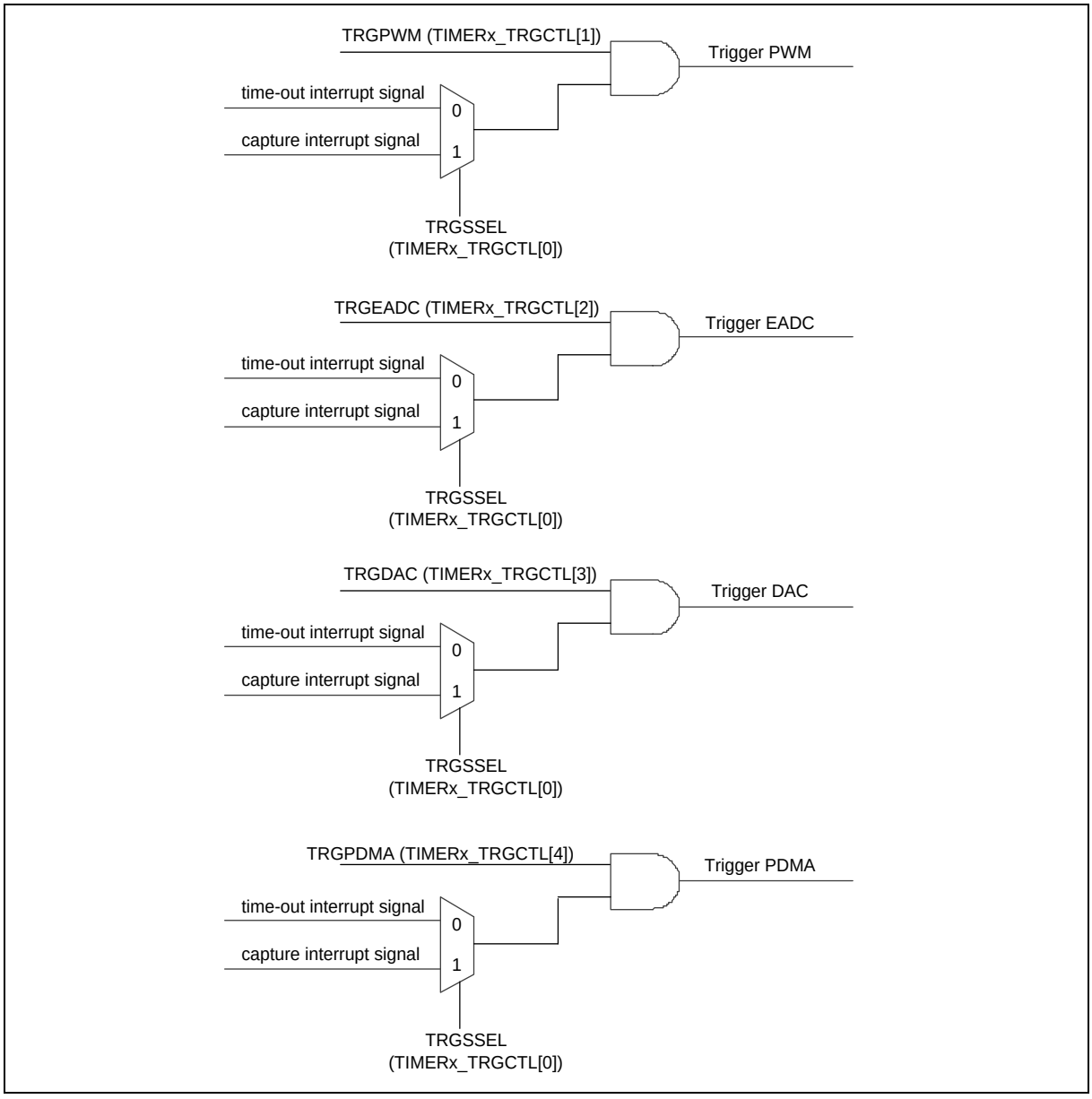


Figure 6.9-14 Internal Timer Trigger

6.9.5.11 Inter-Timer Trigger Capture Mode

In this mode, the Timer0/2/4 will be forced in event counting mode, counting with external event, and will generate an internal signal (INTR_TMR_TRG) to trigger Timer1/3/5 start or stop counting. Also, the Timer1/3/5 will be forced in capture mode and start/stop trigger-counting by Timer0/2/4 counter status.

Setting Timer0 Inter-Timer Trigger Capture enabled, trigger-counting capture function is forced on Timer1. Setting Timer2 Inter-Timer Trigger enabled, trigger-counting capture function is forced on Timer3. Setting Timer4 Inter-Timer Trigger enabled, trigger-counting capture function is forced on Timer5.

Start Trigger

While INTRGEN (TIMERx_CTL[19]) in Timer0/2/4 is set, the Timer0/2/4 will make a rising-edge transition of INTR_TMR_TRG while Timer0/2/4 24-bit counter value (CNT) is counting from 0x0 to 0x1

and Timer1/3/5 counter will start counting immediately and automatically.

Stop Trigger

When Timer0/2/4 CNT reaches the Timer0/2/4 CMPDAT value, the Timer0/2/4 will make a falling-edge transition of INTR_TMR_TRG. Then Timer0/2/4 counter mode function will be disabled and INTRGEN (TIMERx_CTL[19]) will be cleared by hardware then Timer1/3/5 will stop counting also. At the same time, the Timer1/3/5 CNT value will be saved into Timer1/3/5 CAPDAT (TIMERx_CAP[23:0]).

User can use inter-timer trigger mode to measure the period of external event (TMx) more precisely. Figure 6.9-15 shows the sample flow of Inter-Timer Trigger Capture Mode for Timer0 as event counting mode and Timer1 as trigger-counting capture mode.

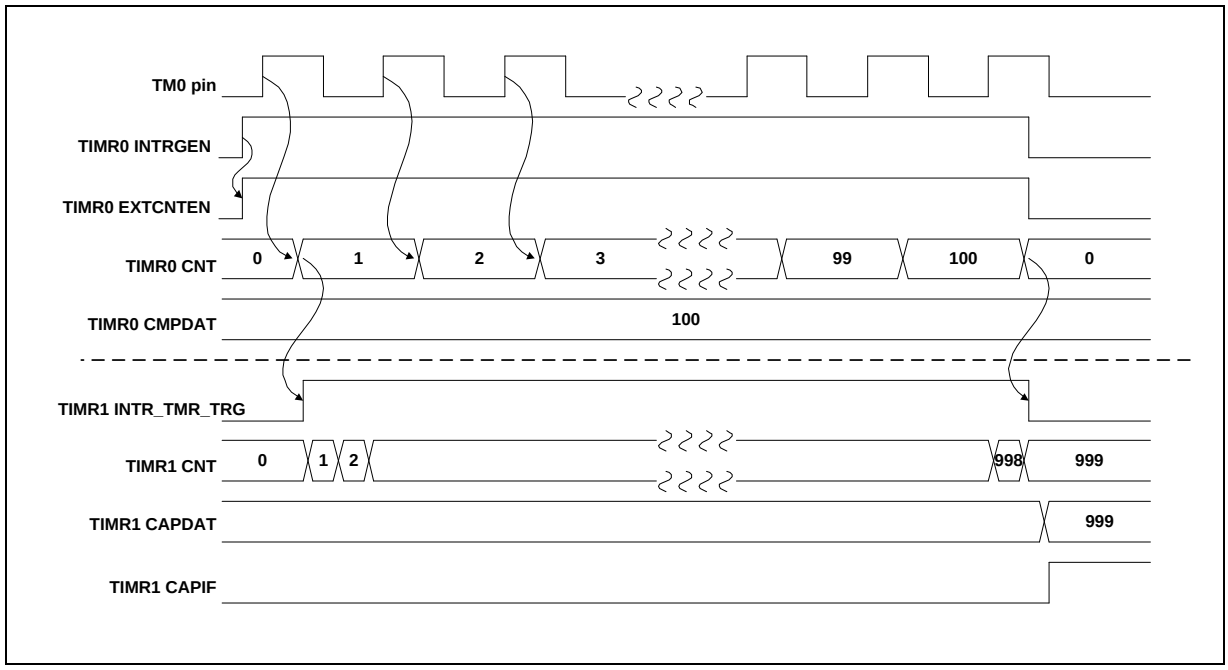


Figure 6.9-15 Inter-Timer Trigger Capture Timing

6.9.6 PWM Functional Description

6.9.6.1 PWM Prescale

The PWM prescale is used to divide clock source, and the clock of PWM counter is divided by (CLKPSC+1). The prescale is set by CLKPSC (TIMERx_PWMCLKPSC[11:0]). Figure 6.9-16 shows an example of PWM prescale waveform in up count type.

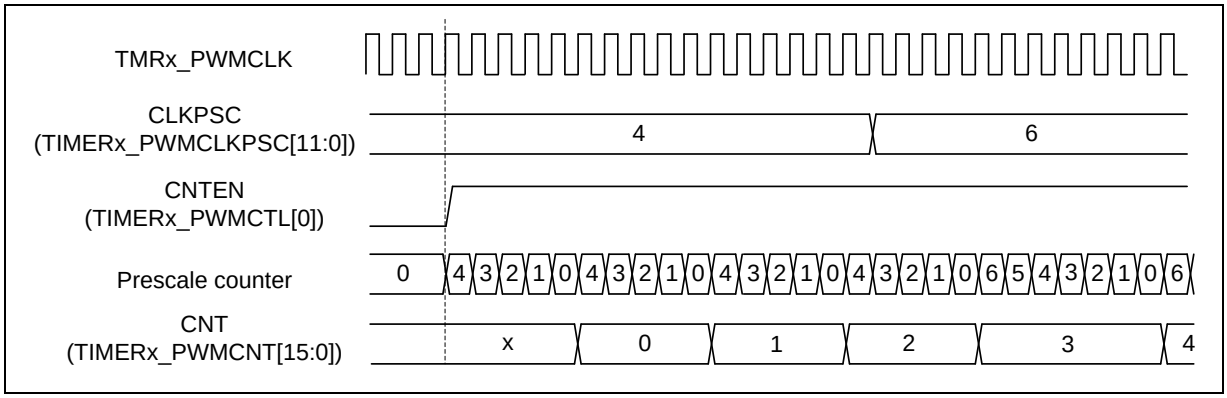


Figure 6.9-16 PWM Prescale Waveform in Up Count Type

6.9.6.2 PWM Counter

The Timer0 ~ Timer3 PWM supports three counter types operation: up count, down count and up-down count types. The Timer4 and Timer5 PWM supports up count type only.

6.9.6.3 Up Count Type

In Timer0 ~ Timer3 PWM up count type

When PWM counter is set to up count type, CNTTYPE (TIMERx_PWMCTL[2:1]) is 0x0, it starts up-counting from 0 to PERIOD (TIMERx_PWMPERIOD[15:0]). The current counter value can be read from the CNT (TIMERx_PWMCNT[15:0]). PWM generates a zero point event when both counter and prescale counts to 0. PWM generates a period point event when the counter counts to PERIOD and prescale counts to 0. Figure 6.9-17 shows an example of PWM up count type, where PWM period time is (PERIOD+1) * (CLKPSC+1) * TMRx_PWMCLK.

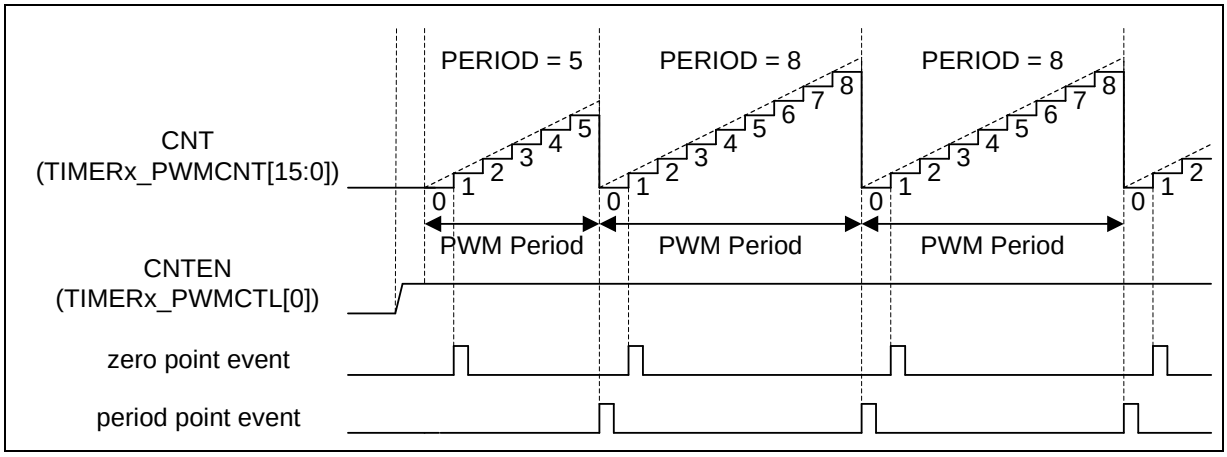


Figure 6.9-17 Timer0 ~ Timer3 PWM Up Count Type

In Timer4 and Timer5 PWM up count type

It starts up-counting from 0 to PERIOD (TIMERx_PWMPERIOD[15:0]). The current counter value can be read from the CNT (TIMERx_PWMCNT[15:0]). PWM generates a period point event when the counter counts to PERIOD and prescale counts to 0. Figure 6.9-18 shows an example of PWM up count type, where PWM period time is (PERIOD+1) * (CLKPSC+1) * TMRx_PWMCLK.

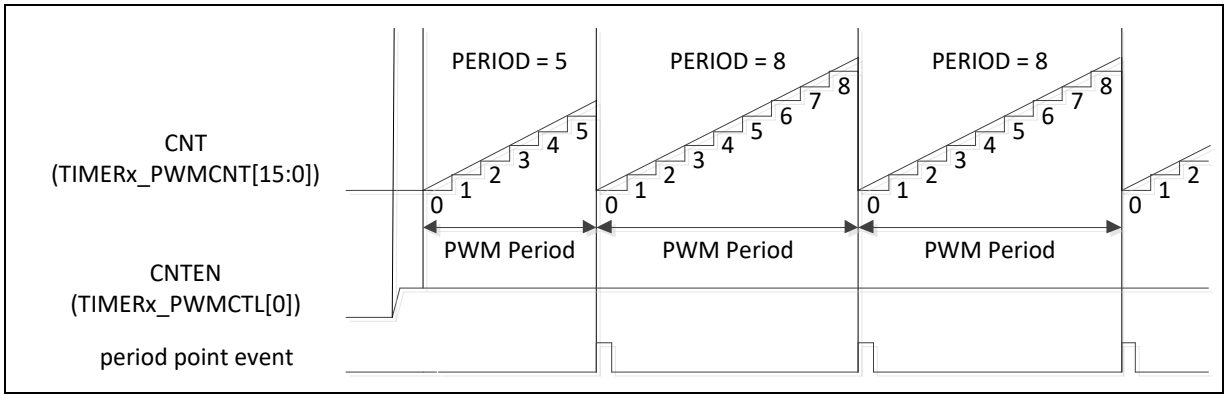


Figure 6.9-18 Timer4 and Timer5 PWM Up Count Type

6.9.6.4 Down Count Type

The down count type is only available in Timer0 ~ Timer3 PWM.

When PWM counter is set to down count type, CNTTYPE (TIMERx_PWMCTL[2:1]) is 0x1, it starts down-counting from PERIOD to 0, current counter value can be read from CNT (TIMERx_PWMCNT[15:0]). PWM generates a zero point event when both counter and prescale counts to 0. PWM generates a period point event when the counter counts to PERIOD and prescale counts to 0. Figure 6.9-19 is an example of PWM down count type, where PWM period time is (PERIOD+1) * (CLKPSC+1) * TMRx_PWMCLK.

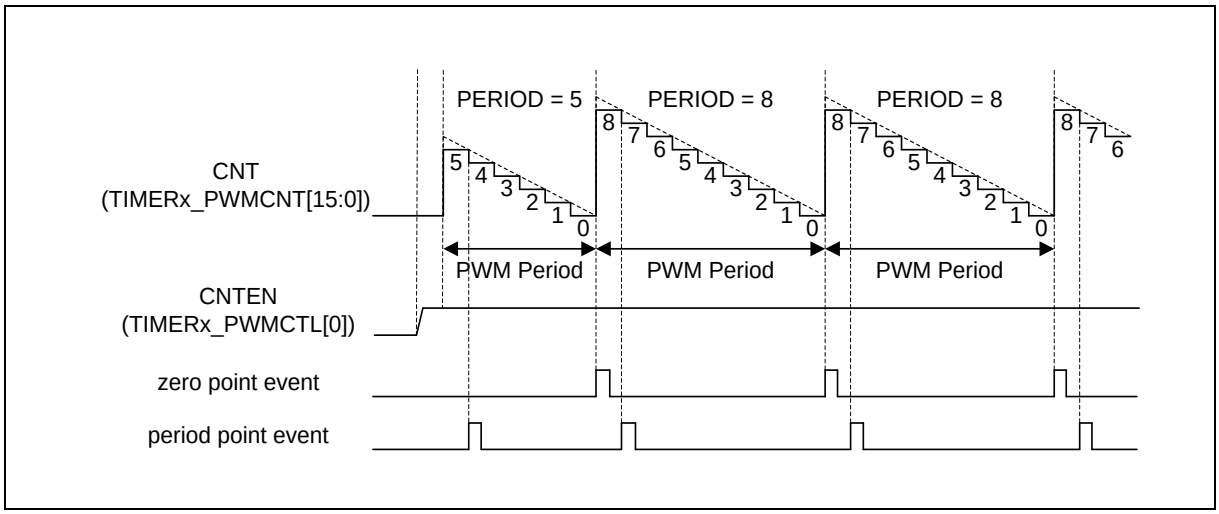
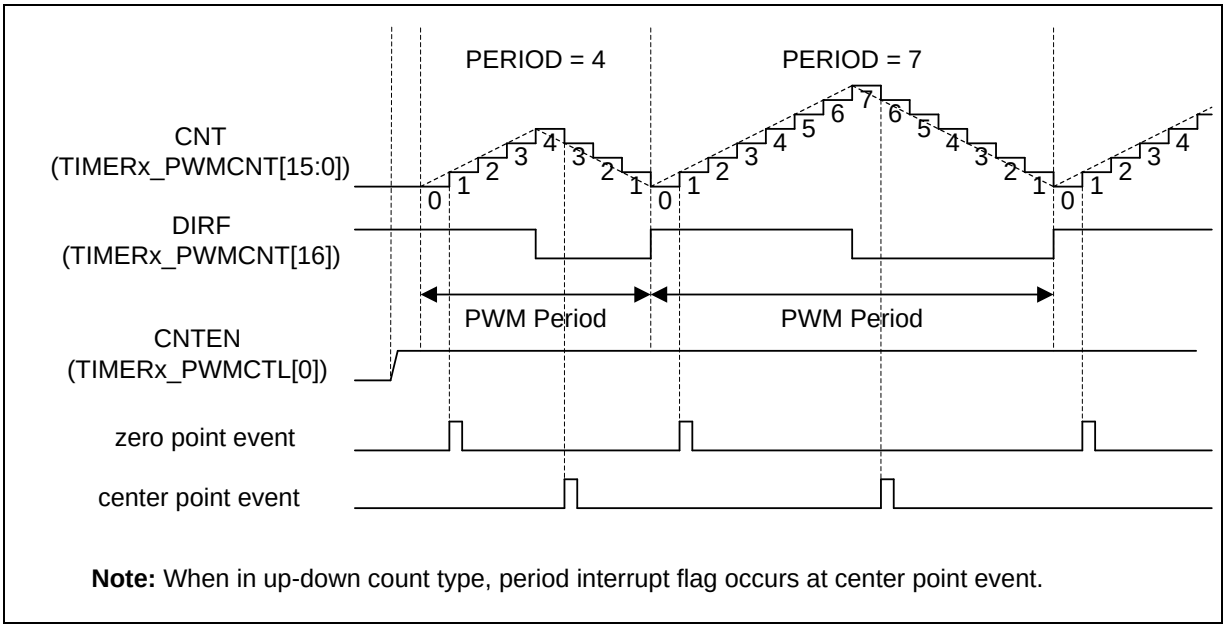


Figure 6.9-19 Timer0 ~ Timer3 PWM Down Count Type

6.9.6.5 Up-Down Count Type

The up-down count type is only available in Timer0 ~ Timer3 PWM.

When PWM counter is set to up-down count type, CNTTYPE (TIMERx_PWMCTL[2:1]) is 0x2, it starts counting up from 0 to PERIOD and then starts counting down to 0. The current counter value can be read from CNT (TIMERx_PWMCNT[15:0]). PWM generates a zero point event when both counter and prescale counts to 0. PWM generates a center point event when the counter counts to PERIOD and prescale counts to 0. Figure 6.9-20 shows an example of PWM up-down count type, where PWM period time is (2 * PERIOD) * (CLKPSC+1) * TMRx_PWMCLK. The DIRF (TIMERx_PWMCNT[16]) is counter direction indicator flag, where 1 is up counting, and 0 is down counting.



Note: When in up-down count type, period interrupt flag occurs at center point event.

Figure 6.9-20 Timer0 ~ Timer3 PWM Up-Down Count Type

6.9.6.6 PWM Counter Operation mode

The PWM counter supports two operation modes: one-shot mode and auto-reload mode. PWM counter will operate in one-shot mode if CNTMODE (TIMERx_PWMCTL[3]) bit is set to 1, and operate in auto-reload mode if CNTMODE bit is set to 0.

In both modes, CMP (TIMERx_PWMCMPDAT[15:0]) and PERIOD (TIMERx_PWMPERIOD[15:0]) should be written first and then set CNTEN (TIMERx_PWMCTL[0]) bit to 1 to start counter running.

In one-shot mode, PWM counter value will reload to default value according count type after one PWM period is completed. User can write CMP to continuous one-shot operation to generate next one-shot pulse once no matter current one-shot counter is running or completed.

In auto-reload mode, PWM counter is continuous running with current active PERIOD and CMP. If user sets PERIOD to zero in auto-reload mode, PWM counter value will reload to default value according count type after one PWM period is completed.

6.9.6.7 PWM Comparator

The CMP (TIMERx_PWMCMPDAT[15:0]) is comparator register of PWM. The CMP value is continuously compared to the corresponding counter value. When the counter is equal to CMP, PWM generates a compared point event.

In Timer0 ~ Timer3 PWM, this event will generate PWM output pulse, interrupt signal or trigger EADC start conversion. In up-down count type, two events will be generated in a PWM period as shown in Figure 6.9-21. The CMPU is up count compared point event and CMPD is down count compared point event.

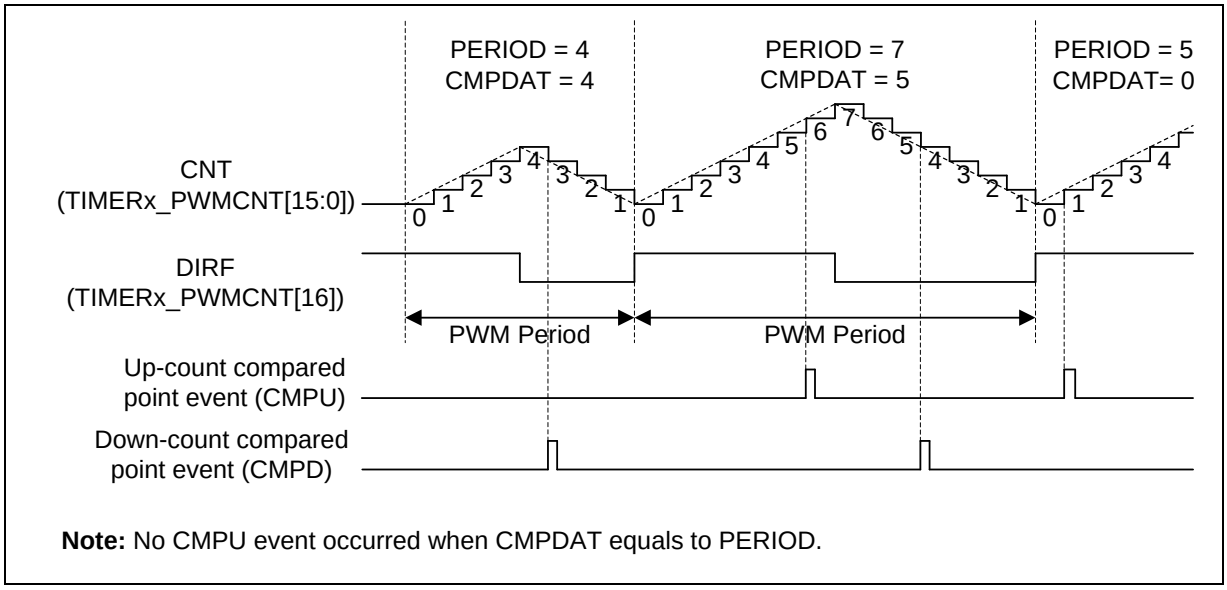


Figure 6.9-21 Timer0 ~ Timer3 PWM Comparator Events in Up-Down Count Type

In Timer4 and Timer5 PWM, this event will generate PWM output pulse, interrupt signal wake-up or trigger EADC, PDMA to start conversion, and the PWM period as shown in Figure 6.9-22.

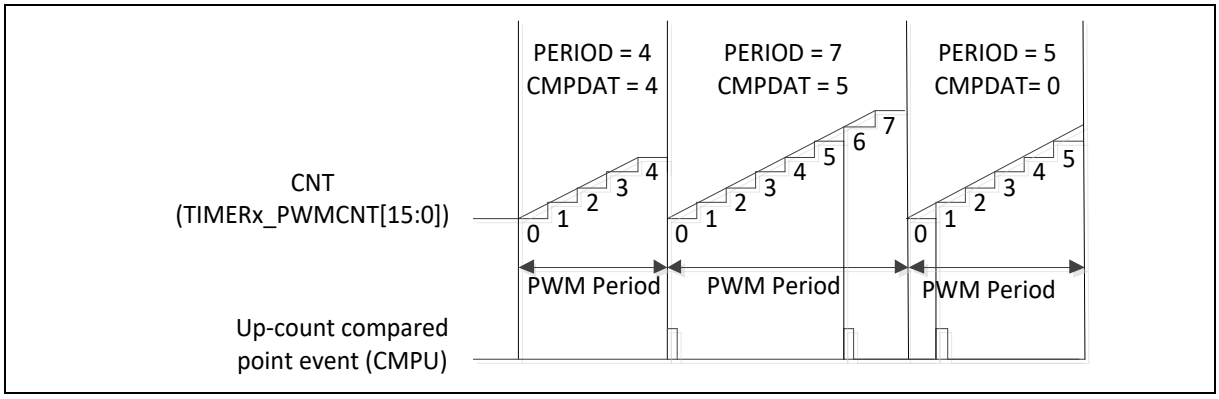


Figure 6.9-22 Timer4 and Timer5 PWM Comparator Events in Up Count Type

6.9.6.8 Period Loading Mode

In Timer0 ~ Timer3 PWM, sets the IMMLDEN (TIMERx_PWMCTL[9]) 0 to select Timer0 ~ Timer3 PWM operates at period loading mode. The loading mode of Timer4 and Timer5 PWM is fixed at period loading mode.

The PWM provides PBUF (TIMERx_PWMPBUF[15:0]) is the active PERIOD buffer register and CMPBUF (TIMERx_PWMCMPBUF[15:0]) is the active CMP buffer register. In period loading mode, both PERIOD (TIMERx_PWMPERIOD[15:0]) and CMP (TIMERx_PWMCMPDAT[15:0]) will load to their active PBUF and CMPBU register while each PWM period is completed. Figure 6.9-23 shows period loading timing of up count type, where PERIOD DATA0 denotes the initial data of PERIOD, PERIOD DATA1 denotes the first updated PERIOD data by user and so on, CMP also follows this rule. The following steps are the sequence of Figure 6.9-23.

1. User writes CMP DATA1 to CMP at point 1.
2. Period loading CMP DATA1 to CMPBUF at the end of PWM period at point 2.

3. User writes PERIOD DATA1 to PERIOD at point 3.
4. Period loading PERIOD DATA1 to PBUF at the end of PWM period at point 4.
5. User writes PERIOD DATA2 to PERIOD at point 5.
6. Period loading PERIOD DATA2 to PBUF at the end of PWM period at point 6.

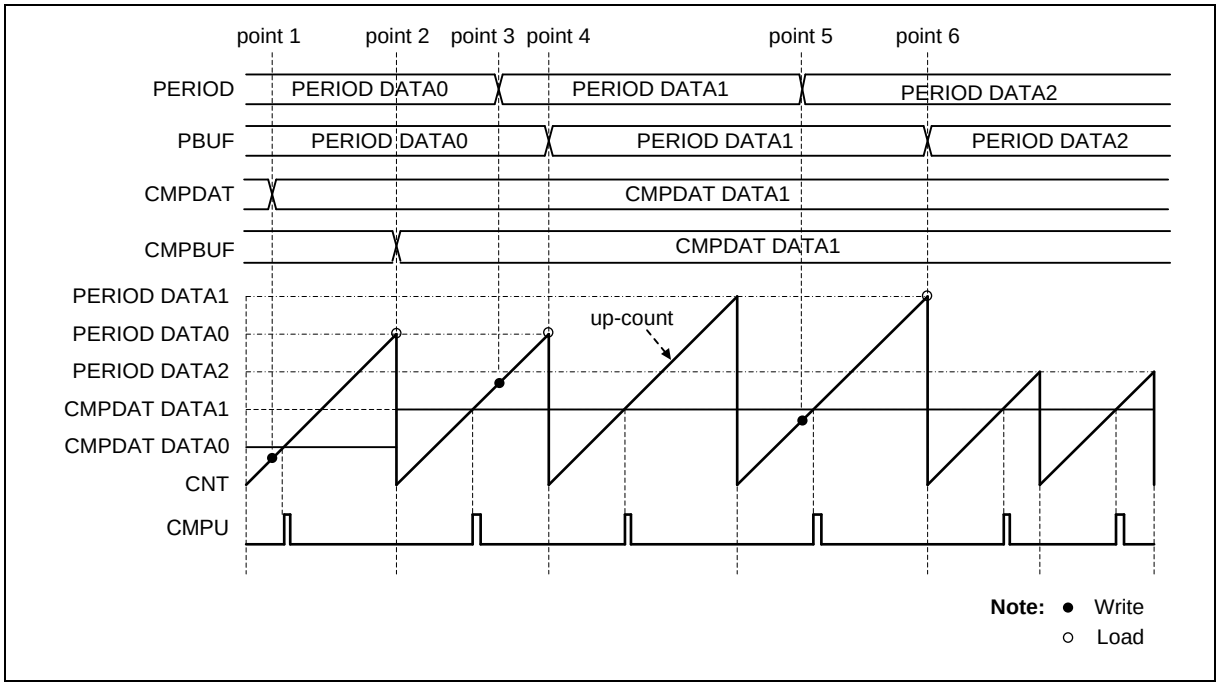


Figure 6.9-23 Timer0 ~ Timer5 PWM Period Loading Mode with Up Count Type

6.9.6.9 Immediately Loading Mode

The immediately loading mode is only available in Timer0 ~ Timer3 PWM.

When the IMMLDEN (TIMERx_PWMCTL[9]) bit set to 1, PWM operates at immediately loading mode. In immediately loading mode, when user update PERIOD (TIMERx_PWMPERIOD[15:0]) or CMP (TIMERx_PWMCMPDAT[15:0]), PERIOD or CMP will be load to active PBUF (TIMERx_PWMPBUF[15:0]) or CMPBUF (TIMERx_PWMCMPBUF[15:0]) after current counter count is completed. If the update PERIOD value is less than current counter value, counter will count wraparound. The following steps are the sequence of Figure 6.9-24.

1. User writes CMP DATA1 at point 1 and hardware will load CMP DATA1 to CMPBUF after current counter count is completed.
2. User writes PERIOD DATA1 at point 2 and PERIOD DATA1 is greater than current counter value, PWM counter will continuously count until equal to PERIOD DATA1 to complete one PWM period.
3. User writes PERIOD DATA2 at point 3 and PERIOD DATA2 is less than the current counter value, PWM counter will continuously count to maximum counter value 0x1FFFF and wraparound from 0x10000 to PERIOD DATA2 to complete one PWM period.

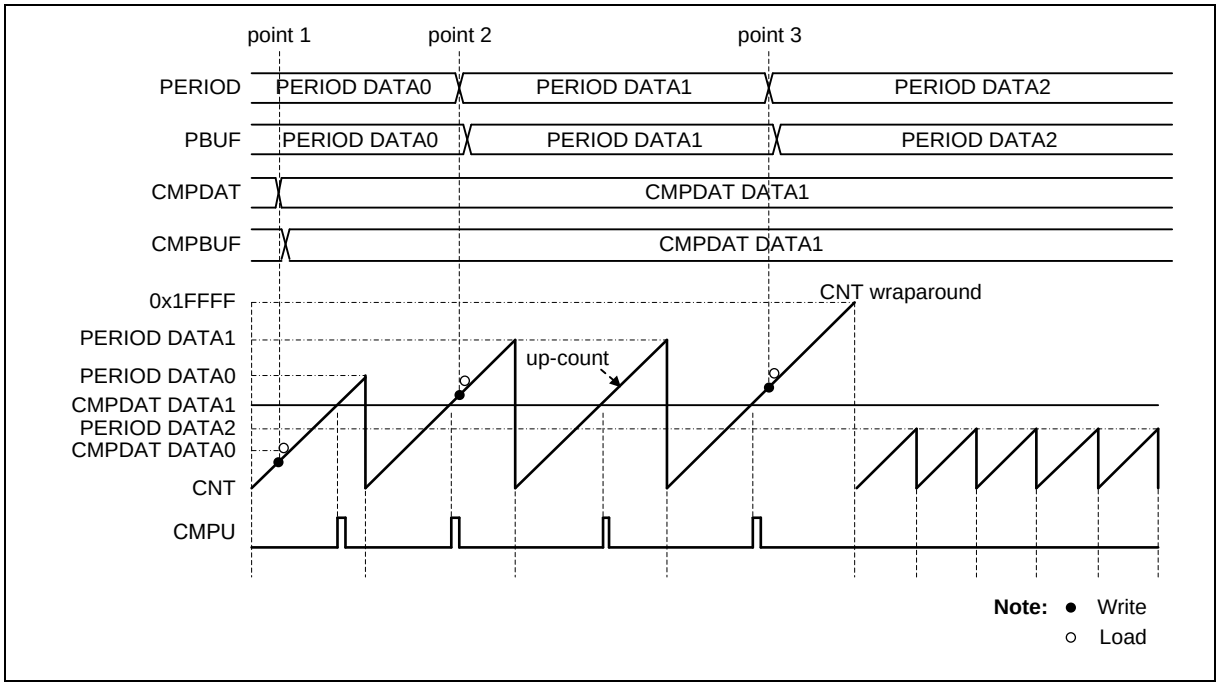


Figure 6.9-24 Timer0 ~ Timer3 PWM Immediately Loading Mode with Up Count Type

6.9.6.10 PWM Pulse Generator

The PWM pulse generator uses counter and comparator events to generate PWM output pulse.

In Timer0 ~ Timer3 PWM, the events are zero point and period point in up count type and down count type, center point in up-down count type and counter equal to comparator point in three count types. Each event point can generate PWM output waveform in different count type as shown in Figure 6.9-25, Figure 6.9-26 and Figure 6.9-27.

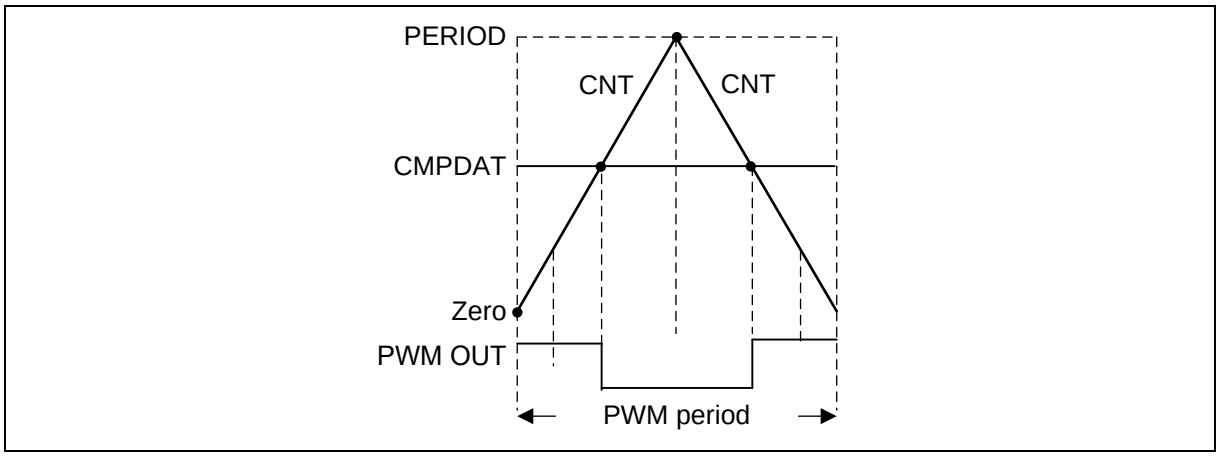


Figure 6.9-25 Timer0 ~ Timer3 PWM Pulse Generation in Up-Down Count Type

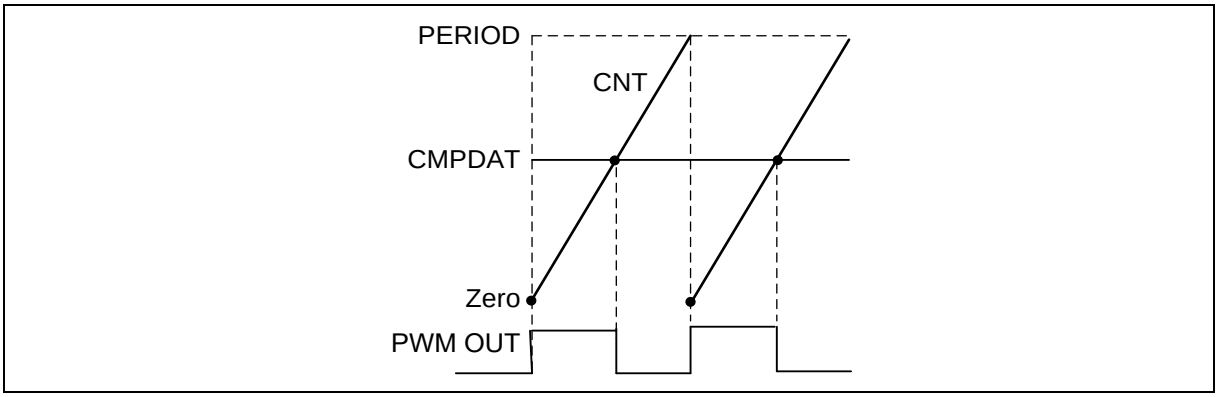


Figure 6.9-26 Timer0 ~ Timer3 PWM Pulse Generation in Up Count Type

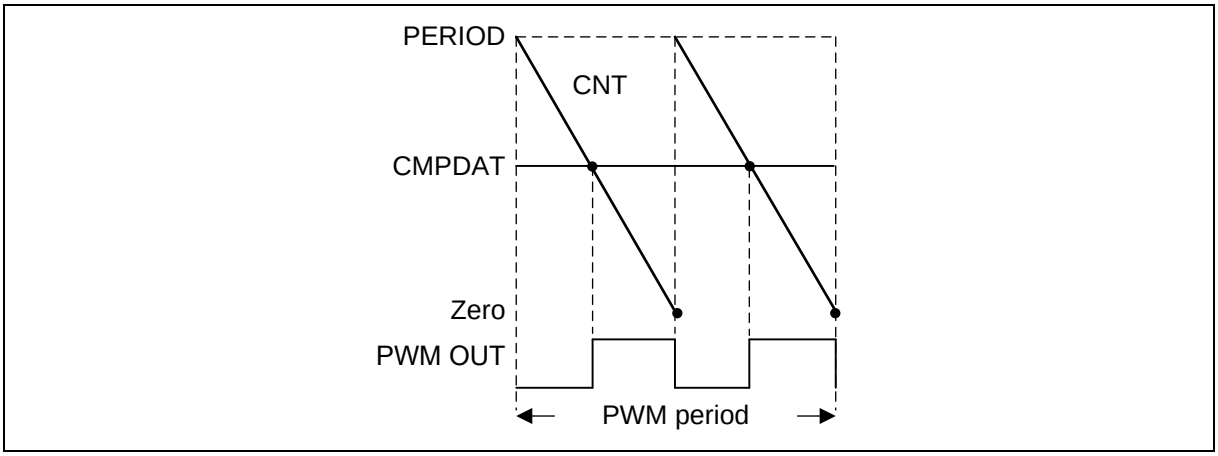


Figure 6.9-27 Timer0 ~ Timer3 PWM Pulse Generation in Down Count Type

The PWM generation events may be sometimes generated at the same time, as the reason, events priority between different counter types should be taken care are listed in Table 6.9-4, Table 6.9-5 and Table 6.9-6, including event priority in up count type, event priority in down count type and event priority in up-down count type.

Priority	Zero And CMPU Point Event (CMP = 0)	PWM Output
1 (High)	Compare up event	Low
2 (Low)	Zero event	High

Table 6.9-4 Timer0 ~ Timer3 PWM Pulse Generation Event Priority in Up Count Type

Priority	Zero And CMPD Point Event (CMP = 0)	PWM Output
1 (High)	Zero event	Low
2 (Low)	Compare down event	High
Priority	Period and CMPD point event	PWM output

	(CMP = PERIOD)	
1 (High)	Compare down event	High
2 (Low)	Period event	Low

Table 6.9-5 Timer0 ~ Timer3 PWM Pulse Generation Event Priority in Down Count Type

Priority	**CMPU And CMPU Point Event (CMP = PERIOD)**	**PWM Output**
1 (High)	Compare down event	High
2 (Low)	Compare up event	Low

Table 6.9-6 Timer0 ~ Timer3 PWM Pulse Generation Event Priority in Up-Down Count Type

According to event priority limitation, the PWM generator can support 0% and 100% duty cycle PWM output waveform only in up count and up-down count type. Figure 6.9-28 is an example about PWM duty cycle from 0% to 100% in up count type and up-down count type where PERIOD is 4 with different CMP value.

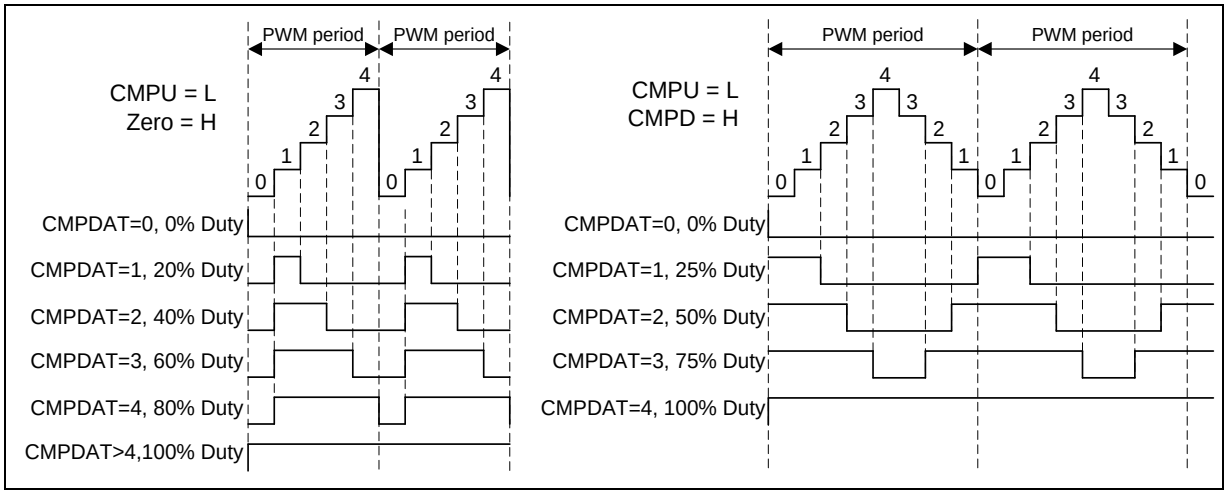


Figure 6.9-28 Timer0 ~ Timer3 PWM 0% to 100% Duty Cycle in Up Count and Up-Down Count Type

In Timer4 and Timer5 PWM, the events are period point and counter equal to comparator point in up count type. Each event point can generate PWM output waveform in up count type as shown in Figure 6.9-29.

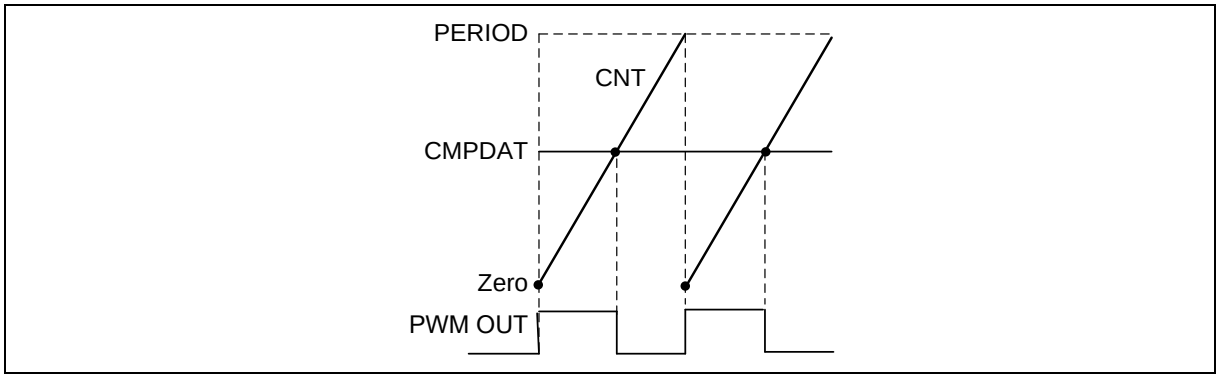


Figure 6.9-29 Timer4 and Timer5 PWM Pulse Generation in Up Count Type

The PWM output pulse is associated with CMPBUF (TIMERx_PWMCMPBUF[15:0]) and PBUF (TIMERx_PWMPBUF[15:0]). The rules are listed as below.

1. If CMPBUF is zero, PWM output is 100% low.
2. If CMPBUF > PBUF, PWM output is 100% high.
3. If 0 < CMPBUF ≦ PBUF, PWM output high/low duty is according to counter value (TIMERx_PWMCNT[15:0]).
   1) If PWM counter is higher than CMPBUF or equal to CMPBUF, PWM output is low.
   2) If PWM counter is lower than CMPBUF, PWM output is high.

The PWM output level table is shown as Table 6.9-7.

Conditions	CMPBUF = 0	CMPBUF > PBUF	CNT ≧ CMPBUF	CNT < CMPBUF
PWM Output	Low	High	Low	High

Table 6.9-7 Timer4 and Timer5 PWM Output Level

Figure 6.9-30 is an example about PWM duty cycle from 0% to 100% in up count type where PERIOD is 4 with different CMP value.

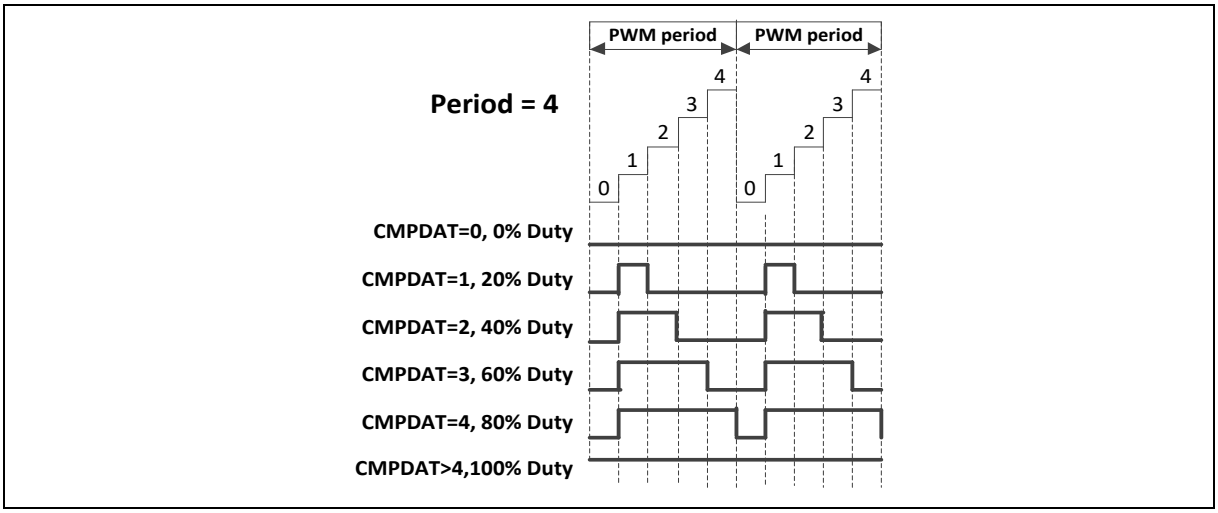


Figure 6.9-30 Timer4 and Timer5 PWM 0% to 100% Duty Cycle in Up Count Type

6.9.6.11 PWM Output Mode

The Timer0 ~ Timer3 PWM supports two output modes: independent mode which may be applied to DC motor system, complementary mode with dead-time insertion which may be used in the application of AC induction motor and permanent magnet synchronous motor.

The Timer4 and Timer5 PWM only supports independent mode.

6.9.6.12 Independent Mode

In Timer0 ~ Timer3 PWM, sets OUTMODE (TIMERx_PWMCTL[16]) bit to 0, PWM output operates in independent mode. In this mode, both PWMx_CH0 and PWMx_CH1 can output the same waveform as shown in Figure 6.9-31.

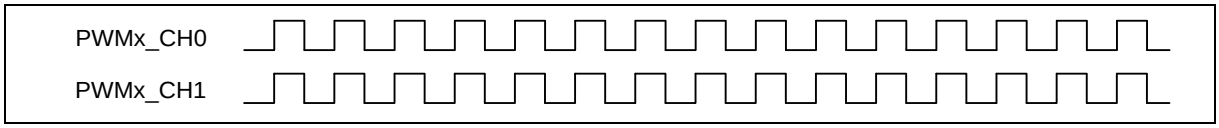


Figure 6.9-31 PWM Independent Mode Output Waveform

In Timer4 ~ Timer5 PWM, only PWMx_CH0 channel can output the waveform in independent mode as shown in Figure 6.9-31.

6.9.6.13 Complementary Mode

The complementary mode is only available in Timer0 ~ Timer3 PWM.

When OUTMODE (TIMERx_PWMCTL[16]) bit is set to 1, PWM output operates in complementary mode. In this mode, both PWMx_CH0 and PWMx_CH1 can output waveform and PWMx_CH1 must always be the complement of PWMx_CH0 as shown in Figure 6.9-32.

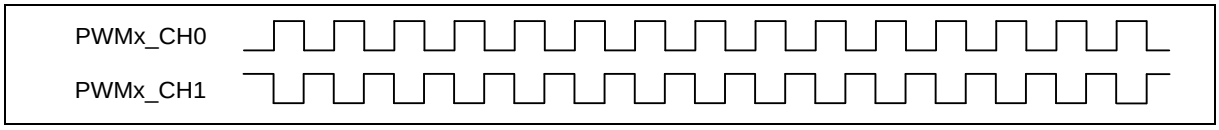


Figure 6.9-32 Timer0 ~ Timer3 PWM Complementary Mode Output Waveform

6.9.6.14 PWM Output Control

In the Timer0 ~ Timer3 PWM, after PWM pulse generator, there are four steps to control output waveform in independent output mode and five control steps in complementary output mode. User can set POEN0 (TIMERx_PWMPOEN[0]) and POEN1 (TIMERx_PWMPOEN[1]) 1 to enable PWMx_CH0 and PWMx_CH1 output waveform.

In Independent mode, there are mask control, brake control, polarity control and output enable control to control output waveform as shown in Figure 6.9-33.

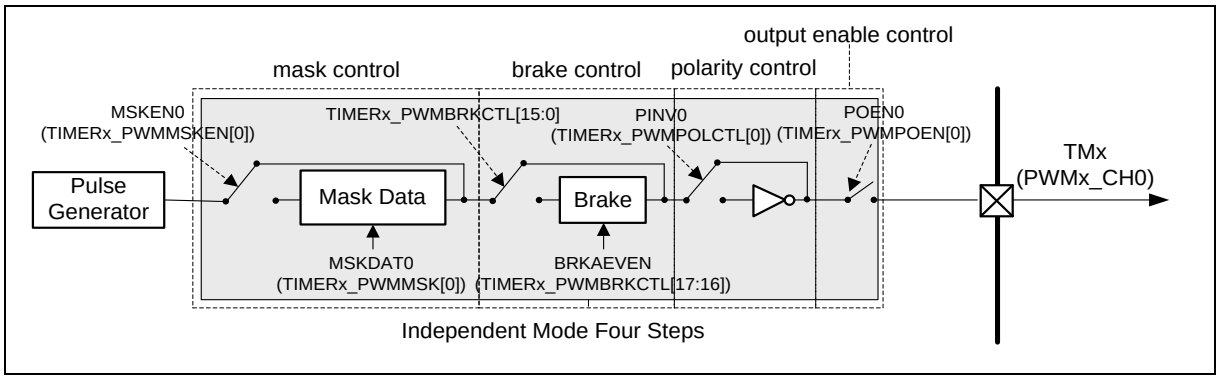


Figure 6.9-33 Timer0 ~ Timer3 PWMx_CH0 Output Control in Independent Mode

In complementary mode, there are dead-time insertion control and four control steps the same as independent mode to control PWMx_CH0 and PWMx_CH1 outputs as shown in Figure 6.9-34.

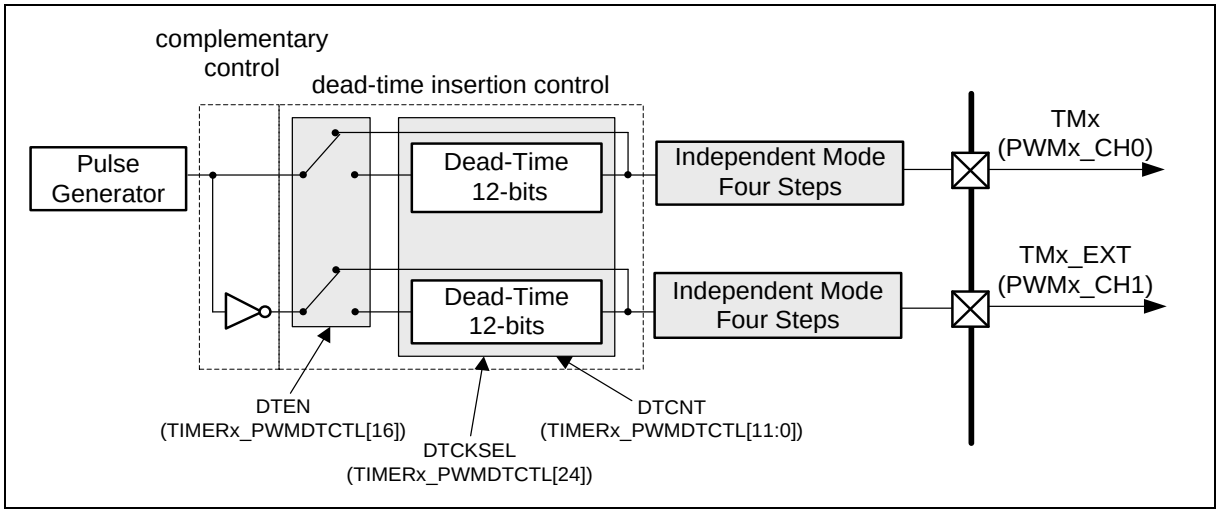


Figure 6.9-34 Timer0 ~ Timer3 PWMx_CH0 and PWMx_CH1 Output Control in Complementary Mode

In Timer4 and Timer5, after PWM pulse generator, there are three steps to control output waveform. User can set POEN0 (TIMERx_PWMPOEN[0]) 1 to enable PWMx_CH0 output waveform and set POSEL (TIMERx_PWMPOEN[8]) to select TMx or TMx_EXT pin as PWMx_CH0 output pin.

There are polarity control, output enable control and output channel select to control output waveform as shown in Figure 6.9-35.

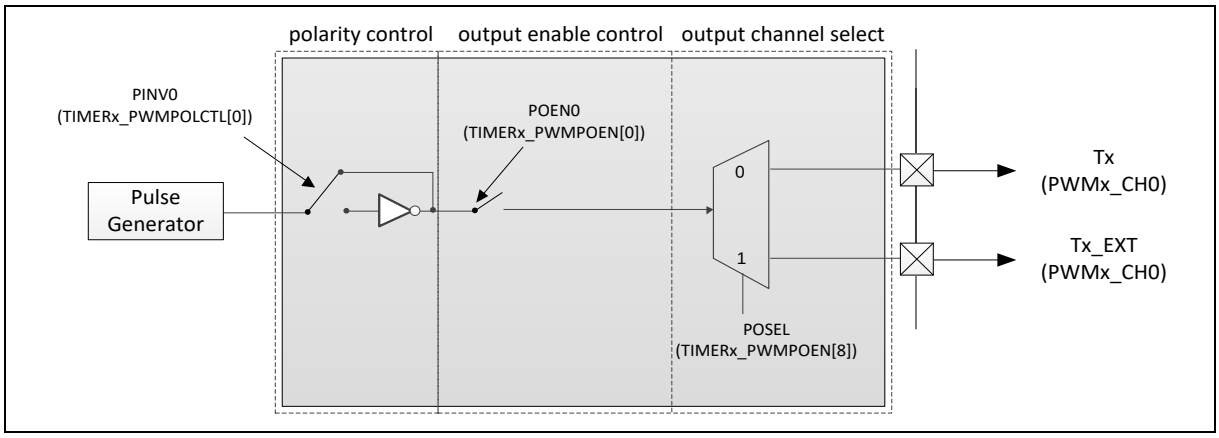


Figure 6.9-35 Timer4 and Timer5 PWM Output PWMx_CH0 Control

6.9.6.15 Dead-Time Insertion Control

The dead-time insertion is only available in Timer0 ~ Timer3 PWM.

In the complementary application, the complement channels may drive the external devices like power switches. The dead-time generator inserts a low level interval between complementary outputs PWMx_CH0 and PWMx_CH1 as shown in Figure 6.9-36. User sets DTEN (TIMERx_PWMDTCTL[16]) bit to enable dead-time control function, DTCNT (TIMERx_PWMDTCTL[11:0]) and DTCKSEL (TIMERx_PWMDTCTL[24]) to control dead-time interval. The dead-time interval can be calculated from the following formula:

Dead-time interval = (DTCNT + 1) * TMRx_PWMCLK period, if DTCKSEL is 0

Dead-time interval = (DTCNT + 1) * TMRx_PWMCLK * (CLKPSC + 1) period, if DTCKSEL is 1

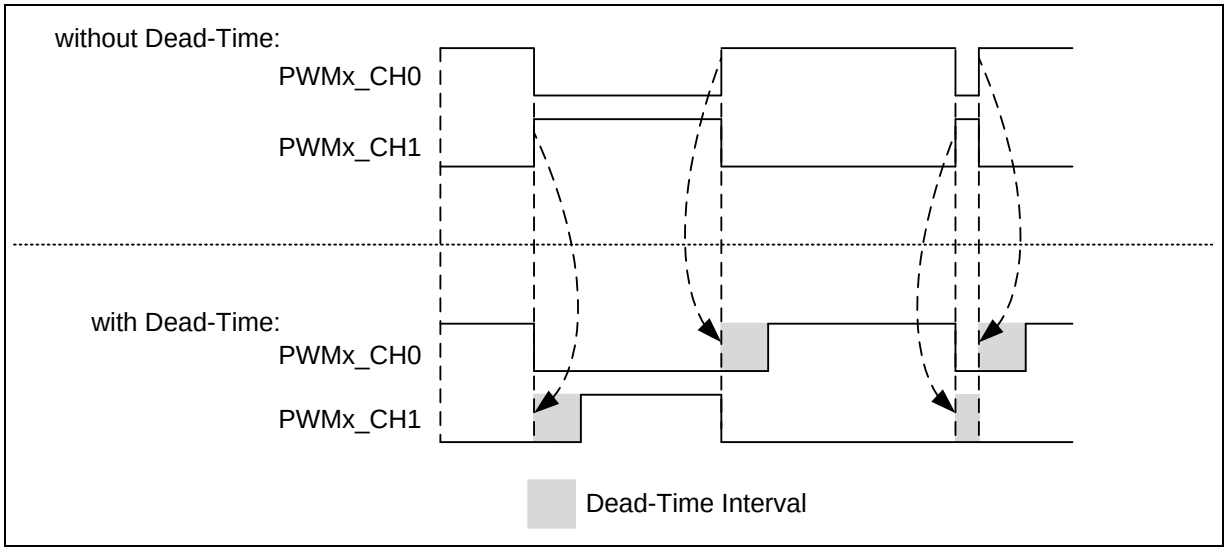


Figure 6.9-36 Timer0 ~ Timer3 PWM Dead-Time Insertion

6.9.6.16 PWM Mask Output Control

The PWM mask output control is only available in Timer0 ~ Timer3 PWM.

PWMx_CH0/CH1 output value can be masked to specified logic states by setting MSKEN0/1 (TIMERx_PWMMSKEN[1:0]) and MSKDAT0/1 (TIMERx_PWMMSK[1:0]). The PWM output mask function is useful when controlling various types of Electrically Commutated Motor (ECM) like a BLDC motor. Figure 6.9-37 shows an example of PWM output mask control in PWMx_CH0 and PWMx_CH1.

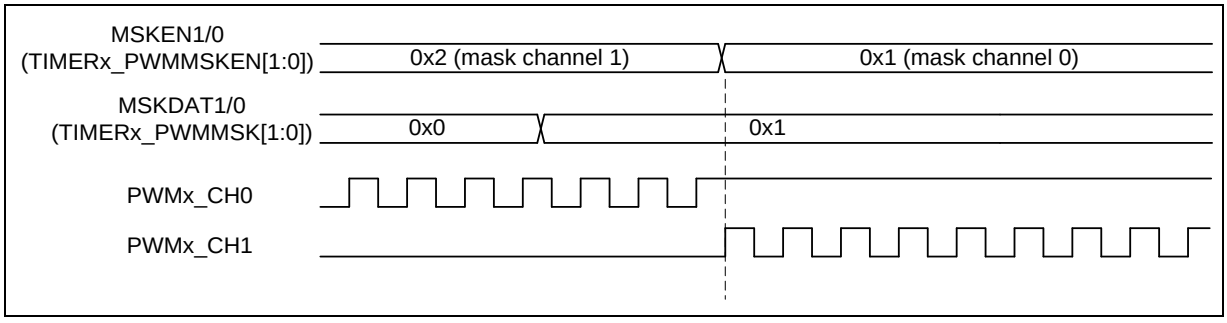


Figure 6.9-37 Timer0 ~ Timer3 PWM Output Mask Control Waveform

6.9.6.17 PWM Brake Control

The PWM brake control is only available in Timer0 ~ Timer3 PWM.

Each PWM generator supports one external input brake pin as PWM brake event source. User can select active brake pin source in BKPINSRC (TIMERx_PWMBNF[17:16]), PWMx_BRAKEy (x=0,1 and y=0,1). There is a 3-bit noise filter counter to filter the external brake pin signal. User can enable BRKNFEN (TIMERx_PWMBNF[0]) to enable the brake pin noise filter function and the noise filter sampling clock can be selected by setting BRKNFSEL (TIMERx_PWMBNF[3:1]) to fit different noise properties. Moreover, by setting BRKFCNT (TIMERx_PWMBNF[6:4]), user can define by how many sampling clock cycles a filter will recognize the effective edge of the brake pin signal. In addition, brake pin polar can be inversed by setting BRKPINV (TIMERx_PWMBNF[7]) to realize the polarity setup for the brake control signals. Set BRKPINV to 0, brake event will occur when PWMx_BRAKEy pin status from low to high; set BRKPINV to 1, brake event will occur when PWMx_BRAKEy pin status from high to low.

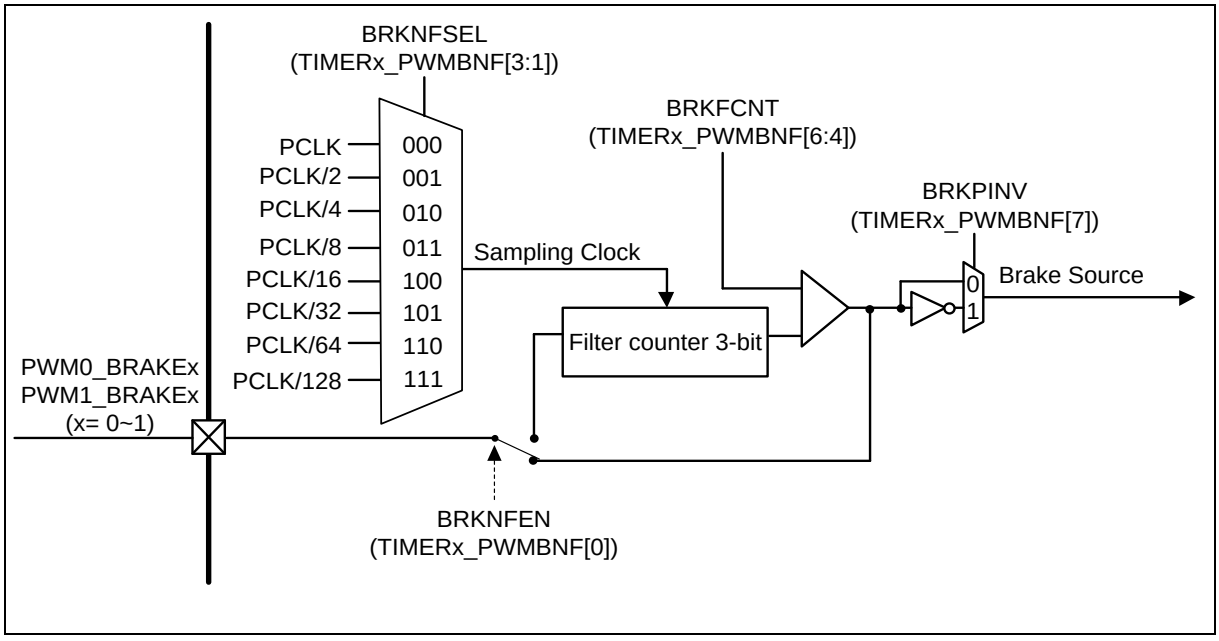


Figure 6.9-38 Timer0 ~ Timer3 PWM Brake Pin Noise Filter Block Diagram

User can set BRKAEVEN (TIMERx_PWMBRKCTL[17:16]) for PWMx_CH0 output state and BRKAODD (TIMERx_PWMBRKCTL[19:18]) for PWMx_CH1 output state when PWM brake event happened. There are two brake detector sources, edge detect brake source and level detect brake source when brake event happened. Figure 6.9-39 shows the brake event block diagram for PWMx_CH0 and PWMx_CH1.

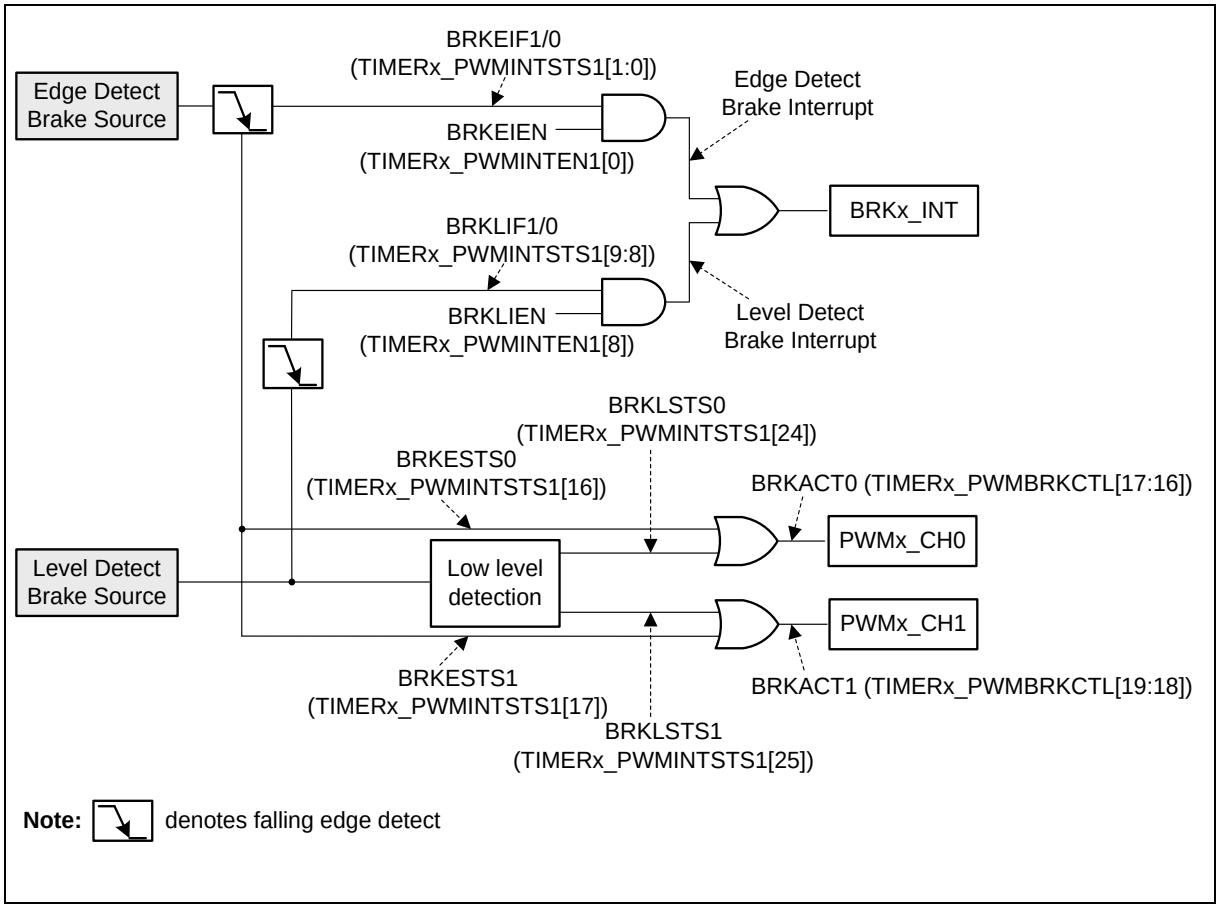


Figure 6.9-39 Timer0 ~ Timer3 PWM Brake Event Block Diagram

When the edge detector detects the brake signal, the brake function generates interrupt status for PWMx_CH1/0 is BRKEIF1/0 (TIMERx_PWMINTSTS1[1:0]) and brake event status for PWMx_CH1/0 is BRKESTS1/0 (TIMERx_PWMINTSTS1[17:16]). The interrupt status BRKEIF1/0 can be cleared by writing 1 to it, and the brake event status BRKESTS1/0 will keep until the next PWM period starts when corresponding BRKEIF1/0 flag has been cleared and PWM generator can resume normal output.

Figure 6.9-40 shows an example of edge detector brake waveform for PWMx_CH0 and PWMx_CH1. In this case, the edge detect brake source has occurred twice for the brake events. When the first brake event occurs, both of the BRKEIF0 and BRKEIF1 flags are set and BRKESTS0 and BRKESTS1 status are also set to indicate brake state of PWMx_CH0 and PWMx_CH1. For the first occurring event, user writes 1 to clear the BRKEIF0. After that, the BRKESTS0 is cleared by hardware at the next start of the PWM period and the PWMx_CH0 outputs the normal waveform even though the edge brake event is still occurring. At the same time, BRKESTS1 keep 1 and PWMx_CH1 keep outputs low in brake state. The second event also triggers the same flags, but at this time, user writes 1 to clear the BRKEIF1. Afterward, PWMx_CH1 outputs normally at the next start of the PWM period.

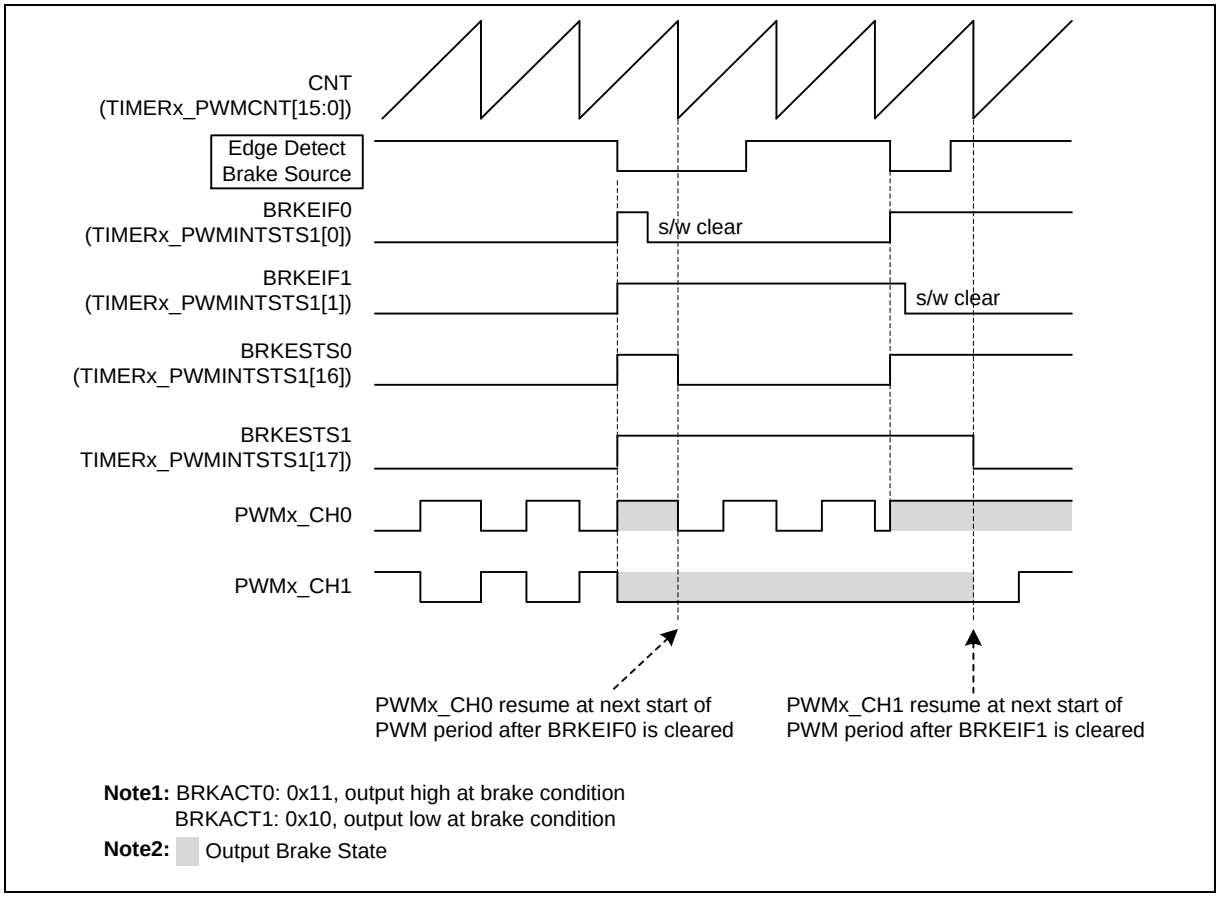


Figure 6.9-40 Timer0 ~ Timer3 PWM Edge Detector Brake Waveform

When the level detector detects the brake signal, the brake function generates interrupt status for PWMx_CH1/0 is BRKLIF1/0 (TIMERx_PWMINTSTS1[9:8]) and brake event status for PWMx_CH1/0 is BRKLSTS1/0 (TIMERx_PWMINTSTS1[25:24]). The interrupt status BRKLIF1/0 can be cleared by writing 1 to it, and the brake event status BRKLSTS1/0 will be cleared only when current period is completed and brake condition removed, then PWM generator can resume normal output when next PWM period starts.

Figure 6.9-41 shows an example of level detector brake waveform for PWMx_CH0 and PWMx_CH1. In this case, the BRKLIF0 and BRKLIF1 can only indicate the brake event has occurred, writes 1 to clear this flags will not affect BRKLSTS0 and BRKLSTS1 brake event status. Both BRKLSTS0 and BRKLSTS1 brake states will automatically cleared at the start of the next PWM period when level brake condition has released no matter BRKLIF0 and BRKLIF1 status.

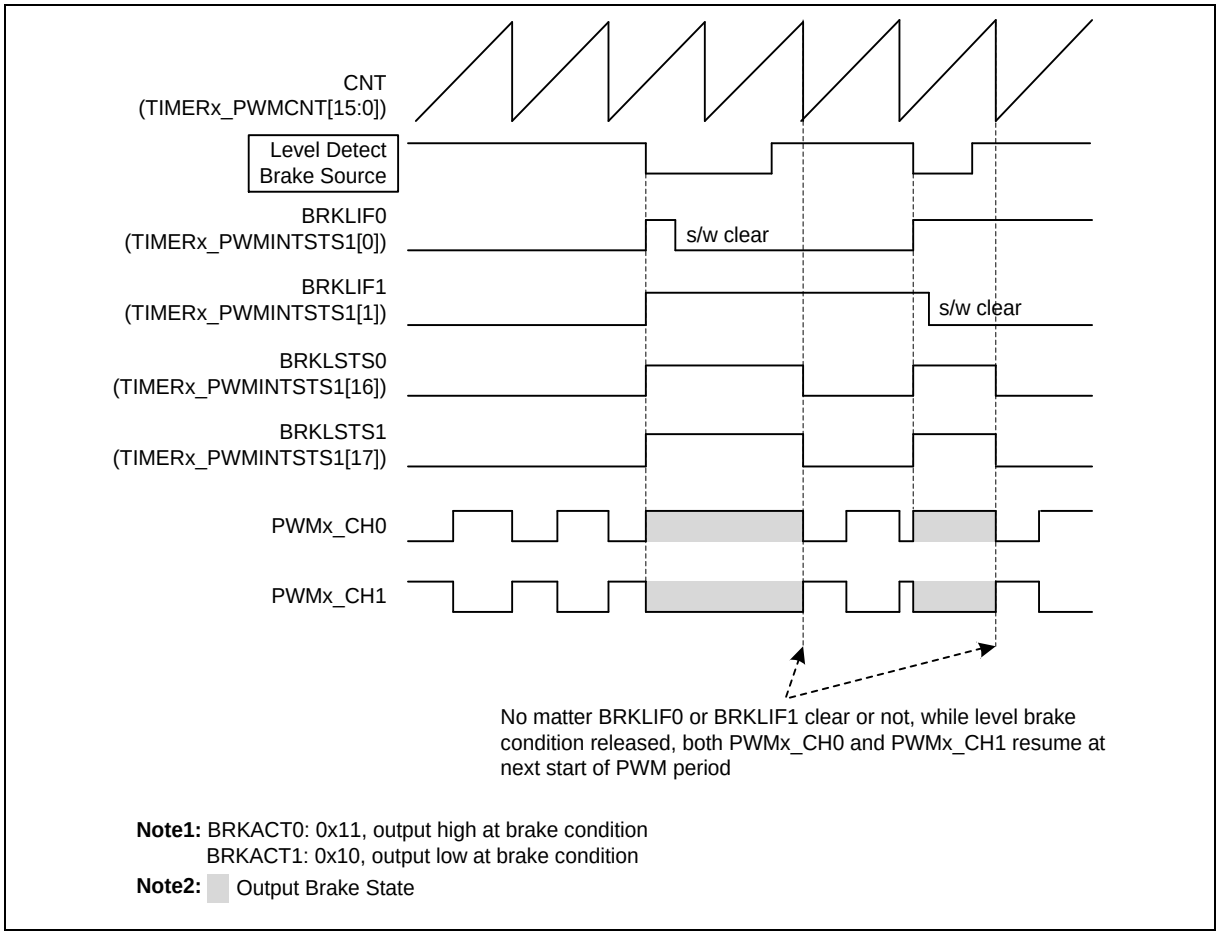


Figure 6.9-41 Timer0 ~ Timer3 PWM Level Detector Brake Waveform

The two kinds of detectors detecting the same five brake sources as shown in Figure 6.9-42 are one from PWMx_BRAKEy (x=0,1 and y=0,1) external input signals, two from internal ACMP comparator signals, one from system fail events and one from software trigger brake event. ACMP brake sources will be detected only when internal ACMP0_O or ACMP1_O signal from low to high.

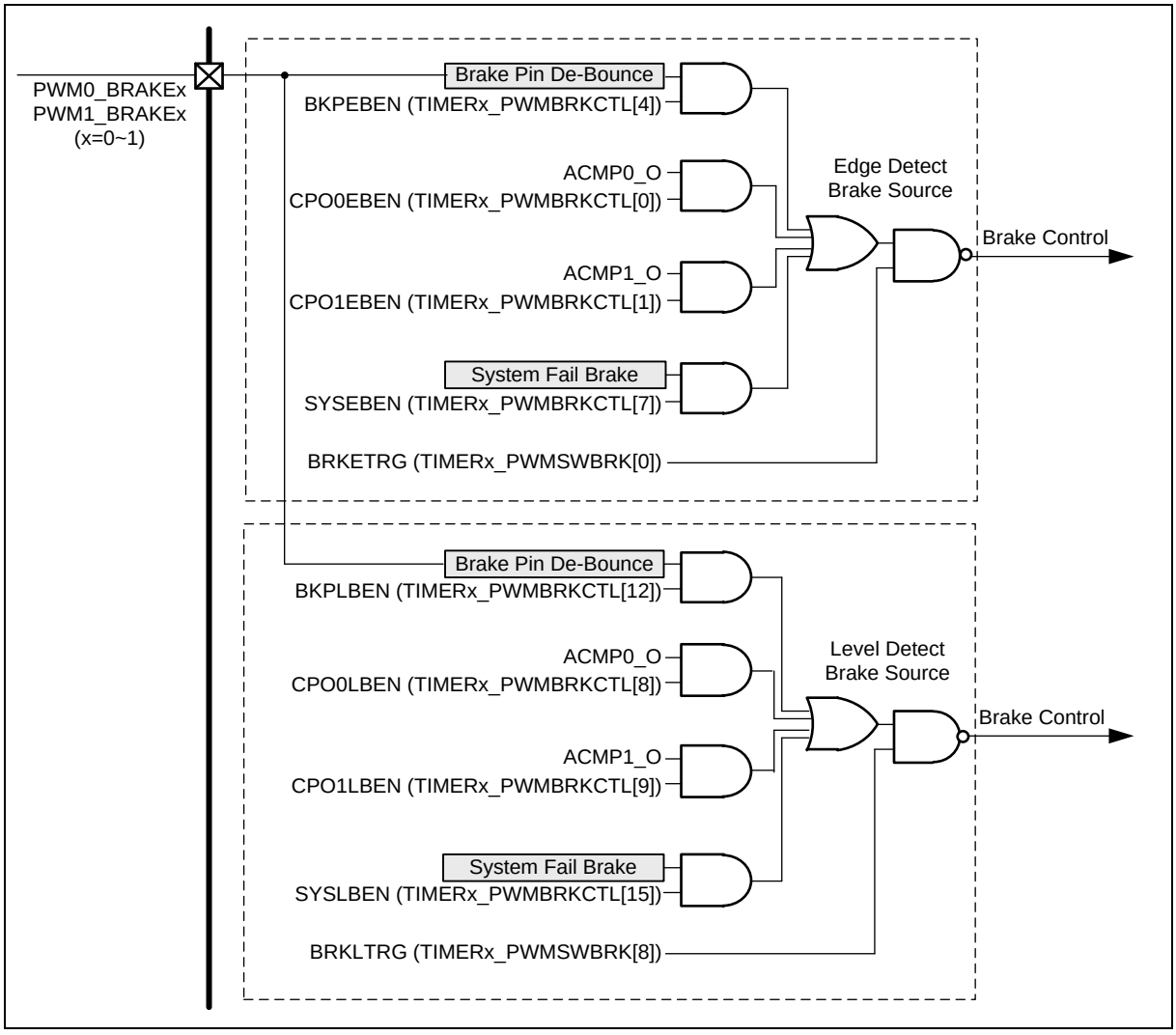


Figure 6.9-42 Timer0 ~ Timer3 PWM Brake Source Block Diagram

Among the above described brake sources, the brake source coming from system fail event can be specified to one of the different system fail conditions, these conditions include clock fail, BOD detect, SRAM parity error and CPU lockup as shown in Figure 6.9-43.

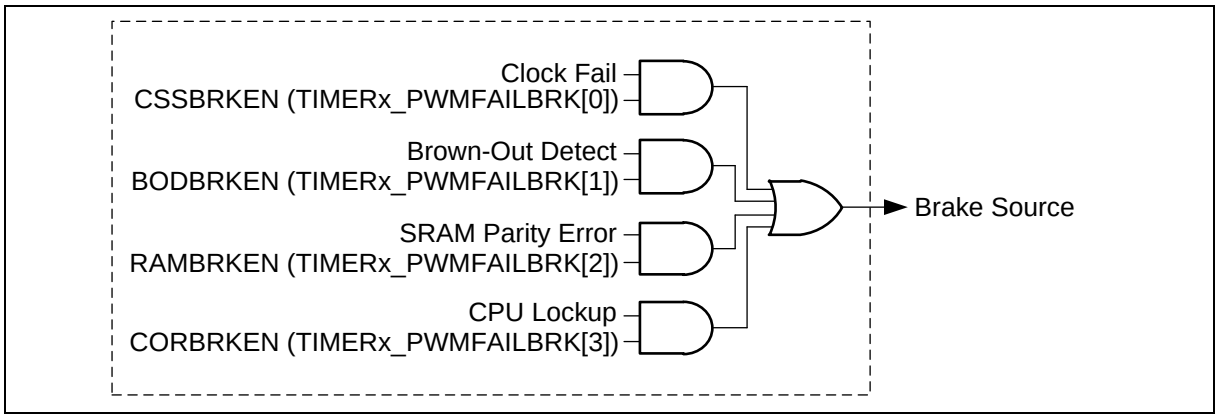
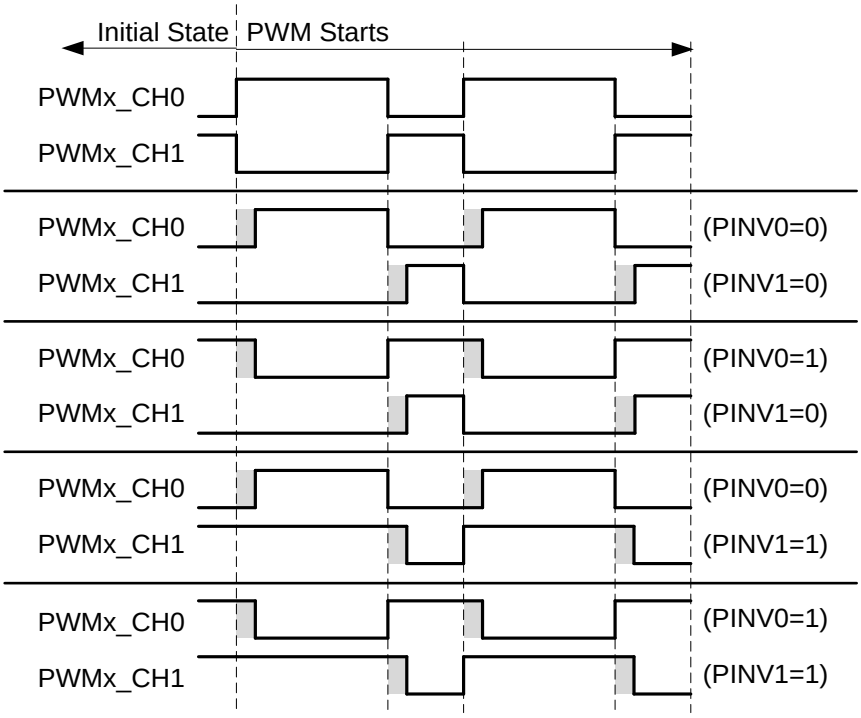


Figure 6.9-43 Timer0 ~ Timer3 System Fail Brake Block Diagram

6.9.6.18 Polarity Control

In Timer0 ~ Timer3 PWM, each PWMx_CH0 and PWMx_CH1 has an independent polarity control to configure the polarity of the active state of PWM output. User can control polarity state of PWMx_CH0 on PINV0 (TIMERx_PWMPOLCTL[0]) and PWMx_CH1 on PINV1 (TIMERx_PWMPOLCTL[1]). Figure 6.9-44 shows the PWMx_CH0 and PWMx_CH1 output with polarity control and dead-time insertion.

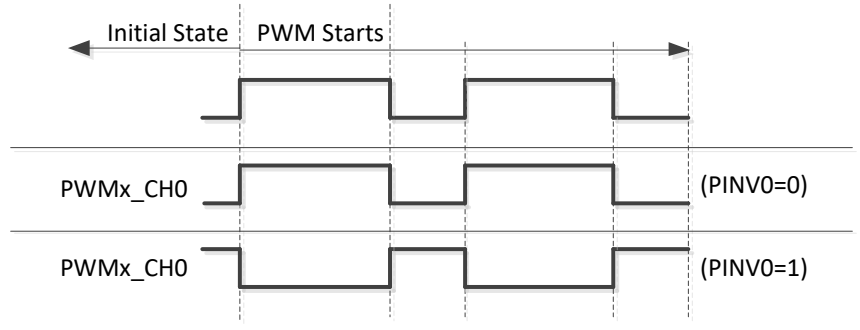


Note1: dead-time insertion

Note2: PINV0/PIV1, it controls the output polar inverse

Figure 6.9-44 Timer0 ~ Timer3 PWM Output Polarity Control with Dead-Time Insertion

In Timer4 and Timer5 PWM, the PWMx_CH0 (TMx or TMx_EXT) has an independent polarity control to configure the polarity of the active state of PWM output. User can control polarity state on PINV0 (TIMERx_PWMPOLCTL[0]). Figure 6.9-45 shows the PWMx_CH0 with polarity control.



Note: PINV controls the output polar inverse

Figure 6.9-45 Timer4 and Timer5 PWMx_CH0 Polarity Control

6.9.6.19 PWM Interrupt Generator

There are independent interrupts (PWMx_INT) for each Timer0 ~ Timer3 PWM as shown in Figure 6.9-46.

The Timer0 ~ Timer3 PWM counter can generate the zero point interrupt flag ZIF (TIMERx_PWMINTSTS0[0]) and the period point interrupt flag PIF (TIMERx_PWMINTSTS0[1]). When counter equals to the comparator value stored in CMP (TIMERx_PWMCMPDAT[15:0]), the different interrupt flags will be triggered depending on the counting direction. If counter and CMP matched occurs at up-count direction, the comparator up interrupt flag CMPUIF (TIMERx_PWMINTSTS0[2]) is set and if matched at down-count direction, the comparator down interrupt flag CMPDIF (TIMERx_PWMINTSTS0[3]) is set. If the corresponding interrupt enable bits are set, the interrupt trigger events will also generate interrupt signals. When PWM brake event occurred, the relatives interrupt event will be triggered according to PWM brake settings.

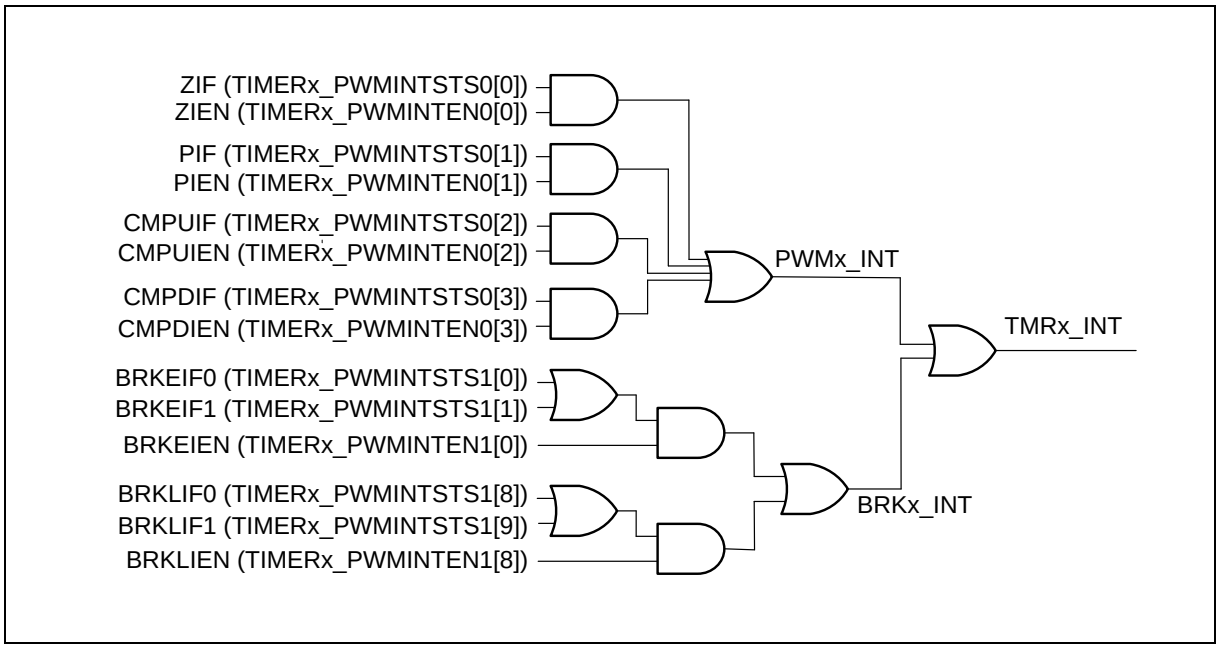


Figure 6.9-46 Timer0 ~ Timer3 PWM Interrupt Architecture Diagram

There are independent interrupts (PWMx_INT) for each Timer4 and Timer5 PWM as shown in Figure 6.9-47.

The Timer4 and Timer5 PWM counter can generate the period point interrupt flag PIF (TIMERx_PWMINTSTS0[1]). When counter equals to the comparator value stored in CMP (TIMERx_PWMCMPDAT[15:0]) at up-count direction, the comparator up interrupt flag CMPUIF (TIMERx_PWMINTSTS0[2]) is set. If the corresponding interrupt enable bits are set, the interrupt trigger events will also generate interrupt signals.

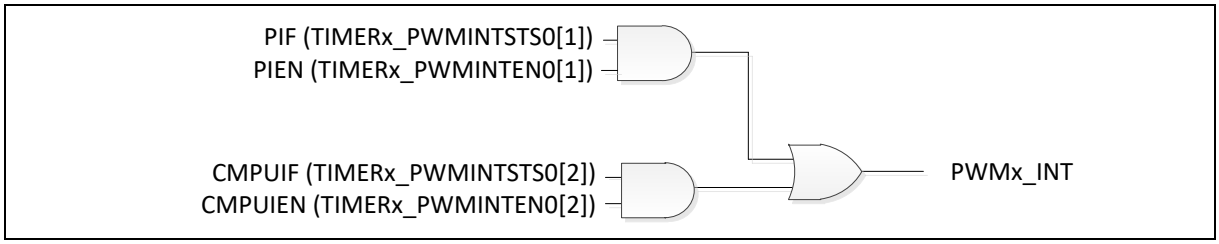


Figure 6.9-47 Timer4 and Timer5 PWM Interrupt Architecture Diagram

6.9.6.20 PWM Wake-up Generator

The PWM wake-up function is only available in Timer4 and Timer5 PWM.

User can sets WKEN (TIMERx_PWMCTL[12]) high to wake up CPU when PWM interrupt occurs. Before enter power-down mode, user must select LXT, LIRC or MIRC as PWM clock source.

When PWM wake-up occurs, WKF (TIMERx_PWMSTATUS[8]) is set to 1. User can write 1 to this bit to clear this flag.

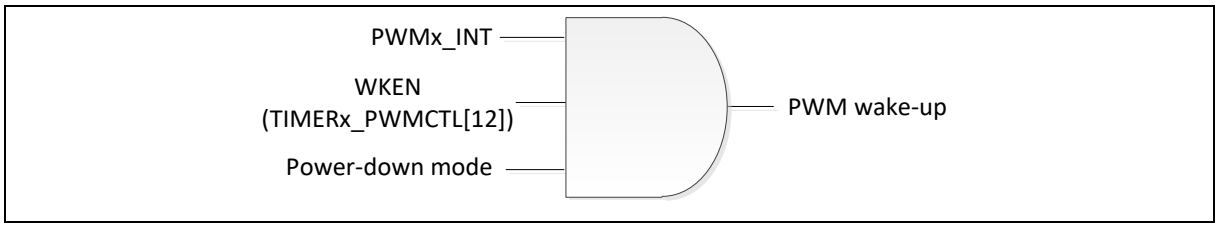


Figure 6.9-48 Timer4 and Timer5 PWM Wake-up Architecture Diagram

6.9.6.21 PWM Trigger EADC, PDMA Generator

The Timer0 ~ Timer3 PWM counter event can be the EADC conversion trigger source. User sets TRGSEL (TIMERx_PWMTRGCTL[2:0]) to select which PWM counter event can trigger EADC conversion after TRGEADC (TIMERx_PWMTRGCTL[7]) is enabled.

There are five PWM counter events that can be selected as the trigger source to start EADC conversion as shown in Figure 6.9-49.

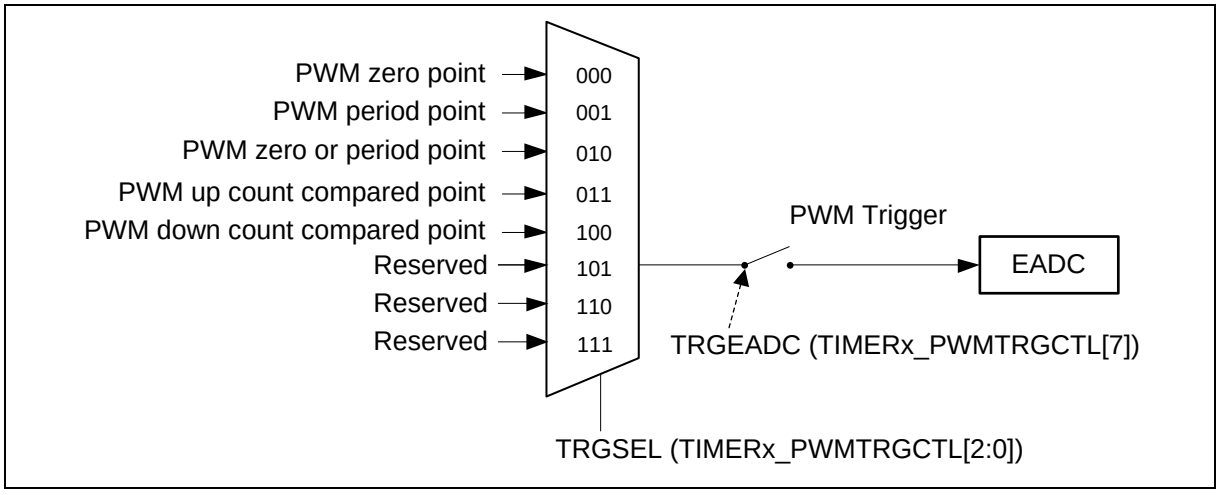


Figure 6.9-49 Timer0 ~ Timer3 PWM Trigger EADC Block Diagram

The Timer4 and Timer5 PWM counter event can be one of the EADC, PDMA conversion trigger source. User sets TRGSEL (TIMERx_PWMTRGCTL[2:0]) to select which PWM counter event can trigger conversion.

When the TRGEADC (TIMERx_PWMTRGCTL[7]) is set to 1, the PWM counter event can trigger EADC conversion. When the TRGPDMA (TIMERx_TRGCTL[9]) is set to 1, the PWM PWM counter event can trigger PDMA conversion.

There are three PWM counter events can be selected as the trigger source to start conversion as shown in Figure 6.9-50.

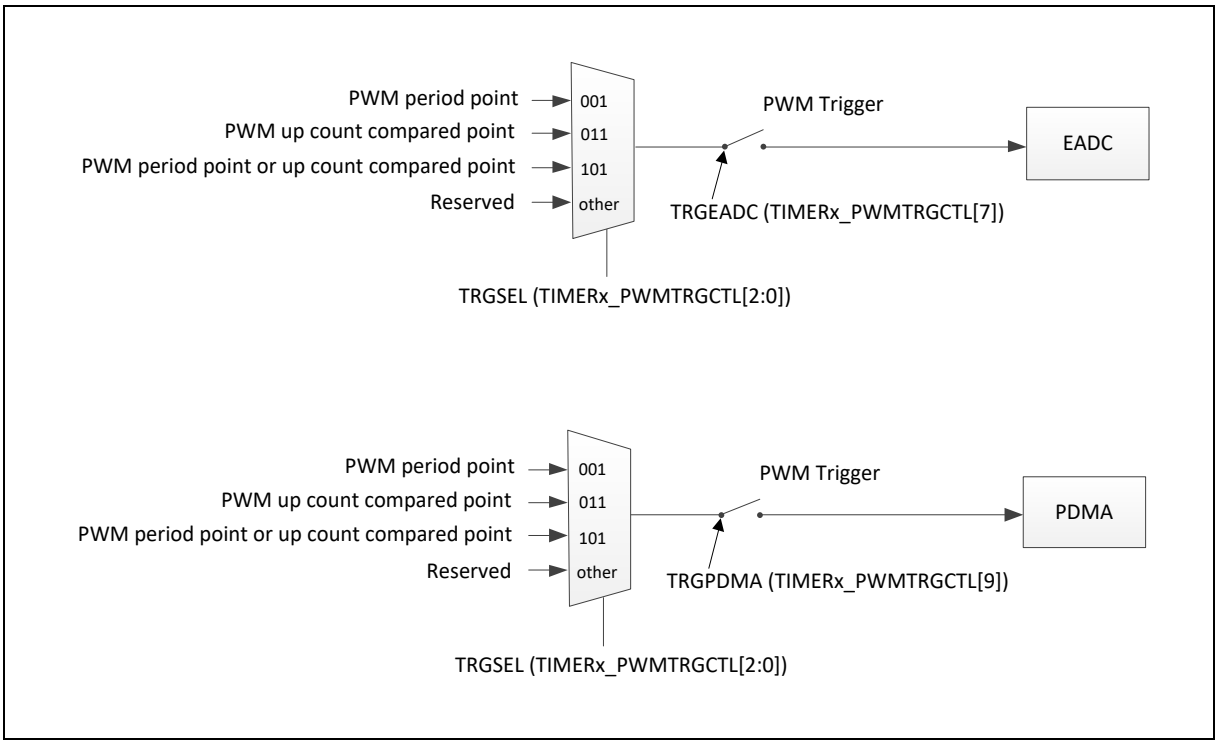


Figure 6.9-50 Timer4 and Timer5 PWM Trigger EADC, PDMA Block Diagram

6.9.7 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
TIMER Base Address: **TMR01_BA = 0x4005_0000** **TMR23_BA = 0x4005_1000** **TMR45_BA = 0x4005_2000** **TMR23 non-secure base address is TMR23_BA + 0x1000_0000** **TMR45 non-secure base address is TMR45_BA + 0x1000_0000**				
TIMER0_CTL	TMR01_BA+0x00	R/W	Timer0 Control Register	0x0000_0005
TIMER0_CMP	TMR01_BA+0x04	R/W	Timer0 Comparator Register	0x0000_0000
TIMER0_INTSTS	TMR01_BA+0x08	R/W	Timer0 Interrupt Status Register	0x0000_0000
TIMER0_CNT	TMR01_BA+0x0C	R/W	Timer0 Data Register	0x0000_0000
TIMER0_CAP	TMR01_BA+0x10	R	Timer0 Capture Data Register	0x0000_0000
TIMER0_EXTCTL	TMR01_BA+0x14	R/W	Timer0 External Control Register	0x0000_0000
TIMER0_EINTSTS	TMR01_BA+0x18	R/W	Timer0 External Interrupt Status Register	0x0000_0000
TIMER0_TRGCTL	TMR01_BA+0x1C	R/W	Timer0 Trigger Control Register	0x0000_0000
TIMER0_ALTCTL	TMR01_BA+0x20	R/W	Timer0 Alternative Control Register	0x0000_0000
TIMER0_PWMCTL	TMR01_BA+0x40	R/W	Timer0 PWM Control Register	0x0000_0000
TIMER0_PWMCLKSRC	TMR01_BA+0x44	R/W	Timer0 PWM Counter Clock Source Register	0x0000_0000
TIMER0_PWMCLKPSC	TMR01_BA+0x48	R/W	Timer0 PWM Counter Clock Pre-scale Register	0x0000_0000
TIMER0_PWMCNTCLR	TMR01_BA+0x4C	R/W	Timer0 PWM Clear Counter Register	0x0000_0000
TIMER0_PWMPERIOD	TMR01_BA+0x50	R/W	Timer0 PWM Period Register	0x0000_0000
TIMER0_PWMCMPDAT	TMR01_BA+0x54	R/W	Timer0 PWM Comparator Register	0x0000_0000
TIMER0_PWMDTCTL	TMR01_BA+0x58	R/W	Timer0 PWM Dead-Time Control Register	0x0000_0000
TIMER0_PWMCNT	TMR01_BA+0x5C	R	Timer0 PWM Counter Register	0x0000_0000
TIMER0_PWMMSKEN	TMR01_BA+0x60	R/W	Timer0 PWM Output Mask Enable Register	0x0000_0000
TIMER0_PWMMSK	TMR01_BA+0x64	R/W	Timer0 PWM Output Mask Data Control Register	0x0000_0000
TIMER0_PWMBNF	TMR01_BA+0x68	R/W	Timer0 PWM Brake Pin Noise Filter Register	0x0000_0000
TIMER0_PWMFAILBRK	TMR01_BA+0x6C	R/W	Timer0 PWM System Fail Brake Control Register	0x0000_0000
TIMER0_PWMBRKCTL	TMR01_BA+0x70	R/W	Timer0 PWM Brake Control Register	0x0000_0000
TIMER0_PWMPOLCTL	TMR01_BA+0x74	R/W	Timer0 PWM Pin Output Polar Control Register	0x0000_0000

TIMER0_PWMPOEN	TMR01_BA+0x78	R/W	Timer0 PWM Pin Output Enable Register	0x0000_0000
TIMER0_PWMSWBRK	TMR01_BA+0x7C	W	Timer0 PWM Software Trigger Brake Control Register	0x0000_0000
TIMER0_PWMINTEN0	TMR01_BA+0x80	R/W	Timer0 PWM Interrupt Enable Register 0	0x0000_0000
TIMER0_PWMINTEN1	TMR01_BA+0x84	R/W	Timer0 PWM Interrupt Enable Register 1	0x0000_0000
TIMER0_PWMINTSTS0	TMR01_BA+0x88	R/W	Timer0 PWM Interrupt Status Register 0	0x0000_0000
TIMER0_PWMINTSTS1	TMR01_BA+0x8C	R/W	Timer0 PWM Interrupt Status Register 1	0x0000_0000
TIMER0_PWMTRGCTL	TMR01_BA+0x90	R/W	Timer0 PWM Trigger Control Register	0x0000_0000
TIMER0_PWMSCTL	TMR01_BA+0x94	R/W	Timer0 PWM Synchronous Control Register	0x0000_0000
TIMER0_PWMSTRG	TMR01_BA+0x98	W	Timer0 PWM Synchronous Trigger Register	0x0000_0000
TIMER0_PWMSTATUS	TMR01_BA+0x9C	R/W	Timer0 PWM Status Register	0x0000_0000
TIMER0_PWMPBUF	TMR01_BA+0xA0	R	Timer0 PWM Period Buffer Register	0x0000_0000
TIMER0_PWMCMPBUF	TMR01_BA+0xA4	R	Timer0 PWM Comparator Buffer Register	0x0000_0000
TIMER1_CTL	TMR01_BA+0x100	R/W	Timer1 Control Register	0x0000_0005
TIMER1_CMP	TMR01_BA+0x104	R/W	Timer1 Comparator Register	0x0000_0000
TIMER1_INTSTS	TMR01_BA+0x108	R/W	Timer1 Interrupt Status Register	0x0000_0000
TIMER1_CNT	TMR01_BA+0x10C	R/W	Timer1 Data Register	0x0000_0000
TIMER1_CAP	TMR01_BA+0x110	R	Timer1 Capture Data Register	0x0000_0000
TIMER1_EXTCTL	TMR01_BA+0x114	R/W	Timer1 External Control Register	0x0000_0000
TIMER1_EINTSTS	TMR01_BA+0x118	R/W	Timer1 External Interrupt Status Register	0x0000_0000
TIMER1_TRGCTL	TMR01_BA+0x11C	R/W	Timer1 Trigger Control Register	0x0000_0000
TIMER1_ALTCTL	TMR01_BA+0x120	R/W	Timer1 Alternative Control Register	0x0000_0000
TIMER1_PWMCTL	TMR01_BA+0x140	R/W	Timer1 PWM Control Register	0x0000_0000
TIMER1_PWMCLKSRC	TMR01_BA+0x144	R/W	Timer1 PWM Counter Clock Source Register	0x0000_0000
TIMER1_PWMCLKPSC	TMR01_BA+0x148	R/W	Timer1 PWM Counter Clock Pre-scale Register	0x0000_0000
TIMER1_PWMCNTCLR	TMR01_BA+0x14C	R/W	Timer1 PWM Clear Counter Register	0x0000_0000
TIMER1_PWMPERIOD	TMR01_BA+0x150	R/W	Timer1 PWM Period Register	0x0000_0000
TIMER1_PWMCMPDAT	TMR01_BA+0x154	R/W	Timer1 PWM Comparator Register	0x0000_0000
TIMER1_PWMDTCTL	TMR01_BA+0x158	R/W	Timer1 PWM Dead-Time Control Register	0x0000_0000
TIMER1_PWMCNT	TMR01_BA+0x15C	R	Timer1 PWM Counter Register	0x0000_0000
TIMER1_PWMMSKEN	TMR01_BA+0x160	R/W	Timer1 PWM Output Mask Enable Register	0x0000_0000

TIMER1_PWMMSK	TMR01_BA+0x164	R/W	Timer1 PWM Output Mask Data Control Register	0x0000_0000
TIMER1_PWMBNF	TMR01_BA+0x168	R/W	Timer1 PWM Brake Pin Noise Filter Register	0x0000_0000
TIMER1_PWMFAILBRK	TMR01_BA+0x16C	R/W	Timer1 PWM System Fail Brake Control Register	0x0000_0000
TIMER1_PWMBRKCTL	TMR01_BA+0x170	R/W	Timer1 PWM Brake Control Register	0x0000_0000
TIMER1_PWMPOLCTL	TMR01_BA+0x174	R/W	Timer1 PWM Pin Output Polar Control Register	0x0000_0000
TIMER1_PWMPOEN	TMR01_BA+0x178	R/W	Timer1 PWM Pin Output Enable Register	0x0000_0000
TIMER1_PWMSWBRK	TMR01_BA+0x17C	W	Timer1 PWM Software Trigger Brake Control Register	0x0000_0000
TIMER1_PWMINTEN0	TMR01_BA+0x180	R/W	Timer1 PWM Interrupt Enable Register 0	0x0000_0000
TIMER1_PWMINTEN1	TMR01_BA+0x184	R/W	Timer1 PWM Interrupt Enable Register 1	0x0000_0000
TIMER1_PWMINTSTS0	TMR01_BA+0x188	R/W	Timer1 PWM Interrupt Status Register 0	0x0000_0000
TIMER1_PWMINTSTS1	TMR01_BA+0x18C	R/W	Timer1 PWM Interrupt Status Register 1	0x0000_0000
TIMER1_PWMTRGCTL	TMR01_BA+0x190	R/W	Timer1 PWM Trigger Control Register	0x0000_0000
TIMER1_PWMSCTL	TMR01_BA+0x194	R/W	Timer1 PWM Synchronous Control Register	0x0000_0000
TIMER1_PWMSTATUS	TMR01_BA+0x19C	R/W	Timer1 PWM Status Register	0x0000_0000
TIMER1_PWMPBUF	TMR01_BA+0x1A0	R	Timer1 PWM Period Buffer Register	0x0000_0000
TIMER1_PWMCMPBUF	TMR01_BA+0x1A4	R	Timer1 PWM Comparator Buffer Register	0x0000_0000
TIMER2_CTL	TMR23_BA+0x00	R/W	Timer2 Control Register	0x0000_0005
TIMER2_CMP	TMR23_BA+0x04	R/W	Timer2 Comparator Register	0x0000_0000
TIMER2_INTSTS	TMR23_BA+0x08	R/W	Timer2 Interrupt Status Register	0x0000_0000
TIMER2_CNT	TMR23_BA+0x0C	R/W	Timer2 Data Register	0x0000_0000
TIMER2_CAP	TMR23_BA+0x10	R	Timer2 Capture Data Register	0x0000_0000
TIMER2_EXTCTL	TMR23_BA+0x14	R/W	Timer2 External Control Register	0x0000_0000
TIMER2_EINTSTS	TMR23_BA+0x18	R/W	Timer2 External Interrupt Status Register	0x0000_0000
TIMER2_TRGCTL	TMR23_BA+0x1C	R/W	Timer2 Trigger Control Register	0x0000_0000
TIMER2_ALTCTL	TMR23_BA+0x20	R/W	Timer2 Alternative Control Register	0x0000_0000
TIMER2_PWMCTL	TMR23_BA+0x40	R/W	Timer2 PWM Control Register	0x0000_0000
TIMER2_PWMCLKSRC	TMR23_BA+0x44	R/W	Timer2 PWM Counter Clock Source Register	0x0000_0000
TIMER2_PWMCLKPSC	TMR23_BA+0x48	R/W	Timer2 PWM Counter Clock Pre-scale Register	0x0000_0000
TIMER2_PWMCNTCLR	TMR23_BA+0x4C	R/W	Timer2 PWM Clear Counter Register	0x0000_0000
TIMER2_PWMPERIOD	TMR23_BA+0x50	R/W	Timer2 PWM Period Register	0x0000_0000

TIMER2_PWMCMPDAT	TMR23_BA+0x54	R/W	Timer2 PWM Comparator Register	0x0000_0000
TIMER2_PWMDTCTL	TMR23_BA+0x58	R/W	Timer2 PWM Dead-Time Control Register	0x0000_0000
TIMER2_PWMCNT	TMR23_BA+0x5C	R	Timer2 PWM Counter Register	0x0000_0000
TIMER2_PWMMSKEN	TMR23_BA+0x60	R/W	Timer2 PWM Output Mask Enable Register	0x0000_0000
TIMER2_PWMMSK	TMR23_BA+0x64	R/W	Timer2 PWM Output Mask Data Control Register	0x0000_0000
TIMER2_PWMBNF	TMR23_BA+0x68	R/W	Timer2 PWM Brake Pin Noise Filter Register	0x0000_0000
TIMER2_PWMFAILBRK	TMR23_BA+0x6C	R/W	Timer2 PWM System Fail Brake Control Register	0x0000_0000
TIMER2_PWMBRKCTL	TMR23_BA+0x70	R/W	Timer2 PWM Brake Control Register	0x0000_0000
TIMER2_PWMPOLCTL	TMR23_BA+0x74	R/W	Timer2 PWM Pin Output Polar Control Register	0x0000_0000
TIMER2_PWMPOEN	TMR23_BA+0x78	R/W	Timer2 PWM Pin Output Enable Register	0x0000_0000
TIMER2_PWMSWBRK	TMR23_BA+0x7C	W	Timer2 PWM Software Trigger Brake Control Register	0x0000_0000
TIMER2_PWMINTEN0	TMR23_BA+0x80	R/W	Timer2 PWM Interrupt Enable Register 0	0x0000_0000
TIMER2_PWMINTEN1	TMR23_BA+0x84	R/W	Timer2 PWM Interrupt Enable Register 1	0x0000_0000
TIMER2_PWMINTSTS0	TMR23_BA+0x88	R/W	Timer2 PWM Interrupt Status Register 0	0x0000_0000
TIMER2_PWMINTSTS1	TMR23_BA+0x8C	R/W	Timer2 PWM Interrupt Status Register 1	0x0000_0000
TIMER2_PWMTRGCTL	TMR23_BA+0x90	R/W	Timer2 PWM Trigger Control Register	0x0000_0000
TIMER2_PWMSCTL	TMR23_BA+0x94	R/W	Timer2 PWM Synchronous Control Register	0x0000_0000
TIMER2_PWMSTRG	TMR23_BA+0x98	W	Timer2 PWM Synchronous Trigger Register	0x0000_0000
TIMER2_PWMSTATUS	TMR23_BA+0x9C	R/W	Timer2 PWM Status Register	0x0000_0000
TIMER2_PWMPBUF	TMR23_BA+0xA0	R	Timer2 PWM Period Buffer Register	0x0000_0000
TIMER2_PWMCMPBUF	TMR23_BA+0xA4	R	Timer2 PWM Comparator Buffer Register	0x0000_0000
TIMER3_CTL	TMR23_BA+0x100	R/W	Timer3 Control Register	0x0000_0005
TIMER3_CMP	TMR23_BA+0x104	R/W	Timer3 Comparator Register	0x0000_0000
TIMER3_INTSTS	TMR23_BA+0x108	R/W	Timer3 Interrupt Status Register	0x0000_0000
TIMER3_CNT	TMR23_BA+0x10C	R/W	Timer3 Data Register	0x0000_0000
TIMER3_CAP	TMR23_BA+0x110	R	Timer3 Capture Data Register	0x0000_0000
TIMER3_EXTCTL	TMR23_BA+0x114	R/W	Timer3 External Control Register	0x0000_0000
TIMER3_EINTSTS	TMR23_BA+0x118	R/W	Timer3 External Interrupt Status Register	0x0000_0000
TIMER3_TRGCTL	TMR23_BA+0x11C	R/W	Timer3 Trigger Control Register	0x0000_0000
TIMER3_ALTCTL	TMR23_BA+0x120	R/W	Timer3 Alternative Control Register	0x0000_0000

TIMER3_PWMCTL	TMR23_BA+0x140	R/W	Timer3 PWM Control Register	0x0000_0000
TIMER3_PWMCLKSRC	TMR23_BA+0x144	R/W	Timer3 PWM Counter Clock Source Register	0x0000_0000
TIMER3_PWMCLKPSC	TMR23_BA+0x148	R/W	Timer3 PWM Counter Clock Pre-scale Register	0x0000_0000
TIMER3_PWMCNTCLR	TMR23_BA+0x14C	R/W	Timer3 PWM Clear Counter Register	0x0000_0000
TIMER3_PWMPERIOD	TMR23_BA+0x150	R/W	Timer3 PWM Period Register	0x0000_0000
TIMER3_PWMCMPDAT	TMR23_BA+0x154	R/W	Timer3 PWM Comparator Register	0x0000_0000
TIMER3_PWMDTCTL	TMR23_BA+0x158	R/W	Timer3 PWM Dead-Time Control Register	0x0000_0000
TIMER3_PWMCNT	TMR23_BA+0x15C	R	Timer3 PWM Counter Register	0x0000_0000
TIMER3_PWMMSKEN	TMR23_BA+0x160	R/W	Timer3 PWM Output Mask Enable Register	0x0000_0000
TIMER3_PWMMSK	TMR23_BA+0x164	R/W	Timer3 PWM Output Mask Data Control Register	0x0000_0000
TIMER3_PWMBNF	TMR23_BA+0x168	R/W	Timer3 PWM Brake Pin Noise Filter Register	0x0000_0000
TIMER3_PWMFAILBRK	TMR23_BA+0x16C	R/W	Timer3 PWM System Fail Brake Control Register	0x0000_0000
TIMER3_PWMBRKCTL	TMR23_BA+0x170	R/W	Timer3 PWM Brake Control Register	0x0000_0000
TIMER3_PWMPOLCTL	TMR23_BA+0x174	R/W	Timer3 PWM Pin Output Polar Control Register	0x0000_0000
TIMER3_PWMPOEN	TMR23_BA+0x178	R/W	Timer3 PWM Pin Output Enable Register	0x0000_0000
TIMER3_PWMSWBRK	TMR23_BA+0x17C	W	Timer3 PWM Software Trigger Brake Control Register	0x0000_0000
TIMER3_PWMINTEN0	TMR23_BA+0x180	R/W	Timer3 PWM Interrupt Enable Register 0	0x0000_0000
TIMER3_PWMINTEN1	TMR23_BA+0x184	R/W	Timer3 PWM Interrupt Enable Register 1	0x0000_0000
TIMER3_PWMINTSTS0	TMR23_BA+0x188	R/W	Timer3 PWM Interrupt Status Register 0	0x0000_0000
TIMER3_PWMINTSTS1	TMR23_BA+0x18C	R/W	Timer3 PWM Interrupt Status Register 1	0x0000_0000
TIMER3_PWMTRGCTL	TMR23_BA+0x190	R/W	Timer3 PWM Trigger Control Register	0x0000_0000
TIMER3_PWMSCTL	TMR23_BA+0x194	R/W	Timer3 PWM Synchronous Control Register	0x0000_0000
TIMER3_PWMSTATUS	TMR23_BA+0x19C	R/W	Timer3 PWM Status Register	0x0000_0000
TIMER3_PWMPBUF	TMR23_BA+0x1A0	R	Timer3 PWM Period Buffer Register	0x0000_0000
TIMER3_PWMCMPBUF	TMR23_BA+0x1A4	R	Timer3 PWM Comparator Buffer Register	0x0000_0000
TIMER4_CTL	TMR45_BA+0x00	R/W	Timer4 Control Register	0x0000_0005
TIMER4_CMP	TMR45_BA+0x04	R/W	Timer4 Comparator Register	0x0000_0000
TIMER4_INTSTS	TMR45_BA+0x08	R/W	Timer4 Interrupt Status Register	0x0000_0000
TIMER4_CNT	TMR45_BA+0x0C	R/W	Timer4 Data Register	0x0000_0000
TIMER4_CAP	TMR45_BA+0x10	R	Timer4 Capture Data Register	0x0000_0000

TIMER4_EXTCTL	TMR45_BA+0x14	R/W	Timer4 External Control Register	0x0000_0000
TIMER4_EINTSTS	TMR45_BA+0x18	R/W	Timer4 External Interrupt Status Register	0x0000_0000
TIMER4_TRGCTL	TMR45_BA+0x1C	R/W	Timer4 Trigger Control Register	0x0000_0000
TIMER4_PWMCTL	TMR45_BA+0x40	R/W	Timer4 PWM Control Register	0x0000_0000
TIMER4_PWMCLKPSC	TMR45_BA+0x48	R/W	Timer4 PWM Counter Clock Pre-scale Register	0x0000_0000
TIMER4_PWMCNTCLR	TMR45_BA+0x4C	R/W	Timer4 PWM Clear Counter Register	0x0000_0000
TIMER4_PWMPERIOD	TMR45_BA+0x50	R/W	Timer4 PWM Period Register	0x0000_0000
TIMER4_PWMCMPDAT	TMR45_BA+0x54	R/W	Timer4 PWM Comparator Register	0x0000_0000
TIMER4_PWMCNT	TMR45_BA+0x5C	R	Timer4 PWM Counter Register	0x0000_0000
TIMER4_PWMPOLCTL	TMR45_BA+0x74	R/W	Timer4 PWM Pin Output Polar Control Register	0x0000_0000
TIMER4_PWMPOEN	TMR45_BA+0x78	R/W	Timer4 PWM Pin Output Enable Register	0x0000_0000
TIMER4_PWMINTEN0	TMR45_BA+0x80	R/W	Timer4 PWM Interrupt Enable Register 0	0x0000_0000
TIMER4_PWMINTSTS0	TMR45_BA+0x88	R/W	Timer4 PWM Interrupt Status Register 0	0x0000_0000
TIMER4_PWMTRGCTL	TMR45_BA+0x90	R/W	Timer4 PWM Trigger Control Register	0x0000_0000
TIMER4_PWMSTATUS	TMR45_BA+0x9C	R/W	Timer4 PWM Status Register	0x0000_0000
TIMER4_PWMPBUF	TMR45_BA+0xA0	R	Timer4 PWM Period Buffer Register	0x0000_0000
TIMER4_PWMCMPBUF	TMR45_BA+0xA4	R	Timer4 PWM Comparator Buffer Register	0x0000_0000
TIMER5_CTL	TMR45_BA+0x100	R/W	Timer5 Control Register	0x0000_0005
TIMER5_CMP	TMR45_BA+0x104	R/W	Timer5 Comparator Register	0x0000_0000
TIMER5_INTSTS	TMR45_BA+0x108	R/W	Timer5 Interrupt Status Register	0x0000_0000
TIMER5_CNT	TMR45_BA+0x10C	R/W	Timer5 Data Register	0x0000_0000
TIMER5_CAP	TMR45_BA+0x110	R	Timer5 Capture Data Register	0x0000_0000
TIMER5_EXTCTL	TMR45_BA+0x114	R/W	Timer5 External Control Register	0x0000_0000
TIMER5_EINTSTS	TMR45_BA+0x118	R/W	Timer5 External Interrupt Status Register	0x0000_0000
TIMER5_TRGCTL	TMR45_BA+0x11C	R/W	Timer5 Trigger Control Register	0x0000_0000
TIMER5_PWMCTL	TMR45_BA+0x140	R/W	Timer5 PWM Control Register	0x0000_0000
TIMER5_PWMCLKPSC	TMR45_BA+0x148	R/W	Timer5 PWM Counter Clock Pre-scale Register	0x0000_0000
TIMER5_PWMCNTCLR	TMR45_BA+0x14C	R/W	Timer5 PWM Clear Counter Register	0x0000_0000
TIMER5_PWMPERIOD	TMR45_BA+0x150	R/W	Timer5 PWM Period Register	0x0000_0000
TIMER5_PWMCMPDAT	TMR45_BA+0x154	R/W	Timer5 PWM Comparator Register	0x0000_0000

TIMER5_PWMCNT	TMR45_BA+0x15C	R	Timer5 PWM Counter Register	0x0000_0000
TIMER5_PWMPOLCTL	TMR45_BA+0x174	R/W	Timer5 PWM Pin Output Polar Control Register	0x0000_0000
TIMER5_PWMPOEN	TMR45_BA+0x178	R/W	Timer5 PWM Pin Output Enable Register	0x0000_0000
TIMER5_PWMINTEN0	TMR45_BA+0x180	R/W	Timer5 PWM Interrupt Enable Register 0	0x0000_0000
TIMER5_PWMINTSTS0	TMR45_BA+0x188	R/W	Timer5 PWM Interrupt Status Register 0	0x0000_0000
TIMER5_PWMTRGCTL	TMR45_BA+0x190	R/W	Timer5 PWM Trigger Control Register	0x0000_0000
TIMER5_PWMSTATUS	TMR45_BA+0x19C	R/W	Timer5 PWM Status Register	0x0000_0000
TIMER5_PWMPBUF	TMR45_BA+0x1A0	R	Timer5 PWM Period Buffer Register	0x0000_0000
TIMER5_PWMCMPBUF	TMR45_BA+0x1A4	R	Timer5 PWM Comparator Buffer Register	0x0000_0000

6.9.8 Register Description

Timer Control Register (TIMERx_CTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_CTL	TMR01_BA+0x00	R/W	Timer0 Control Register	0x0000_0005
TIMER1_CTL	TMR01_BA+0x100	R/W	Timer1 Control Register	0x0000_0005
TIMER2_CTL	TMR23_BA+0x00	R/W	Timer2 Control Register	0x0000_0005
TIMER3_CTL	TMR23_BA+0x100	R/W	Timer3 Control Register	0x0000_0005
TIMER4_CTL	TMR45_BA+0x00	R/W	Timer4 Control Register	0x0000_0005
TIMER5_CTL	TMR45_BA+0x100	R/W	Timer5 Control Register	0x0000_0005

31	30	29	28	27	26	25	24
ICEDEBUG	**CNTEN**	**INTEN**	**OPMODE**		**Reserved**	**ACTSTS**	**EXTCNTEN**
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
WKEN	**CAPSRC**	**TGLPINSEL**	**PERIOSEL**	**INTRGEN**	**Reserved**		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FUNCSEL	**Reserved**						
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PSC							

Bits	Description	
[31]	**ICEDEBUG**	**ICE Debug Mode Acknowledge Disable Bit (Write Protect)** 0 = ICE debug mode acknowledgement effects TIMER counting. TIMER counter will be held while CPU is held by ICE. 1 = ICE debug mode acknowledgement Disabled. TIMER counter will keep going no matter CPU is held by ICE or not. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[30]	**CNTEN**	**Timer Counting Enable Bit** 0 = Stops/Suspends counting. 1 = Starts counting. **Note 1:** In stop status, and then set CNTEN to 1 will enable the 24-bit up counter to keep counting from the last stop counting value. **Note 2:** This bit is auto-cleared by hardware in one-shot mode (TIMER_CTL[28:27] = 00) when the timer time-out interrupt flag TIF (TIMERx_INTSTS[0]) is generated. **Note 3:** Set enable/disable this bit needs 2 * TMR_CLK period to become active, user can read ACTSTS (TIMERx_CTL[25]) to check enable/disable command is completed or not.
[29]	**INTEN**	**Timer Interrupt Enable Bit** 0 = Timer time-out interrupt Disabled.

		1 = Timer time-out interrupt Enabled. **Note:** If this bit is enabled, when the timer time-out interrupt flag TIF is set to 1, the timer interrupt signal is generated and inform to CPU.
[28:27]	**OPMODE**	**Timer Counting Mode Select** 00 = The timer controller is operated in One-shot mode. 01 = The timer controller is operated in Periodic mode. 10 = The timer controller is operated in Toggle-output mode. 11 = The timer controller is operated in Continuous Counting mode.
[26]	**Reserved**	Reserved.
[25]	**ACTSTS**	**Timer Active Status Bit (Read Only)** This bit indicates the 24-bit up counter status. 0 = 24-bit up counter is not active. 1 = 24-bit up counter is active. **Note:** This bit may active when CNT 0 transition to CNT 1.
[24]	**EXTCNTEN**	**Event Counter Mode Enable Bit** This bit is for external counting pin function enabled. 0 = Event counter mode Disabled. 1 = Event counter mode Enabled. **Note:** When timer is used as an event counter, this bit should be set to 1 and select PCLK as timer clock source.
[23]	**WKEN**	**Wake-up Function Enable Bit** If this bit is set to 1, while timer interrupt flag TIF (TIMERx_INTSTS[0]) is 1 and INTEN (TIMERx_CTL[29]) is enabled, the timer interrupt signal will generate a wake-up trigger event to CPU. 0 = Wake-up function Disabled if timer interrupt signal generated. 1 = Wake-up function Enabled if timer interrupt signal generated.
[22]	**CAPSRC**	**Capture Pin Source Selection** 0 = Capture Function source is from TMx_EXT (x= 0~5) pin. 1 = Capture Function source is from internal ACMP output signal, internal clock source (HIRC, LIRC, MIRC) or external clock (HXT, LXT). **Note 1:** When CAPSRC = 1, user can set INTERCAPSEL (TIMERx_EXTCTL[10:8]) to decide which internal ACMP output signal or which clock is as timer capture source. **Note 2:** MIRC clock source is only available in Timer4 ~ Timer5.
[21]	**TGLPINSEL**	**Toggle-output Pin Select** 0 = Toggle mode output to TMx (Timer Event Counter Pin). 1 = Toggle mode output to TMx_EXT (Timer External Capture Pin).
[20]	**PERIOSEL**	**Periodic Mode Behavior Selection Enable Bit** 0 = The behavior selection in periodic mode is Disabled. When user updates CMPDAT while timer is running in periodic mode, CNT will be reset to default value. 1 = The behavior selection in periodic mode is Enabled. When user update CMPDAT while timer is running in periodic mode, the limitations as bellows list, If updated CMPDAT value > CNT, CMPDAT will be updated and CNT keep running continually. If updated CMPDAT value = CNT, timer time-out interrupt will be asserted immediately. If updated CMPDAT value < CNT, CNT will be reset to default value.
[19]	**INTRGEN**	**Inter-timer Trigger Mode Enable Bit** Setting this bit will enable the inter-timer trigger capture function.

		The Timer0/2/4 will be in event counter mode and counting with external clock source or event. Also, Timer1/3/5 will be in trigger-counting mode of capture function. 0 = Inter-Timer Trigger Capture mode Disabled. 1 = Inter-Timer Trigger Capture mode Enabled. **Note:** For Timer1/3/5, this bit is ignored and the read back value is always 0.
[18:16]	**Reserved**	Reserved.
[15]	**FUNCSEL**	**Function Selection** This bit sets the operation mode of Timer4 and Timer5 to PWM function. 0 = Timer controller is used as timer function. 1 = Timer controller is used as PWM function. **Note:** The Timer0 ~ Timer3 function selection is controlled in TIMERx_ALTCTL[0], x = 0 ~ 3.
[14:8]	**Reserved**	Reserved.
[7:0]	**PSC**	**Prescale Counter** Timer input clock or event source is divided by (PSC+1) before it is fed to the timer up counter. If this field is 0 (PSC = 0), then there is no scaling. **Note:** Update prescale counter value will reset internal 8-bit prescale counter and 24-bit up counter value.

Timer Comparator Register (TIMERx_CMP)

Register	Offset	R/W	Description	Reset Value
TIMER0_CMP	TMR01_BA+0x04	R/W	Timer0 Comparator Register	0x0000_0000
TIMER1_CMP	TMR01_BA+0x104	R/W	Timer1 Comparator Register	0x0000_0000
TIMER2_CMP	TMR23_BA+0x04	R/W	Timer2 Comparator Register	0x0000_0000
TIMER3_CMP	TMR23_BA+0x104	R/W	Timer3 Comparator Register	0x0000_0000
TIMER4_CMP	TMR45_BA+0x04	R/W	Timer4 Comparator Register	0x0000_0000
TIMER5_CMP	TMR45_BA+0x104	R/W	Timer5 Comparator Register	0x0000_0000

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

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**CMPDAT**	**Timer Comparator Value** CMPDAT is a 24-bit compared value register. When the internal 24-bit up counter value is equal to CMPDAT value, the TIF (TIMERx_INTSTS[0] Timer Interrupt Flag) will be set to 1. Time-out period = (Period of timer clock input) * (8-bit PSC + 1) * (24-bit CMPDAT). **Note 1:** Never write 0x0 or 0x1 in CMPDAT field, or the core will run into unknown state. **Note 2:** When timer is operating in continuous counting mode, the 24-bit up counter will keep counting continuously even if user writes a new value into CMPDAT field. But if timer is operating at other modes, the 24-bit up counter will restart counting from 0 and use the newest CMPDAT value to be the timer compared value while user writes a new value into the CMPDAT field.

Timer Interrupt Status Register (TIMERx_INTSTS)

Register	Offset	R/W	Description	Reset Value
TIMER0_INTSTS	TMR01_BA+0x08	R/W	Timer0 Interrupt Status Register	0x0000_0000
TIMER1_INTSTS	TMR01_BA+0x108	R/W	Timer1 Interrupt Status Register	0x0000_0000
TIMER2_INTSTS	TMR23_BA+0x08	R/W	Timer2 Interrupt Status Register	0x0000_0000
TIMER3_INTSTS	TMR23_BA+0x108	R/W	Timer3 Interrupt Status Register	0x0000_0000
TIMER4_INTSTS	TMR45_BA+0x08	R/W	Timer4 Interrupt Status Register	0x0000_0000
TIMER5_INTSTS	TMR45_BA+0x108	R/W	Timer5 Interrupt Status Register	0x0000_0000

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

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**TWKF**	**Timer Wake-up Flag** This bit indicates the interrupt wake-up flag status of timer. 0 = Timer does not cause CPU wake-up. 1 = CPU wake-up from Idle or Power-down mode if timer time-out interrupt signal generated. **Note:** This bit is cleared by writing 1 to it.
[0]	**TIF**	**Timer Interrupt Flag** This bit indicates the interrupt flag status of Timer while 24-bit timer up counter CNT (TIMERx_CNT[23:0]) value reaches to CMPDAT (TIMERx_CMP[23:0]) value. 0 = No effect. 1 = CNT value matches the CMPDAT value. **Note:** This bit is cleared by writing 1 to it.

Timer Data Register (TIMERx_CNT)

Register	Offset	R/W	Description	Reset Value
TIMER0_CNT	TMR01_BA+0x0C	R/W	Timer0 Data Register	0x0000_0000
TIMER1_CNT	TMR01_BA+0x10C	R/W	Timer1 Data Register	0x0000_0000
TIMER2_CNT	TMR23_BA+0x0C	R/W	Timer2 Data Register	0x0000_0000
TIMER3_CNT	TMR23_BA+0x10C	R/W	Timer3 Data Register	0x0000_0000
TIMER4_CNT	TMR45_BA+0x0C	R/W	Timer4 Data Register	0x0000_0000
TIMER5_CNT	TMR45_BA+0x10C	R/W	Timer5 Data Register	0x0000_0000

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

Bits	Description	
[31]	**RSTACT**	**Timer Data Register Reset Active (Read Only)** This bit indicates if the counter reset operation active. When user writes this CNT register, timer starts to reset its internal 24-bit timer up-counter to 0 and reload 8-bit pre-scale counter. At the same time, timer set this flag to 1 to indicate the counter reset operation is in progress. Once the counter reset operation done, timer clear this bit to 0 automatically. 0 = Reset operation is done. 1 = Reset operation triggered by writing TIMERx_CNT is in progress.
[30:24]	**Reserved**	Reserved.
[23:0]	**CNT**	**Timer Data Register** Read operation. Read this register to get CNT value. For example: If EXTCNTEN (TIMERx_CTL[24]) is 0, user can read CNT value for getting current 24-bit counter value. If EXTCNTEN (TIMERx_CTL[24]) is 1, user can read CNT value for getting current 24-bit event input counter value. Write operation. Writing any value to this register will reset current CNT value to 0 and reload internal 8-bit prescale counter.

Timer Capture Data Register (TIMERx_CAP)

Register	Offset	R/W	Description	Reset Value
TIMER0_CAP	TMR01_BA+0x10	R	Timer0 Capture Data Register	0x0000_0000
TIMER1_CAP	TMR01_BA+0x110	R	Timer1 Capture Data Register	0x0000_0000
TIMER2_CAP	TMR23_BA+0x10	R	Timer2 Capture Data Register	0x0000_0000
TIMER3_CAP	TMR23_BA+0x110	R	Timer3 Capture Data Register	0x0000_0000
TIMER4_CAP	TMR45_BA+0x10	R	Timer4 Capture Data Register	0x0000_0000
TIMER5_CAP	TMR45_BA+0x110	R	Timer5 Capture Data Register	0x0000_0000

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

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**CAPDAT**	**Timer Capture Data Register** When CAPEN (TIMERx_EXTCTL[3]) bit is set, the transition on the capture source matches the CAPEDGE (TIMERx_EXTCTL[14:12]) setting, CAPIF (TIMERx_EINTSTS[0]) will be set to 1 and the current timer counter value CNT (TIMERx_CNT[23:0]) will be auto-loaded into this CAPDAT field.

Timer External Control Register (TIMERx_EXTCTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_EXTCTL	TMR01_BA+0x14	R/W	Timer0 External Control Register	0x0000_0000
TIMER1_EXTCTL	TMR01_BA+0x114	R/W	Timer1 External Control Register	0x0000_0000
TIMER2_EXTCTL	TMR23_BA+0x14	R/W	Timer2 External Control Register	0x0000_0000
TIMER3_EXTCTL	TMR23_BA+0x114	R/W	Timer3 External Control Register	0x0000_0000
TIMER4_EXTCTL	TMR45_BA+0x14	R/W	Timer4 External Control Register	0x0000_0000
TIMER5_EXTCTL	TMR45_BA+0x114	R/W	Timer5 External Control Register	0x0000_0000

31	30	29	28	27	26	25	24
CAPDIVSCL				Reserved			
23	22	21	20	19	18	17	16
Reserved							ECNTSSEL
15	14	13	12	11	10	9	8
Reserved	CAPEDGE			Reserved	INTERCAPSEL		
7	6	5	4	3	2	1	0
CNTDBEN	CAPDBEN	CAPIEN	CAPFUNCS	CAPEN	Reserved		CNTPHASE

Bits	Description	
[31:28]	**CAPDIVSCL**	**Timer Capture Source Divider Scale** These bits indicate the divide scale for capture source divider. 0000 = Capture source/1. 0001 = Capture source/2. 0010 = Capture source/4. 0011 = Capture source/8. 0100 = Capture source/16. 0101 = Capture source/32. 0110 = Capture source/64. 0111 = Capture source/128. 1000 = Capture source/256. 1001~1111 = Reserved. **Note:** Sets INTERCAPSEL (TIMERx_EXTCTL[10:8]) and CAPSRC (TIMERx_CTL[22]) to select capture source.
[27:17]	**Reserved**	Reserved.
[16]	**ECNTSSEL**	**Event Counter Source Selection to Trigger Event Counter Function** 0 = Event Counter input source is from TMx (x= 0~5) pin. 1 = Event Counter input source is from USB internal SOF output signal.

[15]	**Reserved**	Reserved.
[14:12]	**CAPEDGE**	**Timer Capture Edge Detect** When the first capture event is generated, the CNT (TIMERx_CNT[23:0]) will be reset to 0 and first CAPDAT (TIMERx_CAP[23:0]) should be to 0. 000 = Capture event occurred when detect falling edge transfer on capture source. 001 = Capture event occurred when detect rising edge transfer on capture source. 010 = Capture event occurred when detect both falling and rising edge transfer on capture source, and the first capture event occurred at falling edge transfer. 011 = Capture event occurred when detect both rising and falling edge transfer on capture source, and the first capture event occurred at rising edge transfer. 110 = First capture event occurred at falling edge, follows capture events are at rising edge transfer on capture source. 111 = First capture event occurred at rising edge, follows capture events are at falling edge transfer on capture source. 100, 101 = Reserved. **Note:** Set CAPSRC (TIMERx_CTL[22]) and INTERCAPSEL (TIMERx_EXTCTL[10:8]) to select capture source.
[11]	**Reserved**	Reserved.
[10:8]	**INTERCAPSEL**	**Internal Capture Source Select** 000 = Capture Function source is from internal ACMP0 output signal. 001 = Capture Function source is from internal ACMP1 output signal. 010 = Capture Function source is from HXT. 011 = Capture Function source is from LXT. 100 = Capture Function source is from HIRC. 101 = Capture Function source is from LIRC. 110 = Capture Function source is from MIRC, only available in Timer4 and Timer5. 111 = Reserved. **Note:** These bits only available when CAPSRC (TIMERx_CTL[22]) is 1.
[7]	**CNTDBEN**	**Timer External Counter Pin De-bounce Enable Bit** 0 = TMx (x= 0~5) pin de-bounce Disabled. 1 = TMx (x= 0~5) pin de-bounce Enabled. **Note:** If this bit is enabled, the edge detection of TMx pin is detected with de-bounce circuit.
[6]	**CAPDBEN**	**Timer Capture De-bounce Enable Bit** 0 = TMx_EXT (x= 0~5) pin de-bounce or ACMP output de-bounce Disabled. 1 = TMx_EXT (x= 0~5) pin de-bounce or ACMP output de-bounce Enabled. **Note:** If this bit is enabled, the edge detection of TMx_EXT pin or ACMP output is detected with de-bounce circuit.
[5]	**CAPIEN**	**Timer Capture Interrupt Enable Bit** 0 = TMx_EXT (x= 0~5) pin, ACMP, internal clock, or external clock detection Interrupt Disabled. 1 = TMx_EXT (x= 0~5) pin, ACMP, internal clock, or external clock detection Interrupt Enabled. **Note:** CAPIEN is used to enable timer capture interrupt. If CAPIEN enabled, timer will rise an interrupt when CAPIF (TIMERx_EINTSTS[0]) is 1. For example, while CAPIEN = 1, CAPEN = 1, and CAPEDGE = 00, a 1 to 0 transition on the TMx_EXT pin, ACMP, internal clock, or external clock will cause the CAPIF to be set then the interrupt signal is generated and sent to NVIC to inform CPU.
[4]	**CAPFUNCS**	**Capture Function Selection** 0 = Capture Mode Enabled. 1 = Reset Mode Enabled.

		Note 1: When CAPFUNCS is 0 and CAPIF becomes 1, the current 24-bit timer counter value (CNT value) will be saved to CAPDAT field. **Note 2:** When CAPFUNCS is 1 and CAPIF becomes 1, the current 24-bit timer counter value (CNT value) will be saved to CAPDAT field then CNT value will be reset immediately.
[3]	**CAPEN**	**Timer Capture Function Enable Bit** This bit enables the capture input function. 0 = Timer capture function Disabled. 1 = Timer capture function Enabled.
[2:1]	**Reserved**	Reserved.
[0]	**CNTPHASE**	**Timer External Count Phase** This bit indicates the detection phase of external counting pin TMx (x= 0~5). 0 = A falling edge of external counting pin will be counted. 1 = A rising edge of external counting pin will be counted.

Timer External Interrupt Status Register (TIMERx_EINTSTS)

Register	Offset	R/W	Description	Reset Value
TIMER0_EINTSTS	TMR01_BA+0x18	R/W	Timer0 External Interrupt Status Register	0x0000_0000
TIMER1_EINTSTS	TMR01_BA+0x118	R/W	Timer1 External Interrupt Status Register	0x0000_0000
TIMER2_EINTSTS	TMR23_BA+0x18	R/W	Timer2 External Interrupt Status Register	0x0000_0000
TIMER3_EINTSTS	TMR23_BA+0x118	R/W	Timer3 External Interrupt Status Register	0x0000_0000
TIMER4_EINTSTS	TMR45_BA+0x18	R/W	Timer4 External Interrupt Status Register	0x0000_0000
TIMER5_EINTSTS	TMR45_BA+0x118	R/W	Timer5 External Interrupt Status Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**CAPIF**	**Timer Capture Interrupt Flag** This bit indicates the timer capture interrupt flag status. 0 = TMx_EXT (x= 0~5) pin, ACMP, internal clock, or external clock capture interrupt did not occur. 1 = TMx_EXT (x= 0~5) pin, ACMP, internal clock, or external clock capture interrupt occurred. **Note 1:** This bit is cleared by writing 1 to it. **Note 2:** When the CAPEN (TIMERx_EXTCTL[3]) bit is set, the transition on the capture source matches the CAPEDGE (TIMERx_EXTCTL[14:12]) setting, this bit will be set to 1 by hardware. **Note 3:** There is a new incoming capture event detected before CPU clearing the CAPIF status. If the above condition occurred, the timer will keep register TIMERx_CAP unchanged and drop the new capture value.

Timer Trigger Control Register (TIMERx_TRGCTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_TRGCTL	TMR01_BA+0x1C	R/W	Timer0 Trigger Control Register	0x0000_0000
TIMER1_TRGCTL	TMR01_BA+0x11C	R/W	Timer1 Trigger Control Register	0x0000_0000
TIMER2_TRGCTL	TMR23_BA+0x1C	R/W	Timer2 Trigger Control Register	0x0000_0000
TIMER3_TRGCTL	TMR23_BA+0x11C	R/W	Timer3 Trigger Control Register	0x0000_0000
TIMER4_TRGCTL	TMR45_BA+0x1C	R/W	Timer4 Trigger Control Register	0x0000_0000
TIMER5_TRGCTL	TMR45_BA+0x11C	R/W	Timer5 Trigger Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**3**	**4**	**3**	**2**	**1**	**0**
Reserved			**TRGPDMA**	**TRGDAC**	**TRGEADC**	**TRGPWM**	**TRGSSEL**

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**TRGPDMA**	**Trigger PDMA Enable Bit** If this bit is set to 1, each timer time-out event or capture event can be triggered PDMA transfer. 0 = Timer interrupt trigger PDMA Disabled. 1 = Timer interrupt trigger PDMA Enabled. **Note:** If TRGSSEL (TIMERx_TRGCTL[0]) = 0, time-out interrupt signal will trigger PDMA transfer. If TRGSSEL (TIMERx_TRGCTL[0]) = 1, capture interrupt signal will trigger PDMA transfer.
[3]	**TRGDAC**	**Trigger DAC Enable Bit** If this bit is set to 1, each timer time-out event or capture event can be triggered DAC. 0 = Timer interrupt trigger DAC Disabled. 1 = Timer interrupt trigger DAC Enabled. **Note 1:** This bit is not available in Timer4 and Timer5. **Note 2:** If TRGSSEL (TIMERx_TRGCTL[0]) = 0, time-out interrupt signal will trigger DAC. If TRGSSEL (TIMERx_TRGCTL[0]) = 1, capture interrupt signal will trigger DAC.
[2]	**TRGEADC**	**Trigger EADC Enable Bit** If this bit is set to 1, each timer time-out event or capture event can be triggered EADC conversion. 0 = Timer interrupt trigger EADC Disabled.

		1 = Timer interrupt trigger EADC Enabled. **Note:** If TRGSSEL (TIMERx_TRGCTL[0]) = 0, time-out interrupt signal will trigger EADC conversion. If TRGSSEL (TIMERx_TRGCTL[0]) = 1, capture interrupt signal will trigger EADC conversion.
[1]	**TRGPWM**	**Trigger EPWM and BPWM Enable Bit** If this bit is set to 1, each timer time-out event or capture event can be as EPWM and BPWM counter clock source. 0 = Timer interrupt trigger EPWM and BPWM Disabled. 1 = Timer interrupt trigger EPWM and BPWM Enabled. **Note 1:** This bit is not available in Timer4 and Timer5. **Note 2:** If TRGSSEL (TIMERx_TRGCTL[0]) = 0, time-out interrupt signal as EPWM and BPWM counter clock source. If TRGSSEL (TIMERx_TRGCTL[0]) = 1, capture interrupt signal as EPWM and BPWM counter clock source.
[0]	**TRGSSEL**	**Trigger Source Select Bit** This bit is used to select internal trigger source is form timer time-out interrupt signal or capture interrupt signal. 0 = Time-out interrupt signal is used to internal trigger EPWM/BPWM, PDMA, DAC, and EADC. 1 = Capture interrupt signal is used to internal trigger EPWM/BPWM, PDMA, DAC, and EADC.

Timer Alternative Control Register (TIMERx_ALTCTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_ALTCTL	TMR01_BA+0x20	R/W	Timer0 Alternative Control Register	0x0000_0000
TIMER1_ALTCTL	TMR01_BA+0x120	R/W	Timer1 Alternative Control Register	0x0000_0000
TIMER2_ALTCTL	TMR23_BA+0x20	R/W	Timer2 Alternative Control Register	0x0000_0000
TIMER3_ALTCTL	TMR23_BA+0x120	R/W	Timer3 Alternative Control Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**FUNCSEL**	**Function Selection** This bit sets the operation mode of Timer0 ~ Timer3 to PWM function. 0 = Timer controller is used as timer function. 1 = Timer controller is used as PWM function. **Note 1:** The Timer4 and Timer5 function selection is controlled in TIMERx_CTL[15], x = 4 ~ 5. **Note 2:** When timer is used as PWM, the clock source of time controller will be forced to PCLKx automatically.

Timer PWM Control Register (TIMERx_PWMCTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMCTL	TMR01_BA+0x40	R/W	Timer0 PWM Control Register	0x0000_0000
TIMER1_PWMCTL	TMR01_BA+0x140	R/W	Timer1 PWM Control Register	0x0000_0000
TIMER2_PWMCTL	TMR23_BA+0x40	R/W	Timer2 PWM Control Register	0x0000_0000
TIMER3_PWMCTL	TMR23_BA+0x140	R/W	Timer3 PWM Control Register	0x0000_0000
TIMER4_PWMCTL	TMR45_BA+0x40	R/W	Timer4 PWM Control Register	0x0000_0000
TIMER5_PWMCTL	TMR45_BA+0x140	R/W	Timer5 PWM Control Register	0x0000_0000

31	30	29	28	27	26	25	24
DBGTRIOFF	DBGHALT	Reserved					
23	22	21	20	19	18	17	16
Reserved							OUTMODE
15	14	13	12	11	10	9	8
Reserved			WKEN	Reserved		IMMLDEN	CTRLD
7	6	5	4	3	2	1	0
Reserved				CNTMODE	CNTTYPE		CNTEN

Bits	Description	
[31]	DBGTRIOFF	**ICE Debug Mode Acknowledge Disable Bit (Write Protect)** 0 = ICE debug mode acknowledgement effects PWM output. PWM output pin will be forced as tri-state while ICE debug mode acknowledged. 1 = ICE debug mode acknowledgement disabled. PWM output pin will keep output no matter ICE debug mode acknowledged or not. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[30]	DBGHALT	**ICE Debug Mode Counter Halt (Write Protect)** If debug mode counter halt is enabled, PWM counter will keep current value until exit ICE debug mode. 0 = ICE debug mode counter halt disable. 1 = ICE debug mode counter halt enable. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[29:17]	Reserved	Reserved.
[16]	OUTMODE	**PWM Output Mode** This bit controls the output mode of corresponding PWM channel. 0 = PWM independent mode. 1 = PWM complementary mode. **Note:** This bit is not available in Timer4 and Timer5.
[15:13]	Reserved	Reserved.

[12]	**WKEN**	**PWM Wake-up Enable Bit** If this bit is set to 1, the Timer4 and Timer5 PWM interrupt event will generate a wake-up trigger event to CPU. 0 = PWM interrupt wake-up Disabled. 1 = PWM interrupt wake-up Enabled. **Note:** This bit is only available in Timer4 and Timer5.
[11:10]	**Reserved**	Reserved.
[9]	**IMMLDEN**	**Immediately Load Enable Bit** 0 = PERIOD will load to PBUF when current PWM period is completed no matter CTRLD is enabled/disabled. If CTRLD is disabled, CMP will load to CMPBUF when current PWM period is completed; if CTRLD is enabled in up-down count type, CMP will load to CMPBUF at the center point of current period. 1 = PERIOD/CMP will load to PBUF/CMPBUF immediately when user update PERIOD/CMP. **Note 1:** This bit is not available in Timer4 and Timer5. **Note 2:** If IMMLDEN is enabled, CTRLD will be invalid.
[8]	**CTRLD**	**Center Re-load** In up-down count type, PERIOD will load to PBUF when current PWM period is completed always and CMP will load to CMPBUF at the center point of current period. **Note:** This bit is not available in Timer4 and Timer5.
[7:4]	**Reserved**	Reserved.
[3]	**CNTMODE**	**PWM Counter Mode** 0 = Auto-reload mode. 1 = One-shot mode.
[2:1]	**CNTTYPE**	**PWM Counter Behavior Type** These bits are used to set the count type of Timer0 ~ Timer3. The count type of Timer4 and Timer5 is fixed as the up count type. 00 = Up count type. 01 = Down count type. 10 = Up-down count type. 11 = Reserved. **Note:** These bits are not available in Timer4 and Timer5.
[0]	**CNTEN**	**PWM Counter Enable Bit** 0 = PWM counter and clock prescale Stop Running. 1 = PWM counter and clock prescale Start Running.

Timer PWM Counter Clock Source Register (TIMERx PWMCLKSRC)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMCLKSRC	TMR01_BA+0x44	R/W	Timer0 PWM Counter Clock Source Register	0x0000_0000
TIMER1_PWMCLKSRC	TMR01_BA+0x144	R/W	Timer1 PWM Counter Clock Source Register	0x0000_0000
TIMER2_PWMCLKSRC	TMR23_BA+0x44	R/W	Timer2 PWM Counter Clock Source Register	0x0000_0000
TIMER3_PWMCLKSRC	TMR23_BA+0x144	R/W	Timer3 PWM Counter Clock Source Register	0x0000_0000

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

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2:0]	**CLKSRC**	**PWM Counter Clock Source Select** The PWM counter clock source can be selected from TMRx_CLK or internal timer time-out or capture event in Timer0 ~ Timer3. 000 = TMRx_CLK. 001 = Internal TIMER0 time-out or capture event. 010 = Internal TIMER1 time-out or capture event. 011 = Internal TIMER2 time-out or capture event. 100 = Internal TIMER3 time-out or capture event. Others = Reserved. **Note 1:** These bits are not available in Timer4 and Timer5 **Note 2:** If TIMER0 PWM function is enabled, the PWM counter clock source can be selected from TMR0_CLK, TIMER1 interrupt events, TIMER2 interrupt events, or TIMER3 interrupt events.

Timer PWM Counter Clock Pre-scale Register (TIMERx PWMCLKPSC)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMCLKPSC	TMR01_BA+0x48	R/W	Timer0 PWM Counter Clock Pre-scale Register	0x0000_0000
TIMER1_PWMCLKPSC	TMR01_BA+0x148	R/W	Timer1 PWM Counter Clock Pre-scale Register	0x0000_0000
TIMER2_PWMCLKPSC	TMR23_BA+0x48	R/W	Timer2 PWM Counter Clock Pre-scale Register	0x0000_0000
TIMER3_PWMCLKPSC	TMR23_BA+0x148	R/W	Timer3 PWM Counter Clock Pre-scale Register	0x0000_0000
TIMER4_PWMCLKPSC	TMR45_BA+0x48	R/W	Timer4 PWM Counter Clock Pre-scale Register	0x0000_0000
TIMER5_PWMCLKPSC	TMR45_BA+0x148	R/W	Timer5 PWM Counter Clock Pre-scale Register	0x0000_0000

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

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**CLKPSC**	**PWM Counter Clock Pre-scale** The active clock of PWM counter is decided by counter clock prescale and divided by (CLKPSC + 1). If CLKPSC is 0, then there is no scaling in PWM counter clock source. **Note:** The valid value is 12-bit TIMERx_PWMCLKPSC[11:0] in Timer0 ~ Timer3, and 8-bit TIMERx_PWMCLKPSC[7:0] in Timer4 and Timer5.

Timer PWM Clear Counter Register (TIMERx PWMCNTCLR)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMCNTCLR	TMR01_BA+0x4C	R/W	Timer0 PWM Clear Counter Register	0x0000_0000
TIMER1_PWMCNTCLR	TMR01_BA+0x14C	R/W	Timer1 PWM Clear Counter Register	0x0000_0000
TIMER2_PWMCNTCLR	TMR23_BA+0x4C	R/W	Timer2 PWM Clear Counter Register	0x0000_0000
TIMER3_PWMCNTCLR	TMR23_BA+0x14C	R/W	Timer3 PWM Clear Counter Register	0x0000_0000
TIMER4_PWMCNTCLR	TMR45_BA+0x4C	R/W	Timer4 PWM Clear Counter Register	0x0000_0000
TIMER5_PWMCNTCLR	TMR45_BA+0x14C	R/W	Timer5 PWM Clear Counter Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**CNTCLR**	**Clear PWM Counter Control Bit** It is automatically cleared by hardware. 0 = No effect. 1 = In Timer0 ~ Timer3, clears 16-bit PWM counter to 0x10000 in up and up-down count type and reset counter value to PERIOD in down count type. In Timer4 and Timer5, clears 16-bit PWM counter to 0x0 in up count type.

Timer PWM Period Register (TIMERx PWMPERIOD)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMPERIOD	TMR01_BA+0x50	R/W	Timer0 PWM Period Register	0x0000_0000
TIMER1_PWMPERIOD	TMR01_BA+0x150	R/W	Timer1 PWM Period Register	0x0000_0000
TIMER2_PWMPERIOD	TMR23_BA+0x50	R/W	Timer2 PWM Period Register	0x0000_0000
TIMER3_PWMPERIOD	TMR23_BA+0x150	R/W	Timer3 PWM Period Register	0x0000_0000
TIMER4_PWMPERIOD	TMR45_BA+0x50	R/W	Timer4 PWM Period Register	0x0000_0000
TIMER5_PWMPERIOD	TMR45_BA+0x150	R/W	Timer5 PWM Period Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PERIOD**	**PWM Period Register** In up count type: PWM counter counts from 0 to PERIOD, and restarts from 0. In down count type: PWM counter counts from PERIOD to 0, and restarts from PERIOD. In up-down count type: PWM counter counts from 0 to PERIOD, then decrements to 0 and repeats again. In up and down count type: PWM period time = (PERIOD + 1) * (CLKPSC + 1) * TMRx_PWMCLK. In up-down count type: PWM period time = 2 * PERIOD * (CLKPSC+ 1) * TMRx_PWMCLK. **Note 1:** The count type of Timer4 and Timer5 is fixed as up count type. **Note 2:** User should take care DIRF (TIMERx_PWMCNT[16]) bit in up/down/up-down count type to monitor current counter direction in each count type in Timer0 ~ Timer3.

Timer PWM Comparator Register (TIMERx PWMCMPDAT)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMCMPDAT	TMR01_BA+0x54	R/W	Timer0 PWM Comparator Register	0x0000_0000
TIMER1_PWMCMPDAT	TMR01_BA+0x154	R/W	Timer1 PWM Comparator Register	0x0000_0000
TIMER2_PWMCMPDAT	TMR23_BA+0x54	R/W	Timer2 PWM Comparator Register	0x0000_0000
TIMER3_PWMCMPDAT	TMR23_BA+0x154	R/W	Timer3 PWM Comparator Register	0x0000_0000
TIMER4_PWMCMPDAT	TMR45_BA+0x54	R/W	Timer4 PWM Comparator Register	0x0000_0000
TIMER5_PWMCMPDAT	TMR45_BA+0x154	R/W	Timer5 PWM Comparator Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CMP**	**PWM Comparator Register** PWM CMP is used to compare with PWM CNT to generate PWM output waveform, interrupt events and trigger EADC and PDMA to start conversion.

Timer PWM Dead-time Control Register (TIMERx PWMDTCTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMDTCTL	TMR01_BA+0x58	R/W	Timer0 PWM Dead-Time Control Register	0x0000_0000
TIMER1_PWMDTCTL	TMR01_BA+0x158	R/W	Timer1 PWM Dead-Time Control Register	0x0000_0000
TIMER2_PWMDTCTL	TMR23_BA+0x58	R/W	Timer2 PWM Dead-Time Control Register	0x0000_0000
TIMER3_PWMDTCTL	TMR23_BA+0x158	R/W	Timer3 PWM Dead-Time Control Register	0x0000_0000

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

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24]	**DTCKSEL**	**Dead-time Clock Select (Write Protect)** 0 = Dead-time clock source from TMRx_PWMCLK without counter clock prescale. 1 = Dead-time clock source from TMRx_PWMCLK with counter clock prescale. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[23:17]	**Reserved**	Reserved.
[16]	**DTEN**	**Enable Dead-time Insertion for PWMx_CH0 and PWMx_CH1 (Write Protect)** Dead-time insertion function is only active when PWM complementary mode is enabled. If dead-time insertion is inactive, the outputs of PWMx_CH0 and PWMx_CH1 are complementary without any delay. 0 = Dead-time insertion Disabled on the pin pair. 1 = Dead-time insertion Enabled on the pin pair. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[15:12]	**Reserved**	Reserved.
[11:0]	**DTCNT**	**Dead-time Counter (Write Protect)** The dead-time can be calculated from the following two formulas: Dead-time = (DTCNT[11:0] + 1) * TMRx_PWMCLK, if DTCKSEL is 0. Dead-time = (DTCNT[11:0] + 1) * TMRx_PWMCLK * (CLKPSC + 1), if DTCKSEL is 1. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.

Timer PWM Counter Register (TIMERx PWMCNT)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMCNT	TMR01_BA+0x5C	R	Timer0 PWM Counter Register	0x0000_0000
TIMER1_PWMCNT	TMR01_BA+0x15C	R	Timer1 PWM Counter Register	0x0000_0000
TIMER2_PWMCNT	TMR23_BA+0x5C	R	Timer2 PWM Counter Register	0x0000_0000
TIMER3_PWMCNT	TMR23_BA+0x15C	R	Timer3 PWM Counter Register	0x0000_0000
TIMER4_PWMCNT	TMR45_BA+0x5C	R	Timer4 PWM Counter Register	0x0000_0000
TIMER5_PWMCNT	TMR45_BA+0x15C	R	Timer5 PWM Counter Register	0x0000_0000

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

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**DIRF**	**PWM Counter Direction Indicator Flag (Read Only)** 0 = Counter is active in down count. 1 = Counter is active up count. **Note 1:** This indicator flag is used for Timer0 ~ Timer3 only. **Note 2:** Since the count type of Timer4 ~ Timer5 is fixed as up count, this bit is fixed 0 in Timer4 and Timer5.
[15:0]	**CNT**	**PWM Counter Value Register (Read Only)** User can monitor CNT to know the current counter value in 16-bit period counter.

Timer PWM Output Mask Enable Register (TIMERx PWMMSKEN)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMMSKEN	TMR01_BA+0x60	R/W	Timer0 PWM Output Mask Enable Register	0x0000_0000
TIMER1_PWMMSKEN	TMR01_BA+0x160	R/W	Timer1 PWM Output Mask Enable Register	0x0000_0000
TIMER2_PWMMSKEN	TMR23_BA+0x60	R/W	Timer2 PWM Output Mask Enable Register	0x0000_0000
TIMER3_PWMMSKEN	TMR23_BA+0x160	R/W	Timer3 PWM Output Mask Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**MSKEN1**	**MSKEN0**

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**MSKEN1**	**PWMx_CH1 Output Mask Enable Bit** The PWMx_CH1 output signal will be masked when this bit is enabled. The PWMx_CH1 will output MSKDAT1 (TIMERx_PWMMSK[1]) data. 0 = PWMx_CH1 output signal is non-masked. 1 = PWMx_CH1 output signal is masked and output MSKDAT1 data.
[0]	**MSKEN0**	**PWMx_CH0 Output Mask Enable Bit** The PWMx_CH0 output signal will be masked when this bit is enabled. The PWMx_CH0 will output MSKDAT0 (TIMERx_PWMMSK[0]) data. 0 = PWMx_CH0 output signal is non-masked. 1 = PWMx_CH0 output signal is masked and output MSKDAT0 data.

Timer PWM Output Mask Data Control Register (TIMERx PWMMSK)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMMSK	TMR01_BA+0x64	R/W	Timer0 PWM Output Mask Data Control Register	0x0000_0000
TIMER1_PWMMSK	TMR01_BA+0x164	R/W	Timer1 PWM Output Mask Data Control Register	0x0000_0000
TIMER2_PWMMSK	TMR23_BA+0x64	R/W	Timer2 PWM Output Mask Data Control Register	0x0000_0000
TIMER3_PWMMSK	TMR23_BA+0x164	R/W	Timer3 PWM Output Mask Data Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						MSKDAT1	MSKDAT0

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**MSKDAT1**	**PWMx_CH1 Output Mask Data Control Bit** This bit is used to control the output state of PWMx_CH1 pin when PWMx_CH1 output mask function is enabled (MSKEN1 = 1). 0 = Output logic Low to PWMx_CH1. 1 = Output logic High to PWMx_CH1.
[0]	**MSKDAT0**	**PWMx_CH0 Output Mask Data Control Bit** This bit is used to control the output state of PWMx_CH0 pin when PWMx_CH0 output mask function is enabled (MSKEN0 = 1). 0 = Output logic Low to PWMx_CH0. 1 = Output logic High to PWMx_CH0.

Timer PWM Brake Pin Noise Filter Register (TIMERx PWMBNF)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMBNF	TMR01_BA+0x68	R/W	Timer0 PWM Brake Pin Noise Filter Register	0x0000_0000
TIMER1_PWMBNF	TMR01_BA+0x168	R/W	Timer1 PWM Brake Pin Noise Filter Register	0x0000_0000
TIMER2_PWMBNF	TMR23_BA+0x68	R/W	Timer2 PWM Brake Pin Noise Filter Register	0x0000_0000
TIMER3_PWMBNF	TMR23_BA+0x168	R/W	Timer3 PWM Brake Pin Noise Filter Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						BKPINSRC	
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
BRKPINV	BRKFCNT			BRKNFSEL			BRKNFEN

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17:16]	**BKPINSRC**	**Brake Pin Source Select** 00 = Brake pin source comes from PWM0_BRAKE0 pin. 01 = Brake pin source comes from PWM0_BRAKE1 pin. 10 = Brake pin source comes from PWM1_BRAKE0 pin. 11 = Brake pin source comes from PWM1_BRAKE1 pin.
[15:8]	**Reserved**	Reserved.
[7]	**BRKPINV**	**Brake Pin Detection Control Bit** 0 = Brake pin event will be detected if PWMx_BRAKEy pin status transfer from low to high in edge-detect, or pin status is high in level-detect. 1 = Brake pin event will be detected if PWMx_BRAKEy pin status transfer from high to low in edge-detect, or pin status is low in level-detect .
[6:4]	**BRKFCNT**	**Brake Pin Noise Filter Count** The fields is used to control the active noise filter sample time. Once noise filter sample time = (Period time of BRKNFSEL) * (BRKFCNT + 1).
[3:1]	**BRKNFSEL**	**Brake Pin Noise Filter Clock Selection** 000 = Noise filter clock is PCLKx. 001 = Noise filter clock is PCLKx/2. 010 = Noise filter clock is PCLKx/4. 011 = Noise filter clock is PCLKx/8. 100 = Noise filter clock is PCLKx/16.

		101 = Noise filter clock is PCLKx/32. 110 = Noise filter clock is PCLKx/64. 111 = Noise filter clock is PCLKx/128.
[0]	**BRKNFEN**	**Brake Pin Noise Filter Enable Bit** 0 = Pin noise filter detect of PWMx_BRAKEy Disabled. 1 = Pin noise filter detect of PWMx_BRAKEy Enabled.

Timer PWM System Fail Brake Control Register (TIMERx PWMFAILBRK)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMFAILBRK	TMR01_BA+0x6C	R/W	Timer0 PWM System Fail Brake Control Register	0x0000_0000
TIMER1_PWMFAILBRK	TMR01_BA+0x16C	R/W	Timer1 PWM System Fail Brake Control Register	0x0000_0000
TIMER2_PWMFAILBRK	TMR23_BA+0x6C	R/W	Timer2 PWM System Fail Brake Control Register	0x0000_0000
TIMER3_PWMFAILBRK	TMR23_BA+0x16C	R/W	Timer3 PWM System Fail Brake Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	2	4	3	2	1	0
Reserved				CORBRKEN	RAMBRKEN	BODBRKEN	CSSBRKEN

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**CORBRKEN**	**Core Lockup Detection Trigger PWM Brake Function Enable Bit** 0 = Brake Function triggered by core lockup event Disabled. 1 = Brake Function triggered by core lockup event Enabled.
[2]	**RAMBRKEN**	**SRAM Parity Error Detection Trigger PWM Brake Function Enable Bit** 0 = Brake Function triggered by SRAM parity error detection Disabled. 1 = Brake Function triggered by SRAM parity error detection Enabled.
[1]	**BODBRKEN**	**Brown-out Detection Trigger PWM Brake Function Enable Bit** 0 = Brake Function triggered by BOD event Disabled. 1 = Brake Function triggered by BOD event Enabled.
[0]	**CSSBRKEN**	**Clock Security System Detection Trigger PWM Brake Function Enable Bit** 0 = Brake Function triggered by clock fail detection Disabled. 1 = Brake Function triggered by clock fail detection Enabled.

Timer PWM Brake Control Register (TIMERx PWMBRKCTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMBRKCTL	TMR01_BA+0x70	R/W	Timer0 PWM Brake Control Register	0x0000_0000
TIMER1_PWMBRKCTL	TMR01_BA+0x170	R/W	Timer1 PWM Brake Control Register	0x0000_0000
TIMER2_PWMBRKCTL	TMR23_BA+0x70	R/W	Timer2 PWM Brake Control Register	0x0000_0000
TIMER3_PWMBRKCTL	TMR23_BA+0x170	R/W	Timer3 PWM Brake Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved				BRKAODD		BRKAEVEN	
15	14	13	12	11	10	9	8
SYSLBEN	Reserved		BRKPLEN	Reserved		CPO1LBEN	CPO0LBEN
7	6	5	4	3	2	1	0
SYSEBEN	Reserved		BRKPEEN	Reserved		CPO1EBEN	CPO0EBEN

Bits	Description	
[31:20]	**Reserved**	Reserved.
[19:18]	**BRKAODD**	**PWM Brake Action Select for PWMx_CH1 (Write Protect)** 00 = PWMx_BRAKEy brake event will not affect PWMx_CH1 output. 01 = PWMx_CH1 output tri-state when PWMx_BRAKEy brake event happened. 10 = PWMx_CH1 output low level when PWMx_BRAKEy brake event happened. 11 = PWMx_CH1 output high level when PWMx_BRAKEy brake event happened. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[17:16]	**BRKAEVEN**	**PWM Brake Action Select for PWMx_CH0 (Write Protect)** 00 = PWMx_BRAKEy brake event will not affect PWMx_CH0 output. 01 = PWMx_CH0 output tri-state when PWMx_BRAKEy brake event happened. 10 = PWMx_CH0 output low level when PWMx_BRAKEy brake event happened. 11 = PWMx_CH0 output high level when PWMx_BRAKEy brake event happened. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[15]	**SYSLBEN**	**Enable System Fail As Level-detect Brake Source (Write Protect)** 0 = System fail condition as level-detect brake source Disabled. 1 = System fail condition as level-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[14:13]	**Reserved**	Reserved.
[12]	**BRKPLEN**	**Enable TM_BRAKEx Pin As Level-detect Brake Source (Write Protect)** 0 = PWMx_BRAKEy pin event as level-detect brake source Disabled. 1 = PWMx_BRAKEy pin event as level-detect brake source Enabled.

		Note: This bit is write protected. Refer to SYS_REGLCTL register.
[11:10]	**Reserved**	Reserved.
[9]	**CPO1LBEN**	**Enable Internal ACMP1_O Digital Output As Level-detect Brake Source (Write Protect)** 0 = Internal ACMP1_O signal as level-detect brake source Disabled. 1 = Internal ACMP1_O signal as level-detect brake source Enabled. **Note 1:** Only internal ACMP1_O signal from low to high will be detected as brake event. **Note 2:** This bit is write protected. Refer to SYS_REGLCTL register.
[8]	**CPO0LBEN**	**Enable Internal ACMP0_O Digital Output As Level-detect Brake Source (Write Protect)** 0 = Internal ACMP0_O signal as level-detect brake source Disabled. 1 = Internal ACMP0_O signal as level-detect brake source Enabled. **Note 1:** Only internal ACMP0_O signal from low to high will be detected as brake event. **Note 2:** This bit is write protected. Refer to SYS_REGLCTL register.
[7]	**SYSEBEN**	**Enable System Fail As Edge-detect Brake Source (Write Protect)** 0 = System fail condition as edge-detect brake source Disabled. 1 = System fail condition as edge-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[6:5]	**Reserved**	Reserved.
[4]	**BRKPEEN**	**Enable TM_BRAKEx Pin As Edge-detect Brake Source (Write Protect)** 0 = PWMx_BRAKEy pin event as edge-detect brake source Disabled. 1 = PWMx_BRAKEy pin event as edge-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[3:2]	**Reserved**	Reserved.
[1]	**CPO1EBEN**	**Enable Internal ACMP1_O Digital Output As Edge-detect Brake Source (Write Protect)** 0 = Internal ACMP1_O signal as edge-detect brake source Disabled. 1 = Internal ACMP1_O signal as edge-detect brake source Enabled. **Note 1:** Only internal ACMP1_O signal from low to high will be detected as brake event. **Note 2:** This bit is write protected. Refer to SYS_REGLCTL register.
[0]	**CPO0EBEN**	**Enable Internal ACMP0_O Digital Output As Edge-detect Brake Source (Write Protect)** 0 = Internal ACMP0_O signal as edge-detect brake source Disabled. 1 = Internal ACMP0_O signal as edge-detect brake source Enabled. **Note 1:** Only internal ACMP0_O signal from low to high will be detected as brake event. **Note 2:** This bit is write protected. Refer to SYS_REGLCTL register.

Timer PWM Pin Output Polar Control Register (TIMERx PWMPOLCTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMPOLCTL	TMR01_BA+0x74	R/W	Timer0 PWM Pin Output Polar Control Register	0x0000_0000
TIMER1_PWMPOLCTL	TMR01_BA+0x174	R/W	Timer1 PWM Pin Output Polar Control Register	0x0000_0000
TIMER2_PWMPOLCTL	TMR23_BA+0x74	R/W	Timer2 PWM Pin Output Polar Control Register	0x0000_0000
TIMER3_PWMPOLCTL	TMR23_BA+0x174	R/W	Timer3 PWM Pin Output Polar Control Register	0x0000_0000
TIMER4_PWMPOLCTL	TMR45_BA+0x74	R/W	Timer4 PWM Pin Output Polar Control Register	0x0000_0000
TIMER5_PWMPOLCTL	TMR45_BA+0x174	R/W	Timer5 PWM Pin Output Polar Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**PINV1**	**PINV0**

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**PINV1**	**PWMx_CH1 Output Pin Polar Control Bit** The bit is used to control polarity state of PWMx_CH1 output pin. 0 = PWMx_CH1 output pin polar inverse Disabled. 1 = PWMx_CH1 output pin polar inverse Enabled. **Note:** This bit is not available in Timer4 and Timer5.
[0]	**PINV0**	**PWMx_CH0 Output Pin Polar Control Bit** The bit is used to control polarity state of PWMx_CH0 output pin. 0 = PWMx_CH0 output pin polar inverse Disabled. 1 = PWMx_CH0 output pin polar inverse Enabled. **Note:** In Timer4 and Timer5, the PWMx_CH0 output pin can be selected as TMx or TMx_EXT pin by POSEL (TIMERx_PWMPOEN[8]), x= 4~5.

Timer PWM Pin Output Enable Register (TIMERx PWMPOEN)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMPOEN	TMR01_BA+0x78	R/W	Timer0 PWM Pin Output Enable Register	0x0000_0000
TIMER1_PWMPOEN	TMR01_BA+0x178	R/W	Timer1 PWM Pin Output Enable Register	0x0000_0000
TIMER2_PWMPOEN	TMR23_BA+0x78	R/W	Timer2 PWM Pin Output Enable Register	0x0000_0000
TIMER3_PWMPOEN	TMR23_BA+0x178	R/W	Timer3 PWM Pin Output Enable Register	0x0000_0000
TIMER4_PWMPOEN	TMR45_BA+0x78	R/W	Timer4 PWM Pin Output Enable Register	0x0000_0000
TIMER5_PWMPOEN	TMR45_BA+0x178	R/W	Timer5 PWM Pin Output Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							**POSEL**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**POEN1**	**POEN0**

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**POSEL**	**PWMx_CH0 Output Pin Select** This bit is used to select the output channel of Timer4 and Timer5 PWM. 0 = PWMx_CH0 pin is TMx. 1 = PWMx_CH0 pin is TMx_EXT. **Note:** This bit is only available in Timer4 and Timer5.
[7:2]	**Reserved**	Reserved.
[1]	**POEN1**	**PWMx_CH1 Output Pin Enable Bit** 0 = PWMx_CH1 pin at tri-state mode. 1 = PWMx_CH1 pin in output mode. **Note:** This bit is not available in Timer4 and Timer5.
[0]	**POEN0**	**PWMx_CH0 Output Pin Enable Bit** 0 = PWMx_CH0 pin at tri-state mode. 1 = PWMx_CH0 pin in output mode. **Note:** In Timer4 and Timer5, the PWMx_CH0 output pin can be selected as TMx or TMx_EXT pin by POSEL (TIMERx_PWMPOEN[8]), x= 4~5.

Timer PWM Software Trigger Brake Control Register (TIMERx PWMSWBRK)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMSWBRK	TMR01_BA+0x7C	W	Timer0 PWM Software Trigger Brake Control Register	0x0000_0000
TIMER1_PWMSWBRK	TMR01_BA+0x17C	W	Timer1 PWM Software Trigger Brake Control Register	0x0000_0000
TIMER2_PWMSWBRK	TMR23_BA+0x7C	W	Timer2 PWM Software Trigger Brake Control Register	0x0000_0000
TIMER3_PWMSWBRK	TMR23_BA+0x17C	W	Timer3 PWM Software Trigger Brake Control Register	0x0000_0000

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

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**BRKLTRG**	**Software Trigger Level-detect Brake Source (Write Only) (Write Protect)** Write 1 to this bit will trigger PWM level-detect brake source, then BRKLIF0 and BRKLIF1 will be set to 1 automatically in TIMERx_PWMINTSTS1 register. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[7:1]	**Reserved**	Reserved.
[0]	**BRKETRG**	**Software Trigger Edge-detect Brake Source (Write Only) (Write Protect)** Write 1 to this bit will trigger PWM edge-detect brake source, then BRKEIF0 and BRKEIF1 will be set to 1 automatically in TIMERx_PWMINTSTS1 register. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.

Timer PWM Interrupt Enable Register 0 (TIMERx PWMINTEN0)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMINTEN0	TMR01_BA+0x80	R/W	Timer0 PWM Interrupt Enable Register 0	0x0000_0000
TIMER1_PWMINTEN0	TMR01_BA+0x180	R/W	Timer1 PWM Interrupt Enable Register 0	0x0000_0000
TIMER2_PWMINTEN0	TMR23_BA+0x80	R/W	Timer2 PWM Interrupt Enable Register 0	0x0000_0000
TIMER3_PWMINTEN0	TMR23_BA+0x180	R/W	Timer3 PWM Interrupt Enable Register 0	0x0000_0000
TIMER4_PWMINTEN0	TMR45_BA+0x80	R/W	Timer4 PWM Interrupt Enable Register 0	0x0000_0000
TIMER5_PWMINTEN0	TMR45_BA+0x180	R/W	Timer5 PWM Interrupt Enable Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**CMPDIEN**	**CMPUIEN**	**PIEN**	**ZIEN**

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**CMPDIEN**	**PWM Compare Down Count Interrupt Enable Bit** 0 = Compare down count interrupt Disabled. 1 = Compare down count interrupt Enabled. **Note:** This bit is not available in Timer4 and Timer5
[2]	**CMPUIEN**	**PWM Compare Up Count Interrupt Enable Bit** 0 = Compare up count interrupt Disabled. 1 = Compare up count interrupt Enabled.
[1]	**PIEN**	**PWM Period Point Interrupt Enable Bit** 0 = Period point interrupt Disabled. 1 = Period point interrupt Enabled. **Note:** In up-down count type, period point means the center point of current PWM period.
[0]	**ZIEN**	**PWM Zero Point Interrupt Enable Bit** 0 = Zero point interrupt Disabled. 1 = Zero point interrupt Enabled. **Note:** This bit is not available in Timer4 and Timer5

Timer PWM Interrupt Enable Register 1 (TIMERx PWMINTEN1)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMINTEN1	TMR01_BA+0x84	R/W	Timer0 PWM Interrupt Enable Register 1	0x0000_0000
TIMER1_PWMINTEN1	TMR01_BA+0x184	R/W	Timer1 PWM Interrupt Enable Register 1	0x0000_0000
TIMER2_PWMINTEN1	TMR23_BA+0x84	R/W	Timer2 PWM Interrupt Enable Register 1	0x0000_0000
TIMER3_PWMINTEN1	TMR23_BA+0x184	R/W	Timer3 PWM Interrupt Enable Register 1	0x0000_0000

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

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**BRKLIEN**	**PWM Level-detect Brake Interrupt Enable Bit (Write Protect)** 0 = PWM level-detect brake interrupt Disabled. 1 = PWM level-detect brake interrupt Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[7:1]	**Reserved**	Reserved.
[0]	**BRKEIEN**	**PWM Edge-detect Brake Interrupt Enable Bit (Write Protect)** 0 = PWM edge-detect brake interrupt Disabled. 1 = PWM edge-detect brake interrupt Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.

Timer PWM Interrupt Status Register 0 (TIMERx PWMINTSTS0)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMINTSTS0	TMR01_BA+0x88	R/W	Timer0 PWM Interrupt Status Register 0	0x0000_0000
TIMER1_PWMINTSTS0	TMR01_BA+0x188	R/W	Timer1 PWM Interrupt Status Register 0	0x0000_0000
TIMER2_PWMINTSTS0	TMR23_BA+0x88	R/W	Timer2 PWM Interrupt Status Register 0	0x0000_0000
TIMER3_PWMINTSTS0	TMR23_BA+0x188	R/W	Timer3 PWM Interrupt Status Register 0	0x0000_0000
TIMER4_PWMINTSTS0	TMR45_BA+0x88	R/W	Timer4 PWM Interrupt Status Register 0	0x0000_0000
TIMER5_PWMINTSTS0	TMR45_BA+0x188	R/W	Timer5 PWM Interrupt Status Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				CMPDIF	CMPUIF	PIF	ZIF

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**CMPDIF**	**PWM Compare Down Count Interrupt Flag** This bit is set by hardware when TIMERx_PWM counter in down count direction and reaches CMP. **Note 1:** This bit is not available in Timer4 and Timer5 **Note 2:** If CMP equal to PERIOD, there is no CMPDIF flag in down count type. **Note 3:** This bit is cleared by writing 1 to it.
[2]	**CMPUIF**	**PWM Compare Up Count Interrupt Flag** This bit is set by hardware when TIMERx_PWM counter in up count direction and reaches CMP. **Note 1:** If CMP equal to PERIOD, there is no CMPUIF flag in up count type and up-down count type. **Note 2:** This bit is cleared by writing 1 to it.
[1]	**PIF**	**PWM Period Point Interrupt Flag** This bit is set by hardware when TIMERx_PWM counter reaches PERIOD. **Note 1:** In up-down count type, PIF flag means the center point flag of current PWM period. **Note 2:** This bit is cleared by writing 1 to it.
[0]	**ZIF**	**PWM Zero Point Interrupt Flag** This bit is set by hardware when TIMERx_PWM counter reaches 0. **Note 1:** This bit is not available in Timer4 and Timer5 **Note 2:** This bit is cleared by writing 1 to it.

Timer PWM Interrupt Status Register 1 (TIMERx PWMINTSTS1)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMINTSTS1	TMR01_BA+0x8C	R/W	Timer0 PWM Interrupt Status Register 1	0x0000_0000
TIMER1_PWMINTSTS1	TMR01_BA+0x18C	R/W	Timer1 PWM Interrupt Status Register 1	0x0000_0000
TIMER2_PWMINTSTS1	TMR23_BA+0x8C	R/W	Timer2 PWM Interrupt Status Register 1	0x0000_0000
TIMER3_PWMINTSTS1	TMR23_BA+0x18C	R/W	Timer3 PWM Interrupt Status Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved						BRKLSTS1	BRKLSTS0
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved						BRKESTS1	BRKESTS0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved						BRKLIF1	BRKLIF0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						BRKEIF1	BRKEIF0

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25]	**BRKLSTS1**	**Level-detect Brake Status of PWMx_CH1 (Read Only)** 0 = PWMx_CH1 level-detect brake state is released. 1 = PWMx_CH1 at level-detect brake state. **Note:** If TIMERx_PWM level-detect brake source has released, both PWMx_CH0 and PWMx_CH1 will release brake state when current PWM period finished and resume PWMx_CH0 and PWMx_CH1 output waveform start from next full PWM period.
[24]	**BRKLSTS0**	**Level-detect Brake Status of PWMx_CH0 (Read Only)** 0 = PWMx_CH0 level-detect brake state is released. 1 = PWMx_CH0 at level-detect brake state. **Note:** If TIMERx_PWM level-detect brake source has released, both PWMx_CH0 and PWMx_CH1 will release brake state when current PWM period finished and resume PWMx_CH0 and PWMx_CH1 output waveform start from next full PWM period.
[23:18]	**Reserved**	Reserved.
[17]	**BRKESTS1**	**Edge-detect Brake Status of PWMx_CH1 (Read Only)** 0 = PWMx_CH1 edge-detect brake state is released. 1 = PWMx_CH1 at edge-detect brake state. **Note:** User can set BRKEIF1 1 to clear BRKEIF1 flag and PWMx_CH1 will release brake state when current PWM period finished and resume PWMx_CH1 output waveform start from next full PWM period.
[16]	**BRKESTS0**	**Edge -detect Brake Status of PWMx_CH0 (Read Only)** 0 = PWMx_CH0 edge-detect brake state is released. 1 = PWMx_CH0 at edge-detect brake state. **Note:** User can set BRKEIF0 1 to clear BRKEIF0 flag and PWMx_CH0 will release brake state when

		current PWM period finished and resume PWMx_CH0 output waveform start from next full PWM period.
[15:10]	**Reserved**	Reserved.
[9]	**BRKLIF1**	**Level-detect Brake Interrupt Flag on PWMx_CH1 (Write Protect)** 0 = PWMx_CH1 level-detect brake event do not happen. 1 = PWMx_CH1 level-detect brake event happened. **Note 1:** This bit is cleared by writing 1 to it. **Note 2:** This bit is write protected. Refer to SYS_REGLCTL register.
[8]	**BRKLIF0**	**Level-detect Brake Interrupt Flag on PWMx_CH0 (Write Protect)** 0 = PWMx_CH0 level-detect brake event do not happen. 1 = PWMx_CH0 level-detect brake event happened. **Note 1:** This bit is cleared by writing 1 to it. **Note 2:** This bit is write protected. Refer to SYS_REGLCTL register.
[7:2]	**Reserved**	Reserved.
[1]	**BRKEIF1**	**Edge-detect Brake Interrupt Flag PWMx_CH1 (Write Protect)** 0 = PWMx_CH1 edge-detect brake event do not happen. 1 = PWMx_CH1 edge-detect brake event happened. **Note 1:** This bit is cleared by writing 1 to it. **Note 2:** This bit is write protected. Refer to SYS_REGLCTL register.
[0]	**BRKEIF0**	**Edge-detect Brake Interrupt Flag on PWMx_CH0 (Write Protect)** 0 = PWMx_CH0 edge-detect brake event do not happen. 1 = PWMx_CH0 edge-detect brake event happened. **Note 1:** This bit is cleared by writing 1 to it. **Note 2:** This bit is write protected. Refer to SYS_REGLCTL register.

Timer PWM Trigger Control Register (TIMERx PWMTRGCTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMTRGCTL	TMR01_BA+0x90	R/W	Timer0 PWM Trigger Control Register	0x0000_0000
TIMER1_PWMTRGCTL	TMR01_BA+0x190	R/W	Timer1 PWM Trigger Control Register	0x0000_0000
TIMER2_PWMTRGCTL	TMR23_BA+0x90	R/W	Timer2 PWM Trigger Control Register	0x0000_0000
TIMER3_PWMTRGCTL	TMR23_BA+0x190	R/W	Timer3 PWM Trigger Control Register	0x0000_0000
TIMER4_PWMTRGCTL	TMR45_BA+0x90	R/W	Timer4 PWM Trigger Control Register	0x0000_0000
TIMER5_PWMTRGCTL	TMR45_BA+0x190	R/W	Timer5 PWM Trigger Control Register	0x0000_0000

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

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**TRGPDMA**	**PWM Counter Event Trigger PDMA Conversion Enable Bit** 0 = PWM counter event trigger PDMA Disabled. 1 = PWM counter event trigger PDMA Enabled. **Note 1:** This bit is only available in Timer4 and Timer5. **Note 2:** Set TRGSEL (TIMERx_PWMTRGCTL[2:0]) to select PWM trigger conversion source.
[8]	**Reserved**	Reserved.
[7]	**TRGEADC**	**PWM Counter Event Trigger EADC Conversion Enable Bit** 0 = PWM counter event trigger EADC conversion Disabled. 1 = PWM counter event trigger EADC conversion Enabled. **Note:** Set TRGSEL (TIMERx_PWMTRGCTL[2:0]) to select PWM trigger conversion source.
[6:3]	**Reserved**	Reserved.
[2:0]	**TRGSEL**	**PWM Counter Event Source Select to Trigger Conversion** In Timer0 ~ Timer3, 000 = Trigger conversion at zero point (ZIF). 001 = Trigger conversion at period point (PIF). 010 = Trigger conversion at zero or period point (ZIF or PIF). 011 = Trigger conversion at compare up count point (CMPUIF).

		100 = Trigger conversion at compare down count point (CMPDIF). In Timer4 and Timer5, 001 = Trigger conversion at period point (PIF). 011 = Trigger conversion at compare up count point (CMPUIF). 101 = Trigger conversion at period or compare up count point (PIF or CMPUIF). Others = Reserved.

Timer PWM Synchronous Control Register (TIMERx PWMSCTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMSCTL	TMR01_BA+0x94	R/W	Timer0 PWM Synchronous Control Register	0x0000_0000
TIMER1_PWMSCTL	TMR01_BA+0x194	R/W	Timer1 PWM Synchronous Control Register	0x0000_0000
TIMER2_PWMSCTL	TMR23_BA+0x94	R/W	Timer2 PWM Synchronous Control Register	0x0000_0000
TIMER3_PWMSCTL	TMR23_BA+0x194	R/W	Timer3 PWM Synchronous Control Register	0x0000_0000

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

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**SYNCSRC**	**PWM Synchronous Counter Start/Clear Source Select** 0 = Counter synchronous start/clear by trigger TIMER0_PWMSTRG STRGEN. 1 = Counter synchronous start/clear by trigger TIMER2_PWMSTRG STRGEN. **Note 1:** If TIMER0/1/2/3 PWM counter synchronous source are from TIMER0, TIMER0_PWMSCTL[8], TIMER1_PWMSCTL[8], TIMER2_PWMSCTL[8] and TIMER3_PWMSCTL[8] should be 0. **Note 2:** If TIMER0/1/ PWM counter synchronous source are from TIMER0, TIMER0_PWMSCTL[8] and TIMER1_PWMSCTL[8] should be set 0, and TIMER2/3/ PWM counter synchronous source are from TIMER2, TIME2_PWMSCTL[8] and TIMER3_PWMSCTL[8] should be set 1.
[7:2]	**Reserved**	Reserved.
[1:0]	**SYNCMODE**	**PWM Synchronous Mode Enable Select** 00 = PWM synchronous function Disabled. 01 = PWM synchronous counter start function Enabled. 10 = Reserved. 11 = PWM synchronous counter clear function Enabled.

Timer PWM Synchronous Trigger Register (TIMERx PWMSTRG)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMSTRG	TMR01_BA+0x98	W	Timer0 PWM Synchronous Trigger Register	0x0000_0000
TIMER2_PWMSTRG	TMR23_BA+0x98	W	Timer2 PWM Synchronous Trigger Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**STRGEN**	**PWM Counter Synchronous Trigger Enable Bit (Write Only)** PMW counter synchronous function is used to make selected PWM channels (including TIMER0/1/2/3 PWM, TIMER0/1 PWM and TIMER2/3 PWM) start counting or clear counter at the same time according to TIMERx_PWMSCTL setting. **Note:** This bit is only available in TIMER0 and TIMER2.

Timer PWM Status Register (TIMERx PWMSTATUS)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMSTATUS	TMR01_BA+0x9C	R/W	Timer0 PWM Status Register	0x0000_0000
TIMER1_PWMSTATUS	TMR01_BA+0x19C	R/W	Timer1 PWM Status Register	0x0000_0000
TIMER2_PWMSTATUS	TMR23_BA+0x9C	R/W	Timer2 PWM Status Register	0x0000_0000
TIMER3_PWMSTATUS	TMR23_BA+0x19C	R/W	Timer3 PWM Status Register	0x0000_0000
TIMER4_PWMSTATUS	TMR45_BA+0x9C	R/W	Timer4 PWM Status Register	0x0000_0000
TIMER5_PWMSTATUS	TMR45_BA+0x19C	R/W	Timer5 PWM Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved					PDMATRGF	Reserved	EADCTRGF
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							WKF
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							CNTMAXF

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18]	**PDMATRGF**	**Trigger PDMA Start Conversion Flag** 0 = PWM counter event trigger PDMA start conversion did not occur. 1 = PWM counter event trigger PDMA start conversion occurred. **Note 1:** This bit is only available in Timer4 and Timer5. **Note 2:** This bit is cleared by writing 1 to it.
[17]	**Reserved**	Reserved.
[16]	**EADCTRGF**	**Trigger EADC Start Conversion Flag** 0 = PWM counter event trigger EADC start conversion did not occur. 1 = PWM counter event trigger EADC start conversion occurred. **Note:** This bit is cleared by writing 1 to it.
[15:9]	**Reserved**	Reserved.
[8]	**WKF**	**PWM Wake-up Flag** 0 = PWM interrupt wake-up did not occur. 1 = PWM interrupt wake-up occurred. **Note 1:** This bit is only available in Timer4 and Timer5. **Note 2:** This bit is cleared by writing 1 to it.

[7:1]	**Reserved**	Reserved.
[0]	**CNTMAXF**	**PWM Counter Equal to 0xFFFF Flag** 0 = The PWM counter value never reached its maximum value 0xFFFF. 1 = The PWM counter value has reached its maximum value. **Note:** This bit is cleared by writing 1 to it.

Timer PWM Period Buffer Register (TIMERx PWMPBUF)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMPBUF	TMR01_BA+0xA0	R	Timer0 PWM Period Buffer Register	0x0000_0000
TIMER1_PWMPBUF	TMR01_BA+0x1A0	R	Timer1 PWM Period Buffer Register	0x0000_0000
TIMER2_PWMPBUF	TMR23_BA+0xA0	R	Timer2 PWM Period Buffer Register	0x0000_0000
TIMER3_PWMPBUF	TMR23_BA+0x1A0	R	Timer3 PWM Period Buffer Register	0x0000_0000
TIMER4_PWMPBUF	TMR45_BA+0xA0	R	Timer4 PWM Period Buffer Register	0x0000_0000
TIMER5_PWMPBUF	TMR45_BA+0x1A0	R	Timer5 PWM Period Buffer Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PBUF**	**PWM Period Buffer Register (Read Only)** Used as PERIOD active register.

Timer PWM Comparator Buffer Register (TIMERx PWMCMPBUF)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMCMPBUF	TMR01_BA+0xA4	R	Timer0 PWM Comparator Buffer Register	0x0000_0000
TIMER1_PWMCMPBUF	TMR01_BA+0x1A4	R	Timer1 PWM Comparator Buffer Register	0x0000_0000
TIMER2_PWMCMPBUF	TMR23_BA+0xA4	R	Timer2 PWM Comparator Buffer Register	0x0000_0000
TIMER3_PWMCMPBUF	TMR23_BA+0x1A4	R	Timer3 PWM Comparator Buffer Register	0x0000_0000
TIMER4_PWMCMPBUF	TMR45_BA+0xA4	R	Timer4 PWM Comparator Buffer Register	0x0000_0000
TIMER5_PWMCMPBUF	TMR45_BA+0x1A4	R	Timer5 PWM Comparator Buffer Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CMPBUF**	**PWM Comparator Buffer Register (Read Only)** Used as CMP active register.

6.10 Watchdog Timer (WDT)

6.10.1 Overview

The Watchdog Timer (WDT) is used to perform a system reset when system runs into an unknown state. This prevents system from hanging for an infinite period of time. Besides, this Watchdog Timer supports the function to wake up system from Idle/Power-down mode.

6.10.2 Features

- 20-bit free running up counter for WDT time-out interval
- Selectable time-out interval (2^4 ~ 2^{20}) and the time-out interval is 0.5 ms ~ 32.768 s if WDT_CLK = 32 kHz.
- System kept in reset state for a period of (1 / WDT_CLK) * 63
- Supports selectable WDT reset delay period, including 1026, 130, 18 or 3 WDT_CLK reset delay period
- Supports to force WDT enabled after chip powered on or reset by setting CWDTEN[2:0] in Config0 register
- Supports WDT time-out wake-up function only if WDT clock source is selected as LIRC 32kHz or LXT.

6.10.3 Block Diagram

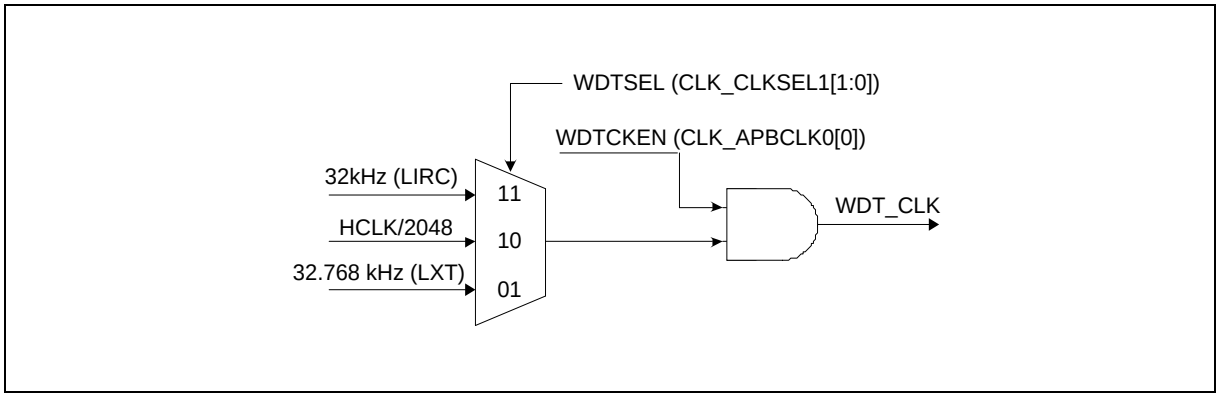
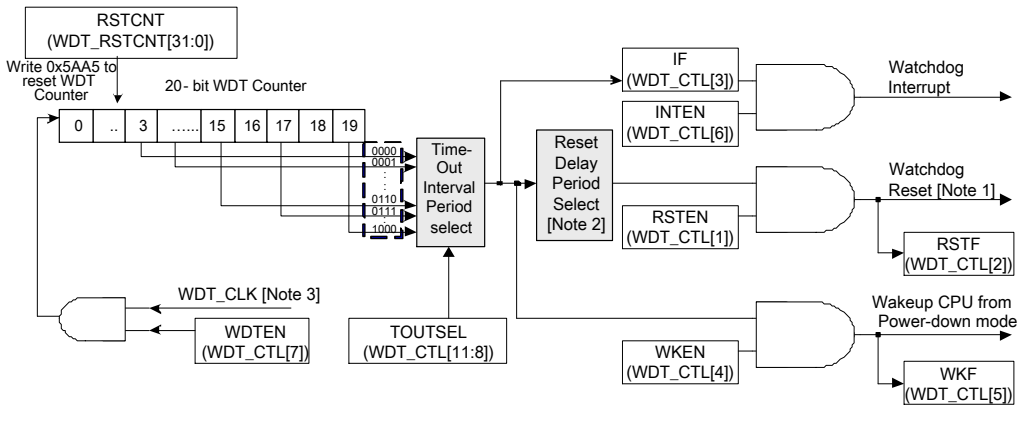


Figure 6.10-1 Watchdog Timer Clock Control



Note1: WDT resets CPU and lasts 63 WDT_CLK.

Note2: The WDT reset delay period can be selected as 3/18/130/1026 WDT_CLK.

Note3: Chip can be woken up by WDT time-out interrupt signal generated only if WDT clock source is selected to LXT or LIRC

Figure 6.10-2 Watchdog Timer Block Diagram

6.10.4 Basic Configuration

- Clock Source Configuration
  - Select the source of WDT peripheral clock on WDTSEL (CLK_CLKSEL1[1:0])
  - Enable WDT peripheral clock in WDTCKEN (CLK_APBCLK0[0]).
  - Force enable and active WDT controller after chip powered on or reset in CWDTEN[2:0] (CWDTEN[2] is Config0[31], CWDTEN[1:0] is Config0[4:3]) while CWDTEN[2:0] is not configure to 111.

6.10.5 Functional Description

The WDT includes an 20-bit free running up counter with programmable time-out intervals. Table 6.10-1 shows the WDT time-out interval period selection and Figure 6.10-1 shows the WDT time-out interval and reset period timing.

6.10.5.1 WDT Time-out Interrupt

Setting WDTEN (WDT_CTL[7]) to 1 will enable the WDT function and the WDT counter to start counting up. The SYNC (WDT_CTL[30]) can be indicated whether enable/disable WDTEN function is completed or not. There are nine time-out interval period can be selected by setting TOUTSEL (WDT_CTL[11:8]). When the WDT up counter reaches the TOUTSEL (WDT_CTL[11:8]) settings, WDT time-out interrupt will occur then WDT time-out interrupt flag IF (WDT_CTL[3]) will be set to 1 immediately. If INTEN (WDT_CTL[6]) is enabled, WDT time-out interrupt will inform CPU.

6.10.5.2 WDT Reset Delay Period and Reset System

There is a specified T_{RSTD} reset delay period follows the IF (WDT_CTL[3]) is setting to 1. User should program 0x00005AA5 to RSTCNT (WDT_RSTCNT[31:0]) to reset the 20-bit WDT up counter value to avoid generate WDT time-out reset signal before the T_{RSTD} reset delay period expires. Moreover, user should set RSTDSEL (WDT_ALTCTL[1:0]) to select reset delay period to clear WDT counter. If the

WDT up counter value has not been cleared after the specific T_{RSTD} delay period expires, the WDT control will set RSTF (WDT_CTL[2]) to 1 if RSTEN (WDT_CTL[1]) bit is enabled, then chip enters to reset state immediately. T_{RST} reset period will keep last 63 WDT clocks then chip restart executing program from reset vector (0x0000_0000). The RSTF (WDT_CTL[2]) will keep 1 after WDT time-out resets the chip, user can check RSTF (WDT_CTL[2]) by software to recognize the system has been reset by WDT time-out reset or not.

TOUTSEL	Time-Out Interval Period T_{TIS}	Reset Delay Period T_{RSTD}
0000	$2^4 * T_{WDT}$	(3/18/130/1026) * T_{WDT}
0001	$2^6 * T_{WDT}$	(3/18/130/1026) * T_{WDT}
0010	$2^8 * T_{WDT}$	(3/18/130/1026) * T_{WDT}
0011	$2^{10} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
0100	$2^{12} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
0101	$2^{14} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
0110	$2^{16} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
0111	$2^{18} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
1000	$2^{20} * T_{WDT}$	(3/18/130/1026) * T_{WDT}

Table 6.10-1 Watchdog Timer Time-out Interval Period Selection

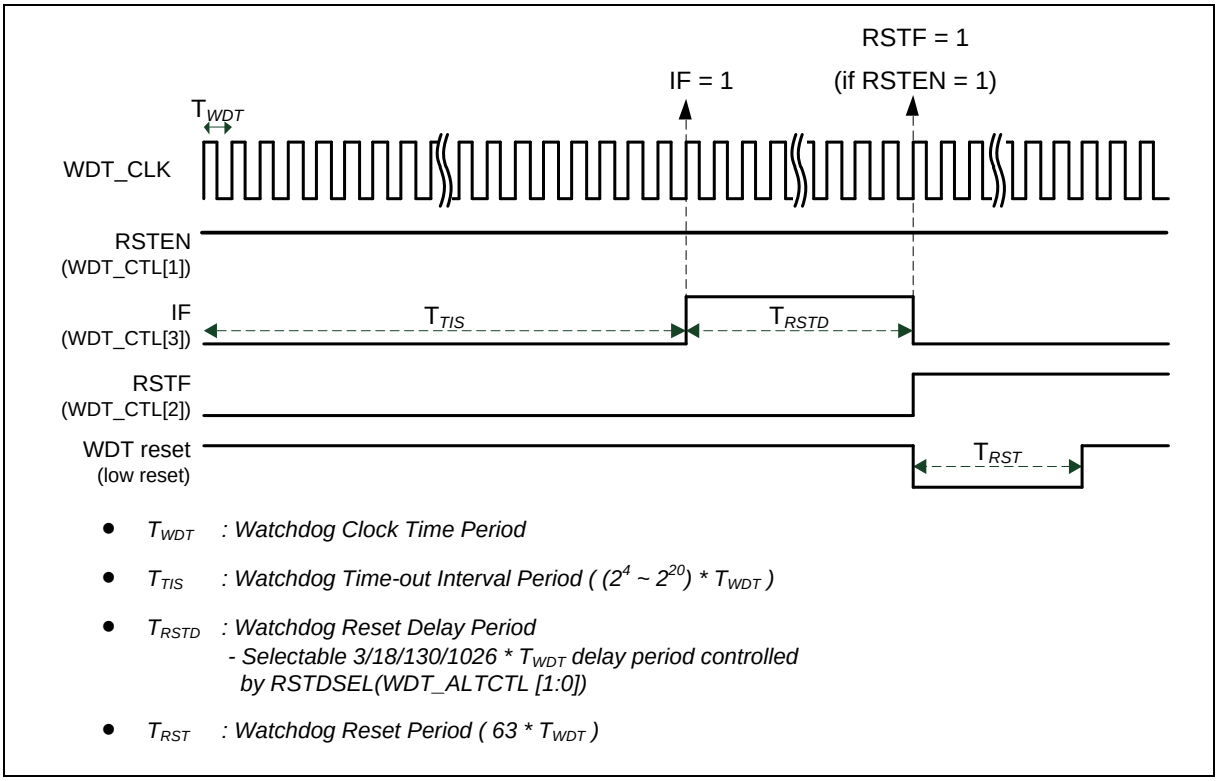


Figure 6.10-1 Watchdog Timer Time-out Interval and Reset Period Timing

6.10.5.3 WDT Wake-up

If WDT clock source is selected to 32 kHz or LXT, system can be woken up from Power-down mode while WDT time-out interrupt signal is generated and WKEN (WDT_CTL[4]) enabled. Note that user should set LXTEN (CLK_PWRCTL[1]) or LIRCEN (CLK_PWRCTL[3]) to select clock source before system enters Power-down mode because the system peripheral clock are disabled when system is in Power-down mode. In the meanwhile, the WKF (WDT_CTL[5]) will be set to 1 automatically, and user can check WKF (WDT_CTL[5]) status by software to recognize the system has been woken up by WDT time-out interrupt or not.

6.10.5.4 WDT ICE Debug

When ICE is connected to MCU, WDT counter is counting or not by ICEDEBUG (WDT_CTL[31]). The default value of ICEDEBUG is 0, WDT counter will stop counting when CPU is held by ICE. If ICEDEBUG is set to 1, WDT counter will keep counting no matter CPU is held by ICE or not.

6.10.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
WDT Base Address: **WDT_BA = 0x4004_0000**				
WDT_CTL	WDT_BA+0x00	R/W	WDT Control Register	0x0000_0800
WDT_ALTCTL	WDT_BA+0x04	R/W	WDT Alternative Control Register	0x0000_0000
WDT_RSTCNT	WDT_BA+0x08	W	WDT Reset Counter Register	0x0000_0000

6.10.7 Register Description

WDT Control Register (WDT_CTL)

Register	Offset	R/W	Description	Reset Value
WDT_CTL	WDT_BA+0x00	R/W	WDT Control Register	0x0000_0800

31	30	29	28	27	26	25	24
ICEDEBUG	SYNC	Reserved					
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				TOUTSEL			
7	6	5	4	3	2	1	0
WDTEN	INTEN	WKF	WKEN	IF	RSTF	RSTEN	Reserved

Bits	Description	
[31]	**ICEDEBUG**	**ICE Debug Mode Acknowledge Disable Bit (Write Protect)** 0 = ICE debug mode acknowledgement affects WDT counting. WDT up counter will be held while CPU is held by ICE. 1 = ICE debug mode acknowledgement Disabled. WDT up counter will keep going no matter CPU is held by ICE or not. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[30]	**SYNC**	**WDT Enable Control SYNC Flag Indicator (Read Only)** If user executes enable/disable WDTEN (WDT_CTL[7]), this flag can be indicated enable/disable WDTEN function is completed or not. 0 = Set WDTEN bit is completed. 1 = Set WDTEN bit is synchronizing and not become active yet. **Note:** Performing enable or disable WDTEN bit needs 2 * WDT_CLK period to become active.
[29:12]	**Reserved**	Reserved.
[11:8]	**TOUTSEL**	**WDT Time-out Interval Selection (Write Protect)** These three bits select the time-out interval period for the WDT. 0000 = 2^4 * WDT_CLK. 0001 = 2^6 * WDT_CLK. 0010 = 2^8 * WDT_CLK. 0011 = 2^{10} * WDT_CLK. 0100= 2^{12} * WDT_CLK. 0101 = 2^{14} * WDT_CLK. 0110 = 2^{16} * WDT_CLK. 0111 = 2^{18} * WDT_CLK. 1000 = 2^{20} * WDT_CLK. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

[7]	**WDTEN**	**WDT Enable Bit (Write Protect)** 0 = WDT Disabled (This action will reset the internal up counter value). 1 = WDT Enabled. **Note 1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note 2:** If CWDTEN[2:0] (combined by Config0[31] and Config0[4:3]) bits is not configured to 111, this bit is forced as 1 and user cannot change this bit to 0.
[6]	**INTEN**	**WDT Time-out Interrupt Enable Bit (Write Protect)** If this bit is enabled, the WDT time-out interrupt signal is generated and inform to CPU. 0 = WDT time-out interrupt Disabled. 1 = WDT time-out interrupt Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[5]	**WKF**	**WDT Time-out Wake-up Flag** This bit indicates the interrupt wake-up flag status of WDT 0 = WDT does not cause chip wake-up. 1 = Chip wake-up from Idle or Power-down mode if WDT time-out interrupt signal generated. **Note:** This bit is cleared by writing 1 to it.
[4]	**WKEN**	**WDT Time-out Wake-up Function Control (Write Protect)** If this bit is set to 1, while WDT time-out interrupt flag IF (WDT_CTL[3]) is generated to 1 and interrupt enable bit INTEN (WDT_CTL[6]) is enabled, the WDT time-out interrupt signal will generate a wake-up trigger event to chip. 0 = Wake-up trigger event Disabled if WDT time-out interrupt signal generated. 1 = Wake-up trigger event Enabled if WDT time-out interrupt signal generated. **Note 1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note 2:** Chip can be woken up by WDT time-out interrupt signal generated only if WDT clock source is selected to 32 kHz internal low speed RC oscillator (LIRC) or LXT.
[3]	**IF**	**WDT Time-out Interrupt Flag** This bit will set to 1 while WDT up counter value reaches the selected WDT time-out interval 0 = WDT time-out interrupt did not occur. 1 = WDT time-out interrupt occurred. **Note:** This bit is cleared by writing 1 to it.
[2]	**RSTF**	**WDT Time-out Reset Flag** This bit indicates the system has been reset by WDT time-out reset or not. 0 = WDT time-out reset did not occur. 1 = WDT time-out reset occurred. **Note:** This bit is cleared by writing 1 to it.
[1]	**RSTEN**	**WDT Time-out Reset Enable Bit (Write Protect)** Setting this bit will enable the WDT time-out reset function if the WDT up counter value has not been cleared after the specific WDT reset delay period expires. 0 = WDT time-out reset function Disabled. 1 = WDT time-out reset function Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[0]	**Reserved**	Reserved.

WDT Alternative Control Register (WDT_ALTCTL)

Register	Offset	R/W	Description	Reset Value
WDT_ALTCTL	WDT_BA+0x04	R/W	WDT Alternative Control Register	0x0000_0000

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

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1:0]	**RSTDSEL**	**WDT Reset Delay Selection (Write Protect)** When WDT time-out happened, user has a time named WDT Reset Delay Period to clear WDT counter by programming 0x5AA5 to RSTCNT to prevent WDT time-out reset happened. User can select a suitable setting of RSTDSEL for different WDT Reset Delay Period. 00 = WDT Reset Delay Period is 1026 * WDT_CLK. 01 = WDT Reset Delay Period is 130 * WDT_CLK. 10 = WDT Reset Delay Period is 18 * WDT_CLK. 11 = WDT Reset Delay Period is 3 * WDT_CLK. **Note 1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note 2:** This register will be reset to 0 if WDT time-out reset happened.

WDT Reset Counter Register (WDT_RSTCNT)

Register	Offset	R/W	Description	Reset Value
WDT_RSTCNT	WDT_BA+0x08	W	WDT Reset Counter Register	0x0000_0000

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

Bits	Description	
[31:0]	**RSTCNT**	**WDT Reset Counter Register** Writing 0x00005AA5 to this field will reset the internal 20-bit WDT up counter value to 0. **Note:** Performing RSTCNT to reset counter needs 2 * WDT_CLK period to become active**.**

6.11 Extra Watchdog Timer (EWDT)

6.11.1 Overview

The Extra Watchdog Timer (EWDT) is used to perform a system reset when system runs into an unknown state. This prevents system from hanging for an infinite period of time. Besides, this Watchdog Timer supports the function to wake up system from Idle/Power-down mode.

6.11.2 Features

- 20-bit free running up counter for EWDT time-out interval
- Selectable time-out interval (2^4 ~ 2^{20}) and the time-out interval is 0.5 ms ~ 32.768 s if EWDT_CLK = 32 kHz.
- System kept in reset state for a period of (1 / EWDT_CLK) * 63
- Supports selectable EWDT reset delay period, including 1026, 130, 18 or 3 EWDT_CLK reset delay period
- Supports EWDT time-out wake-up function only if EWDT clock source is selected as LIRC 32kHz or LXT.

6.11.3 Block Diagram

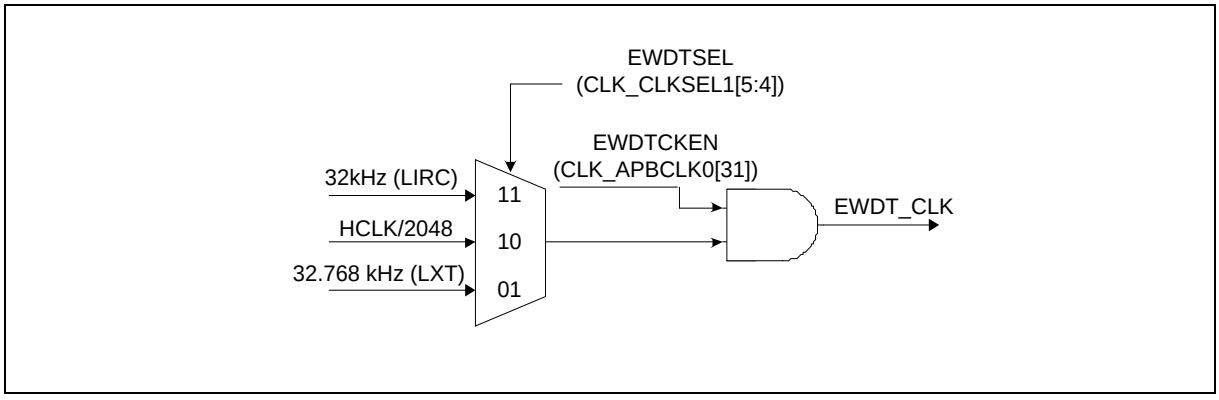
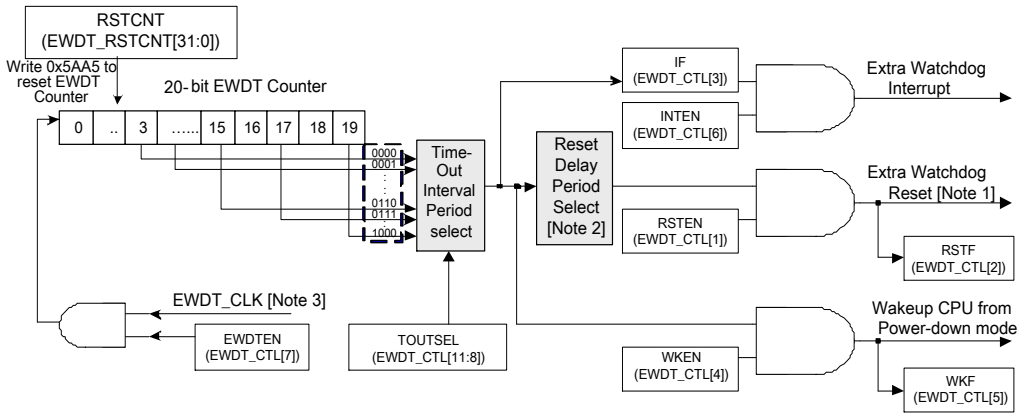


Figure 6.11-1 Extra Watchdog Timer Clock Control



Note1: EWDT resets CPU and lasts 63 EWDT_CLK.

Note2: The EWDT reset delay period can be selected as 3/18/130/1026 EWDT_CLK.

Note3: Chip can be woken up by EWDT time-out interrupt signal generated only if EWDT clock source is selected to LXT or LIRC.

Figure 6.11-2 Extra Watchdog Timer Block Diagram

6.11.4 Basic Configuration

- Clock Source Configuration
  - Select the source of EWDT peripheral clock on EWDTSEL (CLK_CLKSEL1[5:4])
  - Enable EWDT peripheral clock in EWDTCKEN (CLK_APBCLK0[31]).

6.11.5 Functional Description

The EWDT includes an 20-bit free running up counter with programmable time-out intervals. Table 6.10-1 shows the EWDT time-out interval period selection and Figure 6.10-1 shows the EWDT time-out interval and reset period timing.

6.11.5.1 EWDT Time-out Interrupt

Setting EWDTEN (EWDT_CTL[7]) to 1 will enable the EWDT function and the EWDT counter to start counting up. The SYNC (EWDT_CTL[30]) can indicate whether enable/disable EWDTEN function is completed or not. There are nine time-out interval periods that can be selected by setting TOUTSEL (EWDT_CTL[11:8]). When the EWDT up counter reaches the TOUTSEL (EWDT_CTL[11:8]) settings, EWDT time-out interrupt will occur then EWDT time-out interrupt flag IF (EWDT_CTL[3]) will be set to 1 immediately. If INTEN (EWDT_CTL[6]) is enabled, EWDT time-out interrupt will inform CPU.

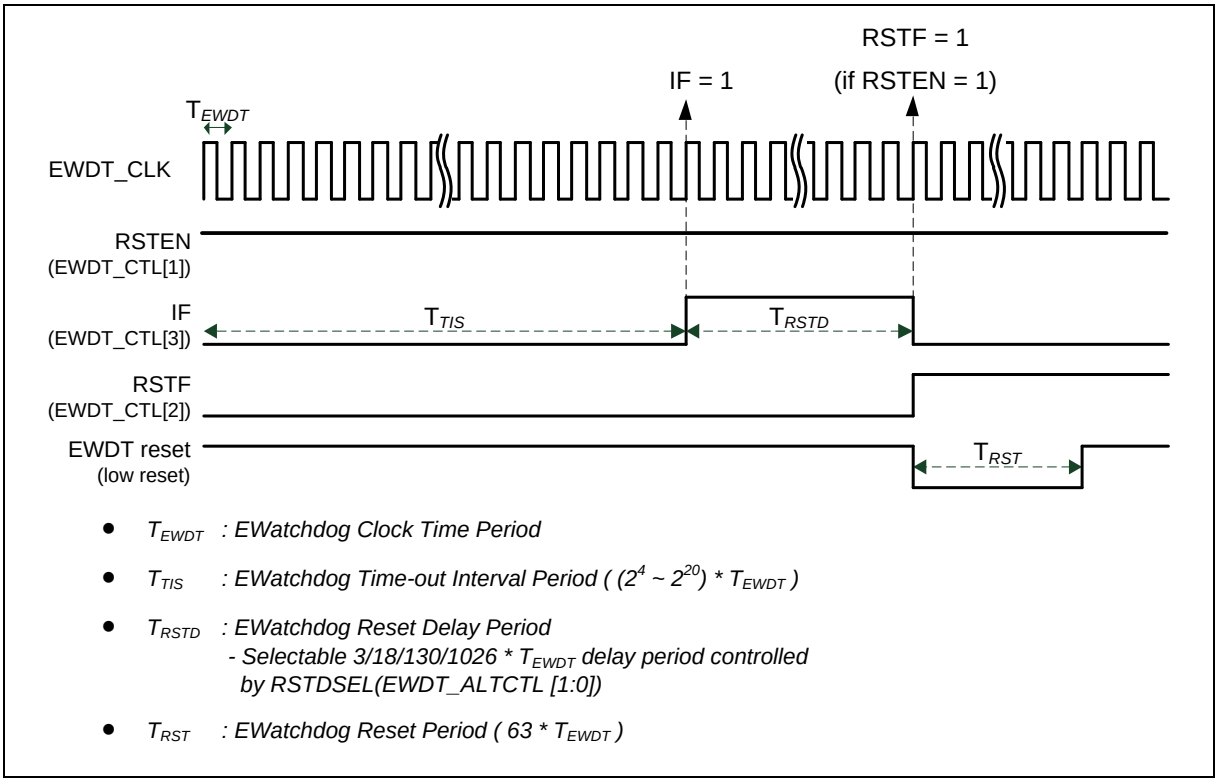
6.11.5.2 EWDT Reset Delay Period and Reset System

There is a specified TRSTD reset delay period follows the IF (EWDT_CTL[3]) is setting to 1. User should program 0x00005AA5 to RSTCNT (EWDT_RSTCNT[31:0]) to reset the 20-bit EWDT up counter value to avoid generating EWDT time-out reset signal before the TRSTD reset delay period expires. Moreover, user should set RSTDSEL (EWDT_ALTCTL[1:0]) to select reset delay period to clear EWDT counter. If the EWDT up counter value has not been cleared after the specific TRSTD delay period expires, the EWDT control will set RSTF (EWDT_CTL[2]) to 1 if RSTEN (EWDT_CTL[1]) bit is enabled, then chip enters to reset state immediately. TRST reset period will keep last 63 EWDT clocks then chip restarts executing program from reset vector (0x0000_0000). The RSTF

(EWDT_CTL[2]) will keep 1 after EWDT time-out resets the chip, and user can check RSTF (EWDT_CTL[2]) by software to recognize the system has been reset by EWDT time-out reset or not.

TOUTSEL	Time-Out Interval Period T_{TIS}	Reset Delay Period T_{RSTD}
0000	$2^4 * T_{EWDT}$	(3/18/130/1026) * T_{EWDT}
0001	$2^6 * T_{EWDT}$	(3/18/130/1026) * T_{EWDT}
0010	$2^8 * T_{EWDT}$	(3/18/130/1026) * T_{EWDT}
0011	$2^{10} * T_{EWDT}$	(3/18/130/1026) * T_{EWDT}
0100	$2^{12} * T_{EWDT}$	(3/18/130/1026) * T_{EWDT}
0101	$2^{14} * T_{EWDT}$	(3/18/130/1026) * T_{EWDT}
0110	$2^{16} * T_{EWDT}$	(3/18/130/1026) * T_{EWDT}
0111	$2^{18} * T_{EWDT}$	(3/18/130/1026) * T_{EWDT}
1000	$2^{20} * T_{EWDT}$	(3/18/130/1026) * T_{EWDT}

Table 6.11-1 Extra Watchdog Timer Time-out Interval Period Selection



- T_{EWDT} : *EWatchdog Clock Time Period*
- T_{TIS} : *EWatchdog Time-out Interval Period ((2^4 ~ 2^{20}) * T_{EWDT})*
- T_{RSTD} : *EWatchdog Reset Delay Period*
  *- Selectable 3/18/130/1026 * T_{EWDT} delay period controlled by RSTDSEL(EWDT_ALTCTL [1:0])*
- T_{RST} : *EWatchdog Reset Period (63 * T_{EWDT})*

Figure 6.11-1 Extra Watchdog Timer Time-out Interval and Reset Period Timing

6.11.5.3 EWDT Wake-up

If EWDT clock source is selected to 32 kHz or LXT, system can be woken up from Power-down mode while EWDT time-out interrupt signal is generated and WKEN (EWDT_CTL[4]) enabled. Note that user should set LXTEN (CLK_PWRCTL[1]) or LIRCEN (CLK_PWRCTL[3]) to select clock source before

system enters Power-down mode because the system peripheral clock are disabled when system is in Power-down mode. In the meanwhile, the WKF (EWDT_CTL[5]) will be set to 1 automatically, and user can check WKF (EWDT_CTL[5]) status by software to recognize the system has been woken up by EWDT time-out interrupt or not.

6.11.5.4 EWDT ICE Debug

When ICE is connected to MCU, EWDT counter is counting or not by ICEDEBUG (EWDT_CTL[31]). The default value of ICEDEBUG is 0, EWDT counter will stop counting when CPU is held by ICE. If ICEDEBUG is set to 1, EWDT counter will keep counting no matter CPU is held by ICE or not.

6.11.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
EWDT Base Address: **EWDT_BA = 0x4004_2000** **EWDT non-secure base address is EWDT _BA + 0x1000_0000**				
EWDT_CTL	EWDT_BA+0x00	R/W	Extra WDT Control Register	0x0000_0800
EWDT_ALTCTL	EWDT_BA+0x04	R/W	Extra WDT Alternative Control Register	0x0000_0000
EWDT_RSTCNT	EWDT_BA+0x08	W	Extra WDT Reset Counter Register	0x0000_0000

6.11.7 Register Description

Extra WDT Control Register (EWDT_CTL)

Register	Offset	R/W	Description	Reset Value
EWDT_CTL	EWDT_BA+0x00	R/W	Extra WDT Control Register	0x0000_0800

31	30	29	28	27	26	25	24
ICEDEBUG	**SYNC**	**Reserved**					
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**TOUTSEL**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WDTEN	**INTEN**	**WKF**	**WKEN**	**IF**	**RSTF**	**RSTEN**	**Reserved**

Bits	Description	
[31]	**ICEDEBUG**	**ICE Debug Mode Acknowledge Disable Bit (Write Protect)** 0 = ICE debug mode acknowledgement affects EWDT counting. EWDT up counter will be held while CPU is held by ICE. 1 = ICE debug mode acknowledgement Disabled. EWDT up counter will keep going no matter CPU is held by ICE or not. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[30]	**SYNC**	**EWDT Enable Control SYNC Flag Indicator (Read Only)** If user executes enable/disable WDTEN (EWDT_CTL[7]), this flag can indicate enable/disable WDTEN function is completed or not. 0 = Set WDTEN bit is completed. 1 = Set WDTEN bit is synchronizing and not become active yet. **Note:** Performing enable or disable WDTEN bit needs 2 * EWDT_CLK period to become active.
[29:12]	**Reserved**	Reserved.
[11:8]	**TOUTSEL**	**WDT Time-out Interval Selection (Write Protect)** These three bits select the time-out interval period for the WDT. 0000 = 2^4 * EWDT_CLK. 0001 = 2^6 * EWDT_CLK. 0010 = 2^8 * EWDT_CLK. 0011 = 2^{10} * EWDT_CLK. 0100= 2^{12} * EWDT_CLK. 0101 = 2^{14} * EWDT_CLK. 0110 = 2^{16} * EWDT_CLK. 0111 = 2^{18} * EWDT_CLK. 1000 = 2^{20} * EWDT_CLK. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

[7]	**WDTEN**	**WDT Enable Bit (Write Protect)** 0 = EWDT Disabled (this action will reset the internal up counter value). 1 = EWDT Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[6]	**INTEN**	**EWDT Time-out Interrupt Enable Bit (Write Protect)** If this bit is enabled, the EWDT time-out interrupt signal is generated and inform to CPU. 0 = EWDT time-out interrupt Disabled. 1 = EWDT time-out interrupt Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[5]	**WKF**	**WDT Time-out Wake-up Flag** This bit indicates the interrupt wake-up flag status of EWDT 0 = EWDT does not cause chip wake-up. 1 = Chip wake-up from Idle or Power-down mode if EWDT time-out interrupt signal generated. **Note:** This bit is cleared by writing 1 to it.
[4]	**WKEN**	**WDT Time-out Wake-up Function Control (Write Protect)** If this bit is set to 1, while EWDT time-out interrupt flag IF (EWDT_CTL[3]) is generated to 1 and interrupt enable bit INTEN (EWDT_CTL[6]) is enabled, the EWDT time-out interrupt signal will generate a wake-up trigger event to chip. 0 = Wake-up trigger event Disabled if EWDT time-out interrupt signal generated. 1 = Wake-up trigger event Enabled if EWDT time-out interrupt signal generated. **Note 1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note 2:** Chip can be woken up by EWDT time-out interrupt signal generated only if EWDT clock source is selected to 32 kHz internal low speed RC oscillator (LIRC) or LXT.
[3]	**IF**	**EWDT Time-out Interrupt Flag** This bit will set to 1 while EWDT up counter value reaches the selected EWDT time-out interval 0 = EWDT time-out interrupt did not occur. 1 = EWDT time-out interrupt occurred. **Note:** This bit is cleared by writing 1 to it.
[2]	**RSTF**	**EWDT Time-out Reset Flag** This bit indicates the system has been reset by EWDT time-out reset or not. 0 = EWDT time-out reset did not occur. 1 = EWDT time-out reset occurred. **Note:** This bit is cleared by writing 1 to it.
[1]	**RSTEN**	**EWDT Time-out Reset Enable Bit (Write Protect)** Setting this bit will enable the EWDT time-out reset function if the EWDT up counter value has not been cleared after the specific EWDT reset delay period expires. 0 = EWDT time-out reset function Disabled. 1 = EWDT time-out reset function Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[0]	**Reserved**	Reserved.

Extra WDT Alternative Control Register (EWDT_ALTCTL)

Register	Offset	R/W	Description	Reset Value
EWDT_ALTCTL	EWDT_BA+0x04	R/W	Extra WDT Alternative Control Register	0x0000_0000

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

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1:0]	**RSTDSEL**	**WDT Reset Delay Selection (Write Protect)** When EWDT time-out happened, user has a time named EWDT Reset Delay Period to clear EWDT counter by programming 0x5AA5 to RSTCNT to prevent EWDT time-out reset happened. User can select a suitable setting of RSTDSEL for different EWDT Reset Delay Period. 00 = EWDT Reset Delay Period is 1026 * EWDT_CLK. 01 = EWDT Reset Delay Period is 130 * EWDT_CLK. 10 = EWDT Reset Delay Period is 18 * EWDT_CLK. 11 = EWDT Reset Delay Period is 3 * EWDT_CLK. **Note 1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note 2:** This register will be reset to 0 if EWDT time-out reset happened.

Extra WDT Reset Counter Register (EWDT_RSTCNT)

Register	Offset	R/W	Description	Reset Value
EWDT_RSTCNT	EWDT_BA+0x08	W	Extra WDT Reset Counter Register	0x0000_0000

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

Bits	Description	
[31:0]	**RSTCNT**	**WDT Reset Counter Register** Writing 0x00005AA5 to this field will reset the internal 20-bit EWDT up counter value to 0. **Note:** Performing RSTCNT to reset counter needs 2 * EWDT_CLK period to become active**.**

6.12 Window Watchdog Timer (WWDT)

6.12.1 Overview

The Window Watchdog Timer (WWDT) is used to perform a system reset within a specified window period to prevent software running to uncontrollable status by any unpredictable condition.

6.12.2 Features

- 6-bit down counter value (CNTDAT, WWDT_CNT[5:0]) and 6-bit compare value (CMPDAT, WWDT_CTL[21:16]) to make the WWDT time-out window period flexible
- Supports 4-bit value (PSCSEL, WWDT_CTL[11:8]) to programmable maximum 11-bit prescale counter period of WWDT counter
- WWDT counter suspends in Idle/Power-down mode

6.12.3 Block Diagram

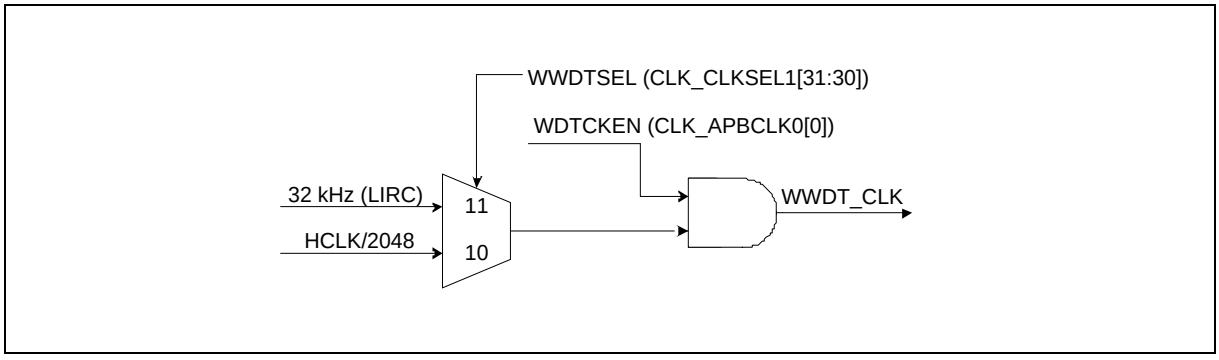


Figure 6.12-1 WWDT Clock Control

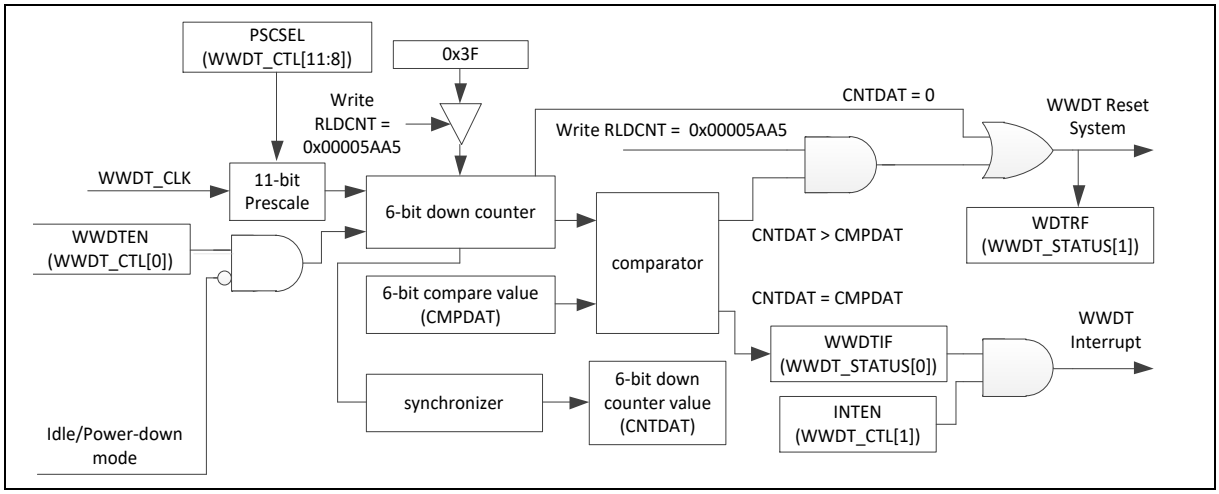


Figure 6.12-2 WWDT Block Diagram

6.12.4 Basic Configuration

- Clock Source Configuration
  - Select the source of WWDT peripheral clock on WWDTSEL (CLK_CLKSEL1[31:30])

- Enable WWDT peripheral clock in WDTCKEN (CLK_APBCLK0[0]).

6.12.5 Functional Description

The WWDT includes a 6-bit down counter with programmable prescale value to define different WWDT time-out intervals. The clock source of 6-bit WWDT is based on system clock divide 2048 (HCLK/2048) or 32 kHz internal low speed RC oscillator (LIRC) with a programmable 11-bit prescale counter value which controlled by PSCSEL (WWDT_CTL[11:8]). Also, the correlate of PSCSEL (WWDT_CTL[11:8]) and prescale value are listed in Table 6.12-1.

PSCSEL	Prescale Value	Max. Time-Out Period	Max. Time-Out Interval (WWDT_CLK = LIRC 32 kHz)
0000	1	1 * 64 * T_{WWDT}	2 ms
0001	2	2 * 64 * T_{WWDT}	4 ms
0010	4	4 * 64 * T_{WWDT}	8 ms
0011	8	8 * 64 * T_{WWDT}	16 ms
0100	16	16 * 64 * T_{WWDT}	32 ms
0101	32	32 * 64 * T_{WWDT}	64 ms
0110	64	64 * 64 * T_{WWDT}	128 ms
0111	128	128 * 64 * T_{WWDT}	256 ms
1000	192	192 * 64 * T_{WWDT}	384 ms
1001	256	256 * 64 * T_{WWDT}	512 ms
1010	384	384 * 64 * T_{WWDT}	768 ms
1011	512	512 * 64 * T_{WWDT}	1024 ms
1100	768	768 * 64 * T_{WWDT}	1536 ms
1101	1024	1024 * 64 * T_{WWDT}	2048 ms
1110	1536	1536 * 64 * T_{WWDT}	3072 ms
1111	2048	2048 * 64 * T_{WWDT}	4096 ms

Table 6.12-1 WWDT Prescaler Value Selection

6.12.5.1 WWDT Counting

When the WWDTEN (WWDT_CTL[0]) is set, WWDT down counter will start counting from 0x3F to 0. To prevent program runs to disable WWDT counter counting unexpected, the WWDT_CTL register can only be written once after chip is powered on or reset. User cannot disable WWDT counter counting (WWDTEN), change counter prescale period (PSCSEL) or change window compare value (CMPDAT) while WWDTEN (WWDT_CTL[0]) has been enabled by user unless chip is reset.

To avoid the system is reset while CPU clock is disabled, the WWDT counter will stop counting when CPU enters Idle/Power-down mode. After CPU enters normal mode, the WWDT counter will resume down counting.

6.12.5.2 WWDT Compare Match Interrupt

During down counting by the WWDT counter, the WWDTIF (WWDT_STATUS[0]) is set to 1 while the WWDT counter value (CNTDAT) is equal to window compare value (CMPDAT) and WWDTIF can be

cleared by user; if INTEN (WWDT_CTL[1]) is also set to 1 by user, the WWDT compare match interrupt signal is generated also while WWDTIF is set to 1 by hardware.

6.12.5.3 WWDT Reset System

Figure 6.12-1 shows three cases of WWDT reset and reload behavior.

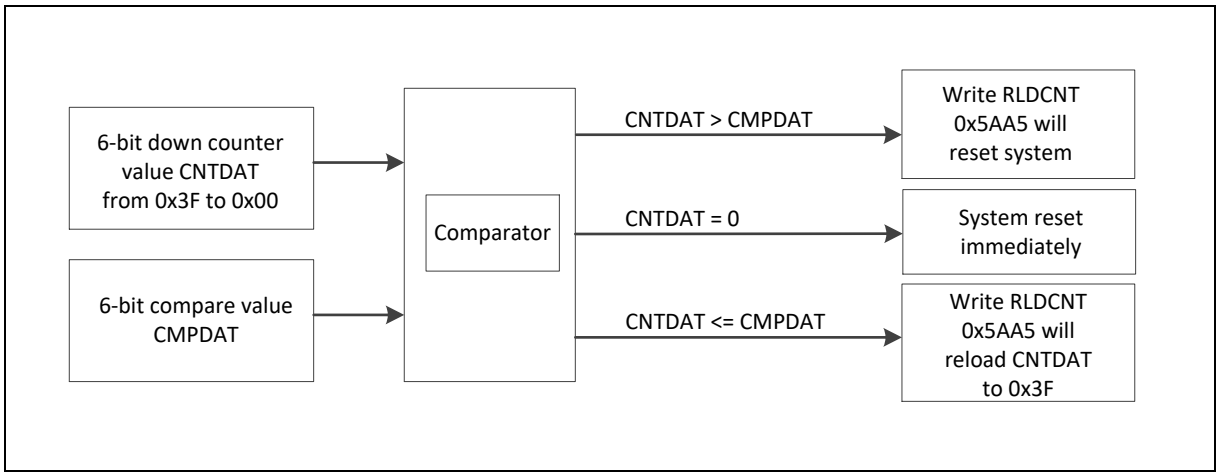


Figure 6.12-1 WWDT Reset and Reload Behavior

If the current CNTDAT (WWDT_CNT[5:0]) is larger than CMPDAT (WWDT_CTL[21:16]) and user writes 0x00005AA5 to the WWDT_RLDCNT register, the WWDT reset system signal will be generated to cause chip reset also. The waveform of WWDT reload counter when CNTDAT > CMPDAT is shown in Figure 6.12-2.

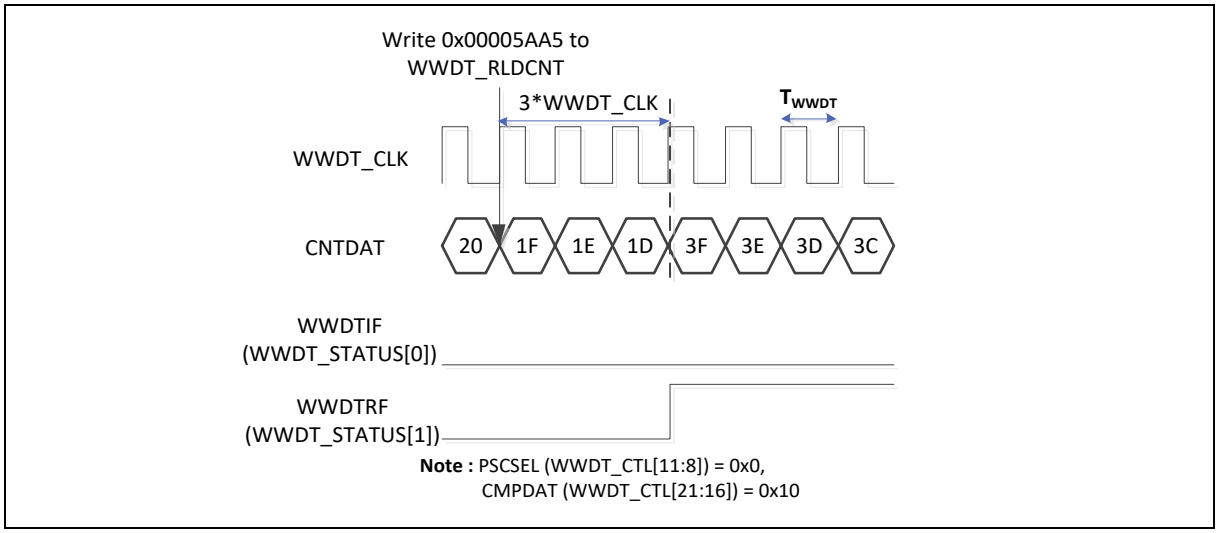


Figure 6.12-2 WWDT Reload Counter When CNTDAT > CMPDAT

When WWDTIF (WWDT_STATUS[0]) is generated, user must reload WWDT counter value to 0x3F by writing 0x00005AA5 to WWDT_RLDCNT register, and also to prevent WWDT counter value reached to 0 and generate WWDT reset system signal to inform system reset. Figure 6.12-3 shows the waveform of WWDT reload counter when CNTDAT < CMPDAT and Figure 6.12-4 shows WWDT generates reset system signal (WWDTRF) if user doesn't write 0x00005AA5 to WWDT_RLDCNT before WWDT counter value reaches 0.

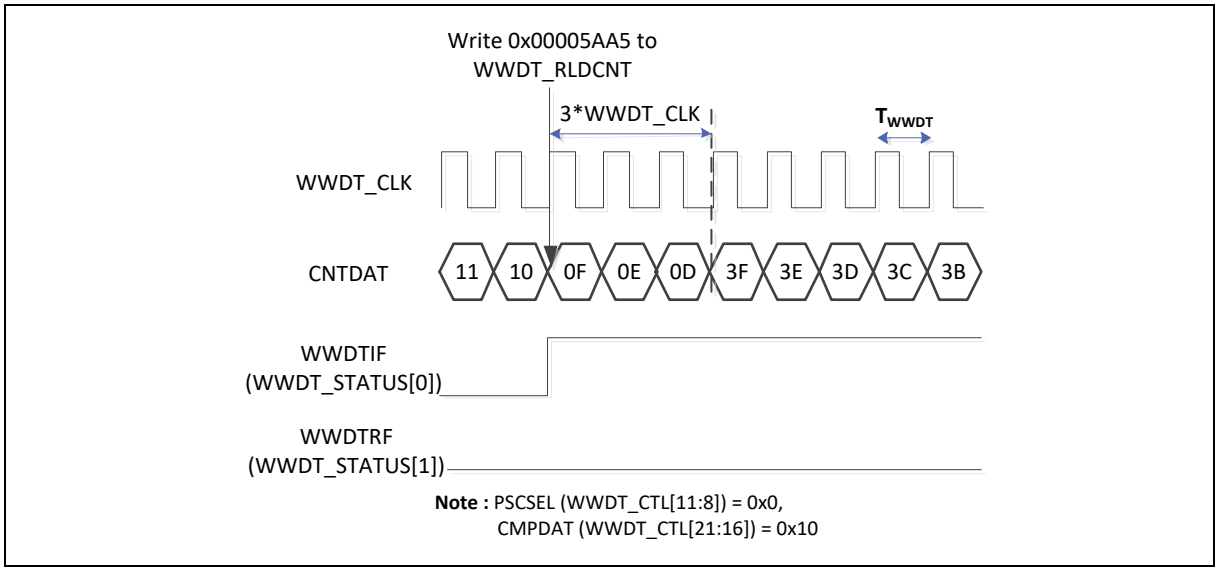


Figure 6.12-3 WWDT Reload Counter When CNTDAT < CMPDAT

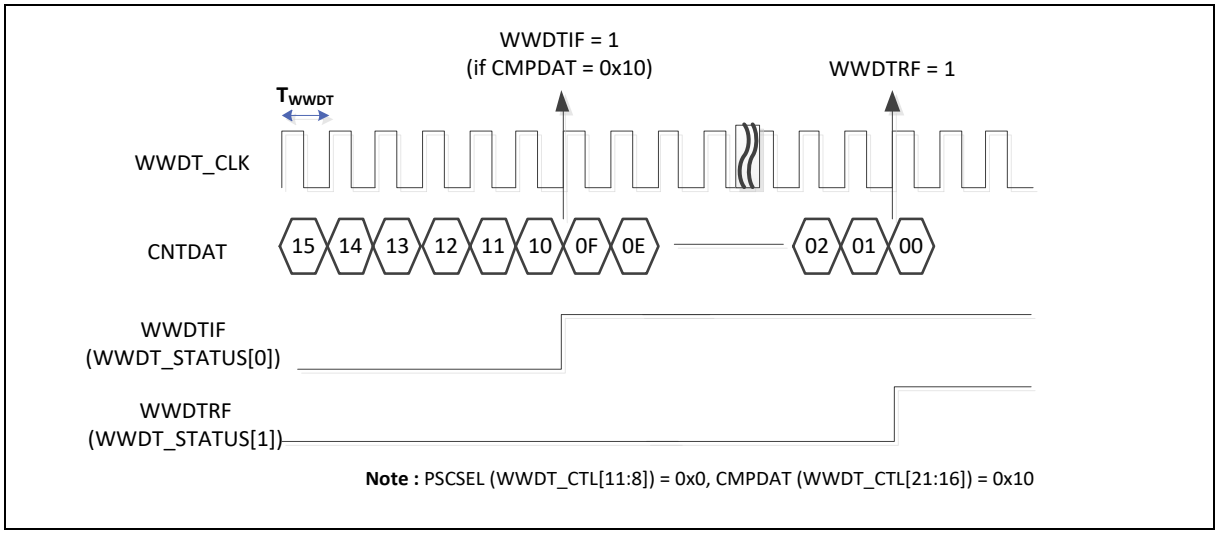


Figure 6.12-4 WWDT Interrupt and Reset Signals

6.12.5.4 WWDT Window Setting Limitation

When user writes 0x00005AA5 to WWDT_RLDCNT register to reload WWDT counter value to 0x3F, it needs 3 WWDT clocks to sync the reload command to actually perform reload action. Note that if user sets PSCSEL (WWDT_CTL[11:8]) to 0000, the counter prescale value should be as 1, and the CMPDAT (WWDT_CTL[21:16]) must be larger than 2. Otherwise, writing WWDT_RLDCNT register to reload WWDT counter value to 0x3F is unavailable, WWDTIF (WWDT_STATUS[0]) is generated, and WWDT reset system event always happened. The WWDT CMPDAT setting limitation is shown in Table 6.12-2.

If user sets CMPDATA as 0x3F and 0x0, the interrupt doesn't occur. The reset occurs when WWDT counts to 0x0, so the interrupt doesn't occur when CMPDATA is 0x0.

PSCSEL	Prescale Value	Valid CMPDAT Value
0000	1	0x3 ~ 0x3E

0001	2	0x2 ~ 0x3E
Others	Others	0x1 ~ 0x3E

Table 6.12-2 CMPDAT Setting Limitation

6.12.5.5 WWDT ICE Debug

When ICE is connected to MCU, the WWDT counter is counting or not by ICEDEBUG (WWDT_CTL[31]). The default value of ICEDEBUG is 0. The WWDT counter will stop counting when CPU is held by ICE. If ICEDEBUG is set to 1, WWDT counter will keep counting no matter CPU is held by ICE or not.

6.12.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
WWDT Base Address: **WWDT_BA = 0x4004_0100**				
WWDT_RLDCNT	WWDT_BA+0x00	W	WWDT Reload Counter Register	0x0000_0000
WWDT_CTL	WWDT_BA+0x04	R/W	WWDT Control Register	0x003F_0800
WWDT_STATUS	WWDT_BA+0x08	R/W	WWDT Status Register	0x0000_0000
WWDT_CNT	WWDT_BA+0x0C	R	WWDT Counter Value Register	0x0000_003F

6.12.7 Register Description

WWDT Reload Counter Register (WWDT_RLDCNT)

Register	Offset	R/W	Description	Reset Value
WWDT_RLDCNT	WWDT_BA+0x00	W	WWDT Reload Counter Register	0x0000_0000

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

Bits	Description	
[31:0]	**RLDCNT**	**WWDT Reload Counter Register** Writing 0x00005AA5 to this register will reload the WWDT counter value to 0x3F. **Note:** User can only write WWDT_RLDCNT register to reload WWDT counter value when current WWDT counter value between 0 and CMPDAT (WWDT_CTL[21:16]). If user writes WWDT_RLDCNT when current WWDT counter value is larger than CMPDAT, WWDT reset signal will be generated.

WWDT Control Register (WWDT_CTL)

Register	Offset	R/W	Description	Reset Value
WWDT_CTL	WWDT_BA+0x04	R/W	WWDT Control Register	0x003F_0800

Note: This register can be written only one time after chip is powered on or reset.

31	30	29	28	27	26	25	24
ICEDEBUG	**Reserved**						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved		**CMPDAT**					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**PSCSEL**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**INTEN**	**WWDTEN**

Bits	Description	
[31]	**ICEDEBUG**	**ICE Debug Mode Acknowledge Disable Bit** 0 = ICE debug mode acknowledgement effects WWDT counting. WWDT down counter will be held while CPU is held by ICE. 1 = ICE debug mode acknowledgement Disabled. **Note:** WWDT down counter will keep going no matter CPU is held by ICE or not.
[30:22]	**Reserved**	Reserved.
[21:16]	**CMPDAT**	**WWDT Window Compare Register** Set this register to adjust the valid reload window. **Note:** User can only write WWDT_RLDCNT register to reload WWDT counter value when current WWDT counter value between 0 and CMPDAT. If user writes WWDT_RLDCNT register when current WWDT counter value larger than CMPDAT, WWDT reset signal will generate.
[15:12]	**Reserved**	Reserved.
[11:8]	**PSCSEL**	**WWDT Counter Prescale Period Selection** 0000 = Pre-scale is 1; Max time-out period is 1 * 64 * WWDT_CLK. 0001 = Pre-scale is 2; Max time-out period is 2 * 64 * WWDT_CLK. 0010 = Pre-scale is 4; Max time-out period is 4 * 64 * WWDT_CLK. 0011 = Pre-scale is 8; Max time-out period is 8 * 64 * WWDT_CLK. 0100 = Pre-scale is 16; Max time-out period is 16 * 64 * WWDT_CLK. 0101 = Pre-scale is 32; Max time-out period is 32 * 64 * WWDT_CLK. 0110 = Pre-scale is 64; Max time-out period is 64 * 64 * WWDT_CLK. 0111 = Pre-scale is 128; Max time-out period is 128 * 64 * WWDT_CLK. 1000 = Pre-scale is 192; Max time-out period is 192 * 64 * WWDT_CLK. 1001 = Pre-scale is 256; Max time-out period is 256 * 64 * WWDT_CLK. 1010 = Pre-scale is 384; Max time-out period is 384 * 64 * WWDT_CLK. 1011 = Pre-scale is 512; Max time-out period is 512 * 64 * WWDT_CLK.

		1100 = Pre-scale is 768; Max time-out period is 768 * 64 * WWDT_CLK. 1101 = Pre-scale is 1024; Max time-out period is 1024 * 64 * WWDT_CLK. 1110 = Pre-scale is 1536; Max time-out period is 1536 * 64 * WWDT_CLK. 1111 = Pre-scale is 2048; Max time-out period is 2048 * 64 * WWDT_CLK.
[7:2]	**Reserved**	Reserved.
[1]	**INTEN**	**WWDT Interrupt Enable Bit** If this bit is enabled, the WWDT counter compare match interrupt signal is generated and inform to CPU. 0 = WWDT counter compare match interrupt Disabled. 1 = WWDT counter compare match interrupt Enabled.
[0]	**WWDTEN**	**WWDT Enable Bit** 0 = WWDT counter is stopped. 1 = WWDT counter starts counting.

WWDT Status Register (WWDT_STATUS)

Register	Offset	R/W	Description	Reset Value
WWDT_STATUS	WWDT_BA+0x08	R/W	WWDT Status Register	0x0000_0000

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

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**WWDTRF**	**WWDT Timer-out Reset Flag** This bit indicates the system has been reset by WWDT time-out reset or not. 0 = WWDT time-out reset did not occur. 1 = WWDT time-out reset occurred. **Note:** This bit is cleared by writing 1 to it.
[0]	**WWDTIF**	**WWDT Compare Match Interrupt Flag** This bit indicates the interrupt flag status of WWDT while WWDT counter value matches CMPDAT (WWDT_CTL[21:16]). 0 = No effect. 1 = WWDT counter value matches CMPDAT. **Note:** This bit is cleared by writing 1 to it.

WWDT Counter Value Register (WWDT_CNT)

Register	Offset	R/W	Description	Reset Value
WWDT_CNT	WWDT_BA+0x0C	R	WWDT Counter Value Register	0x0000_003F

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

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5:0]	**CNTDAT**	**WWDT Counter Value** CNTDAT will be updated continuously to monitor 6-bit WWDT down counter value.

6.13 Extra Window Watchdog Timer (EWWDT)

6.13.1 Overview

The Extra Window Watchdog Timer (EWWDT) is used to perform a system reset within a specified window period to prevent software running to uncontrollable status by any unpredictable condition.

6.13.2 Features

- 6-bit down counter value (CNTDAT, EWWDT_CNT[5:0]) and 6-bit compare value (CMPDAT, EWWDT_CTL[21:16]) to make the EWWDT time-out window period flexible
- Supports 4-bit value (PSCSEL, EWWDT_CTL[11:8]) to programmable maximum 11-bit prescale counter period of EWWDT counter
- EWWDT counter suspends in Idle/Power-down mode

6.13.3 Block Diagram

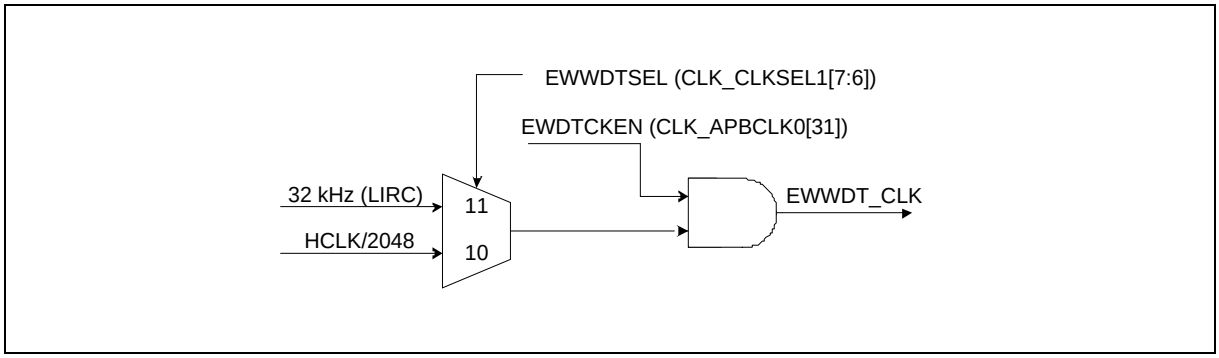


Figure 6.13-1 Extra WWDT Clock Control

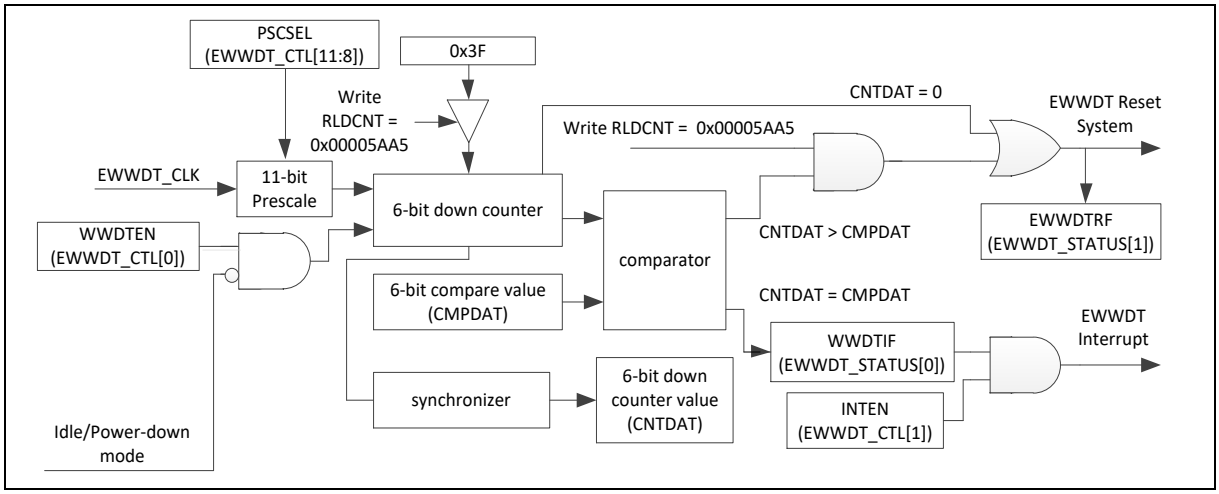


Figure 6.13-2 Extra WWDT Block Diagram

6.13.4 Basic Configuration

- Clock Source Configuration
  - Select the source of WWDT peripheral clock on EWWDTSEL (CLK_CLKSEL1[7:6])

- Enable EWWDT peripheral clock in EWDTCKEN (CLK_APBCLK0[31]).

6.13.5 Functional Description

The EWWDT includes a 6-bit down counter with programmable prescale value to define different EWWDT time-out intervals. The clock source of 6-bit EWWDT is based on system clock divide 2048 (HCLK/2048) or 32 kHz internal low speed RC oscillator (LIRC) with a programmable 11-bit prescale counter value which is controlled by PSCSEL (EWWDT_CTL[11:8]). Also, the correlate of PSCSEL (EWWDT_CTL[11:8]) and prescale value are listed in Table 6.13-1

PSCSEL	Prescaler Value	Max. Time-Out Period	Max. Time-Out Interval (EWWDT_CLK = LIRC 32 kHz)
0000	1	1 * 64 * T_{EWWDT}	2 ms
0001	2	2 * 64 * T_{EWWDT}	4 ms
0010	4	4 * 64 * T_{EWWDT}	8 ms
0011	8	8 * 64 * T_{EWWDT}	16 ms
0100	16	16 * 64 * T_{EWWDT}	32 ms
0101	32	32 * 64 * T_{EWWDT}	64 ms
0110	64	64 * 64 * T_{EWWDT}	128 ms
0111	128	128 * 64 * T_{EWWDT}	256 ms
1000	192	192 * 64 * T_{EWWDT}	384 ms
1001	256	256 * 64 * T_{EWWDT}	512 ms
1010	384	384 * 64 * T_{EWWDT}	768 ms
1011	512	512 * 64 * T_{EWWDT}	1024 ms
1100	768	768 * 64 * T_{EWWDT}	1536 ms
1101	1024	1024 * 64 * T_{EWWDT}	2048 ms
1110	1536	1536 * 64 * T_{EWWDT}	3072 ms
1111	2048	2048 * 64 * T_{EWWDT}	4096 ms

Table 6.13-1 Extra WWDT Prescaler Value Selection

6.13.5.1 EWWDT Counting

When the WWDTEN (EWWDT_CTL[0]) is set, EWWDT down counter will start counting from 0x3F to 0. To prevent program runs to disable EWWDT counter counting unexpectedly, the EWWDT_CTL register can only be written once after chip is powered on or reset. User cannot disable EWWDT counter counting (WWDTEN), change counter prescale period (PSCSEL) or change window compare value (CMPDAT) while WWDTEN (EWWDT_CTL[0]) has been enabled by user unless chip is reset.

To avoid the system is reset while CPU clock is disabled, the EWWDT counter will stop counting when CPU enters Idle/Power-down mode. After CPU enters normal mode, the EWWDT counter will resume down counting.

6.13.5.2 EWWDT Compare Match Interrupt

During down counting by the EWWDT counter, the WWDTIF (EWWDT_STATUS[0]) is set to 1 while the EWWDT counter value (CNTDAT) is equal to window compare value (CMPDAT) and WWDTIF

can be cleared by user; if INTEN (EWWDT_CTL[1]) is also set to 1 by user, the EWWDT compare match interrupt signal is generated also while WWDTIF is set to 1 by hardware.

6.13.5.3 EWWDT Reset System

Figure 6.12-1 shows three cases of EWWDT reset and reload behavior.

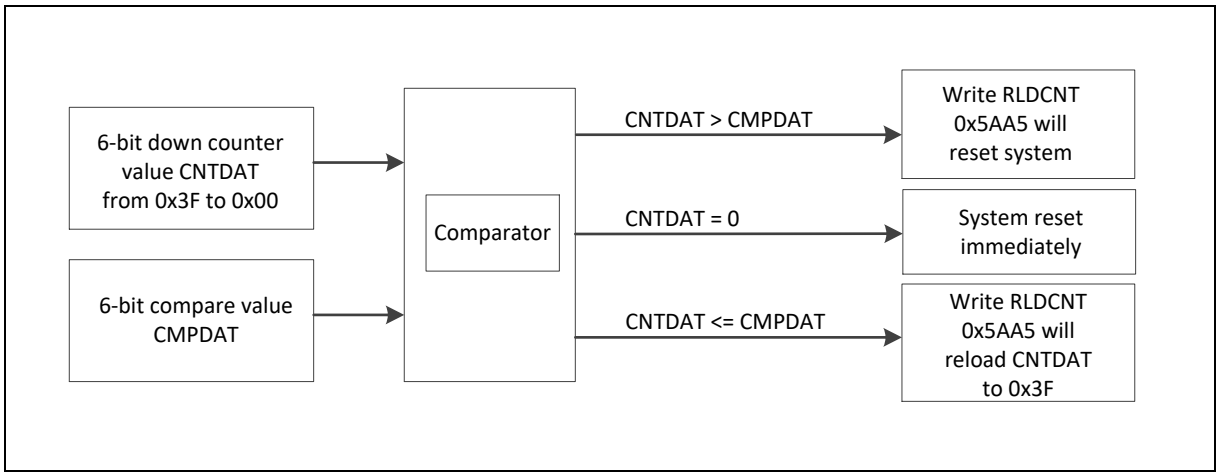


Figure 6.13-1 Extra WWDT Reset and Reload Behavior

If the current CNTDAT (EWWDT_CNT[5:0]) is larger than CMPDAT (EWWDT_CTL[21:16]) and user writes 0x00005AA5 to the EWWDT_RLDCNT register, the EWWDT reset system signal will be generated to cause chip reset also. The waveform of EWWDT reload counter when CNTDAT > CMPDAT is shown in **Figure 6.12-2**.

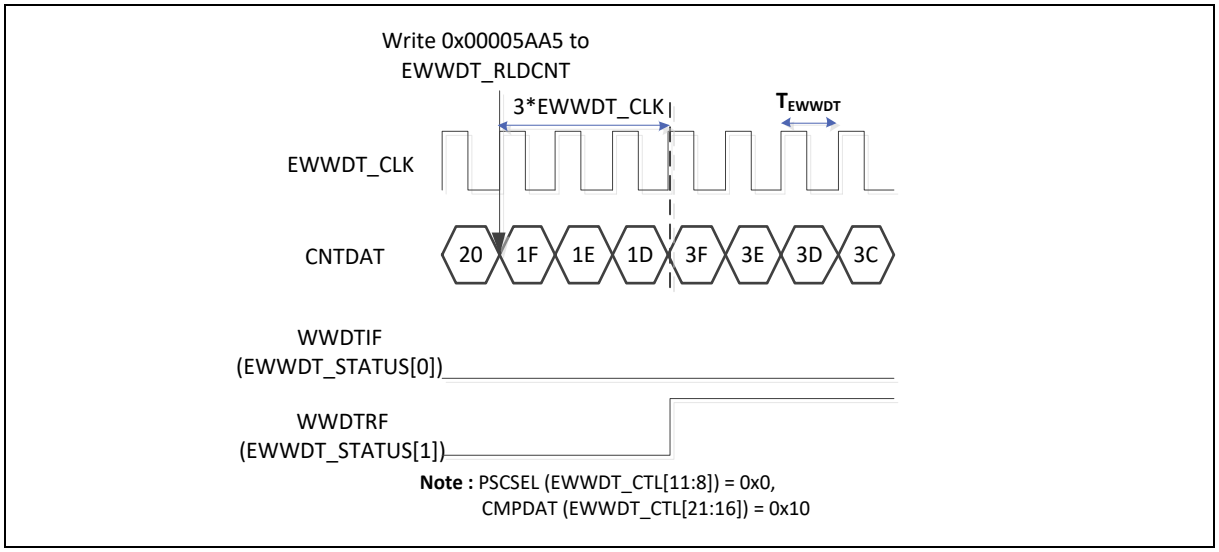


Figure 6.13-2 WWDT Reload Counter When CNTDAT > CMPDAT

When WWDTIF (EWWDT_STATUS[0]) is generated, user must reload EWWDT counter value to 0x3F by writing 0x00005AA5 to EWWDT_RLDCNT register, and also to prevent EWWDT counter value reached to 0 and generate EWWDT reset system signal to inform system reset. **Figure 6.12-3** shows the waveform of EWWDT reload counter when CNTDAT < CMPDAT and **Figure 6.12-4** shows EWWDT generates reset system signal (WWDTRF) if user doesn't write 0x00005AA5 to EWWDT_RLDCNT before EWWDT counter value reaches 0.

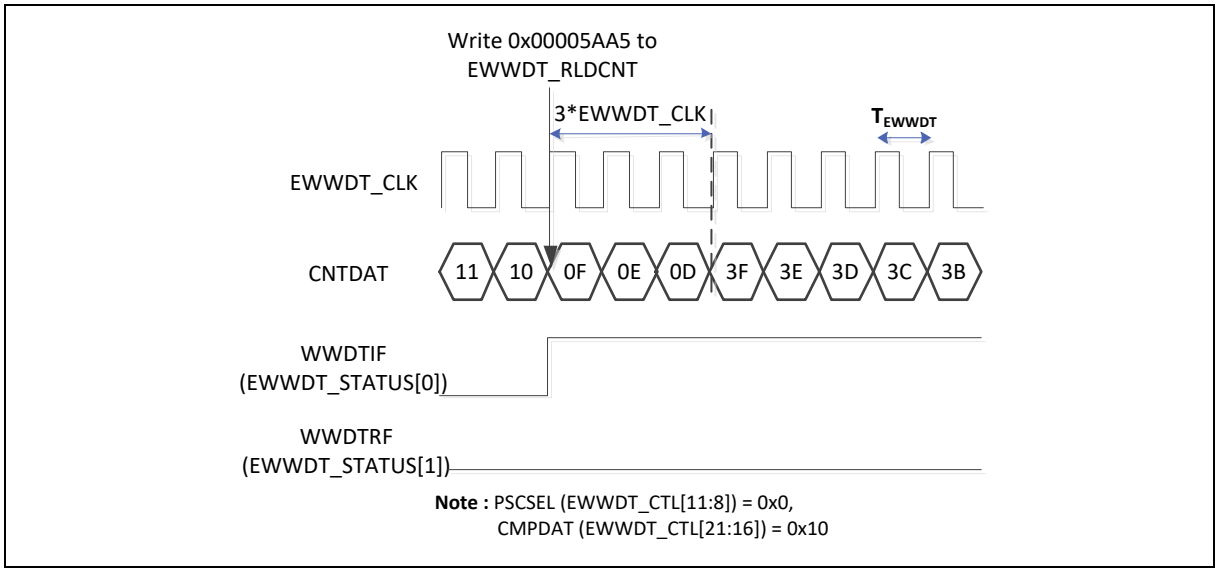


Figure 6.13-3 Extra WWDT Reload Counter When CNTDAT < CMPDAT

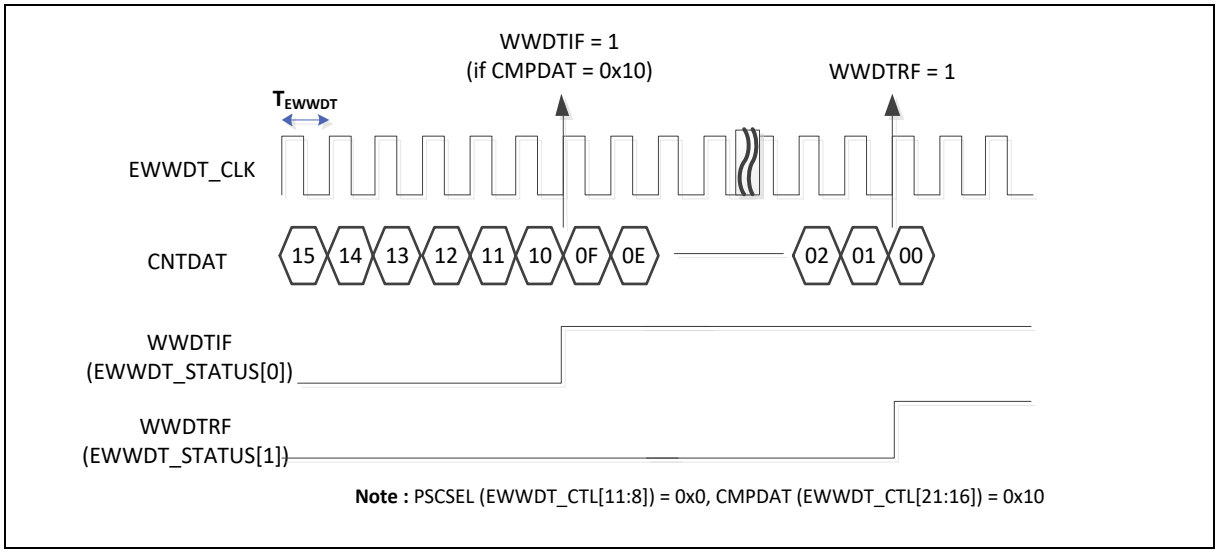


Figure 6.13-4 Extra WWDT Interrupt and Reset Signals

6.13.5.4 EWWDT Window Setting Limitation

When user writes 0x00005AA5 to EWWDT_RLDCNT register to reload EWWDT counter value to 0x3F, it needs 3 EWWDT clocks to sync the reload command to actually perform reload action. Note that if user sets PSCSEL (EWWDT_CTL[11:8]) to 0000, the counter prescale value should be as 1, and the CMPDAT (EWWDT_CTL[21:16]) must be larger than 2. Otherwise, writing EWWDT_RLDCNT register to reload EWWDT counter value to 0x3F is unavailable, WWDTIF (EWWDT_STATUS[0]) is generated, and EWWDT reset system event always happened. The EWWDT CMPDAT setting limitation is shown in **Table 6.12-2**.

If user sets CMPDATA as 0x3F and 0x0, the interrupt doesn't occur. The reset occurs when EWWDT counts to 0x0, so the interrupt doesn't occur when CMPDATA is 0x0.

PSCSEL	Prescale Value	Valid CMPDAT Value

0000	1	0x3 ~ 0x3E
0001	2	0x2 ~ 0x3E
Others	Others	0x1 ~ 0x3E

Table 6.13-2 CMPDAT Setting Limitation

6.13.5.5 EWWDT ICE Debug

When ICE is connected to MCU, the EWWDT counter is counting or not by ICEDEBUG (EWWDT_CTL[31]). The default value of ICEDEBUG is 0. The EWWDT counter will stop counting when CPU is held by ICE. If ICEDEBUG is set to 1, EWWDT counter will keep counting no matter CPU is held by ICE or not.

6.13.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
EWWDT Base Address: **EWWDT_BA = 0x4004_2100** **EWWDT non-secure base address is EWWDT _BA + 0x1000_0000**				
EWWDT_RLDCNT	EWWDT_BA+0x00	W	Extra WWDT Reload Counter Register	0x0000_0000
EWWDT_CTL	EWWDT_BA+0x04	R/W	Extra WWDT Control Register	0x003F_0800
EWWDT_STATUS	EWWDT_BA+0x08	R/W	Extra WWDT Status Register	0x0000_0000
EWWDT_CNT	EWWDT_BA+0x0C	R	Extra WWDT Counter Value Register	0x0000_003F

6.13.7 Register Description

Extra WWDT Reload Counter Register (EWWDT_RLDCNT)

Register	Offset	R/W	Description	Reset Value
EWWDT_RLDCNT	EWWDT_BA+0x00	W	Extra WWDT Reload Counter Register	0x0000_0000

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

Bits	Description	
[31:0]	**RLDCNT**	**EWWDT Reload Counter Register** Writing 0x00005AA5 to this register will reload the EWWDT counter value to 0x3F. **Note:** User can only write EWWDT_RLDCNT register to reload EWWDT counter value when current EWWDT counter value between 0 and CMPDAT (EWWDT_CTL[21:16]). If user writes EWWDT_RLDCNT when current EWWDT counter value is larger than CMPDAT, EWWDT reset signal will be generated.

Extra WWDT Control Register (EWWDT_CTL)

Register	Offset	R/W	Description	Reset Value
EWWDT_CTL	EWWDT_BA+0x04	R/W	Extra WWDT Control Register	0x003F_0800

Note: This register can be written only one time after chip is powered on or reset.

31	30	29	28	27	26	25	24
ICEDEBUG	**Reserved**						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved		**CMPDAT**					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**PSCSEL**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**INTEN**	**WWDTEN**

Bits	Description	
[31]	**ICEDEBUG**	**ICE Debug Mode Acknowledge Disable Bit** 0 = ICE debug mode acknowledgement effects EWWDT counting. WWDT down counter will be held while CPU is held by ICE. 1 = ICE debug mode acknowledgement Disabled. **Note:** EWWDT down counter will keep going no matter CPU is held by ICE or not.
[30:22]	**Reserved**	Reserved.
[21:16]	**CMPDAT**	**EWWDT Window Compare Register** Set this register to adjust the valid reload window. **Note:** User can only write EWWDT_RLDCNT register to reload EWWDT counter value when the current EWWDT counter value is between 0 and CMPDAT. If user writes EWWDT_RLDCNT register when the current EWWDT counter value is greater than CMPDAT, EWWDT reset signal will be generated.
[15:12]	**Reserved**	Reserved.
[11:8]	**PSCSEL**	**EWWDT Counter Prescale Period Selection** 0000 = Pre-scale is 1; Max time-out period is 1 * 64 * EWWDT_CLK. 0001 = Pre-scale is 2; Max time-out period is 2 * 64 * EWWDT_CLK. 0010 = Pre-scale is 4; Max time-out period is 4 * 64 * EWWDT_CLK. 0011 = Pre-scale is 8; Max time-out period is 8 * 64 * EWWDT_CLK. 0100 = Pre-scale is 16; Max time-out period is 16 * 64 * EWWDT_CLK. 0101 = Pre-scale is 32; Max time-out period is 32 * 64 * EWWDT_CLK. 0110 = Pre-scale is 64; Max time-out period is 64 * 64 * EWWDT_CLK. 0111 = Pre-scale is 128; Max time-out period is 128 * 64 * EWWDT_CLK. 1000 = Pre-scale is 192; Max time-out period is 192 * 64 * EWWDT_CLK. 1001 = Pre-scale is 256; Max time-out period is 256 * 64 * EWWDT_CLK. 1010 = Pre-scale is 384; Max time-out period is 384 * 64 * EWWDT_CLK. 1011 = Pre-scale is 512; Max time-out period is 512 * 64 * EWWDT_CLK.

		1100 = Pre-scale is 768; Max time-out period is 768 * 64 * EWWDT_CLK. 1101 = Pre-scale is 1024; Max time-out period is 1024 * 64 * EWWDT_CLK. 1110 = Pre-scale is 1536; Max time-out period is 1536 * 64 * EWWDT_CLK. 1111 = Pre-scale is 2048; Max time-out period is 2048 * 64 * EWWDT_CLK.
[7:2]	**Reserved**	Reserved.
[1]	**INTEN**	**EWWDT Interrupt Enable Bit** If this bit is enabled, the EWWDT counter compare match interrupt signal is generated and inform to CPU. 0 = EWWDT counter compare match interrupt Disabled. 1 = EWWDT counter compare match interrupt Enabled.
[0]	**WWDTEN**	**EWWDT Enable Bit** 0 = EWWDT counter is stopped. 1 = EWWDT counter starts counting.

Extra WWDT Status Register (EWWDT_STATUS)

Register	Offset	R/W	Description	Reset Value
EWWDT_STATUS	EWWDT_BA+0x08	R/W	Extra WWDT Status Register	0x0000_0000

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

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**WWDTRF**	**EWWDT Timer-out Reset Flag** This bit indicates the system has been reset by EWWDT time-out reset or not. 0 = EWWDT time-out reset did not occur. 1 = EWWDT time-out reset occurred. **Note:** This bit is cleared by writing 1 to it.
[0]	**WWDTIF**	**EWWDT Compare Match Interrupt Flag** This bit indicates the interrupt flag status of EWWDT while EWWDT counter value matches CMPDAT (EWWDT_CTL[21:16]). 0 = No effect. 1 = EWWDT counter value matches CMPDAT. **Note:** This bit is cleared by writing 1 to it.

Extra WWDT Counter Value Register (EWWDT_CNT)

Register	Offset	R/W	Description	Reset Value
EWWDT_CNT	EWWDT_BA+0x0C	R	Extra WWDT Counter Value Register	0x0000_003F

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

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5:0]	**CNTDAT**	**EWWDT Counter Value** CNTDAT will be updated continuously to monitor 6-bit EWWDT down counter value.

6.14 Real Time Clock (RTC)

6.14.1 Overview

The Real Time Clock (RTC) controller provides the real time and calendar message. The RTC offers programmable time tick and alarm match interrupts. The data format of time and calendar messages are expressed in BCD format. A digital frequency compensation feature is available to compensate external crystal oscillator frequency accuracy.

6.14.2 Features

- Supports external power pin V_{BAT}.
- Supports real time counter in RTC_TIME (hour, minute, second) and calendar counter in RTC_CAL (year, month, day) for RTC time and calendar check.
- Supports alarm time (hour, minute, second) and calendar (year, month, day) settings in RTC_TALM and RTC_CALM.
- Supports alarm time (hour, minute, second) and calendar (year, month, day) mask enable in RTC_TAMSK and RTC_CAMSK.
- Selectable 12-hour or 24-hour time scale in RTC_CLKFMT register.
- Supports Leap Year indication in RTC_LEAPYEAR register.
- Supports Day of the Week counter in RTC_WEEKDAY register.
- Frequency of RTC clock source compensate by RTC_FREQADJ register.
- All time and calendar message expressed in BCD format.
- Supports periodic RTC Time Tick interrupt with 8 period interval options 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 and 1 second.
- Supports RTC Time Tick and Alarm Match interrupt.
- Supports 1 Hz clock output.
- Supports chip wake-up from Idle or Power-down mode while a RTC interrupt signal is generated.
- Supports Daylight Saving Time software control in RTC_DSTCTL.
- Supports up 3 pairs dynamic loop tamper pin or 6 individual tamper pin.
- Built-in LXT frequency monitor.
- Supports 80 bytes spare registers and tamper pins detection to clear the content of these spare registers.
- Supports Flash mass erase operate will also clear the 80 bytes spare registers content.

6.14.3 Block Diagram

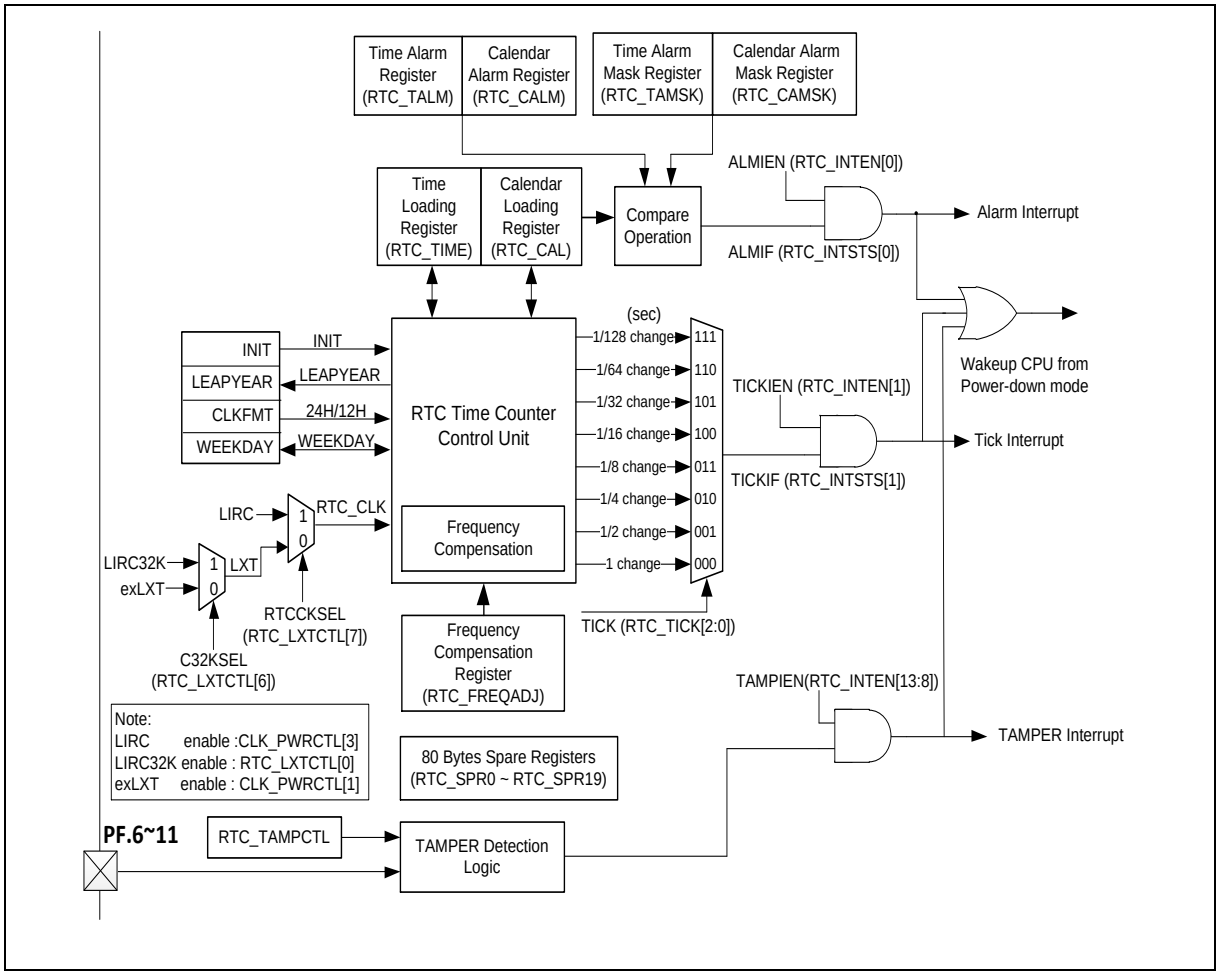


Figure 6.14-1 RTC Block Diagram

6.14.4 Basic Configuration

- Clock Source Configuration
  - The RTC controller clock source is enabled by RTCCKEN (APBCLK0[1]).
  - The RTC Time Counter source can be selected as
    - Internal LIRC if RTCCKSEL (RTC_LXTCTL[7]) = 1
    - External LXT if RTCCKSEL (RTC_LXTCTL[7]) = 0 and C32KSEL (RTC_LXTCTL[6]) = 0
    - Internal LIRC32K if RTCCKSEL (RTC_LXTCTL[7]) = 0 and C32KSEL (RTC_LXTCTL[6]) = 1
- Pin Configuration

Group	Pin Name	GPIO	MFP
X32	X32_OUT	PF.4	MFP10
	X32_IN	PF.5	MFP10
TAMPER0	TAMPER0	PF.6	MFP10

TAMPER1	TAMPER1	PF.7	MFP10
TAMPER2	TAMPER2	PF.8	MFP10
TAMPER3	TAMPER3	PF.9	MFP10
TAMPER4	TAMPER4	PF.10	MFP10
TAMPER5	TAMPER5	PF.11	MFP10

6.14.5 Functional Description

6.14.5.1 RTC Initiation

When a RTC block is powered on, RTC is at reset state. User has to write a number 0xa5eb1357 to RTC initial register INIT (RTC_INIT[31:0]) to make RTC leaving reset state. Once the INIT register is written as 0xa5eb1357, the RTC will be in normal active state permanently. User can read ACTIVE (RTC_INIT[0]) to check whether the RTC is at normal active state or reset state.

6.14.5.2 RTC Control Registers Access Attribute

The RTC control registers access attributes are shown in Table 6.14-1.

Register	INIR = 0	INIR = 1
RTC_INIT	R/W	R/W
RTC_FREQADJ	R/W	R/W
RTC_TIME	Not available	R/W
RTC_CAL	Not available	R/W
RTC_CLKFMT	Not available	R/W
RTC_WEEKDAY	Not available	R/W
RTC_TALM	Not available	R/W
RTC_CALM	Not available	R/W
RTC_LEAPYEAR	Not available	R
RTC_INTEN	R/W	R/W
RTC_INTSTS	R/W	R/W
RTC_TICK	Not available	R/W
RTC_TAMSK	Not available	R/W
RTC_CAMSK	Not available	R/W
RTC_SPRCTL	R/W	R/W
RTC_SPRx	R/W	R/W
RTC_LXTCTL	R/W	R/W
RTC_GPIOCTL0	R/W	R/W
RTC_GPIOCTL1	R/W	R/W
RTC_DSTCTL	Not available	R/W
RTC_TAMPCTL	R/W	R/W

RTC_TAMPSEED	R/W	R/W
RTC_CLKDCTL	R/W	R/W
RTC_CLBR	R/W	R/W

Table 6.14-1 RTC Read/Write Enable

6.14.5.3 Frequency Compensation

The frequency compensation circuit supports dynamic compensation to adjust compensation value without reset the compensation circuit. The enable bit for dynamic compensation is DCOMPEN (RTC_CLKFMT[16]).

If enable dynamic compensation, the minimal interval to continuous writing RTC_FREQADJ is 0.24 seconds, and FCRBUSY (RTC_FREQADJ[31]) flag is used to indicate that new register write operation is prohibited. And the compensate value will load into compensation circuit after 0.24~1.16 seconds when write to RTC_FREQADJ.

The RTC_FREQADJ register allows user to make digital compensation to a clock input. Please follow the example and formula below to write the actual frequency of 32k crystal to RTC_FREQADJ register. Following are the compensation examples for higher or lower than 32768 Hz.

Example 1:

Frequency counter measurement: 32773.65 Hz

Integer Part: 32773 => Setting INTEGER (RTC_FREQADJ[12:8]) = 0x15

Fraction Part: 0.65 X 64 = 41.6 (0x2A) => Setting FRACTION (RTC_FREQADJ[5:0]) = 0x2A

Example 2:

Frequency counter measurement: 32763.25 Hz

Integer part: 32763 => Setting INTEGER (RTC_FREQADJ[12:8]) = 0x0B

Fraction part: 0.25 X 64 = 16 (0x10) => Setting FRACTION (RTC_FREQADJ[5:0]) = 0x10

Note: The value of RTC_FREQADJ register will be the default value (0x0000_1000) while the compensation is not executed. User can utilize a frequency counter to measure RTC clock source via clock output function in manufacturing. In the meanwhile, user can use clock output function to check the result of RTC frequency compensation.

6.14.5.4 Time and Calendar Counter

RTC_TIME and RTC_CAL are used to load the real time and calendar. RTC_TALM and RTC_CALM are used for setup alarm time and calendar.

6.14.5.5 12/24 hour Time Scale Selection

The 12/24 hour time scale selection depends on 24HEN (RTC_CLKFMT[0]). When RTC runs as 12-hour time scale mode, RTC_TIME[21] (the high bit of TENHR[1:0]) means AM/PM indication, if RTC_TIME[21] is 1, it indicates PM time message and RTC_TIME[21] is 0 indicates AM time message. Table 6.14-2 shows RTC_TIME mapping table of 12/24 hour time scale selection.

Note: The Hour Value Write Into RTC_TIME[21:16], Messages Are Expressed In BCD Format.			
24-Hour Time Scale (24HEN = 1)		12-Hour Time Scale (PM Time + 0x20) (24HEN = 0)	
0x00 (AM12)	0x12 (PM12)	0x12 (AM12)	0x32 (PM12)
0x01 (AM01)	0x13 (PM01)	0x01 (AM01)	0x21 (PM01)

0x02 (AM02)	0x14 (PM02)	0x02 (AM02)	0x22 (PM02)
0x03 (AM03)	0x15 (PM03)	0x03 (AM03)	0x23 (PM03)
0x04 (AM04)	0x16 (PM04)	0x04 (AM04)	0x24 (PM04)
0x05 (AM05)	0x17 (PM05)	0x05 (AM05)	0x25 (PM05)
0x06 (AM06)	0x18 (PM06)	0x06 (AM06)	0x26 (PM06)
0x07 (AM07)	0x19 (PM07)	0x07 (AM07)	0x27 (PM07)
0x08 (AM08)	0x20 (PM08)	0x08 (AM08)	0x28 (PM08)
0x09 (AM09)	0x21 (PM09)	0x09 (AM09)	0x29 (PM09)
0x10 (AM10)	0x22 (PM10)	0x10 (AM10)	0x30 (PM10)
0x11 (AM11)	0x23 (PM11)	0x11 (AM11)	0x31 (PM11)

Table 6.14-212/24 Hour Time Scale Selection

6.14.5.6 Day of the Week Counter

The RTC controller provides day of week in WEEKDAY (RTC_WEEKDAY[2:0]) bits. The value is defined from 0 to 6 to represent Sunday to Saturday respectively.

6.14.5.7 Periodic Time Tick Interrupt

The periodic time tick interrupt has 8 period interval options 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 and 1 second that are selected by TICK (RTC_TICK[2:0]) bits. When Periodic Time Tick interrupt is enabled by setting TICKIEN (RTC_INTEN[1]) to 1, the Periodic Time Tick interrupt is requested periodically in the period selected by RTC_TICK[2:0] settings.

6.14.5.8 Alarm Interrupt

When the real time and calendar message in RTC_TIME and RTC_CAL registers are equal to alarm time and calendar values in RTC_TALM and RTC_CALM registers, the RTC alarm interrupt flag ALMIF (RTC_INTSTS[0]) is set to 1 and the RTC alarm interrupt signal assert if the alarm interrupt enable ALMIEN (RTC_INTEN[0]) is enabled.

The RTC controller provides Time Alarm Mask Register (RTC_TAMSK) and Calendar Alarm Mask Register (RTC_CAMSK) to mask the specified digit and generate periodic interrupt without changing the alarm match condition in RTC_TALM and RTC_CALM registers in each alarm interrupt service routine.

6.14.5.9 Daylight Saving Time

The RTC controller also provides RTC_DSTCTL register to store the control settings of daylight saving time application. User can read DSBAK (RTC_DSTCTL[2]) value to check current RTC date/time counter runs in daylight saving time mode or normal mode.

6.14.5.10 1 Hz Clock Output

The RTC controller provides 1Hz clock output to CLKO function pin. User can set CLK1HZEN (CLK_CLKOCTL[6]) to 1 and enable RTC, 1Hz clock will output to CLKO function pin.

6.14.5.11 Application Note

1. All data in RTC_TALM, RTC_CALM, RTC_TIME and RTC_CAL registers are all expressed in BCD format.
2. User has to make sure that the loaded values are reasonable. For example, Load RTC_CAL as 201a (year), 13 (month), 00 (day), or RTC_CAL does not match with RTC_WEEKDAY, etc.

3. In RTC_CAL and RTC_CALM, only 2 BCD digits are used to express “year”. The 2 BCD digits of xy means 20xy, rather than 19xy or 21xy.
4. Example of 12-Hour Time Setting
   If current RTC time is PM12:59:30 in 12-Hour Time Scale mode, the RTC_TIME setting as:

   HOUR:

   RTC_TIME[21:16]: 0x32 (0x12+0x20) combined by TENHR (RTC_TIME[21:20]) is 0x3, HR (RTC_TIME[19:16]) is 0x2.

   MIN:
   RTC_TIME[14:8]: 0x59 combined by TENMIN (RTC_TIME[14:12]) is 0x5, MIN (RTC_TIME[11:8]) is 0x9.

   SEC:
   RTC_TIME[6:0]: 0x30 combined by TENSEC (RTC_TIME[6:4]) is 0x3, SEC (RTC_TIME[3:0]) is 0x0.
5. Table 6.14-3 shows registers value after both core power and battery power are first powered on.

Register	Reset State
RTC_INIT	0
RTC_CAL	15/8/8 (year/month/day)
RTC_TIME	00:00:00 (hour : minute : second)
RTC_CALM	00/00/00 (year/month/day)
RTC_TALM	00:00:00 (hour : minute : second)
RTC_CLKFMT	1 (24-hour mode)
RTC_WEEKDAY	6 (Saturday)
RTC_INTEN	0
RTC_INTSTS	0
RTC_LEAPYEAR	0
RTC_TICK	0
RTC_DSTCTL	0

Table 6.14-3 Registers Value after Powered On

6.14.5.12 Spare Registers and Tamper Detector

The RTC module is equipped with 80 bytes spare registers to store user’s important information. These spare registers are located in RTC domain, user needs to enable SPRRWEN (RTC_SPRCTL[2]) before writing one of 20 spare registers (RTC_SPR0 ~ RTC_SPR19).

When the transition condition defined in RTC_TAMPCTL is detected, tamper detected interrupt flag TAMPxIF (RTC_INTSTS[13:8]) will be generated. Meanwhile, the 80 bytes spare registers (RTC_SPR0 ~ RTC_SPR19) content will be cleared automatically by hardware to prevent the security data be disclosure and current RTC time and calendar will be loaded to RTC_TAMPTIME and RTC_TAMPCAL registers, these values only can be update again when all TAMPxIF are cleared to 0. And if TAMPxIF is set to 1, the interrupt is generated to NVIC if the tamper detect interrupt enable RTC_TAMPxIEN (RTC_INTEN[13:8]) is enabled.

After Flash mass erase operate, these 80 bytes spare registers content will be cleared automatically also.

The RTC supports 3 pair dynamic loop tamper pins or 6 individual tamper pins. The DYNRATE (RTC_TAMPCTL[7:5]) determine a reference pattern bit duration and it is shown in Figure 6.14-2. In Table 6.14-4, setting DYNSRC (RTC_TAMPCTL[3]) can select detect signal is new value which generated by hardware random value generator or user defined SEED value (RTC_TAMPSEED[31:0]) for dynamic loop tamper pin. Set DYN1ISS (RTC_TAMPCTL[0]) 1 can be select Pair1 detection source is from tamper 0, and DYN12SS (RTC_TAMPCTL[1]) 1 can be select Pair2 detection source is also from tamper 0. The TAMPxLV (RTC_TAMPCTL[4x+9], x=0, 1, ...,5) depend on level attribute of TAMPERx pin for individual tamper detection. The tamper control effect is shown in Table 6.14-5 and Table 6.14-6.

DYNRATE	Duration
000	1/32 sec
001	1/16 sec
010	1/8 sec
011	1/4 sec
100	1/2 sec
101	1 sec
110	2 sec
111	4 sec

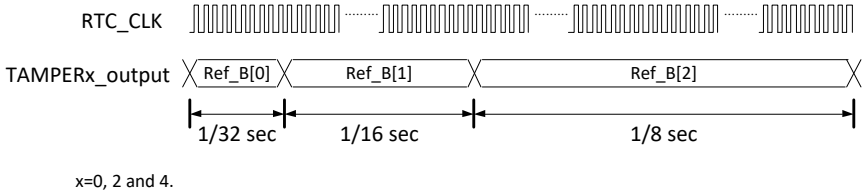


Figure 6.14-2 Dynamic Rate Definition

DYNPRxEN	DYNSRC	The Detect Signal Description
0	-	The TAMPERn is static tamper detection (n=0 ~ 5)
1	0	Generating new random value as reference pattern when the value runs out
1	1	Repeated user defined SEED value as reference pattern when the value runs out

Table 6.14-4 Dynamic Pattern Source Selection

Tamper Configuration			Tamper Control Bit Effect					
DYNPR0EN	TAMPER0EN	TAMPER1EN	TAMP0LV	TAMP1LV	TAMP0DBEN	TAMP1DBEN	TAMPER 0	TAMPER 1
0	0	0	-	-	-	-	-	-
0	0	1	-	V	-	V	-	Static
0	1	0	V	-	V	-	Static	-
0	1	1	V	V	V	V	Static	Static
1	-	0	-	-	-	-	Dynamic Out	-
1	-	1	-	-	-	-	Dynamic Out	Dynamic In

Table 6.14-5 Tamper Control Bit Effect for Pair 0

Tamper Configuration				Tamper Control Bit Effect					
DYNPRxE	DYNxISS	TAMPERnE	TAMPEmRE	TAMPnL	TAMPmL	TAMPnDBE	TAMPmDBE	TAMPER n	TAMPER m

N		N	N	V	V	N	N		
0	-	0	0	-	-	-	-	-	-
0	-	0	1	-	V	-	V	-	Static
0	-	1	0	V	-	V	-	Static	-
0	-	1	1	V	V	V	V	Static	Static
1	0	-	0	-	-	-	-	Dynamic Out	-
1	0	-	1	-	-	-	-	Dynamic Out	Dynamic In
1	1	0	1	-	-	-	-	-	Dynamic Input form TAMPER 0
1	1	1	1	V	V	-	-	Static	Dynamic Input form TAMPER 0
x= 1 and 2		n= 2 and 4, m= 3 and 5							

Table 6.14-6 Tamper Control Bit Effect for Pair 1 and 2

Static Tamper Programming Sequence Example:

1. Clean the TAMPxIF (RTC_INTSTS[x+8], x=0, 1, …, 5).
2. Set TAMPxLV (RTC_TAMPCTL[4x+9], x=0, 1, ..., 5) and TAMPxDBEN (RTC_TAMPCTL[4x+10], x=0, 1, ..., 5).
3. Enable TAMPxEN (RTC_TAMPCTL[4x+8], x=0, 1, ..., 5).

Dynamic Tamper Programming Sequence Example:

1. Clean the TAMPxIF (RTC_INTSTS[x+8], x=0, 1, …, 5).
2. Fill the SEED (RTC_TAMPSEED[31:0]).
3. Setting DYNSRC (RTC_TAMPCTL[3]), DYNxISS (RTC_TAMPCTL[x], x=0, 1) and DYNRATE (RTC_TAMPCTL[7:5]).
4. Enable DYNPRxEN (RTC_TAMPCTL[8x+16], x=0, 1, 2).
5. Enable TAMPxEN (RTC_TAMPCTL[4x+8], x=0, 1, ..., 5).
6. Set SEEDRLD (RTC_TAMPCTL[4]).

6.14.5.13 Backup Domain GPIO Function

When PF.4/X32_OUT and PF.5/X32_IN pins are not used as low speed 32K oscillator function or PF.6~11 are not used for TAPMER function, they can be used as GPIO pin function. The IOCTLSEL (RTC_LXTCTL[8]) is used to select the PF.4~11 pins are controlled by RTC_GPIOCTL0/1 registers by V_{BAT} power domain or relative GPIO control registers by GPIO module. Figure 6.14-3 shows backup I/O control diagram.

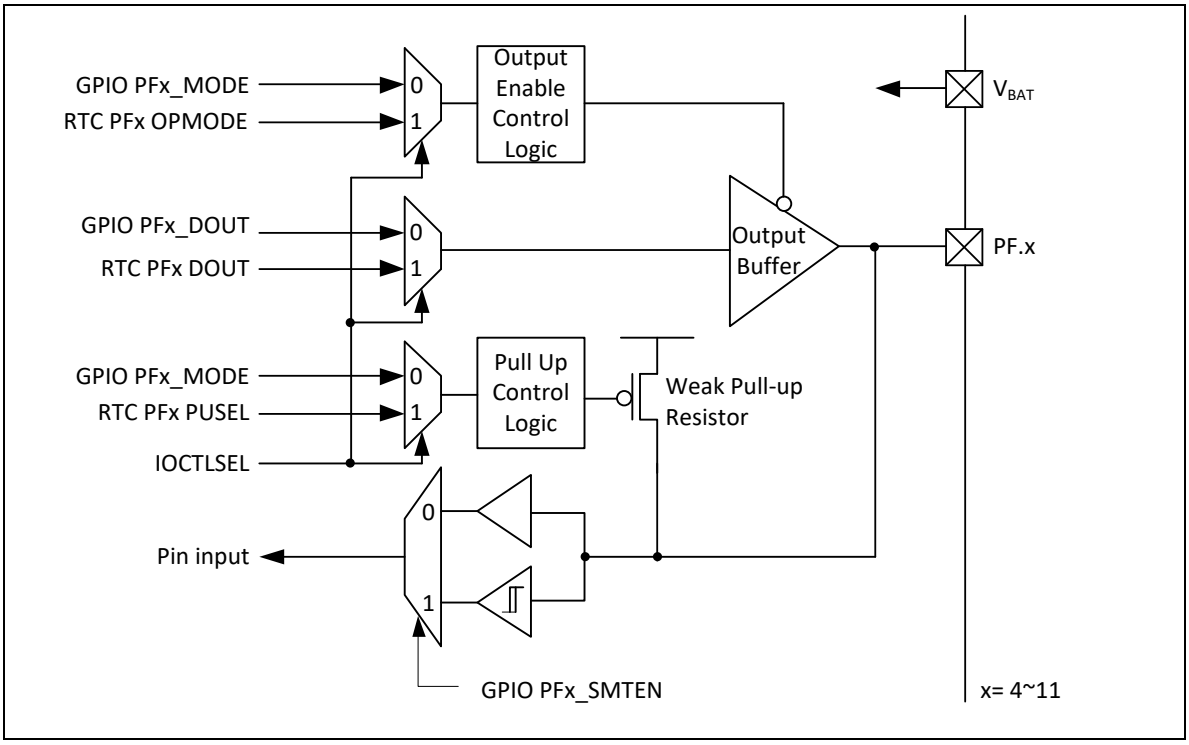


Figure 6.14-3 Backup I/O Control Diagram

6.14.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
RTC Base Address: **RTC_BA = 0x4004_1000**				
RTC_INIT	RTC_BA+0x00	R/W	RTC Initiation Register	0x0000_0000
RTC_FREQADJ	RTC_BA+0x08	R/W	RTC Frequency Compensation Register	0x0000_1000
RTC_TIME	RTC_BA+0x0C	R/W	RTC Time Loading Register	0x0000_0000
RTC_CAL	RTC_BA+0x10	R/W	RTC Calendar Loading Register	0x0015_0808
RTC_CLKFMT	RTC_BA+0x14	R/W	RTC Time Scale Selection Register	0x0000_0001
RTC_WEEKDAY	RTC_BA+0x18	R/W	RTC Day of the Week Register	0x0000_0006
RTC_TALM	RTC_BA+0x1C	R/W	RTC Time Alarm Register	0x0000_0000
RTC_CALM	RTC_BA+0x20	R/W	RTC Calendar Alarm Register	0x0000_0000
RTC_LEAPYEAR	RTC_BA+0x24	R	RTC Leap Year Indicator Register	0x0000_0000
RTC_INTEN	RTC_BA+0x28	R/W	RTC Interrupt Enable Register	0x0000_0000
RTC_INTSTS	RTC_BA+0x2C	R/W	RTC Interrupt Status Register	0x0000_0000
RTC_TICK	RTC_BA+0x30	R/W	RTC Time Tick Register	0x0000_0000
RTC_TAMSK	RTC_BA+0x34	R/W	RTC Time Alarm Mask Register	0x0000_0000
RTC_CAMSK	RTC_BA+0x38	R/W	RTC Calendar Alarm Mask Register	0x0000_0000
RTC_SPRCTL	RTC_BA+0x3C	R/W	RTC Spare Functional Control Register	0x0000_0000
RTC_SPR0	RTC_BA+0x40	R/W	RTC Spare Register 0	0x0000_0000
RTC_SPR1	RTC_BA+0x44	R/W	RTC Spare Register 1	0x0000_0000
RTC_SPR2	RTC_BA+0x48	R/W	RTC Spare Register 2	0x0000_0000
RTC_SPR3	RTC_BA+0x4C	R/W	RTC Spare Register 3	0x0000_0000
RTC_SPR4	RTC_BA+0x50	R/W	RTC Spare Register 4	0x0000_0000
RTC_SPR5	RTC_BA+0x54	R/W	RTC Spare Register 5	0x0000_0000
RTC_SPR6	RTC_BA+0x58	R/W	RTC Spare Register 6	0x0000_0000
RTC_SPR7	RTC_BA+0x5C	R/W	RTC Spare Register 7	0x0000_0000
RTC_SPR8	RTC_BA+0x60	R/W	RTC Spare Register 8	0x0000_0000
RTC_SPR9	RTC_BA+0x64	R/W	RTC Spare Register 9	0x0000_0000
RTC_SPR10	RTC_BA+0x68	R/W	RTC Spare Register 10	0x0000_0000

RTC_SPR11	RTC_BA+0x6C	R/W	RTC Spare Register 11	0x0000_0000
RTC_SPR12	RTC_BA+0x70	R/W	RTC Spare Register 12	0x0000_0000
RTC_SPR13	RTC_BA+0x74	R/W	RTC Spare Register 13	0x0000_0000
RTC_SPR14	RTC_BA+0x78	R/W	RTC Spare Register 14	0x0000_0000
RTC_SPR15	RTC_BA+0x7C	R/W	RTC Spare Register 15	0x0000_0000
RTC_SPR16	RTC_BA+0x80	R/W	RTC Spare Register 16	0x0000_0000
RTC_SPR17	RTC_BA+0x84	R/W	RTC Spare Register 17	0x0000_0000
RTC_SPR18	RTC_BA+0x88	R/W	RTC Spare Register 18	0x0000_0000
RTC_SPR19	RTC_BA+0x8C	R/W	RTC Spare Register 19	0x0000_0000
RTC_LXTCTL	RTC_BA+0x100	R/W	RTC 32.768 kHz Oscillator Control Register	0xXXXX_XX0E
RTC_GPIOCTL0	RTC_BA+0x104	R/W	RTC GPIO Control 0 Register	0x0000_0000
RTC_GPIOCTL1	RTC_BA+0x108	R/W	RTC GPIO Control 1 Register	0x0000_0000
RTC_DSTCTL	RTC_BA+0x110	R/W	RTC Daylight Saving Time Control Register	0x0000_0000
RTC_TAMPCTL	RTC_BA+0x120	R/W	RTC Tamper Pin Control Register	0x0000_0000
RTC_TAMPSEED	RTC_BA+0x128	R/W	RTC Tamper Dynamic Seed Register	0x0000_0000
RTC_TAMPTIME	RTC_BA+0x130	R	RTC Tamper Time Register	0x0000_0000
RTC_TAMPCAL	RTC_BA+0x134	R	RTC Tamper Calendar Register	0x0000_0000
RTC_CLKDCTL	RTC_BA+0x140	R/W	Clock Fail Detector Control Register	0x0000_0000
RTC_CDBR	RTC_BA+0x144	R/W	Clock Frequency Detector Boundary Register	0x00F0_000F
RTC_TEST	RTC_BA+0x1F0	R/W	RTC Test Control Register	0x0000_0000

6.14.7 Register Description

RTC Initiation Register (RTC_INIT)

Register	Offset	R/W	Description	Reset Value
RTC_INIT	RTC_BA+0x00	R/W	RTC Initiation Register	0x0000_0000

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

Bits	Description	
[31:1]	INIT	**RTC Initiation (Write Only)** When RTC block is powered on, RTC is at reset state. User has to write a number (0x a5eb1357) to INIT to make RTC leave reset state. Once the INIT is written as 0xa5eb1357, the RTC will be in un-reset state permanently. The INIT is a write-only field and read value will be always 0.
[0]	ACTIVE	**RTC Active Status (Read Only)** 0 = RTC is at reset state. 1 = RTC is at normal active state.

RTC Frequency Compensation Register (RTC_FREQADJ)

Register	Offset	R/W	Description	Reset Value
RTC_FREQADJ	RTC_BA+0x08	R/W	RTC Frequency Compensation Register	0x0000_1000

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

Bits	Description	
[31]	**FCRBUSY**	**Frequency Compensation Register Write Operation Busy (Read Only)** 0 = The new register write operation is acceptable. 1 = The last write operation is in progress and new register write operation prohibited. **Note:** This bit is only used when DCOMPEN(RTC_CLKFMT[16]) is enabled.
[30:13]	**Reserved**	Reserved.
[12:8]	**INTEGER**	**Integer Part** 00000 = Integer part of detected value is 32752. 00001 = Integer part of detected value is 32753. 00010 = Integer part of detected value is 32754. 00011 = Integer part of detected value is 32755. 00100 = Integer part of detected value is 32756. 00101 = Integer part of detected value is 32757. 00110 = Integer part of detected value is 32758. 00111 = Integer part of detected value is 32759. 01000 = Integer part of detected value is 32760. 01001 = Integer part of detected value is 32761. 01010 = Integer part of detected value is 32762. 01011 = Integer part of detected value is 32763. 01100 = Integer part of detected value is 32764. 01101 = Integer part of detected value is 32765. 01110 = Integer part of detected value is 32766. 01111 = Integer part of detected value is 32767. 10000 = Integer part of detected value is 32768. 10001 = Integer part of detected value is 32769. 10010 = Integer part of detected value is 32770. 10011 = Integer part of detected value is 32771.

		10100 = Integer part of detected value is 32772. 10101 = Integer part of detected value is 32773. 10110 = Integer part of detected value is 32774. 10111 = Integer part of detected value is 32775. 11000 = Integer part of detected value is 32776. 11001 = Integer part of detected value is 32777. 11010 = Integer part of detected value is 32778. 11011 = Integer part of detected value is 32779. 11100 = Integer part of detected value is 32780. 11101 = Integer part of detected value is 32781. 11110 = Integer part of detected value is 32782. 11111 = Integer part of detected value is 32783.
[7:6]	**Reserved**	Reserved.
[5:0]	**FRACTION**	**Fraction Part** Formula: FRACTION = (fraction part of detected value) X 64. **Note:** Digit in FCR must be expressed as hexadecimal number.

Note: If the dynamic compensation is not enabled, i.e. DCOMPEN(RTC_CLKFMT[16]) = 0, the FREQADJ counter will be reset for starting to compensate when writing RTC_FREQADJ. It will impact if the RTC counter will be restarted.

RTC Time Loading Register (RTC_TIME)

Register	Offset	R/W	Description	Reset Value
RTC_TIME	RTC_BA+0x0C	R/W	RTC Time Loading Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		TENHR		HR			
15	14	13	12	11	10	9	8
Reserved	TENMIN			MIN			
7	6	5	4	3	2	1	0
Reserved	TENSEC			SEC			

Bits	Description	
[31:22]	**Reserved**	Reserved.
[21:20]	**TENHR**	10-Hour Time Digit (0~2) When RTC runs as 12-hour time scale mode, RTC_TIME[21] (the high bit of TENHR[1:0]) means AM/PM indication (If RTC_TIME[21] is 1, it indicates PM time message.)
[19:16]	**HR**	1-Hour Time Digit (0~9)
[15]	**Reserved**	Reserved.
[14:12]	**TENMIN**	10-Min Time Digit (0~5)
[11:8]	**MIN**	1-Min Time Digit (0~9)
[7]	**Reserved**	Reserved.
[6:4]	**TENSEC**	10-Sec Time Digit (0~5)
[3:0]	**SEC**	1-Sec Time Digit (0~9)

Note:

1. RTC_TIME is a BCD digit counter and RTC will not check loaded data.
2. The reasonable value range is listed in the parenthesis.
3. The FREQADJ counter will be reset for starting to compensate when writing RTC_TIME, RTC_CAL, RTC_WEEKDAY. It will impact if the RTC counter will be restarted.

RTC Calendar Loading Register (RTC_CAL)

Register	Offset	R/W	Description	Reset Value
RTC_CAL	RTC_BA+0x10	R/W	RTC Calendar Loading Register	0x0015_0808

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TENYEAR				**YEAR**			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved			**TENMON**	**MON**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**TENDAY**		**DAY**			

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:20]	**TENYEAR**	10-Year Calendar Digit (0~9)
[19:16]	**YEAR**	1-Year Calendar Digit (0~9)
[15:13]	**Reserved**	Reserved.
[12]	**TENMON**	10-Month Calendar Digit (0~1)
[11:8]	**MON**	1-Month Calendar Digit (0~9)
[7:6]	**Reserved**	Reserved.
[5:4]	**TENDAY**	10-Day Calendar Digit (0~3)
[3:0]	**DAY**	1-Day Calendar Digit (0~9)

Note:

1. RTC_CAL is a BCD digit counter and RTC will not check loaded data.
2. The reasonable value range is listed in the parenthesis.
3. The FREQADJ counter will be reset for starting to compensate when writing RTC_TIME, RTC_CAL, RTC_WEEKDAY. It will impact if the RTC counter will be restarted.

RTC Time Scale Selection Register (RTC_CLKFMT)

Register	Offset	R/W	Description	Reset Value
RTC_CLKFMT	RTC_BA+0x14	R/W	RTC Time Scale Selection Register	0x0000_0001

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

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**DCOMPEN**	**Dynamic Compensation Enable Bit** 0 = Dynamic Compensation Disabled. 1 = Dynamic Compensation Enabled.
[15:1]	**Reserved**	Reserved.
[0]	**24HEN**	**24-hour / 12-hour Time Scale Selection** The RTC_TIME and RTC_TALM are in 24-hour time scale or 12-hour time scale. 0 = 12-hour time scale with AM and PM indication selected. 1 = 24-hour time scale selected.

RTC Day of the Week Register (RTC_WEEKDAY)

Register	Offset	R/W	Description	Reset Value
RTC_WEEKDAY	RTC_BA+0x18	R/W	RTC Day of the Week Register	0x0000_0006

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

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2:0]	**WEEKDAY**	**Day of the Week Register** 000 = Sunday. 001 = Monday. 010 = Tuesday. 011 = Wednesday. 100 = Thursday. 101 = Friday. 110 = Saturday. 111 = Reserved.

Note: The FREQADJ counter will be reset for starting to compensate when writing RTC_TIME, RTC_CAL, RTC_WEEKDAY. It will impact if the RTC counter will be restarted.

RTC Time Alarm Register (RTC_TALM)

Register	Offset	R/W	Description	Reset Value
RTC_TALM	RTC_BA+0x1C	R/W	RTC Time Alarm Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		TENHR		HR			
15	14	13	12	11	10	9	8
Reserved	TENMIN			MIN			
7	6	5	4	3	2	1	0
Reserved	TENSEC			SEC			

Bits	Description	
[31:22]	**Reserved**	Reserved.
[21:20]	**TENHR**	10-Hour Time Digit of Alarm Setting (0~2) When RTC runs as 12-hour time scale mode, RTC_TIME[21] (the high bit of TENHR[1:0]) means AM/PM indication (If RTC_TIME[21] is 1, it indicates PM time message.)
[19:16]	**HR**	1-Hour Time Digit of Alarm Setting (0~9)
[15]	**Reserved**	Reserved.
[14:12]	**TENMIN**	10-Min Time Digit of Alarm Setting (0~5)
[11:8]	**MIN**	1-Min Time Digit of Alarm Setting (0~9)
[7]	**Reserved**	Reserved.
[6:4]	**TENSEC**	10-Sec Time Digit of Alarm Setting (0~5)
[3:0]	**SEC**	1-Sec Time Digit of Alarm Setting (0~9)

Note:

1. RTC_TALM is a BCD digit counter.
2. The reasonable value range is listed in the parenthesis.

RTC Calendar Alarm Register (RTC_CALM)

Register	Offset	R/W	Description	Reset Value
RTC_CALM	RTC_BA+0x20	R/W	RTC Calendar Alarm Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TENYEAR				**YEAR**			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved			**TENMON**	**MON**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**TENDAY**		**DAY**			

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:20]	**TENYEAR**	10-Year Calendar Digit of Alarm Setting (0~9)
[19:16]	**YEAR**	1-Year Calendar Digit of Alarm Setting (0~9)
[15:13]	**Reserved**	Reserved.
[12]	**TENMON**	10-Month Calendar Digit of Alarm Setting (0~1)
[11:8]	**MON**	1-Month Calendar Digit of Alarm Setting (0~9)
[7:6]	**Reserved**	Reserved.
[5:4]	**TENDAY**	10-Day Calendar Digit of Alarm Setting (0~3)
[3:0]	**DAY**	1-Day Calendar Digit of Alarm Setting (0~9)

Note:

1. RTC_CALM is a BCD digit counter.
2. The reasonable value range is listed in the parenthesis.

RTC Leap Year Indication Register (RTC_LEAPYEAR)

Register	Offset	R/W	Description	Reset Value
RTC_LEAPYEAR	RTC_BA+0x24	R	RTC Leap Year Indicator Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**LEAPYEAR**	**Leap Year Indication (Read Only)** 0 = This year is not a leap year. 1 = This year is leap year.

RTC Interrupt Enable Register (RTC_INTEN)

Register	Offset	R/W	Description	Reset Value
RTC_INTEN	RTC_BA+0x28	R/W	RTC Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved						CLKSTIEN	CLKFIEN
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		TAMP5IEN	TAMP4IEN	TAMP3IEN	TAMP2IEN	TAMP1IEN	TAMP0IEN
7	6	5	4	3	2	1	0
Reserved						TICKIEN	ALMIEN

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25]	**CLKSTIEN**	**LXT Clock Frequency Monitor Stop Interrupt Enable Bit** 0 = LXT Frequency Stop interrupt Disabled. 1 = LXT Frequency Stop interrupt Enabled.
[24]	**CLKFIEN**	**LXT Clock Frequency Monitor Fail Interrupt Enable Bit** 0 = LXT Frequency Fail interrupt Disabled. 1 = LXT Frequency Fail interrupt Enabled.
[23:14]	**Reserved**	Reserved.
[13]	**TAMP5IEN**	**Tamper 5 or Pair 2 Interrupt Enable Bit** Set TAMP5IEN to 1 can also enable chip wake-up function when tamper 5 interrupt event is generated. 0 = Tamper 5 or Pair 2 interrupt Disabled. 1 = Tamper 5 or Pair 2 interrupt Enabled.
[12]	**TAMP4IEN**	**Tamper 4 Interrupt Enable Bit** Set TAMP4IEN to 1 can also enable chip wake-up function when tamper 4 interrupt event is generated. 0 = Tamper 4 interrupt Disabled. 1 = Tamper 4 interrupt Enabled.
[11]	**TAMP3IEN**	**Tamper 3 or Pair 1 Interrupt Enable Bit** Set TAMP3IEN to 1 can also enable chip wake-up function when tamper 3 interrupt event is generated. 0 = Tamper 3 or Pair 1 interrupt Disabled. 1 = Tamper 3 or Pair 1 interrupt Enabled.
[10]	**TAMP2IEN**	**Tamper 2 Interrupt Enable Bit** Set TAMP2IEN to 1 can also enable chip wake-up function when tamper 2 interrupt event is generated. 0 = Tamper 2 interrupt Disabled. 1 = Tamper 2 interrupt Enabled.

[9]	**TAMP1IEN**	**Tamper 1 or Pair 0 Interrupt Enable Bit** Set TAMP1IEN to 1 can also enable chip wake-up function when tamper 1 interrupt event is generated. 0 = Tamper 1 or Pair 0 interrupt Disabled. 1 = Tamper 1 or Pair 0 interrupt Enabled.
[8]	**TAMP0IEN**	**Tamper 0 Interrupt Enable Bit** Set TAMP0IEN to 1 can also enable chip wake-up function when tamper 0 interrupt event is generated. 0 = Tamper 0 interrupt Disabled. 1 = Tamper 0 interrupt Enabled.
[7:2]	**Reserved**	Reserved.
[1]	**TICKIEN**	**Time Tick Interrupt Enable Bit** Set TICKIEN to 1 can also enable chip wake-up function when RTC tick interrupt event is generated. 0 = RTC Time Tick interrupt Disabled. 1 = RTC Time Tick interrupt Enabled.
[0]	**ALMIEN**	**Alarm Interrupt Enable Bit** Set ALMIEN to 1 can also enable chip wake-up function when RTC alarm interrupt event is generated. 0 = RTC Alarm interrupt Disabled. 1 = RTC Alarm interrupt Enabled.

RTC Interrupt Status Register (RTC_INTSTS)

Register	Offset	R/W	Description	Reset Value
RTC_INTSTS	RTC_BA+0x2C	R/W	RTC Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved						CLKSTIF	CLKFIF
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		TAMP5IF	TAMP4IF	TAMP3IF	TAMP2IF	TAMP1IF	TAMP0IF
7	6	5	4	3	2	1	0
Reserved						TICKIF	ALMIF

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25]	**CLKSTIF**	**LXT Clock Frequency Monitor Stop Interrupt Flag** 0 = LXT frequency is normal. 1 = LXT frequency is almost stop. **Note 1:** Write 1 to clear the bit to 0. **Note 2:** LXT detector will automatically disable when CLKSTIF/CLKFIF flag rise, resume after Fail/Stop Flag clear.
[24]	**CLKFIF**	**LXT Clock Frequency Monitor Fail Interrupt Flag** 0 = LXT frequency is normal. 1 = LXT frequency is abnormal. **Note 1:** Write 1 to clear the bit to 0. **Note 2:** LXT detector will automatically disable when CLKSTIF/CLKFIF flag rise, resume after Fail/Stop Flag clear.
[23:14]	**Reserved**	Reserved.
[13]	**TAMP5IF**	**Tamper 5 or Pair 2 Interrupt Flag** This bit is set when TAMPER5 detected level non-equal TAMP5LV (RTC_TAMPCTL[29]) or TAMPER4 and TAMPER5 disconnected during DYNPR2EN (RTC_TAMPCTL[31]) is activated or TAMPER0 and TAMPER5 disconnected during DYNPR2EN (RTC_TAMPCTL[31]) and DYN2ISS (RTC_TAMPCTL[1]) are activated. 0 = No Tamper 5 or Pair 2 interrupt flag is generated. 1 = Tamper 5 or Pair 2 interrupt flag is generated. **Note 1:** Write 1 to clear this bit. **Note 2:** This interrupt flag will be generated again when Tamper setting condition is not restoration. **Note 3:** Need to clear all TAMPxIF, after that, new RTC_TAMPTIME and RTC_TAMPCAL values are loaded while a tamper event occurred.
[12]	**TAMP4IF**	**Tamper 4 Interrupt Flag** This bit is set when TAMPER4 detected level non-equal TAMP4LV (RTC_TAMPCTL[25]). 0 = No Tamper 4 interrupt flag is generated.

		1 = Tamper 4 interrupt flag is generated. **Note 1:** Write 1 to clear this bit. **Note 2:** This interrupt flag will be generated again when Tamper setting condition is not restoration. **Note 3:** Need to clear all TAMPxIF, after that, new RTC_TAMPTIME and RTC_TAMPCAL values are loaded while a tamper event occurred.
[11]	**TAMP3IF**	**Tamper 3 or Pair 1 Interrupt Flag** This bit is set when TAMPER3 detected level non-equal TAMP3LV (RTC_TAMPCTL[21]) or TAMPER2 and TAMPER3 disconnected during DYNPR1EN (RTC_TAMPCTL[23]) is activated or TAMPER0 and TAMPER3 disconnected during DYNPR1EN (RTC_TAMPCTL[23]) and DYN1ISS (RTC_TAMPCTL[0]) are activated. 0 = No Tamper 3 or Pair 1 interrupt flag is generated. 1 = Tamper 3 or Pair 1 interrupt flag is generated. **Note 1:** Write 1 to clear this bit. **Note 2:** This interrupt flag will be generated again when Tamper setting condition is not restoration. **Note 3:** Need to clear all TAMPxIF, after that, new RTC_TAMPTIME and RTC_TAMPCAL values are loaded while a tamper event occurred.
[10]	**TAMP2IF**	**Tamper 2 Interrupt Flag** This bit is set when TAMPER2 detected level non-equal TAMP2LV (RTC_TAMPCTL[17]). 0 = No Tamper 2 interrupt flag is generated. 1 = Tamper 2 interrupt flag is generated. **Note 1:** Write 1 to clear this bit. **Note 2:** This interrupt flag will be generated again when Tamper setting condition is not restoration. **Note 3:** Need to clear all TAMPxIF, after that, new RTC_TAMPTIME and RTC_TAMPCAL values are loaded while a tamper event occurred.
[9]	**TAMP1IF**	**Tamper 1 or Pair 0 Interrupt Flag** This bit is set when TAMPER1 detected level non-equal TAMP1LV (RTC_TAMPCTL[13]) or TAMPER0 and TAMPER1 disconnected during DYNPR0EN (RTC_TAMPCTL[15]) is activated. 0 = No Tamper 1 or Pair 0 interrupt flag is generated. 1 = Tamper 1 or Pair 0 interrupt flag is generated. **Note 1:** Write 1 to clear this bit. **Note 2:** This interrupt flag will be generated again when Tamper setting condition is not restoration. **Note 3:** Need to clear all TAMPxIF, after that, new RTC_TAMPTIME and RTC_TAMPCAL values are loaded while a tamper event occurred.
[8]	**TAMP0IF**	**Tamper 0 Interrupt Flag** This bit is set when TAMPER0 detected level non-equal TAMP0LV (RTC_TAMPCTL[9]). 0 = No Tamper 0 interrupt flag is generated. 1 = Tamper 0 interrupt flag is generated. **Note 1:** Write 1 to clear this bit. **Note 2:** This interrupt flag will be generated again when Tamper setting condition is not restoration. **Note 3:** Need to clear all TAMPxIF, after that, new RTC_TAMPTIME and RTC_TAMPCAL values are loaded while a tamper event occurred.
[7:2]	**Reserved**	Reserved.
[1]	**TICKIF**	**RTC Time Tick Interrupt Flag** 0 = Tick condition did not occur. 1 = Tick condition occurred. **Note:** Write 1 to clear this bit.
[0]	**ALMIF**	**RTC Alarm Interrupt Flag** 0 = Alarm condition is not matched.

		1 = Alarm condition is matched. **Note:** Write 1 to clear this bit.

RTC Time Tick Register (RTC_TICK)

Register	Offset	R/W	Description	Reset Value
RTC_TICK	RTC_BA+0x30	R/W	RTC Time Tick Register	0x0000_0000

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

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2:0]	**TICK**	**Time Tick Register** These bits are used to select RTC time tick period for Periodic Time Tick Interrupt request. 000 = Time tick is 1 second. 001 = Time tick is 1/2 second. 010 = Time tick is 1/4 second. 011 = Time tick is 1/8 second. 100 = Time tick is 1/16 second. 101 = Time tick is 1/32 second. 110 = Time tick is 1/64 second. 111 = Time tick is 1/128 second.

RTC Time Alarm MASK Register (RTC_TAMSK)

Register	Offset	R/W	Description	Reset Value
RTC_TAMSK	RTC_BA+0x34	R/W	RTC Time Alarm Mask Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**MTENHR**	**MHR**	**MTENMIN**	**MMIN**	**MTENSEC**	**MSEC**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**MTENHR**	Mask 10-Hour Time Digit of Alarm Setting (0~2)
[4]	**MHR**	Mask 1-Hour Time Digit of Alarm Setting (0~9)
[3]	**MTENMIN**	Mask 10-Min Time Digit of Alarm Setting (0~5)
[2]	**MMIN**	Mask 1-Min Time Digit of Alarm Setting (0~9)
[1]	**MTENSEC**	Mask 10-Sec Time Digit of Alarm Setting (0~5)
[0]	**MSEC**	Mask 1-Sec Time Digit of Alarm Setting (0~9)

Note:

1. RTC_TALM is a BCD digit counter and RTC will not check loaded data.
2. The reasonable value range is listed in the parenthesis.
3. MTENHR/MHR is based on 24 hour Time Scale.

RTC Calendar Alarm MASK Register (RTC_CAMSK)

Register	Offset	R/W	Description	Reset Value
RTC_CAMSK	RTC_BA+0x38	R/W	RTC Calendar Alarm Mask Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**MTENYEAR**	**MYEAR**	**MTENMON**	**MMON**	**MTENDAY**	**MDAY**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**MTENYEAR**	Mask 10-Year Calendar Digit of Alarm Setting (0~9)
[4]	**MYEAR**	Mask 1-Year Calendar Digit of Alarm Setting (0~9)
[3]	**MTENMON**	Mask 10-Month Calendar Digit of Alarm Setting (0~1)
[2]	**MMON**	Mask 1-Month Calendar Digit of Alarm Setting (0~9)
[1]	**MTENDAY**	Mask 10-Day Calendar Digit of Alarm Setting (0~3)
[0]	**MDAY**	Mask 1-Day Calendar Digit of Alarm Setting (0~9)

Note:

1. RTC_CALM is a BCD digit counter and RTC will not check loaded data.
2. The reasonable value range is listed in the parenthesis.

RTC Spare Functional Control Register (RTC_SPRCTL)

Register	Offset	R/W	Description	Reset Value
RTC_SPRCTL	RTC_BA+0x3C	R/W	RTC Spare Functional Control Register	0x0000_0000

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

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**LXTFCLR**	**LXT Clock Fail/Stop to Clear Spare Enable Bit** 0 = LXT Fail/Stop to clear Spare register content Disabled. 1 = LXT Fail/Stop to clear Spare register content Enabled.
[15:6]	**Reserved**	Reserved.
[5]	**SPRCSTS**	**SPR Clear Flag** This bit indicates if the RTC_SPR0 ~RTC_SPR19 content is cleared when specify tamper event is detected. 0 = Spare register content is not cleared. 1 = Spare register content is cleared. **Note 1:** Write 1 to clear this bit. **Note 2:** This bit keeps 1 when RTC_INTSTS[13:8] is not equal to 0.
[4:3]	**Reserved**	Reserved.
[2]	**SPRRWEN**	**Spare Register Enable Bit** 0 = Spare register Disabled. 1 = Spare register Enabled. **Note:** When spare register is disabled, RTC_SPR0 ~ RTC_SPR19 cannot be accessed.
[1:0]	**Reserved**	Reserved.

RTC Spare Register (RTC_SPRx)

Register	Offset	R/W	Description	Reset Value
RTC_SPR0	RTC_BA+0x40	R/W	RTC Spare Register 0	0x0000_0000
RTC_SPR1	RTC_BA+0x44	R/W	RTC Spare Register 1	0x0000_0000
RTC_SPR2	RTC_BA+0x48	R/W	RTC Spare Register 2	0x0000_0000
RTC_SPR3	RTC_BA+0x4C	R/W	RTC Spare Register 3	0x0000_0000
RTC_SPR4	RTC_BA+0x50	R/W	RTC Spare Register 4	0x0000_0000
RTC_SPR5	RTC_BA+0x54	R/W	RTC Spare Register 5	0x0000_0000
RTC_SPR6	RTC_BA+0x58	R/W	RTC Spare Register 6	0x0000_0000
RTC_SPR7	RTC_BA+0x5C	R/W	RTC Spare Register 7	0x0000_0000
RTC_SPR8	RTC_BA+0x60	R/W	RTC Spare Register 8	0x0000_0000
RTC_SPR9	RTC_BA+0x64	R/W	RTC Spare Register 9	0x0000_0000
RTC_SPR10	RTC_BA+0x68	R/W	RTC Spare Register 10	0x0000_0000
RTC_SPR11	RTC_BA+0x6C	R/W	RTC Spare Register 11	0x0000_0000
RTC_SPR12	RTC_BA+0x70	R/W	RTC Spare Register 12	0x0000_0000
RTC_SPR13	RTC_BA+0x74	R/W	RTC Spare Register 13	0x0000_0000
RTC_SPR14	RTC_BA+0x78	R/W	RTC Spare Register 14	0x0000_0000
RTC_SPR15	RTC_BA+0x7C	R/W	RTC Spare Register 15	0x0000_0000
RTC_SPR16	RTC_BA+0x80	R/W	RTC Spare Register 16	0x0000_0000
RTC_SPR17	RTC_BA+0x84	R/W	RTC Spare Register 17	0x0000_0000
RTC_SPR18	RTC_BA+0x88	R/W	RTC Spare Register 18	0x0000_0000
RTC_SPR19	RTC_BA+0x8C	R/W	RTC Spare Register 19	0x0000_0000

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

Bits	Description	
[31:0]	**SPARE**	**Spare Register** This field is used to store back-up information defined by user. This field will be cleared by hardware automatically in the following conditions, a tamper pin event is detected, LXT clock fail/stop event occurs if LXTFCLR(RTC_SPRCTL[16]) is 1, or after Flash mass operation.

RTC 32K Oscillator Control Register (RTC_LXTCTL)

Register	Offset	R/W	Description	Reset Value
RTC_LXTCTL	RTC_BA+0x100	R/W	RTC 32.768 kHz Oscillator Control Register	0xXXXX_XX0E

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							IOCTLSEL
7	6	5	4	3	2	1	0
RTCCKSEL	C32KSEL	Reserved		GAIN			LIRC32KEN

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**IOCTLSEL**	**I/O Pin Backup Control Selection** When low speed 32 kHz oscillator is disabled or TAMPxEN is disabled, PF.4 pin (X32_OUT pin), PF.5 pin (X32_IN pin) or PF.6~11 pin (TAMPERx pin) can be used as GPIO function. User can program IOCTLSEL to decide PF.4~11 I/O function is controlled by system power domain GPIO module or V_{BAT} power domain RTC_GPIOCTL0/1 control register. 0 = PF.4~11 pin I/O function is controlled by GPIO module. 1 = PF.4~11 pin I/O function is controlled by V_{BAT} power domain. **Note:** IOCTLSEL will automatically be set by hardware to 1 when system power is off and any writable RTC registers has been written at RTCCKEN(CLK_APBCLK0[1]) enabled.
[7]	**RTCCKSEL**	**RTC Clock Source Selection** 0 = Clock source from external low speed crystal oscillator (LXT) or internal low speed RC 32K oscillator (LIRC32K) depended on C32KSEL value. 1 = Clock source from internal low speed RC oscillator (LIRC).
[6]	**C32KSEL**	**Clock 32K Source Selection** 0 = Clock source from external low speed crystal oscillator (LXT). 1 = Clock source from internal low speed RC 32K oscillator (LIRC32K).
[5:4]	**Reserved**	Reserved.
[3:1]	**GAIN**	**Oscillator Gain Option** User can select oscillator gain according to crystal external loading and operating temperature range. The larger gain value corresponding to stronger driving capability and higher power consumption. 000 = reserved. 001 = L1 mode. 010 = reserved. 011 = L3 mode. 100 = L4 mode.

		101 = reserved. 110 = L6 mode. 111 = L7 mode(Default).
[0]	**LIRC32KEN**	**Enable LIRC32K Source** 0 = LIRC32K Disabled. 1 = LIRC32K Enabled.

RTC GPIO Control Register0 (RTC_GPIOCTL0)

Register	Offset	R/W	Description	Reset Value
RTC_GPIOCTL0	RTC_BA+0x104	R/W	RTC GPIO Control 0 Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		PUSEL3		DINOFF3	DOUT3	OPMODE3	
23	22	21	20	19	18	17	16
Reserved		PUSEL2		DINOFF2	DOUT2	OPMODE2	
15	14	13	12	11	10	9	8
Reserved		PUSEL1		DINOFF1	DOUT1	OPMODE1	
7	6	5	4	3	2	1	0
Reserved		PUSEL0		DINOFF0	DOUT0	OPMODE0	

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:28]	**PUSEL3**	**I/O Pull-up and Pull-down Enable Bits** Determine PF.7 I/O pull-up or pull-down. 00 = PF.7 pull-up and pull-down Disabled. 01 = PF.7 pull-up Enabled. 10 = PF.7 pull-down Enabled. 11 = PF.7 pull-up and pull-down Disabled. **Note:** Basically, the pull-up control and pull-down control has following behavior limitation. The independent pull-up / pull-down control register is only valid when OPMODE3 is set as input tri-state and open-drain mode.
[27]	**DINOFF3**	**I/O Pin Digital Input Path Disable Bit** 0 = PF.7 digital input path Enabled. 1 = PF.7 digital input path Disabled (digital input tied to low).
[26]	**DOUT3**	**I/O Output Data** 0 = PF.7 output low. 1 = PF.7 output high.
[25:24]	**OPMODE3**	**I/O Operation Mode** 00 = PF.7 is input only mode. 01 = PF.7 is output push pull mode. 10 = PF.7 is open drain mode. 11 = PF.7 is quasi-bidirectional mode.
[23:22]	**Reserved**	Reserved.
[21:20]	**PUSEL2**	**I/O Pull-up and Pull-down Enable Bits** Determine PF.6 I/O Pull-up or Pull-down. 00 = PF.6 pull-up and pull-down Disabled.

		01 = PF.6 pull-up Enabled. 10 = PF.6 pull-down Enabled. 11 = PF.6 pull-up and pull-down Disabled. **Note:** Basically, the pull-up control and pull-down control has following behavior limitation. The independent pull-up / pull-down control register is only valid when OPMODE2 is set as input tri-state and open-drain mode.
[19]	**DINOFF2**	**I/O Pin Digital Input Path Disable Bit** 0 = PF.6 digital input path Enabled. 1 = PF.6 digital input path Disabled (digital input tied to low).
[18]	**DOUT2**	**I/O Output Data** 0 = PF.6 output low. 1 = PF.6 output high.
[17:16]	**OPMODE2**	**I/O Operation Mode** 00 = PF.6 is input only mode. 01 = PF.6 is output push pull mode. 10 = PF.6 is open drain mode. 11 = PF.6 is quasi-bidirectional mode.
[15:14]	**Reserved**	Reserved.
[13:12]	**PUSEL1**	**I/O Pull-up and Pull-down Enable Bits** Determine PF.5 I/O pull-up or pull-down. 00 = PF.5 pull-up and pull-down Disabled. 01 = PF.5 pull-up Enabled. 10 = PF.5 pull-down Enabled. 11 = PF.5 pull-up and pull-down Disabled. **Note:** Basically, the pull-up control and pull-down control has following behavior limitation. The independent pull-up / pull-down control register is only valid when OPMODE1 is set as input tri-state and open-drain mode.
[11]	**DINOFF1**	**I/O Pin Digital Input Path Disable Bit** 0 = PF.5 digital input path Enabled. 1 = PF.5 digital input path Disabled (digital input tied to low).
[10]	**DOUT1**	**I/O Output Data** 0 = PF.5 output low. 1 = PF.5 output high.
[9:8]	**OPMODE1**	**I/O Operation Mode** 00 = PF.5 is input only mode. 01 = PF.5 is output push pull mode. 10 = PF.5 is open drain mode. 11 = PF.5 is quasi-bidirectional mode.
[7:6]	**Reserved**	Reserved.
[5:4]	**PUSEL0**	**I/O Pull-up and Pull-down Enable Bits** Determine PF.4 I/O pull-up or pull-down. 00 = PF.4 pull-up and pull-down Disabled. 01 = PF.4 pull-up Enabled. 10 = PF.4 pull-down Enabled.

		11 = PF.4 pull-up and pull-down Disabled. **Note:** Basically, the pull-up control and pull-down control has following behavior limitation. The independent pull-up / pull-down control register is only valid when OPMODE0 is set as input tri-state and open-drain mode.
[3]	**DINOFF0**	**I/O Pin Digital Input Path Disable Bit** 0 = PF.4 digital input path Enabled. 1 = PF.4 digital input path Disabled (digital input tied to low).
[2]	**DOUT0**	**I/O Output Data** 0 = PF.4 output low. 1 = PF.4 output high.
[1:0]	**OPMODE0**	**I/O Operation Mode** 00 = PF.4 is input only mode. 01 = PF.4 is output push pull mode. 10 = PF.4 is open drain mode. 11 = PF.4 is quasi-bidirectional mode.

Note: RTC GPIO will be input mode when not programmed, and user needs to avoid the pin floating problem.

RTC GPIO Control Register1 (RTC_GPIOCTL1)

Register	Offset	R/W	Description	Reset Value
RTC_GPIOCTL1	RTC_BA+0x108	R/W	RTC GPIO Control 1 Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		PUSEL7		DINOFF7	DOUT7	OPMODE7	
23	22	21	20	19	18	17	16
Reserved		PUSEL6		DINOFF6	DOUT6	OPMODE6	
15	14	13	12	11	10	9	8
Reserved		PUSEL5		DINOFF5	DOUT5	OPMODE5	
7	6	5	4	3	2	1	0
Reserved		PUSEL4		DINOFF4	DOUT4	OPMODE4	

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:28]	**PUSEL7**	**I/O Pull-up and Pull-down Enable Bits** Determine PF.11 I/O pull-up or pull-down. 00 = PF.11 pull-up and pull-down Disabled. 01 = PF.11 pull-up Enabled. 10 = PF.11 pull-down Enabled. 11 = PF.11 pull-up and pull-down Disabled. **Note:** Basically, the pull-up control and pull-down control has following behavior limitation. The independent pull-up / pull-down control register is only valid when OPMODE7 is set as input tri-state and open-drain mode.
[27]	**DINOFF7**	**I/O Pin Digital Input Path Disable Bit** 0 = PF.11 digital input path Enabled. 1 = PF.11 digital input path Disabled (digital input tied to low).
[26]	**DOUT7**	**I/O Output Data** 0 = PF.11 output low. 1 = PF.11 output high.
[25:24]	**OPMODE7**	**I/O Operation Mode** 00 = PF.11 is input only mode. 01 = PF.11 is output push pull mode. 10 = PF.11 is open drain mode. 11 = PF.11 is quasi-bidirectional mode.
[23:22]	**Reserved**	Reserved.
[21:20]	**PUSEL6**	**I/O Pull-up and Pull-down Enable Bits** Determine PF.10 I/O pull-up or pull-down. 00 = PF.10 pull-up and pull-down Disabled.

		01 = PF.10 pull-up Enabled. 10 = PF.10 pull-down Enabled. 11 = PF.10 pull-up and pull-down Disabled. **Note:** Basically, the pull-up control and pull-down control has following behavior limitation. The independent pull-up / pull-down control register is only valid when OPMODE6 is set as input tri-state and open-drain mode.
[19]	**DINOFF6**	**I/O Pin Digital Input Path Disable Bit** 0 = PF.10 digital input path Enabled. 1 = PF.10 digital input path Disabled (digital input tied to low).
[18]	**DOUT6**	**I/O Output Data** 0 = PF.10 output low. 1 = PF.10 output high.
[17:16]	**OPMODE6**	**I/O Operation Mode** 00 = PF.10 is input only mode. 01 = PF.10 is output push pull mode. 10 = PF.10 is open drain mode. 11 = PF.10 is quasi-bidirectional mode .
[15:14]	**Reserved**	Reserved.
[13:12]	**PUSEL5**	**I/O Pull-up and Pull-down Enable Bits** Determine PF.9 I/O pull-up or pull-down. 00 = PF.9 pull-up and pull-down Disabled. 01 = PF.9 pull-up Enabled. 10 = PF.9 pull-down Enabled. 11 = PF.9 pull-up and pull-down Disabled. **Note:** Basically, the pull-up control and pull-down control has following behavior limitation. The independent pull-up / pull-down control register is only valid when OPMODE5 is set as input tri-state and open-drain mode.
[11]	**DINOFF5**	**I/O Pin Digital Input Path Disable Bit** 0 = PF.9 digital input path Enabled. 1 = PF.9 digital input path Disabled (digital input tied to low).
[10]	**DOUT5**	**I/O Output Data** 0 = PF.9 output low. 1 = PF.9 output high.
[9:8]	**OPMODE5**	**I/O Operation Mode** 00 = PF.9 is input only mode. 01 = PF.9 is output push pull mode. 10 = PF.9 is open drain mode. 11 = PF.9 is quasi-bidirectional mode .
[7:6]	**Reserved**	Reserved.
[5:4]	**PUSEL4**	**I/O Pull-up and Pull-down Enable Bits** Determine PF.8 I/O pull-up or pull-down. 00 = PF.8 pull-up and pull-down Disabled. 01 = PF.8 pull-up Enabled. 10 = PF.8 pull-down Enabled.

		11 = PF.8 pull-up and pull-down Disabled. **Note:** Basically, the pull-up control and pull-down control has following behavior limitation. The independent pull-up / pull-down control register is only valid when OPMODE4 is set as input tri-state and open-drain mode.
[3]	**DINOFF4**	**I/O Pin Digital Input Path Disable Bit** 0 = PF.8 digital input path Enabled. 1 = PF.8 digital input path Disabled (digital input tied to low).
[2]	**DOUT4**	**I/O Output Data** 0 = PF.8 output low. 1 = PF.8 output high.
[1:0]	**OPMODE4**	**I/O Operation Mode** 00 = PF.8 is input only mode. 01 = PF.8 is output push pull mode. 10 = PF.8 is open drain mode. 11 = PF.8 is quasi-bidirectional mode.

Note: RTC GPIO will be input mode when not programmed, and user needs to avoid the pin floating problem.

RTC Daylight Saving Time Control Register (RTC_DSTCTL)

Register	Offset	R/W	Description	Reset Value
RTC_DSTCTL	RTC_BA+0x110	R/W	RTC Daylight Saving Time Control Register	0x0000_0000

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

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**DSBAK**	**Daylight Saving Back** 0= Daylight Saving Change is not performed. 1= Daylight Saving Change is performed.
[1]	**SUBHR**	**Subtract 1 Hour** 0 = No effect. 1 = RTC hour digit has been subtracted one hour for winter time change.
[0]	**ADDHR**	**Add 1 Hour** 0 = No effect. 1 = RTC hour digit has been added one hour for summer time change.

RTC Tamper Pin Control Register (RTC_TAMPCTL)

Register	Offset	R/W	Description	Reset Value
RTC_TAMPCTL	RTC_BA+0x120	R/W	RTC Tamper Pin Control Register	0x0000_0000

31	30	29	28	27	26	25	24
DYNPR2EN	TAMP5DEN	TAMP5LV	TAMP5EN	Reserved	TAMP4DEN	TAMP4LV	TAMP4EN
23	22	21	20	19	18	17	16
DYNPR1EN	TAMP3DEN	TAMP3LV	TAMP3EN	Reserved	TAMP2DEN	TAMP2LV	TAMP2EN
15	14	13	12	11	10	9	8
DYNPR0EN	TAMP1DEN	TAMP1LV	TAMP1EN	Reserved	TAMP0DEN	TAMP0LV	TAMP0EN
7	6	5	4	3	2	1	0
DYNRATE			SEEDRLD	DYNSRC	Reserved	DYN2ISS	DYN1ISS

Bits	Description	
[31]	**DYNPR2EN**	**Dynamic Pair 2 Enable Bit** 0 = Static detect. 1 = Dynamic detect.
[30]	**TAMP5DEN**	**Tamper 5 De-bounce Enable Bit** 0 = Tamper 5 de-bounce Disabled. 1 = Tamper 5 de-bounce Enabled, tamper detection pin will sync 1 RTC counter clock.
[29]	**TAMP5LV**	**Tamper 5 Level** This bit depends on level attribute of tamper pin for static tamper detection. 0 = Detect voltage level is low. 1 = Detect voltage level is high.
[28]	**TAMP5EN**	**Tamper 5 Detect Enable Bit** 0 = Tamper 5 detect Disabled. 1 = Tamper 5 detect Enabled. **Note:** The detection reference clock is RTC counter clock. Tamper detector need to sync 2 ~ 3 RTC counter clock.
[27]	**Reserved**	Reserved.
[26]	**TAMP4DEN**	**Tamper 4 De-bounce Enable Bit** 0 = Tamper 4 de-bounce Disabled. 1 = Tamper 4 de-bounce Enabled, tamper detection pin will sync 1 RTC counter clock.
[25]	**TAMP4LV**	**Tamper 4 Level** This bit depends on level attribute of tamper pin for static tamper detection. 0 = Detect voltage level is low. 1 = Detect voltage level is high.
[24]	**TAMP4EN**	**Tamper4 Detect Enable Bit**

		0 = Tamper 4 detect Disabled. 1 = Tamper 4 detect Enabled. **Note:** The detection reference clock is RTC counter clock. Tamper detector need to sync 2 ~ 3 RTC counter clock.
[23]	**DYNPR1EN**	**Dynamic Pair 1 Enable Bit** 0 = Static detect. 1 = Dynamic detect.
[22]	**TAMP3DEN**	**Tamper 3 De-bounce Enable Bit** 0 = Tamper 3 de-bounce Disabled. 1 = Tamper 3 de-bounce Enabled, tamper detection pin will sync 1 RTC counter clock.
[21]	**TAMP3LV**	**Tamper 3 Level** This bit depends on level attribute of tamper pin for static tamper detection. 0 = Detect voltage level is low. 1 = Detect voltage level is high.
[20]	**TAMP3EN**	**Tamper 3 Detect Enable Bit** 0 = Tamper 3 detect Disabled. 1 = Tamper 3 detect Enabled. **Note:** The detection reference clock is RTC counter clock. Tamper detector need to sync 2 ~ 3 RTC counter clock.
[19]	**Reserved**	Reserved.
[18]	**TAMP2DEN**	**Tamper 2 De-bounce Enable Bit** 0 = Tamper 2 de-bounce Disabled. 1 = Tamper 2 de-bounce Enabled, tamper detection pin will sync 1 RTC counter clock.
[17]	**TAMP2LV**	**Tamper 2 Level** This bit depends on level attribute of tamper pin for static tamper detection. 0 = Detect voltage level is low. 1 = Detect voltage level is high.
[16]	**TAMP2EN**	**Tamper 2 Detect Enable Bit** 0 = Tamper 2 detect Disabled. 1 = Tamper 2 detect Enabled. **Note:** The detection reference clock is RTC counter clock. Tamper detector need to sync 2 ~ 3 RTC counter clock.
[15]	**DYNPR0EN**	**Dynamic Pair 0 Enable Bit** 0 = Static detect. 1 = Dynamic detect.
[14]	**TAMP1DEN**	**Tamper 1 De-bounce Enable Bit** 0 = Tamper 1 de-bounce Disabled. 1 = Tamper 1 de-bounce Enabled, tamper detection pin will sync 1 RTC counter clock.
[13]	**TAMP1LV**	**Tamper 1 Level** This bit depends on level attribute of tamper pin for static tamper detection. 0 = Detect voltage level is low. 1 = Detect voltage level is high.
[12]	**TAMP1EN**	**Tamper 1 Detect Enable Bit** 0 = Tamper 1 detect Disabled.

		1 = Tamper 1 detect Enabled. **Note:** The detection reference clock is RTC counter clock. Tamper detector needs to sync 2 ~ 3 RTC counter clock.
[11]	**Reserved**	Reserved.
[10]	**TAMP0DEN**	**Tamper 0 De-bounce Enable Bit** 0 = Tamper 0 de-bounce Disabled. 1 = Tamper 0 de-bounce Enabled, tamper detection pin will sync 1 RTC counter clock.
[9]	**TAMP0LV**	**Tamper 0 Level** This bit depends on level attribute of tamper pin for static tamper detection. 0 = Detect voltage level is low. 1 = Detect voltage level is high.
[8]	**TAMP0EN**	**Tamper0 Detect Enable Bit** 0 = Tamper 0 detect Disabled. 1 = Tamper 0 detect Enabled. **Note:** The detection reference clock is RTC counter clock. Tamper detector needs to sync 2 ~ 3 RTC counter clock.
[7:5]	**DYNRATE**	**Dynamic Change Rate** This item is choice the dynamic tamper output change rate. 000 = 2^{10} * RTC_CLK. 001 = 2^{11} * RTC_CLK. 010 = 2^{12} * RTC_CLK. 011 = 2^{13} * RTC_CLK. 100 = 2^{14} * RTC_CLK. 101 = 2^{15} * RTC_CLK. 110 = 2^{16} * RTC_CLK. 111 = 2^{17} * RTC_CLK. **Note:** After revising this field, set SEEDRLD (RTC_TAMPCTL[4]) can reload change rate immediately.
[4]	**SEEDRLD**	**Reload New Seed for PRNG Engine** Setting this bit, the tamper configuration will be reloaded. 0 = Generating key based on the current seed. 1 = Reload new seed. **Note 1:** Before this bit is set, the tamper configuration should be set to complete and this bit will be auto clear to 0 after reload new seed completed. **Note 2:** The detection reference clock is RTC counter clock. Tamper detector needs to sync 2 ~ 3 RTC counter clock.
[3]	**DYNSRC**	**Dynamic Reference Pattern** This fields determine the new reference pattern when current pattern run out in dynamic pair mode. 0 = The new reference pattern is generated by random number generator when the reference pattern run out. 1 = The new reference pattern is repeated from SEED (RTC_TAMPSEED[31:0]) when the reference pattern run out. **Note:** After this bit is modified, the SEEDRLD (RTC_TAMPCTL[4]) should be set.
[2]	**Reserved**	Reserved.
[1]	**DYN2ISS**	**Dynamic Pair 2 Input Source Select** This bit determine Tamper 5 input is from Tamper 4 or Tamper 0 in dynamic mode. 0 = Tamper input is from Tamper 4.

		1 = Tamper input is from Tamper 0. **Note:** This bit has effect only when DYNPR2EN (RTC_TAMPCTL[24]) and DYNPR0EN (RTC_TAMPCTL[15]) are set.
[0]	**DYN1ISS**	**Dynamic Pair 1 Input Source Select** This bit determine Tamper 3 input is from Tamper 2 or Tamper 0 in dynamic mode. 0 = Tamper input is from Tamper 2. 1 = Tamper input is from Tamper 0. **Note:** This bit is effective only when DYNPR1EN (RTC_TAMPCTL[16]) and DYNPR0EN (RTC_TAMPCTL[15]) are set.

RTC Tamper Dynamic Seed Register (RTC_TAMPSEED)

Register	Offset	R/W	Description	Reset Value
RTC_TAMPSEED	RTC_BA+0x128	R/W	RTC Tamper Dynamic Seed Register	0x0000_0000

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

Bits	Description	
[31:0]	**SEED**	Seed Value

RTC Tamper Time Register (RTC_TAMPTIME)

Register	Offset	R/W	Description	Reset Value
RTC_TAMPTIME	RTC_BA+0x130	R	RTC Tamper Time Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved		**TENHR**		**HR**			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved	**TENMIN**			**MIN**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**TENSEC**			**SEC**			

Bits	Description	
[31:22]	**Reserved**	Reserved.
[21:20]	**TENHR**	**10-hour Time Digit of Tamper Time (0~2)** **Note:** 24-hour time scale only.
[19:16]	**HR**	1-Hour Time Digit of Tamper Time (0~9)
[15]	**Reserved**	Reserved.
[14:12]	**TENMIN**	10-Min Time Digit of Tamper Time (0~5)
[11:8]	**MIN**	1-Min Time Digit of Tamper Time (0~9)
[7]	**Reserved**	Reserved.
[6:4]	**TENSEC**	10-Sec Time Digit of Tamper Time (0~5)
[3:0]	**SEC**	1-Sec Time Digit of Tamper Time (0~9)

Note:

1. RTC_TAMPTIME is a BCD digit counter.
2. The reasonable value range is listed in the parenthesis.
3. This fields cannot be updated until all TAMPxIF are cleared.

RTC Tamper Calendar Register (RTC_TAMPCAL)

Register	Offset	R/W	Description	Reset Value
RTC_TAMPCAL	RTC_BA+0x134	R	RTC Tamper Calendar Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TENYEAR				**YEAR**			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved			**TENMON**	**MON**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**TENDAY**		**DAY**			

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:20]	**TENYEAR**	10-Year Calendar Digit of Tamper Calendar (0~9)
[19:16]	**YEAR**	1-Year Calendar Digit of Tamper Calendar (0~9)
[15:13]	**Reserved**	Reserved.
[12]	**TENMON**	10-Month Calendar Digit of Tamper Calendar (0~1)
[11:8]	**MON**	1-Month Calendar Digit of Tamper Calendar (0~9)
[7:6]	**Reserved**	Reserved.
[5:4]	**TENDAY**	10-Day Calendar Digit of Tamper Calendar (0~3)
[3:0]	**DAY**	1-Day Calendar Digit of Tamper Calendar (0~9)

Note:

1. RTC_TAMPCAL is a BCD digit counter.
2. The reasonable value range is listed in the parenthesis.
3. This fields cannot be updated until all TAMPxIF are cleared

RTC Clock Fail Detector Control Register (RTC_CLKDCTL)

Register	Offset	R/W	Description	Reset Value
RTC_CLKDCTL	RTC_BA+0x140	R/W	Clock Fail Detector Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						LXTSLOWF	SWLIRCF
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					LXTSTSW	LXTFSW	LXTFDEN

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**LXTSLOWF**	**LXT Slower Than LIRC32K Flag (Read Only)** 0 = LXT frequency faster than LIRC32K. 1 = LXT frequency is slowly. **Note:** LXTSLOWF is valid during CLKSTIF (RTC_INTSTS[25]) or CLKFIF (RTC_INTSTS[24]) rising.
[16]	**SWLIRCF**	**LXT Clock Detector Fail/Stop Switch LIRC32K Flag (Read Only)** 0 = Indicate RTC clock source from LXT. 1 = Indicate RTC clock source from LIRC32K.
[15:3]	**Reserved**	Reserved.
[2]	**LXTSTSW**	**LXT Clock Stop Detector Switch LIRC32K Enable Bit** 0 = LXT clock Stop switch LIRC32K Disabled. 1 = LXT clock Stop detector rise, RTC clock source switch from LIRC32K.
[1]	**LXTFSW**	**LXT Clock Fail Detector Switch LIRC32K Enable Bit** 0 = LXT clock Fail switch LIRC32K Disabled. 1 = LXT clock Fail detector rises, and RTC clock source switch from LIRC32K.
[0]	**LXTFDEN**	**LXT Clock Fail/Stop Detector Enable Bit** 0 = LXT clock Fail/Stop detector Disabled. 1 = LXT clock Fail/Stop detector Enabled. **Note:** LXT detector will automatically be disabled when CLKSTIF/CLKFIF flag rises, and resumes after Fail/Stop Flag is cleared.

RTC Clock Frequency Detector Boundary Register (RTC_CDBR)

Register	Offset	R/W	Description	Reset Value
RTC_CDBR	RTC_BA+0x144	R/W	Clock Frequency Detector Boundary Register	0x00F0_000F

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

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:16]	**FAILBD**	**LXT Clock Frequency Detector Fail Boundary** The bits define the fail value of frequency monitor window. When LXT frequency monitor counter lower than FAILBD, the LXT frequency detect fail interrupt flag will set to 1. **Note:** The boundary is defined as the minimum value of LXT among 256 LIRC32K clock time.
[15:8]	**Reserved**	Reserved.
[7:0]	**STOPBD**	**LXT Clock Stop Frequency Detector Stop Boundary** The bits define the stop value of frequency monitor window. When LXT frequency monitor counter lower than STOPBD, the LXT frequency detect Stop interrupt flag will set to 1. **Note:** The boundary is defined as the maximum value of LXT among 256 LIRC32K clock time.

RTC Test Control Register (RTC_TEST)

Register	Offset	R/W	Description	Reset Value
RTC_TEST	RTC_BA+0x1F0	R/W	RTC Test Control Register	0x0000_0000

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[4]	**BATDETEN**	**Battery Detect Enable Bit** 0 = Battery power detect disable. 1 = Battery power detect enable.
[3:0]	**Reserved**	Reserved.

6.15 EPWM Generator and Capture Timer (EPWM)

6.15.1 Overview

The chip provides two EPWM generators — EPWM0 and EPWM1. Each EPWM supports 6 channels of EPWM output or input capture. There is a 12-bit prescaler to support flexible clock to the 16-bit EPWM counter with 16-bit comparator. The EPWM counter supports up, down and up-down counter types. EPWM uses comparator compared with counter to generate events. These events are used to generate EPWM pulse, interrupt and trigger signal for EADC/DAC to start conversion.

The EPWM generator supports two standard EPWM output modes: Independent mode and Complementary mode, they have different architecture. There are two output functions based on standard output modes: Group function and Synchronous function. Group function can be enabled under Independent mode or complementary mode. Synchronous function only enabled under complementary mode. Complementary mode has two comparators to generate various EPWM pulse with 12-bit dead-time generator and another free trigger comparator to generate trigger signal for EADC. For EPWM output control unit, it supports polarity output, independent pin mask and brake functions.

The EPWM generator also supports input capture function. It supports latch EPWM counter value to corresponding register when input channel has a rising transition, falling transition or both transition is happened. Capture function also support PDMA to transfer captured data to memory.

6.15.2 Features

6.15.2.1 EPWM Function Features

- Supports maximum clock frequency up to maximum PLL frequency
- Supports up to two EPWM modules, each module provides 6 output channels
- Supports independent mode for EPWM output/Capture input channel
- Supports complementary mode for 3 complementary paired EPWM output channel
  - Dead-time insertion with 12-bit resolution
  - Synchronous function for phase control
  - Two compared values during one period
- Supports 12-bit prescaler from 1 to 4096
- Supports 16-bit resolution EPWM counter
  - Up, down and up/down counter operation type
- Supports one-shot or auto-reload counter operation mode
- Supports group function
- Supports synchronous function
- Supports mask function and tri-state enable for each EPWM pin
- Supports brake function
  - Brake source from pin, analog comparator and system safety events (clock failed, SRAM parity error, Brown-out detection and CPU lockup).
  - Noise filter for brake source from pin
  - Leading edge blanking (LEB) function for brake source from analog comparator
  - Edge detect brake source to control brake state until brake interrupt cleared

- Level detect brake source to auto recover function after brake condition removed

- Supports interrupt on the following events:
  - EPWM counter matches 0, period value or compared value
  - Brake condition happened
- Supports trigger EADC/DAC on the following events:
  - EPWM counter matches 0, period value or compared value
  - EPWM counter match free trigger comparator compared value (only for EADC)
  - Supports EPWM trigger EADC event prescaler feature
- Supports EPWM output accumulator stop counter mode
- Supports Fault Detect Function.

6.15.2.2 Capture Function Features

- Supports up to 12 capture input channels with 16-bit resolution
- Supports rising or falling capture condition
- Supports input rising/falling capture interrupt
- Supports rising/falling capture with counter reload option
- Supports PDMA transfer function for EPWM all channels

6.15.3 Block Diagram

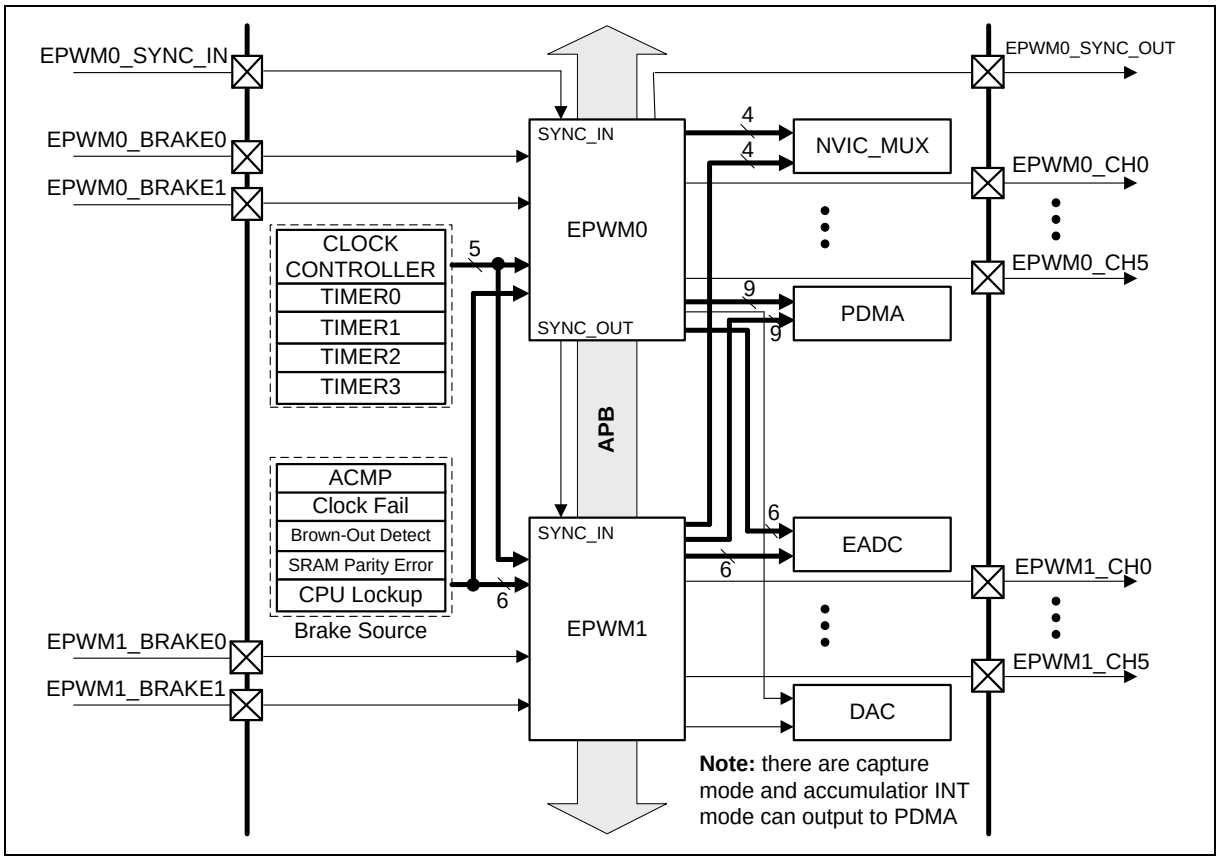


Figure 6.15-1 EPWM Generator Overview Block Diagram

EPWM Clock frequency can be set equal to PCLK frequency as Figure 6.15-2. For the detailed register setting, please refer to Table 6.15-1. Each EPWM generator has three clock source inputs, each clock source can be selected from EPWM Clock or four TIMER trigger EPWM outputs as Figure 6.15-3 by ECLKSRC0 (EPWM_CLKSRC[2:0]) for EPWM_CLK0, ECLKSRC2 (EPWM_CLKSRC[10:8]) for EPWM_CLK2 and ECLKSRC4 (EPWM_CLKSRC[18:16]) for EPWM_CLK4. If the clock source of EPWM counter is selected from TIMERn interrupt events, the TRGPWM (TIMERn_TRGCTL[1], n=0,1..3) bit must be set as 1.

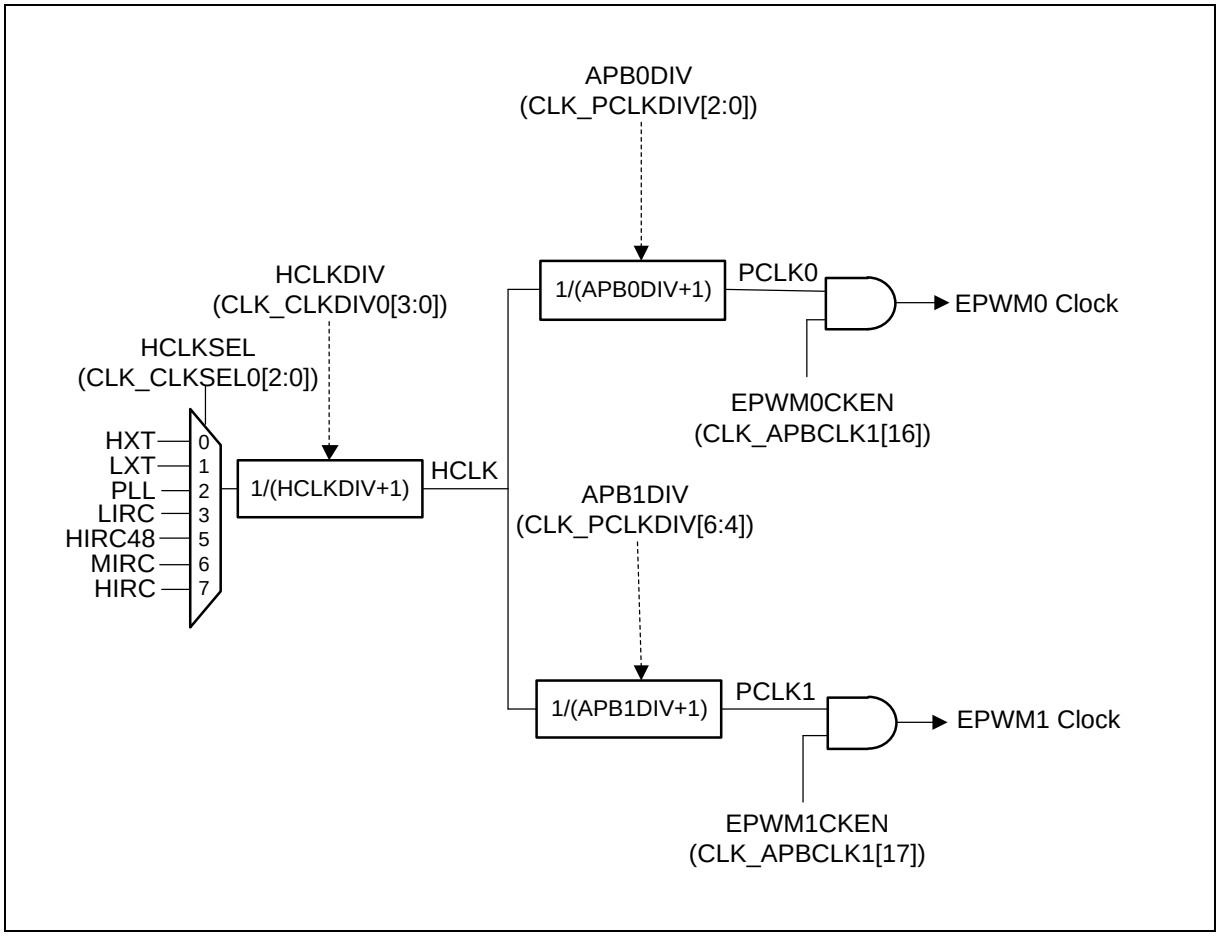


Figure 6.15-2 EPWM Clock Source Control

Frequency Ratio PCLK:EPWM Clock	HCLK	PCLK	EPWM Clock	HCLKSEL CLK_CLKSEL0[2:0]	HCLKDIV CLK_CLKDIV0[3:0]	APBnDIV (CLK_PCLKDIVn [2+4n:4n]), N Denotes 0 Or 1	EPWMnSEL (CLK_CLKSEL2[N]), N Denotes 0 Or 1
1:1	HCLK	PCLK	PCLK	Don't care	Don't care	Don't care	1

Table 6.15-1 EPWM Clock Source Control Registers Setting Table

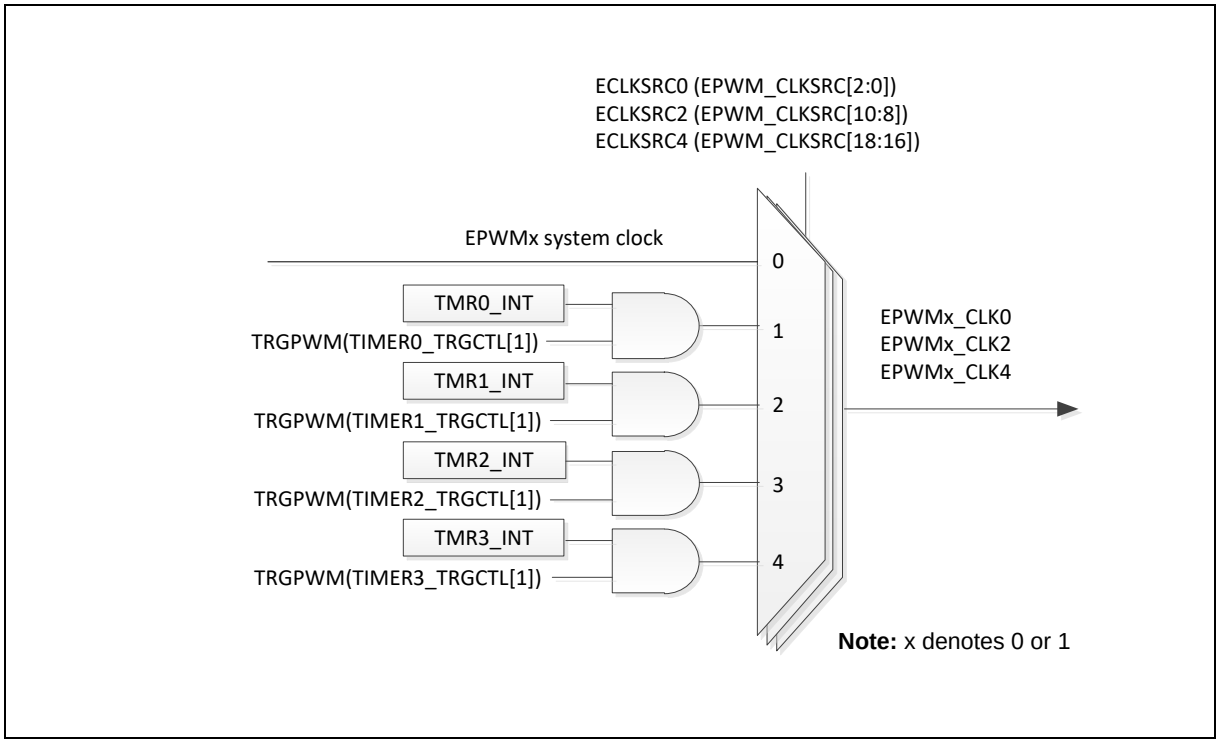


Figure 6.15-3 EPWM Clock Source Control

Figure 6.15-4 and Figure 6.15-5 illustrate the architecture of EPWM independent mode and complementary mode. No matter independent mode or complementary mode, paired channels' (EPWM_CH0 and EPWM_CH1, EPWM_CH2 and EPWM_CH3, EPWM_CH4 and EPWM_CH5) counters both come from the same clock source and prescaler. When counter count to 0, PERIOD (EPWM_PERIODn[15:0]) or equal to comparator, events will be generated. These events are passed to corresponding generators to generate EPWM pulse, interrupt signal and trigger signal for EADC/DAC to start conversion. Output control is used to change EPWM pulse output state; brake function in output control also generates interrupt events. In complementary mode, synchronize function is available and even channel use odd channel comparator to generate events, free trigger comparator events only used to generate trigger EADC signals.

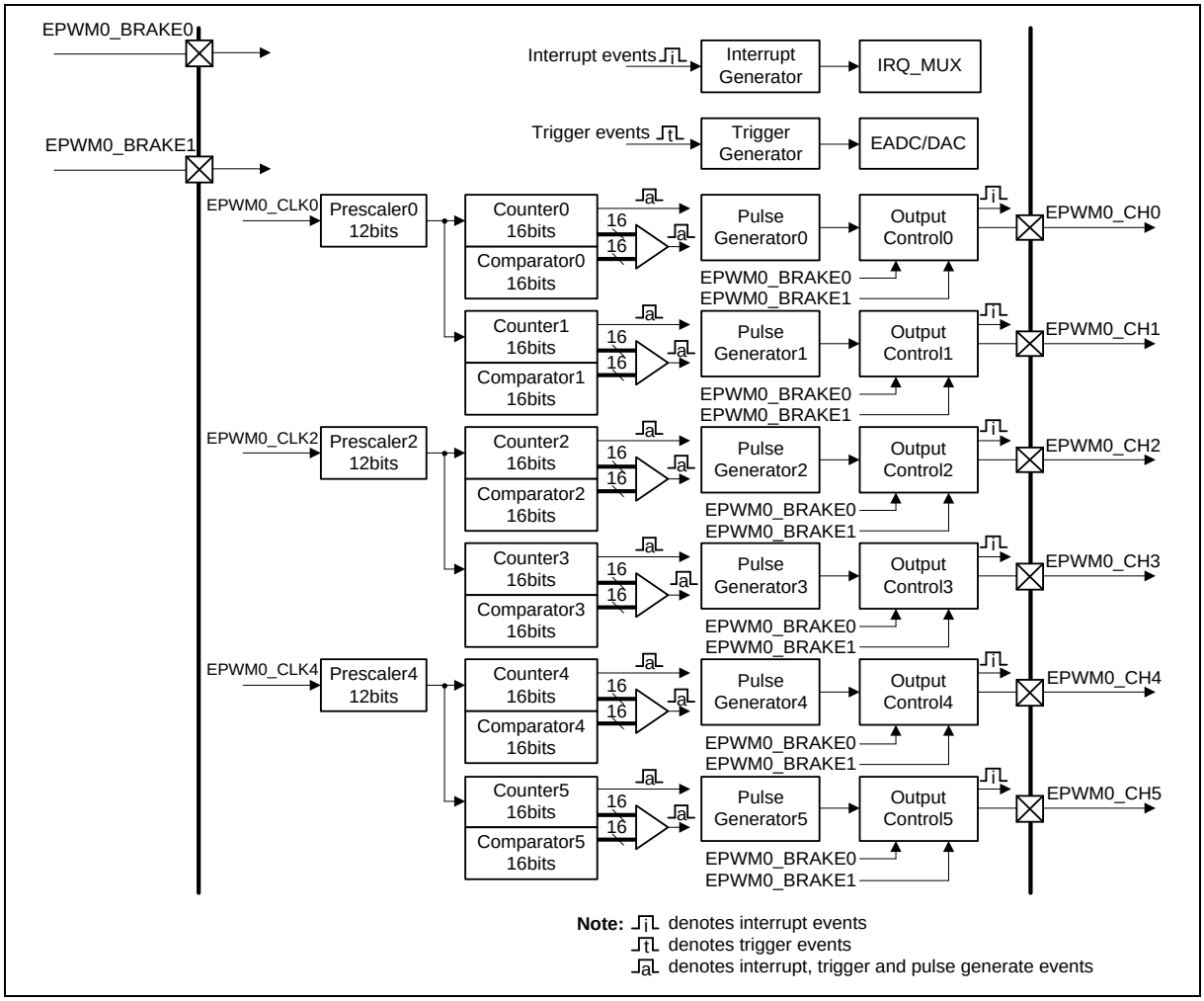


Figure 6.15-4 EPWM Independent Mode Architecture Diagram

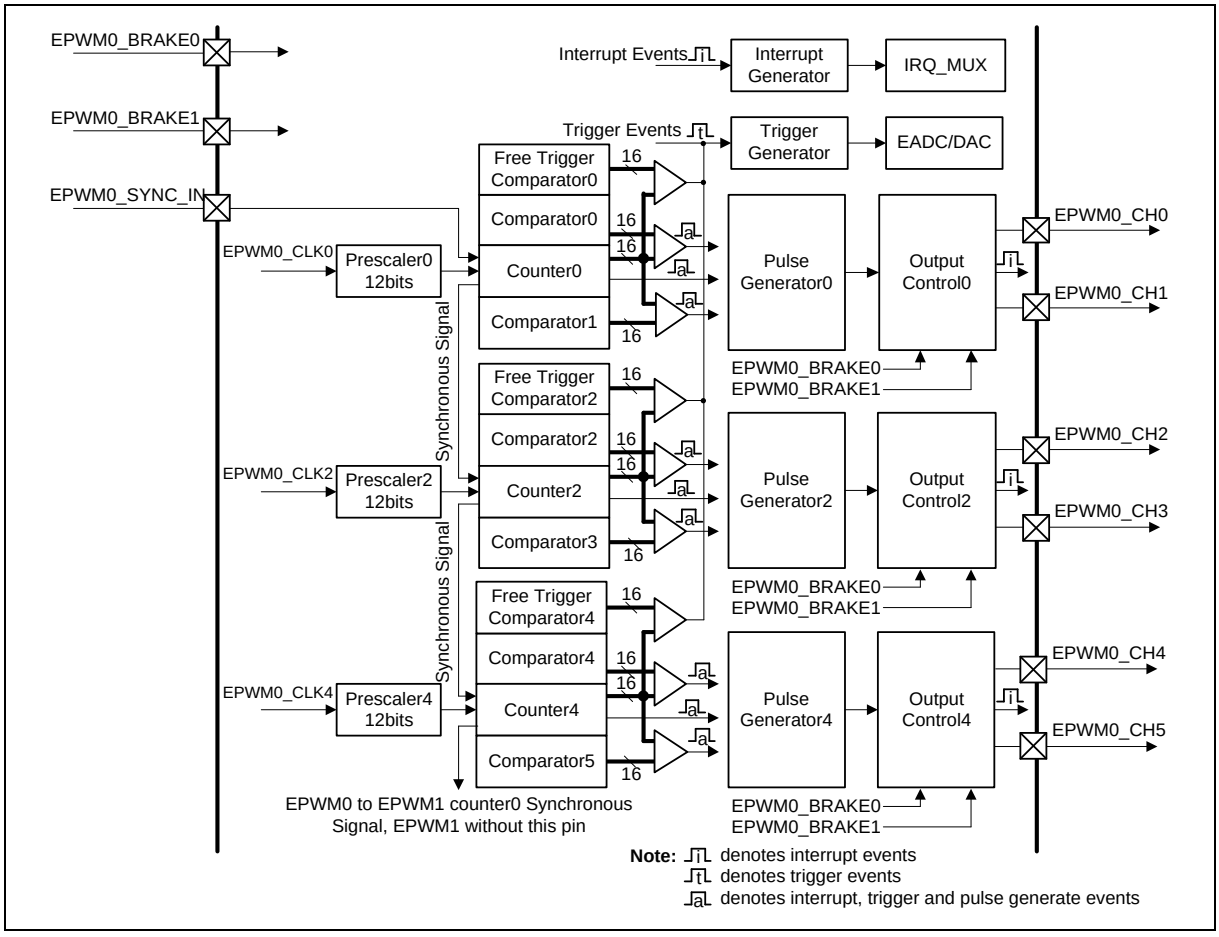


Figure 6.15-5 EPWM Complementary Mode Architecture Diagram

6.15.4 Basic Configuration

6.15.4.1 EPWM0 Basic Configuration

- Clock Source Configuration
  - Select the source of EPWM0 peripheral clock on EPWM0SEL (CLK_CLKSEL2[0])
  - Enable EPWM0 peripheral clock in EPWM0CKEN (CLK_APBCLK1[16])
- Reset Configuration
  - Reset EPWM0 in EPWM0RST (SYS_IPRST2[16])
- Pin Configuration

Group	Pin Name	GPIO	MFP
EPWM0	EPWM0_BRAKE0	PE.8	MFP11
		PB.1	MFP13
	EPWM0_BRAKE1	PE.9	MFP11
		PB.0	MFP13

	EPWM0_CH0	PE.8	MFP10
		PB.5	MFP11
		PE.7	MFP12
		PA.5	MFP13
	EPWM0_CH1	PE.9	MFP10
		PB.4	MFP11
		PE.6	MFP12
		PA.4	MFP13
	EPWM0_CH2	PE.10	MFP10
		PB.3	MFP11
		PE.5	MFP12
		PA.3	MFP13
	EPWM0_CH3	PE.11	MFP10
		PB.2	MFP11
		PE.4	MFP12
		PA.2	MFP13
	EPWM0_CH4	PE.12	MFP10
		PB.1, PD.14	MFP11
		PE.3	MFP12
		PA.1	MFP13
	EPWM0_CH5	PE.13	MFP10
		PB.0, PH.11	MFP11
		PE.2	MFP12
		PA.0	MFP13
	EPWM0_SYNC_IN	PA.15	MFP12
	EPWM0_SYNC_OUT	PF.5	MFP9
		PA.11	MFP10

6.15.4.2 EPWM1 Basic Configuration

- Clock Source Configuration
  - Select the source of EPWM1 peripheral clock on EPWM1SEL (CLK_CLKSEL2[1])
  - Enable EPWM1 peripheral clock in EPWM1CKEN (CLK_APBCLK1[17])
- Reset Configuration
  - Reset EPWM1 in EPWM1RST (SYS_IPRST2[17])
- Pin Configuration

Group	Pin Name	GPIO	MFP
EPWM1	EPWM1_BRAKE0	PB.7, PE.10	MFP11
	EPWM1_BRAKE1	PB.6, PE.11	MFP11
	EPWM1_CH0	PB.15, PE.13	MFP11
		PC.5, PC.12	MFP12
	EPWM1_CH1	PB.14, PC.8	MFP11
		PC.4, PC.11	MFP12
	EPWM1_CH2	PB.13, PC.7	MFP11
		PC.3, PC.10	MFP12
	EPWM1_CH3	PB.12, PC.6	MFP11
		PC.2, PC.9	MFP12
	EPWM1_CH4	PA.7	MFP11
		PB.1, PB.7, PC.1	MFP12
	EPWM1_CH5	PA.6	MFP11
		PB.0, PB.6, PC.0	MFP12

6.15.5 Functional Description

6.15.5.1 EPWM Prescaler

The EPWM prescaler is used to divide clock source, prescaler counting CLKPSC +1 times, EPWM counter only count once. The prescale double buffer is setting by CLKPSC (EPWM_CLKPSCn_m[11:0], n = 0, 2, 4, m = 1, 3, 5) bits. Figure 6.15-6 is an example of EPWM channel 0 prescale waveform. The prescale counter will reload CLKPSC at the begin of the next prescale counter down-count.

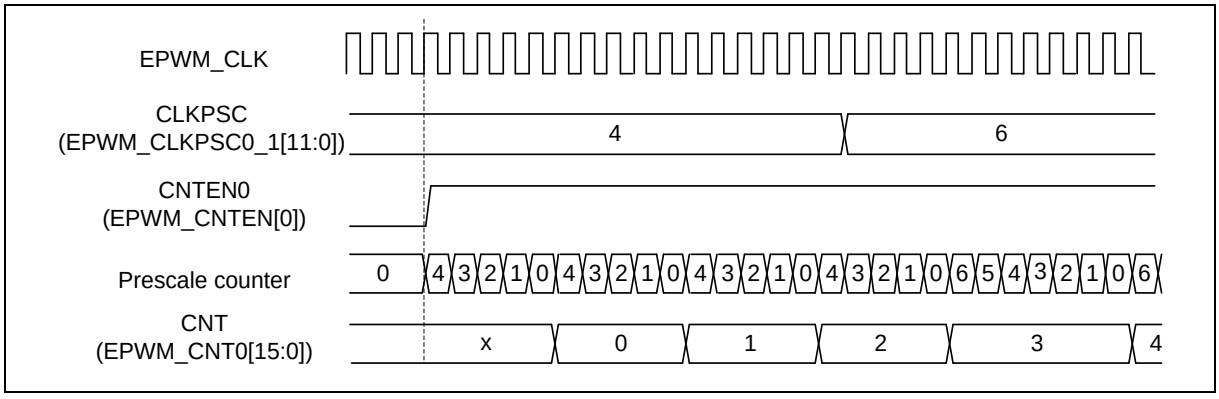


Figure 6.15-6 EPWM_CH0 Prescaler Waveform in Up Counter Type

6.15.5.2 EPWM Counter

The EPWM supports 3 counter types operation: Up Counter, Down Counter and Up-Down Counter types.

For EPWM channel0, CNT(EPWM_CNT0[15:0]) can clear to 0x00 by CNTCLR0 (EPWM_CNTCLR[0]). CNT will be cleared when prescale counter count to 0, and CNTCLR0 will be set 0 by hardware

automatically.

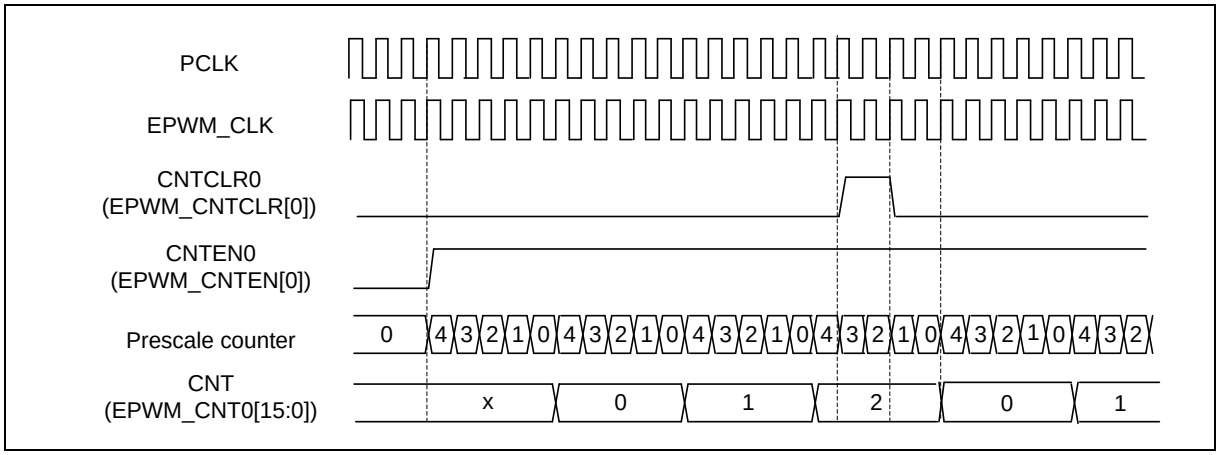


Figure 6.15-7 EPWM Counter Waveform when Setting Clear Counter

6.15.5.3 Up Counter Type

When EPWM counter is set to up counter type, CNTTYPEn (EPWM_CTL1[2n+1:2n], n = 0,1..5) is 0x0, it starts up-counting from 0 to PERIOD (EPWM_PERIODn[15:0], where n denotes channel number) to complete a EPWM period. The current counter value can be read from CNT (EPWM_CNTn[15:0]) bits. EPWM generates zero point event when the counter counts to 0 and prescale counts to 0. EPWM generates period point event when the counter counts to PERIOD and prescale counts to 0. Figure 6.15-8 shows an example of up counter, wherein

EPWM period time = (PERIOD+1) *(CLKPSC+1)* EPWMx_CLK.

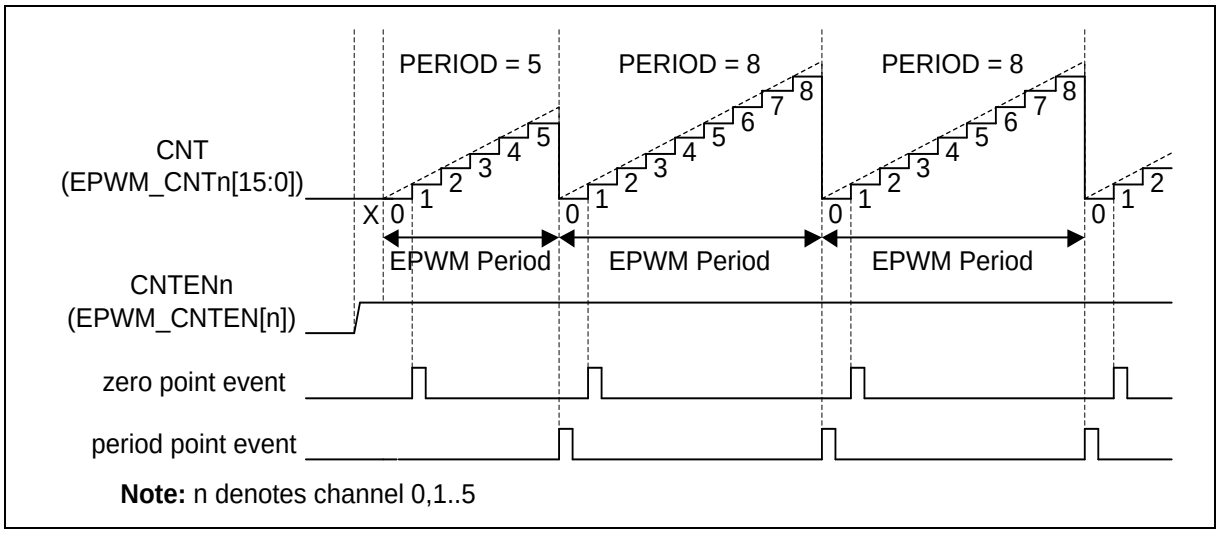


Figure 6.15-8 EPWM Up Counter Type

6.15.5.4 Down Counter Type

When EPWM counter is set to down counter type, CNTTYPEn (EPWM_CTL1[2n+1:2n], n = 0,1..5) is

0x1, it starts down-counting from PERIOD to 0 to complete a EPWM period. The current counter value can be read from CNT (EPWM_CNTn[15:0]) bits. EPWM generates zero point event when the counter counts to 0 and prescale counts to 0. EPWM generates period point event when the counter counts to PERIOD and prescale counts to 0. Figure 6.15-9 shows an example of down counter, wherein

EPWM period time = (PERIOD+1) * (CLKPSC+1) * EPWMx_CLK.

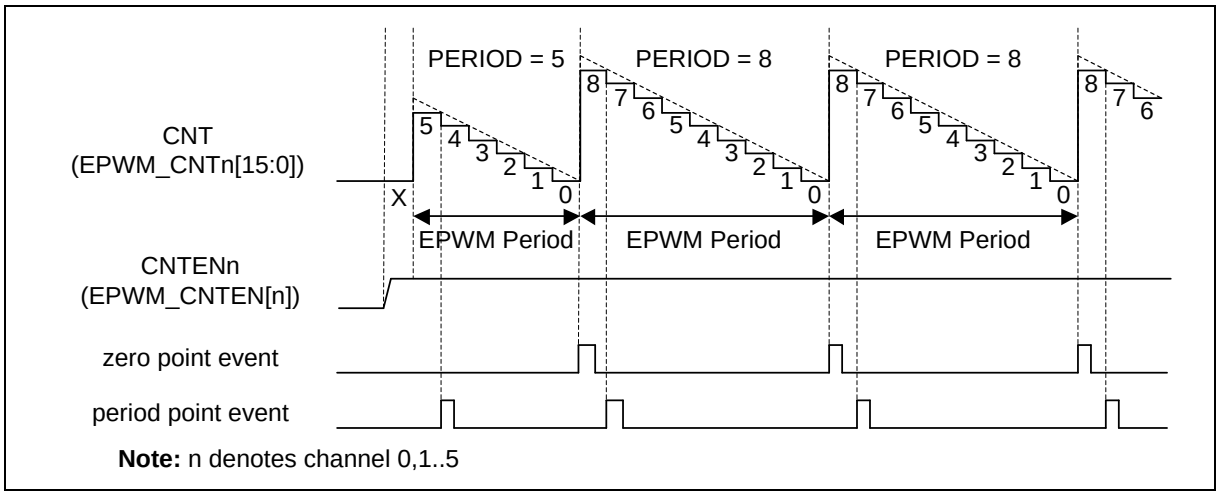


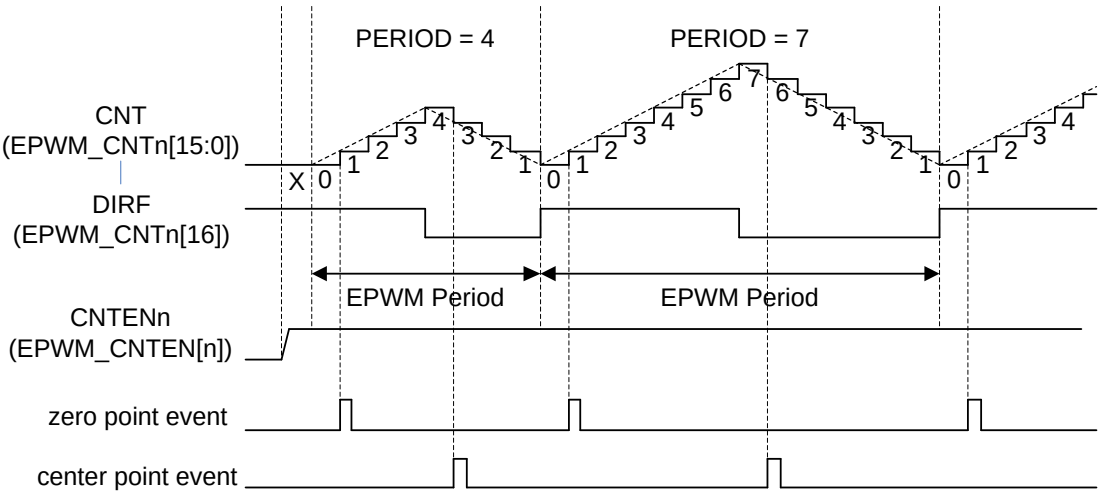
Figure 6.15-9 EPWM Down Counter Type

6.15.5.5 Up-Down Counter Type

When EPWM counter is set to up-down count type, CNTTYPEn (EPWM_CTL1[2n+1:2n], n = 0,1..5) is 0x2, it starts counting-up from 0 to PERIOD and then starts counting down to 0 to complete a EPWM period. The current counter value can be read from CNT (EPWM_CNTn[15:0]) bits. EPWM generates zero point event when the counter counts to 0 and prescale counts to 0. EPWM generates center point event which is equal to period point event when the counter counts to PERIOD. Figure 6.15-10 shows an example of up-down counter, wherein

EPWM period time =(2*PERIOD) * (CLKPSC+1) * EPWMx_CLK.

The DIRF (EPWM_CNTn[16]) bit is counter direction indicator flag, where high is up counting, and low is down counting.



Note1: When in up-down count type, period interrupt flag occurs at center point event.
Note2: n denotes channel 0,1..5

Figure 6.15-10 EPWM Up-Down Counter Type

6.15.5.6 EPWM Comparator

There are two kinds of comparator registers : one is EPWM_CMPDATn(n = 0,1..5), and the other is EPWM_FTCMPDATn_m(n = 0,2,4, m = 1,3,5) register. EPWM_CMPDATn is a basic comparator register of EPWM channel n; In Independent mode each channel only has one comparator, the value of EPWM_CMPDATn register is continuously compared to the corresponding channel's counter value. In Complementary mode each paired channels has two comparators, and the value of EPWM_CMPDATn and EPWM_CMPDATm (n = 0,2,4, m = 1,3,5) registers are continuously compared to the complementary even channel's counter value, because of odd channel's counter is useless. For example, channel 0 and channel 1 are complementary channels, in Complementary mode, channel 1's comparator is continuously compared to channel 0's counter, but not channel 1's. When the counter is equal to value of EPWM_CMPDAT0 register, EPWM generates a compared point event and uses the event to generate EPWM pulse, interrupt or used to trigger EADC/DAC. In up-down counter type, two events will be generated in a EPWM period as shown in Figure 6.15-11. The CMPU is up count compared point event and CMPD is down count compared point event.

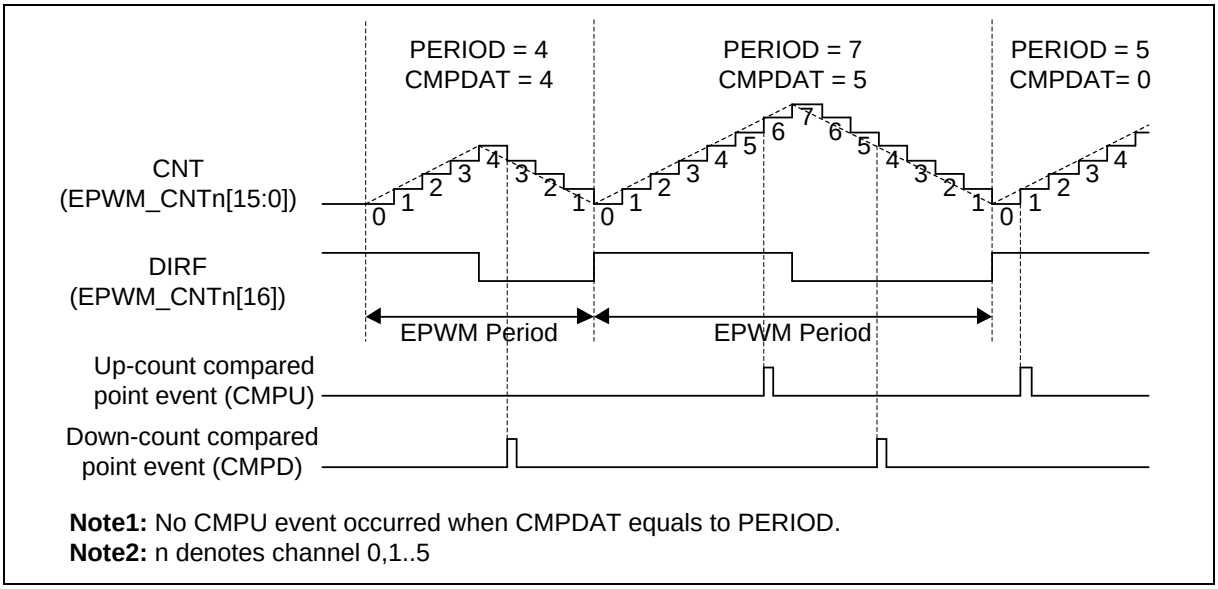


Figure 6.15-11 EPWM Compared point Events in Up-Down Counter Type

EPWM_FTCMPDATn_m is a free trigger comparator register. Each complementary paired channel only supports one free trigger comparator. The value of EPWM_FTCMPDATn_m(n = 0,2,4, m = 1,3,5) register is continuously compared to even channel's counter value. When counter is equal to the value of EPWM_FTCMPDATn_m register, FTCMDn (EPWM_FTCI[10:8], n=0,2,4) indicator is set in down count type and FTCMUn (EPWM_FTCI[2:0], n=0,2,4) indicator is set in up count type. In addition, EPWM generates an event and only uses to trigger EADC.

6.15.5.7 EPWM Double Buffering

The double buffering uses double buffers to separate software writing and hardware action operation timing. There are four loading modes for loading values to buffer: period loading mode, immediately loading mode, window loading mode and center loading mode. After registers are modified through software, hardware will load register value to the buffer register according to the loading mode timing. The hardware action is based on the buffer value. This can prevent asynchronously operation problem due to software and hardware asynchronism.

The EPWM provides PBUF (EPWM_PBUFn[15:0]) as the active EPWM_PERIODn buffer register, CMPBUF (EPWM_CMPBUFn[15:0]) as the active EPWM_CMPDATn buffer register, FTCMPBUF (EPWM_FTCBUFn_m[15:0]) as the active EPWM_FTCMPDATn_m buffer register and CPSCBUF (EPWM_CPSCBUFn_m[15:0]) as the active EPWM_CLKPSCn_m buffer register. The concept of double buffering is used in loading modes, which are described in the following sections. For example, as shown Figure 6.15-12, in period loading mode, writing PERIOD (EPWM_PERIODn[15:0]), CMP (EPWM_CMPDATn[15:0]) and FTCMP (EPWM_FTCMPDATn_m[15:0]) through software, EPWM will load new values to their buffer PBUF (EPWM_PBUFn[15:0]), CMPBUF (EPWM_CMPBUFn[15:0]) and FTCMPBUF (EPWM_FTCBUFn_m[15:0]) at start of the next period without affecting the current period counter operation. FTCMPU denotes up-count free trigger compared point event and FTCMPD denotes down-count free trigger compared event.

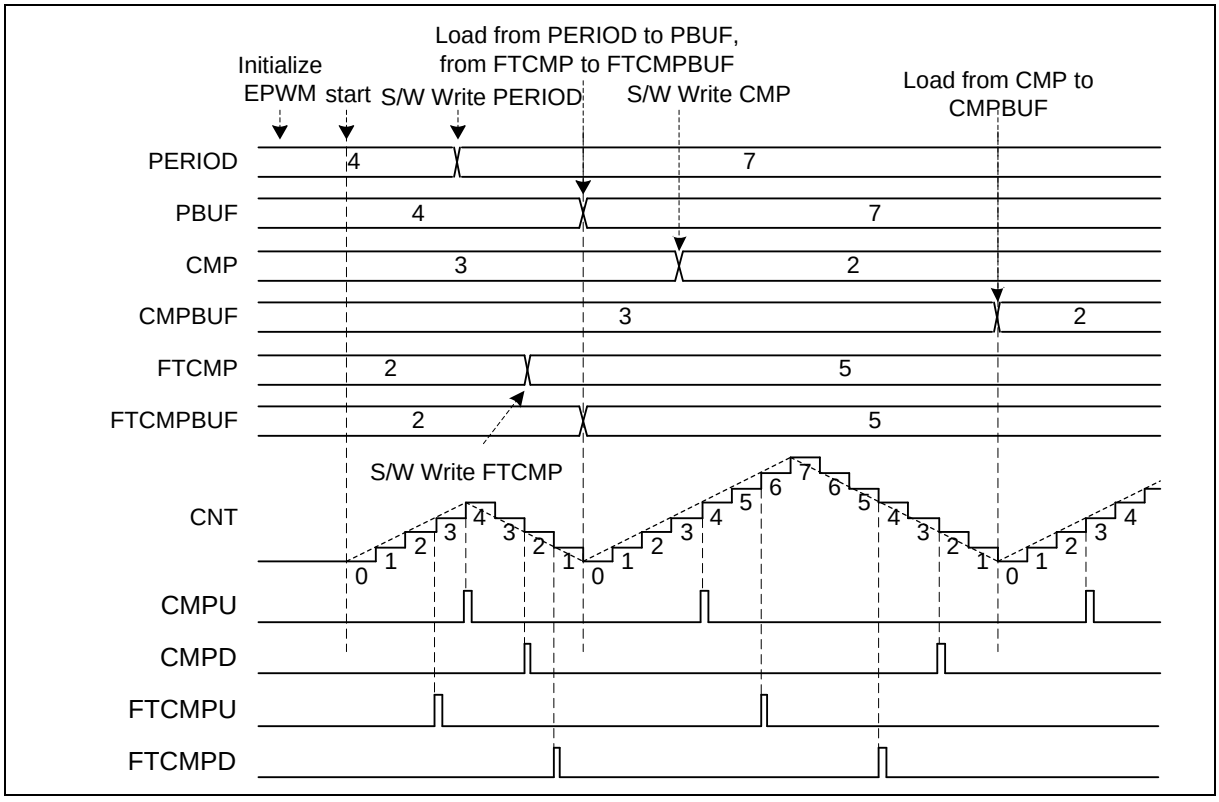


Figure 6.15-12 EPWM Double Buffering Illustration

6.15.5.8 Period Loading Mode

When the immediately loading mode, window loading mode and center loading mode are disabled that IMMLDENn bits, WINLDENn bits and CTRLDn bits of EPWM_CTL0 register are set to 0, EPWM operates in period Loading mode. In period Loading mode, CLKPSC(EPWM_CLKPSCn_m[11:0]), PERIOD(EPWM_PERIODn[15:0]) and CMP(EPWM_CMPDATn[15:0]) will all be loaded to their active CPSCBUF(EPWM_CPSCBUFn_m[11:0]), PBUF(EPWM_PBUFn[15:0]), CMPBUF (EPWM_CMPBUFn[15:0]) and FTCMPBUF(EPWM_FTCBUFn_m[15:0]) buffers while each period is completed. For example, after EPWM counter up counts from 0 to PERIOD in the up-counter operation or down counts from PERIOD to 0 in the down-counter operation or counts up from 0 to PERIOD and then counts down to 0 in the up-down counter operation.

Figure 6.15-13 shows period loading timing of up-count operation, where PERIOD DATA0 denotes the initial data of PERIOD, PERIOD DATA1 denotes the first updated PERIOD data by software and so on. CMP also follows this rule. The following describes steps sequence of Figure 6.15-13. User can know the PERIOD and CMP update condition, by watching EPWM period and CMPU event.

1. Software writes CMP DATA1 to CMP at point 1.
2. Hardware loads CMP DATA1 to CMPBUF at the end of EPWM period at point 2.
3. Software writes PERIOD DATA1 to PERIOD at point 3.
4. Hardware loads PERIOD DATA1 to PBUF at the end of EPWM period at point 4.
5. Software writes PERIOD DATA2 to PERIOD at point 5.
6. Hardware loads PERIOD DATA2 to PBUF at the end of EPWM period at point 6.

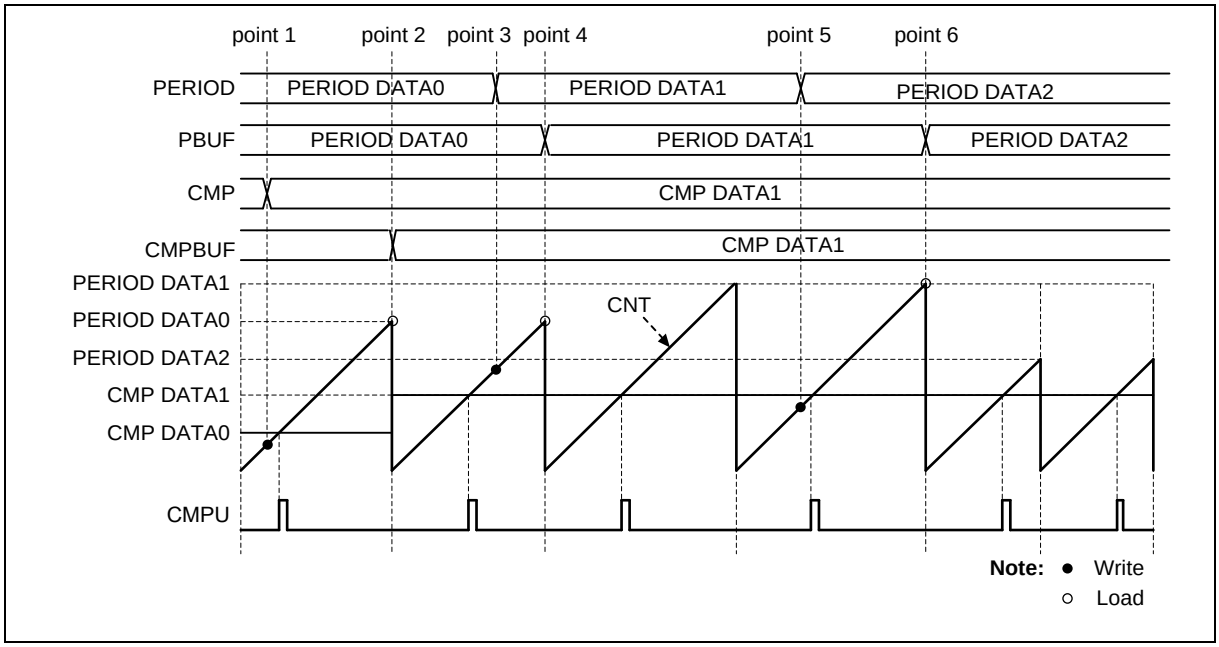


Figure 6.15-13 Period Loading in Up-Count Mode

6.15.5.9 Immediately Loading Mode

If the IMMLDENn (EPWM_CTL0[21:16]) bit is set to 1, EPWM operates at immediately loading mode. In immediately loading mode, when user updates CLKPSC(EPWM_CLKPSCn_m[11:0]), PERIOD(EPWM_PERIODn[15:0]), CMP(EPWM_CMPDATn[15:0]) or FTCMPDAT (EPWM_FTCMPDATn_m[15:0]), CLKPSC, PERIOD, CMP or FTCMPDAT will be load to active CPSCBUF (EPWM_CPSCBUFn_m[15:0]), PBUF (EPWM_PBUFn[15:0]), CMPBUF (EPWM_CMPBUFn[15:0]) or FTCMPBUF (EPWM_FTCBUF[15:0]) after current counter count is completed. If the updated PERIOD value is less than current counter value, counter will count to 0xFFFF, when counter count to 0xFFFF and prescale count to 0, the flag CNTMAXF(EPWMx_STATUS[5:0]) will raise, and then counter will count wraparound. Immediately loading mode has the highest priority. If IMMLDENn has been set, other loading mode for channel n will become invalid. Figure 6.15-14 shows an example and its steps sequence is described below.

1. Software writes CMP DATA1 and hardware immediately loading CMP DATA1 to CMPBUF at point 1.
2. Software writes PERIOD DATA1 which is greater than current counter value at point 2; counter will continue counting until equal to PERIOD DATA1 to finish a period loading.
3. Software writes PERIOD DATA2 which is less than the current counter value at point 3; counter will continue counting to its maximum value 0xFFFF and count wraparound from 0 to PERIOD DATA2 to finish this period loading.

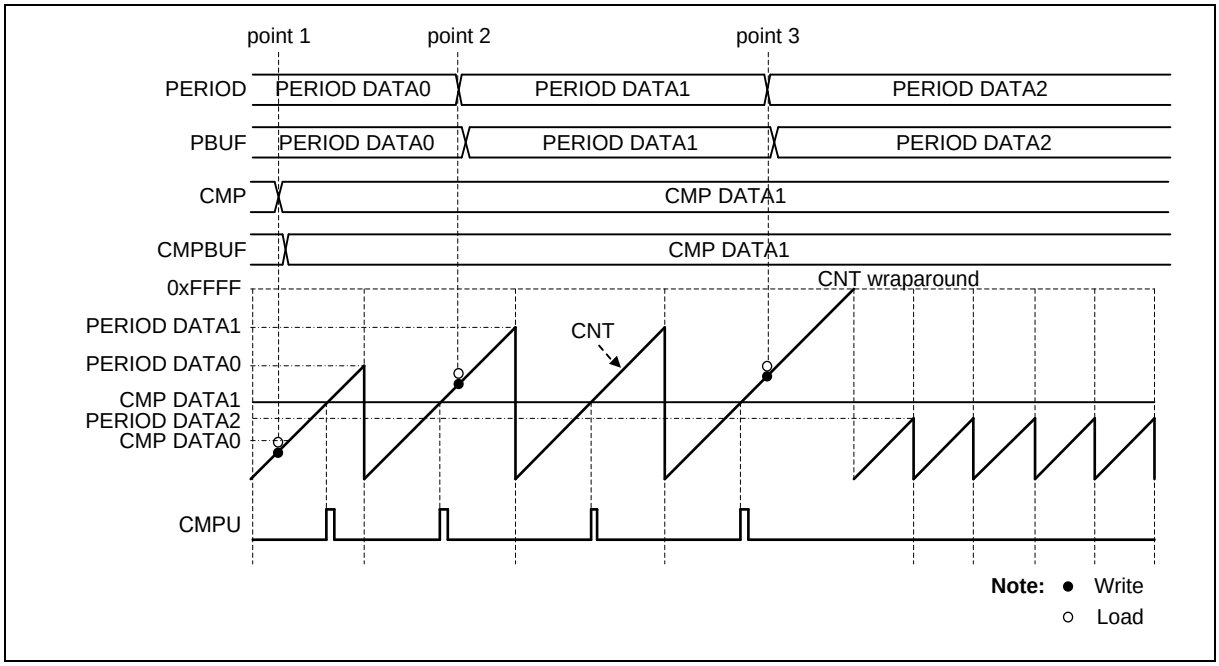


Figure 6.15-14 Immediately Loading in Up-Count Mode

6.15.5.10 Window Loading Mode

When the WINLDENn (EPWM_CTL0[13:8]) bit is set to 1, EPWM operates at window loading mode. In Window loading mode, CLKPSC(EPWM_CLKPSCn_m[11:0]), PERIOD(EPWM_PERIODn[15:0]), CMP(EPWM_CMPDATn[15:0]) and FTCMP(EPWM_FTCMPDATn_m[15:0]) will all be loaded to their active CPSCBUF(EPWM_CPSCBUFn_m[11:0]), PBUF(EPWM_PBUFn[15:0]), CMPBUF (EPWM_CMPBUFn[15:0]) and FTCMPBUF(EPWM_FTCBUFn_m[15:0]) buffers while each period is completed, but CMPBUF loading are valid only when load window is opened. Every channel n's load window is opened by setting the corresponding LOADn (EPWM_LOAD[5:0]) to 1, and hardware will close the window at the end of EPWM period. Figure 6.15-15 shows an example and its steps sequence is described below.

1. Software writes CMP DATA1 at point 1, and the load window is not opened at this period so CMP will not load to CMPBUF.
2. Software writes LOAD to open the load window at point2.
3. Software writes CLKPSC DATA1 and PERIOD DATA1 at point 3.
4. At point 4, load window has been opened, hardware loads CLKPSC DATA1, PERIOD DATA1 and CMP DATA1 to their buffer and closes the load window at the end of EPWM period.
5. Software writes CLKPSC DATA2 and PERIOD DATA2 at point 5.
6. Hardware loads CLKPSC DATA2 and PERIOD DATA2 to their buffer at the end of EPWM period at point 6.
7. Software writes CLKPSC DATA3 and PERIOD DATA3 at point 7.
8. Software writes LOAD to open the load window at point8.
9. Hardware loads CLKPSC DATA3 and PERIOD DATA3 to their buffer and closes the load window at the end of EPWM period at point 9.

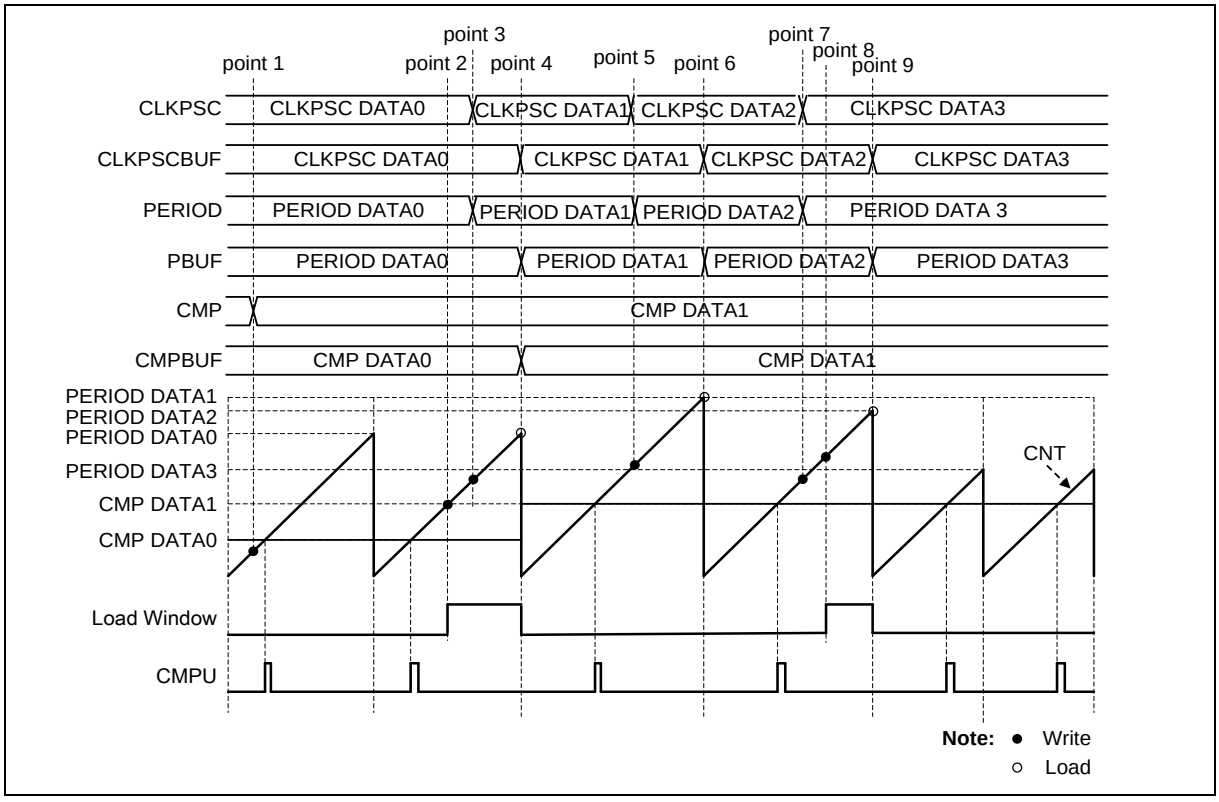


Figure 6.15-15 Window Loading in Up-Count Mode

6.15.5.11 Center Loading Mode

When the CTRLDn (EPWM_CTL0[5:0]) bit is set to 1 and EPWM counter is set to up-down count type, CNTTYPEn (EPWM_CTL1[2n+1:2n], n = 0,1..5) is 0x2, EPWM operates at center loading mode. In center loading mode, CMP(EPWM_CMPDATn[15:0]) and FTCMPDAT (EPWM_FTCMPDATn_m[15:0]) will be loaded to active CMPBUF(EPWM_CMPBUFn[15:0]) and FTCMPBUF(EPWM_FTCBUFn_m[15:0]) buffer in center of each period, that is, counter counts to PERIOD. CLKPSC(EPWM_CLKPSCn_m[11:0]) and PERIOD(EPWM_PERIODn[15:0]) will all load to their active CPSCBUF(EPWM_CPSCBUFn_m[11:0]) and PBUF(EPWM_PBUFn[15:0]) buffers while each period is completed. Center loading mode can work with window loading mode, the CMP(EPWM_CMPDATn[15:0]) will load to active CMPBUF(EPWM_CMPBUFn[15:0]) buffer in center of each period, but it is valid only at the interval of load window. Figure 6.15-16 shows an example and its steps sequence is described below.

1. Software writes CMP DATA1 at point 1.
2. Hardware loads CMP DATA1 to CMPBUF at center of EPWM period at point 2.
3. Software writes PERIOD DATA1 at point 3.
4. Hardware loads PERIOD DATA1 to PBUF at the end of EPWM period at point 4.
5. Software writes CMP DATA2 at point 5.
6. Hardware loads CMP DATA2 to CMPBUF at center of EPWM period at point 6.
7. Software writes PERIOD DATA2 at point 7.
8. Hardware loads PERIOD DATA2 to PBUF at the end of EPWM period at point 8.

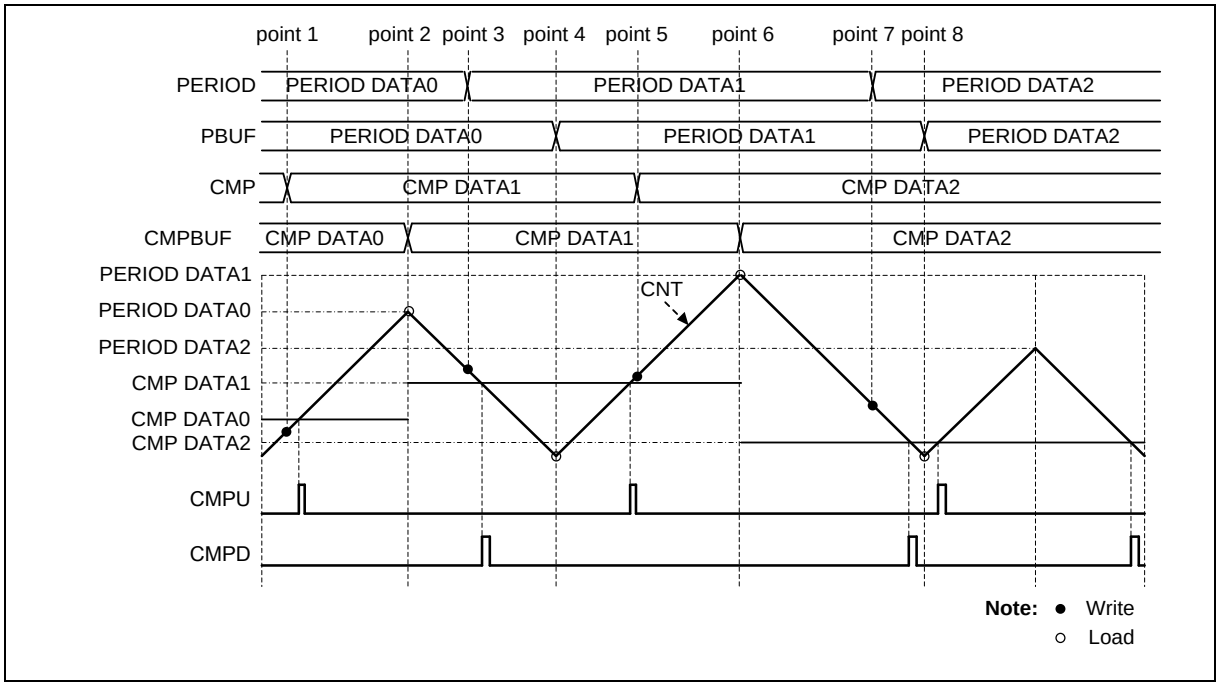


Figure 6.15-16 Center Loading in Up-Down-Count Mode

6.15.5.12 EPWM Counter Operation Mode

The EPWM counter supports two operation modes: One-shot mode and Auto-reload mode. EPWM counter will operate in One-shot mode if CNTMODEn (EPWM_CTL1[21:16]) bit is set to 1, and operate in Auto-reload mode if set to 0.

In One-shot mode, EPWM_CMPDATn and EPWM_PERIODn registers should be written first and then set CNTENn (EPWM_CNTEN[5:0]) bit as 1 to enable EPWM prescaler and counter start running. After EPWM counter counted a period, counter value will keep 0.

User can re-start next one-shot by writing new value to CMP(EPWM_CMPDATn[15:0]) bits. If one-shot counter still running, to update EPWM_CMPDATn register will cause next one-shot as continuous one-shot. Besides, to write EPWM_CMPDATn register twice under continuous one-shot operation, latest value in EPWM_CMPDATn register is valid at next one-shot period and only generate one-shot pulse once. Moreover, if user wants to clear counter within one-shot operation and starts next one-shot, user should monitor counter value to check counter has cleared and then writes EPWM_CMPDATn register. Figure 6.15-17 is an example and following is steps sequence.

1. Software writes PERIOD DATA1 and hardware immediately loading PERIOD DATA1 to PBUF at point 1.
2. Software writes CMP DATA1 which is equal to CMP DATA0 at point 2 and hardware immediately loading CMP DATA1 to CMPBUF, this event also trigger one-shot.
3. Software writes CMP DATA2 and re-trigger next one-shot (continuous one-shot) at point 3.
4. Software writes CMP DATA3 to cover CMP DATA2 and re-trigger next one-shot at point 4.
5. Period loading CMP DATA3 to CMPBUF at point 5.
6. There are no new CMP write in the previous period, and the counter value is kept as 0 at point 6.

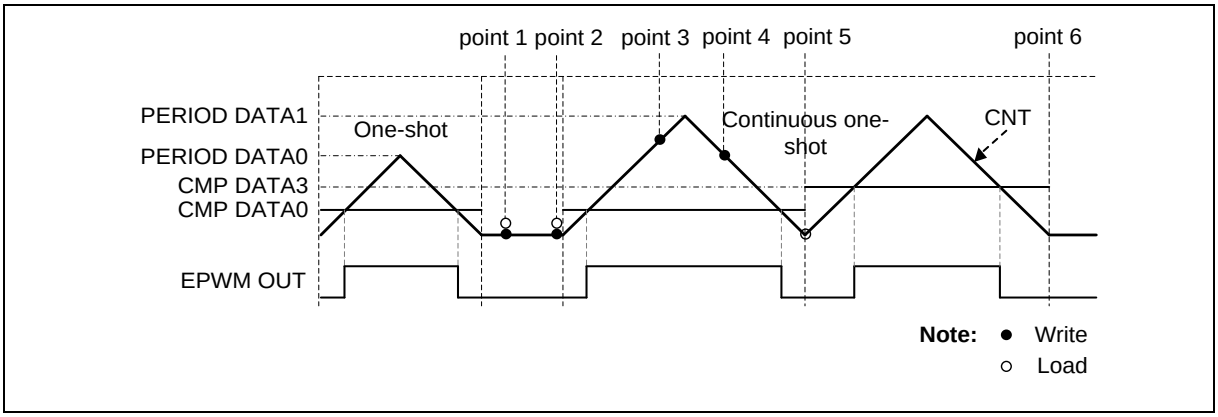


Figure 6.15-17 EPWM One-shot Mode Output Waveform

In Auto-reload mode, EPWM_CMPDATn and EPWM_PERIODn registers should be written first and then the CNTENn(EPWM_CNTEN[n]) bit is set to 1 to enable EPWM prescaler and start to run counter. The value of CLKPSC(EPWM_CLKPSCn_m[11:0]), PERIOD(EPWM_PERIODn[15:0]) and CMP(EPWM_CMPDATn[15:0]) will auto reload to their active buffer according different loading mode. If PERIOD(EPWM_PERIODn[15:0]) is set to 0, EPWM counter will be set to 0.

6.15.5.13 EPWM Pulse Generator

The EPWM pulse generator uses counter and comparator events to generate EPWM pulse. The events are: zero point, period point in up counter type and down counter type, center point in up-down counter type and counter equal to comparator point in three types. As to up-down counter type, there are two counter equal comparator points, one at up count and the other at down count. Besides, Complementary mode has two comparators compared with counter, and thus comparing equal points will become four in up-down counter type and two for up or down counter type.

Each event point can decide EPWM waveform to do nothing (X), set Low (L), set High (H) or toggle (T) by setting the EPWM_WGCTL0 and EPWM_WGCTL1 registers. Using these points can easily generate asymmetric EPWM pulse or variant waveform as shown in Figure 6.15-18. In the figure, EPWM is in complementary mode, there are two comparators n and m to generate EPWM pulse. n denotes even channel number 0, 2, or 4, and m denotes odd channel number 1, 3, or 5. n channel and m channel are complementary paired. Complementary mode uses two channels (CH0 and CH1, CH2 and CH3, or CH4 and CH5) as a pair of EPWM outputs to generate complement paired waveforms. CMPU denotes CNT(EPWM_CNTn[15:0]) is equal to CMP(EPWM_CMPDATn[15:0]) when counting up. CMPD denotes CNT bits is equal to CMP bits when counting down.

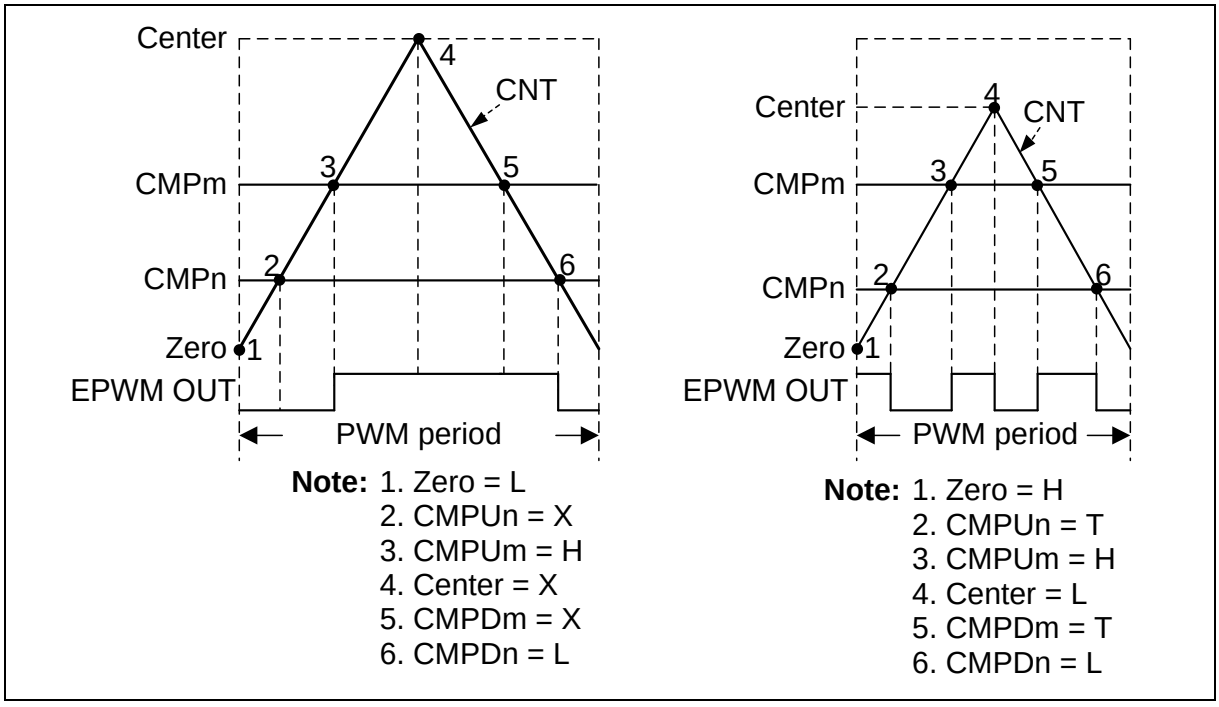


Figure 6.15-18 EPWM Pulse Generation

The generation events may sometimes set to the same value, as the reason, events priority between different counter types are list below, up counter type(Table 6.15-2), down counter type (Table 6.15-3) and up-down counter type (Table 6.15-4). By using event priority, user can easily generate 0% to 100% duty pulse as shown in Figure 6.15-19.

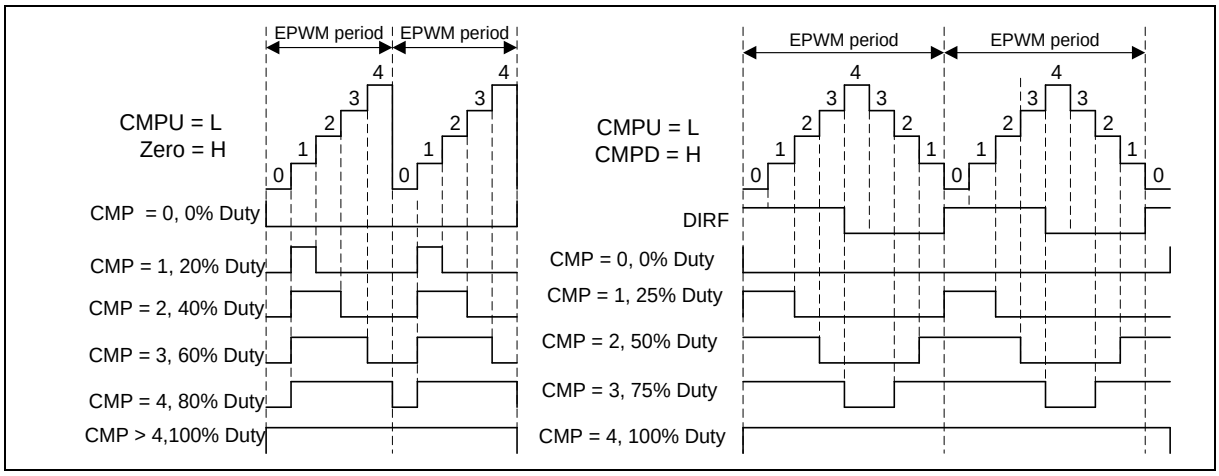


Figure 6.15-19 EPWM 0% to 100% Pulse Generation

Priority	Up Event
1 (Highest)	Period event (CNT = PERIOD)
2	Compare up event of odd channel (CNT = CMPUm)
3	Compare up event of even channel (CNT = CMPUn)
4 (Lowest)	Zero event (CNT = 0)

Table 6.15-2 EPWM Pulse Generation Event Priority for Up-Counter

Priority	Down Event
1 (Highest)	Zero event (CNT = 0)
2	Compare down event of odd channel (CNT = CMPDm)
3	Compare down event of even channel (CNT = CMPDn)
4 (Lowest)	Period event (CNT = PERIOD)

Table 6.15-3 EPWM Pulse Generation Event Priority for Down-Counter

Priority	Up Event	Down Event
1 (Highest)	Compare up event of odd channel (CNT = CMPUm	Compare down event of odd channel (CNT = CMPDm)
2	Compare up event of even channel (CNT = CMPUn)	Compare down event of even channel (CNT = CMPDn)
3	Zero event (CNT = 0)	Period (center) event (CNT = PERIOD)
4	Compare down event of odd channel (CNT = CMPDm)	Compare up event of odd channel (CNT = CMPUm)
5 (Lowest)	Compare down event of even channel (CNT = CMPDn)	Compare up event of even channel (CNT = CMPUn)

Table 6.15-4 EPWM Pulse Generation Event Priority for Up-Down-Counter

6.15.5.14 EPWM Output Mode

The EPWM supports two output modes: Independent mode which may be applied to DC motor system, Complementary mode with dead-time insertion which may be used in the application of AC induction motor and permanent magnet synchronous motor.

6.15.5.15 Independent mode

By default, the EPWM is operating in independent mode, independent mode is enabled when channel n corresponding OUTMODEn (EPWM_CTL1[26:24]) bit is set to 0. In this mode six EPWM channels: EPWM_CH0, EPWM_CH1, EPWM_CH2, EPWM_CH3, EPWM_CH4 and EPWM_CH5 are running off its own period and duty as shown in Figure 6.15-20.

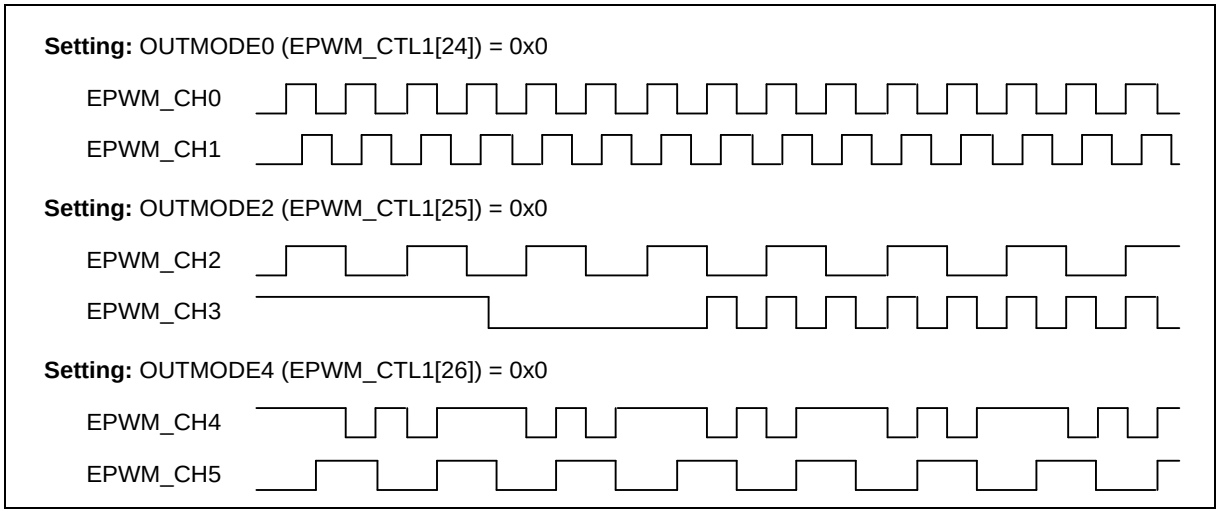


Figure 6.15-20 EPWM Independent Mode Waveform

6.15.5.16 Complementary Mode

Complementary mode is enabled when the pair channel corresponding OUTMODEn (EPWM_CTL1[26:24]) bit set to 1. In this mode there are 3 EPWM generators utilized for complementary mode, with total of 3 EPWM output paired pins in this module. In Complimentary modes, the internal odd EPWM signal must always be the complement of the corresponding even EPWM signal. EPWM_CH1 will be the complement of EPWM_CH0. EPWM_CH3 will be the complement of EPWM_CH2 and EPWM_CH5 will be the complement of EPWM_CH4 as shown in Figure 6.15-21.

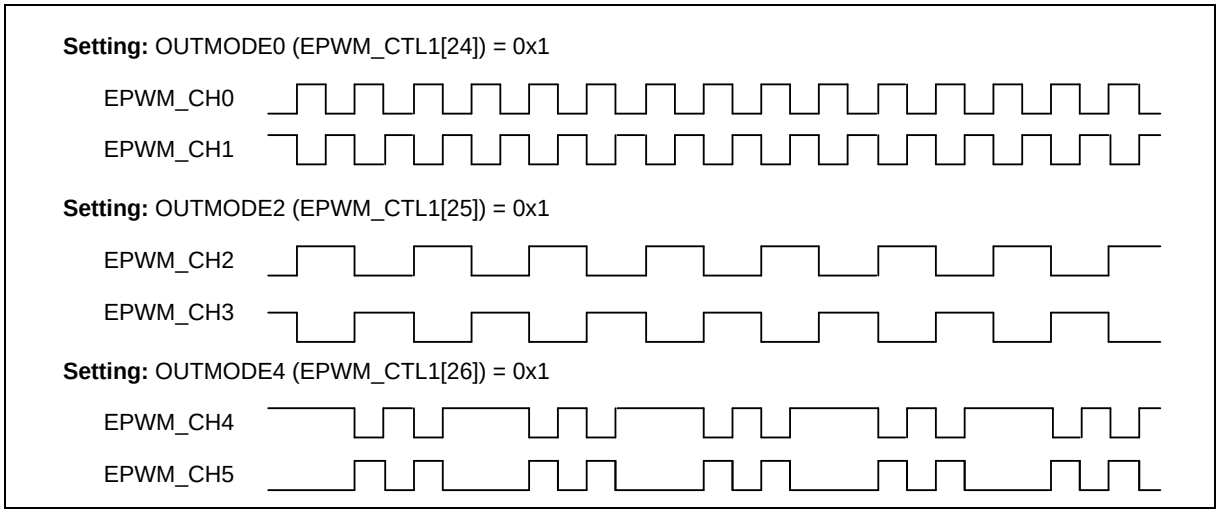


Figure 6.15-21 EPWM Complementary Mode Waveform

6.15.5.17 EPWM Output Function

Based on the output mode, there are two output functions: group and synchronous functions for advanced output control. Group function, forces the EPWM_CH2 and EPWM_CH4 synchronous with EPWM_CH0 generator and forces the EPWM_CH3 and EPWM_CH5 synchronous with EPWM_CH1, may simplify updating duty control in DC and BLDC motor applications. Besides, Synchronous function makes any channel of EPWM0 and EPWM1 in phase, user can control phase value and direction.

6.15.5.18 Group Function

Group function is enabled when GROUPEN (EPWM_CTL0[24]) is set to 1, no matter in independent or complementary mode. This control allows all even EPWM channels output to be controllable by EPWM_PERIOD0 and EPWM_CMPDAT0 registers and all odd EPWM channels output to be controllable by EPWM_PERIOD1 and EPWM_CMPDAT1 registers. That is, user only needs to set EPWM_CH0 to get EPWM_CH0, EPWM_CH2 and EPWM_CH4 output the same pulse, and set EPWM_CH1 to get EPWM_CH1, EPWM_CH3 and EPWM_CH5 output the same pulse, as shown in Figure 6.15-22. When operating group function, OUTMODE0, OUTMODE2 and OUTMODE4 bits of CTL1 register must all set to 0 for independent mode or all set to 1 for complementary mode.

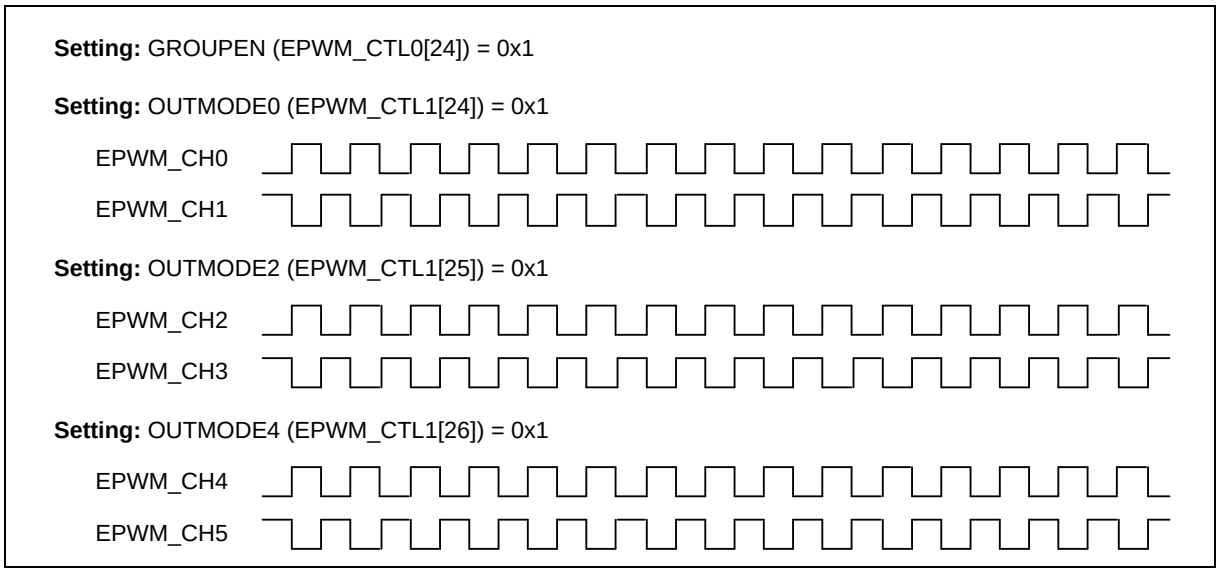


Figure 6.15-22 EPWM Group Function Waveform

6.15.5.19 Synchronous Function

Synchronous function can only be enabled when complementary mode is enabled. Figure 6.15-24 is counter synchronous function block diagram. Every counter of EPWM pairs has a SYNC_IN and a SYNC_OUT signals. The SYNC_IN signal for the first EPWM0 pair counter comes from EPWM0_SYNC_IN pin, and the others come from the SYNC_OUT signal of the previous EPWM pair counter. The input signal from EPWM0_SYNC_IN pin will be filtered by a 3-bit noise filter as Figure 6.15-23. In addition, it can be inversed by setting the bit SINPINV (EPWM_SYNC[23]) to realize the polarity setup for the input signal. The noise filter sampling clock can be selected by setting bits SFLTCSEL (EPWM_SYNC[19:17]) to fit different noise properties. Moreover, by setting the bits SFLTCNT (EPWM_SYNC[22:20]). User can define how many sampling clock cycles a filter will recognize the effective edge of the SYNC_IN signal. Configuring the SNFLTEN (EPWM_SYNC[16]) will enable the noise filter function. By default, it is disabled.

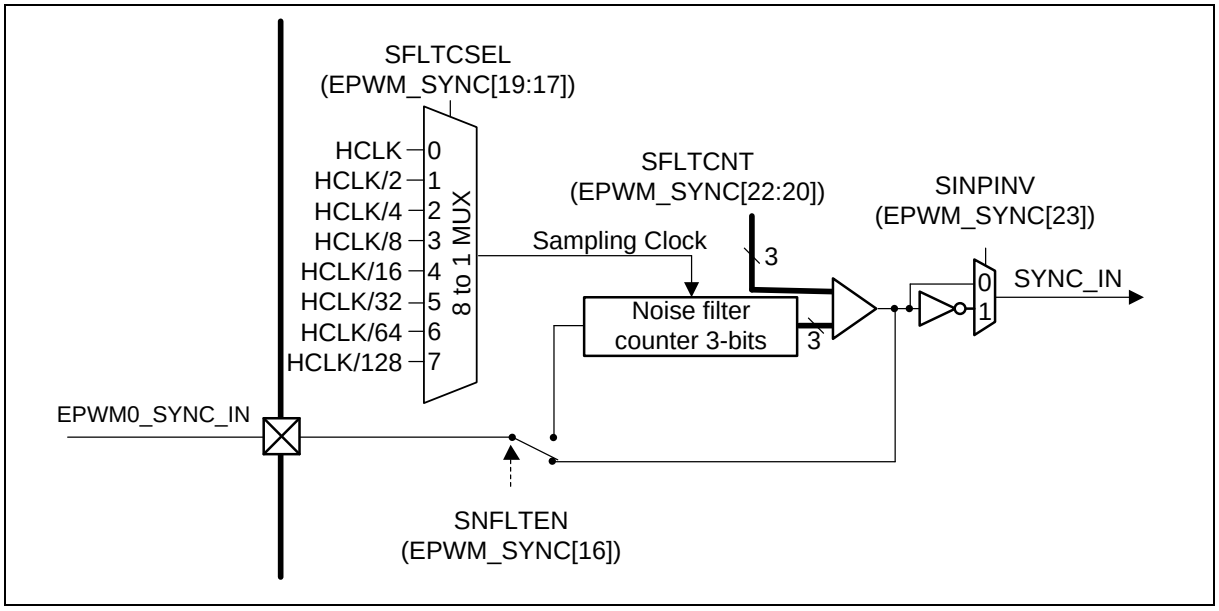


Figure 6.15-23 EPWM SYNC_IN Noise Filter Block Diagram

User can use SINSRCn (EPWM_SYNC[13:8]) bits to select the synchronize source. When SINSRCn bits is set to 0, user can generate SYNC_IN signal for the next counter's synchronization when EPWM0_SYNC_IN pin is high or setting SWSYNCn (EPWM_SWSYNC[2:0]) to 1. Synchronizing source can also be selected as CNT = 0 or CNT = EPWM_CMPDATm register (if being the up-down counter type, it will synchronize twice in a EPWM period) to trigger a sync event or to disable SYNC_OUT signal.

When the PHSENn (EPWM_SYNC[2:0]) is enabled and the synchronous source has a happening event, the counter will load a value from the PHS (EPWM_PHSn_m[15:0]) register. This method synchronizes counters to different phase in the same time. In the up-down counter type, user can set the value in PHSDIRn (EPWM_SYNC[26:24]) to control the counter direction after synchronization. Although the Synchronous function can synchronize channels in phase, it can't work from the beginning of EPWM enable. To start EPWM and BPWM counters in the same time, user has to set the EPWM Synchronous Start Control Register (EPWM_SSCTL[5:0]) to enable the channel counters which are planned to start counting together, and select the SSRC(EPWM_SSCTL[9:8]) to choose the Synchronous Start source, followed by setting the EPWM Synchronous Start Trigger Register CNTSEN (EPWM_SSTRG[0]).

For applications, please do not use Group and Synchronous function simultaneously because the Synchronous function will be inactive.

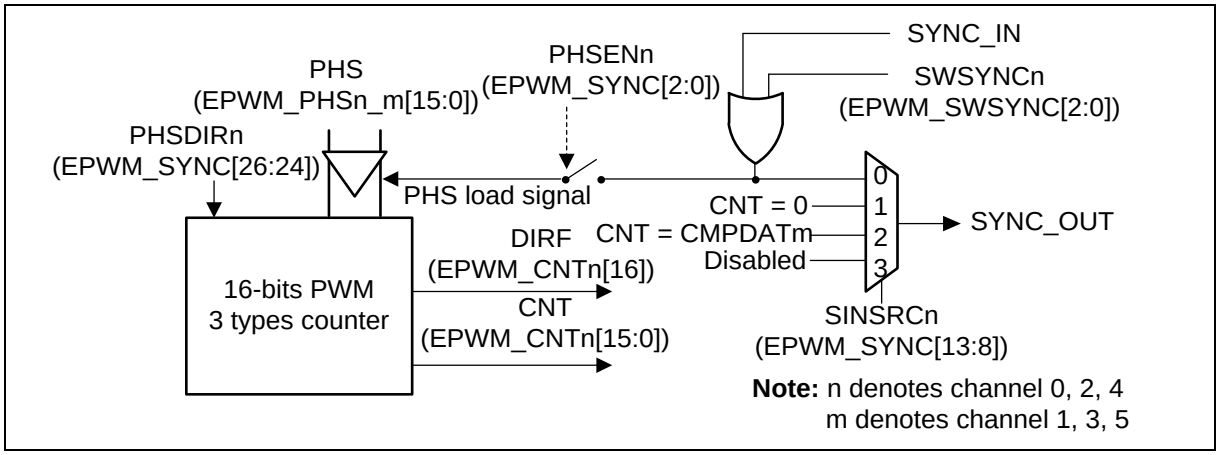


Figure 6.15-24 EPWM Counter Synchronous Function Block Diagram

Figure 6.15-25 is an example of the synchronous function in the up-down counter type. In the example, synchronizing source comes from the external EPWM SYNC_IN signal. At the beginning, the output waveform of EPWM_CH0, EPWM_CH2 and EPWM_CH4 are in the same phase. Then at Point A, the EPWM SYNC input signal comes as a sync event, resulting in phase shifts and counting direction changes for all of the counters. To realize the altered counter behaviors before the sync event coming, user has to setup the corresponding phase value in the PHS of(EPWM_PHSn_m[15:0]) as well as the counting direction in the PHSDIRn (EPWM_SYNC[26:24]). In this case, one third of phase shifts are made. by setting the corresponding channel n's counter counting direction after synchronizing, as illustrated around the left side of Figure 6.15-25.

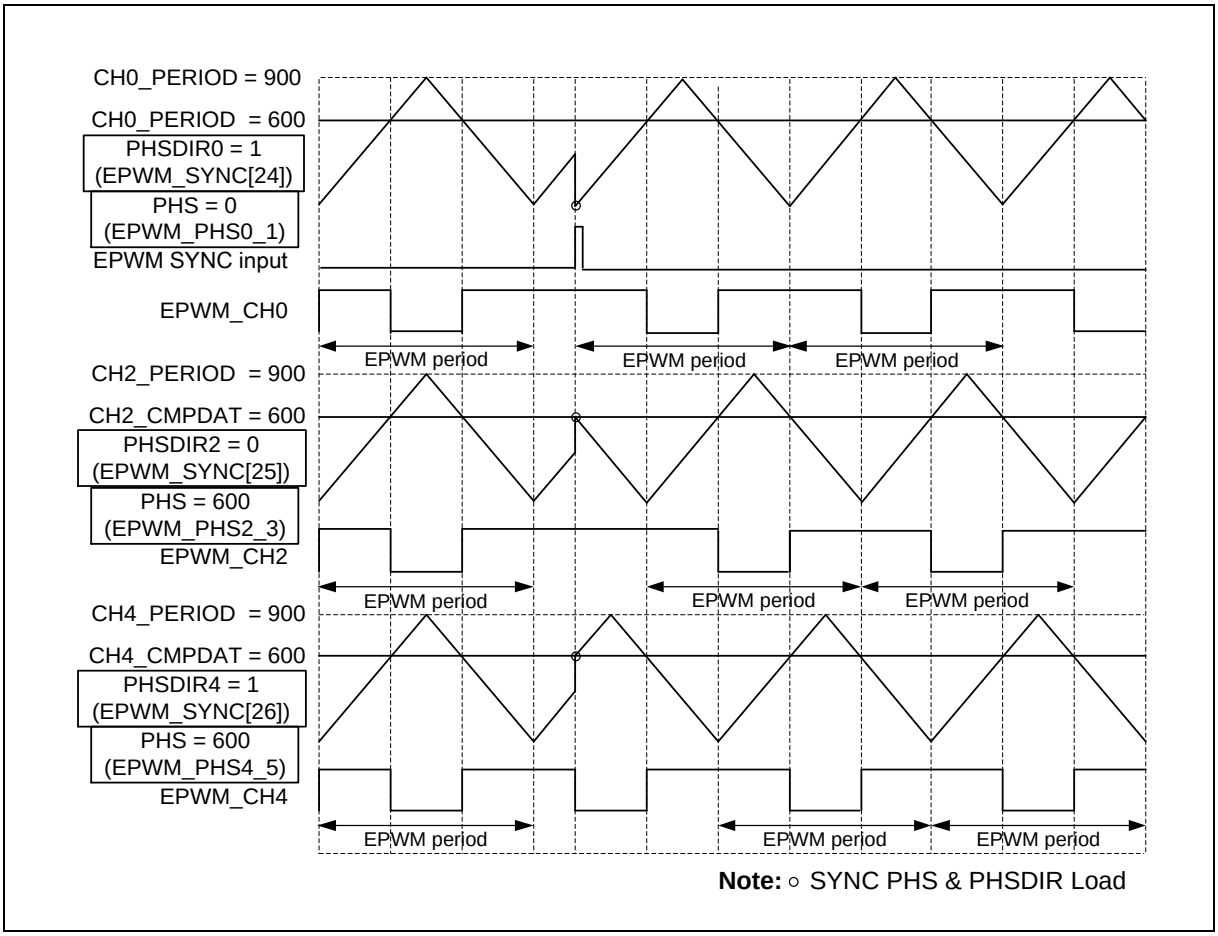


Figure 6.15-25 EPWM Synchronous Function with Synchronize source from SYNC_IN Signal

6.15.5.20 EPWM Output Control

After EPWM pulse generation, there are four to six steps to control the output of EPWM channels. In independent mode, there are Mask, Brake, Pin Polarity and Output Enable four steps as shown in Figure 6.15-26. In complementary mode, it needs two more steps to precede these four steps, Complementary channels and Dead-Time Insertion as shown in Figure 6.15-27.

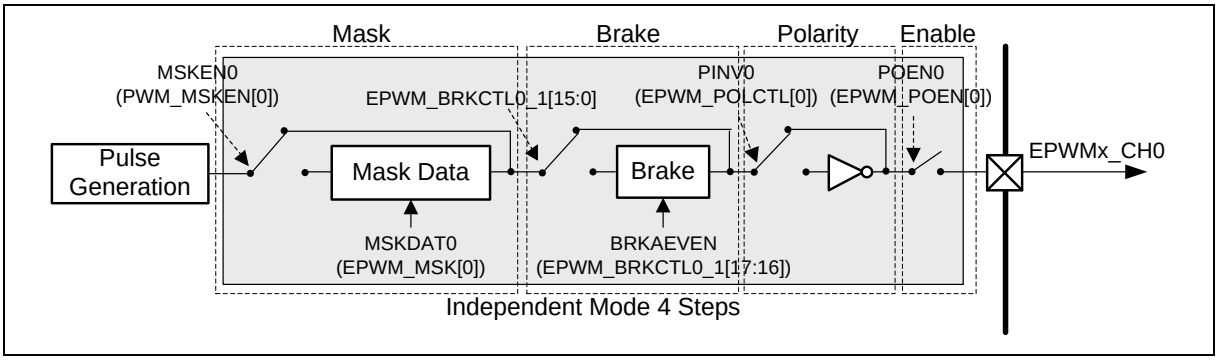


Figure 6.15-26 EPWMx_CH0 Output Control in Independent Mode

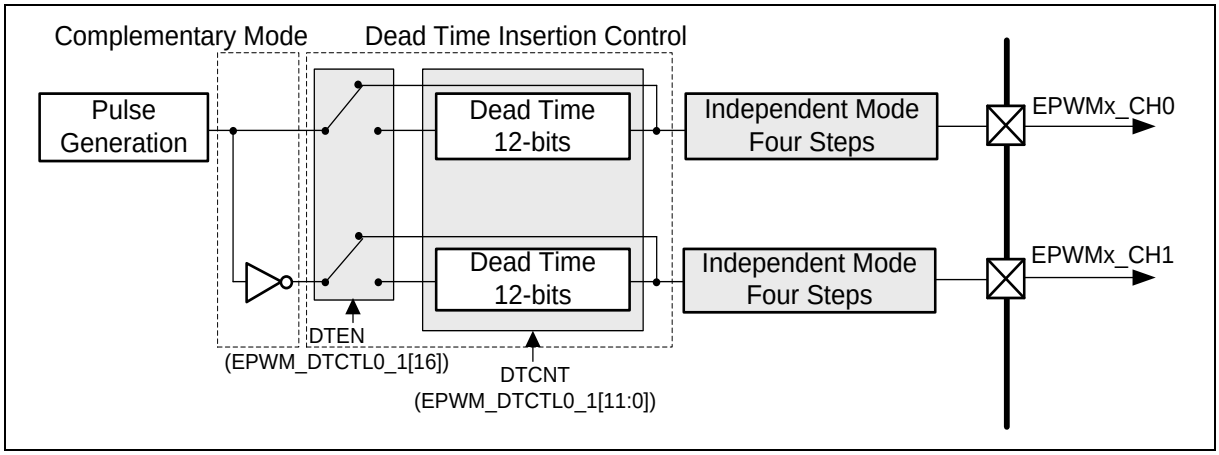


Figure 6.15-27 EPWMx_CH0 and EPWMx_CH1 Output Control in Complementary Mode

6.15.5.21 Dead-Time Insertion

In the complementary application, the complement channels may drive the external devices like power switches. The dead-time generator inserts a low level period called “dead-time” between complementary outputs to drive these devices safely and to prevent system or devices from the burn-out damage. Hence the dead-time control is a crucial mechanism to the proper operation of the complementary system. By setting corresponding channel n DTEN (EPWM_DTCTLn_m[16]) bit to enable dead-time function and DTCNT (EPWM_DTCTLn_m[11:0]) to control dead-time period, the dead-time can be calculated from the following formula:

Dead-time = (DTCNT (EPWM_DTCTLn_m[11:0])+1) * EPWMx_CLK period

Dead-time insertion clock source can be selected from prescaler output by setting DTCKSEL (EPWM_DTCTLn_m[24]) to 1. By default, clock source comes from EPWM_CLK, which is prescaler input. Then the dead-time can be calculated from the following formula:

Dead-time = (DTCNT (EPWM_DTCTLn_m[11:0])+1) *

(CLKPSC (EPWM_CLKPSCn_m [11:0])+1)*EPWMx_CLK period

Please note that the EPWM_DTCTLn_m are write-protected registers.

Figure 6.15-28 indicates the dead-time insertion for one pair of EPWM signals.

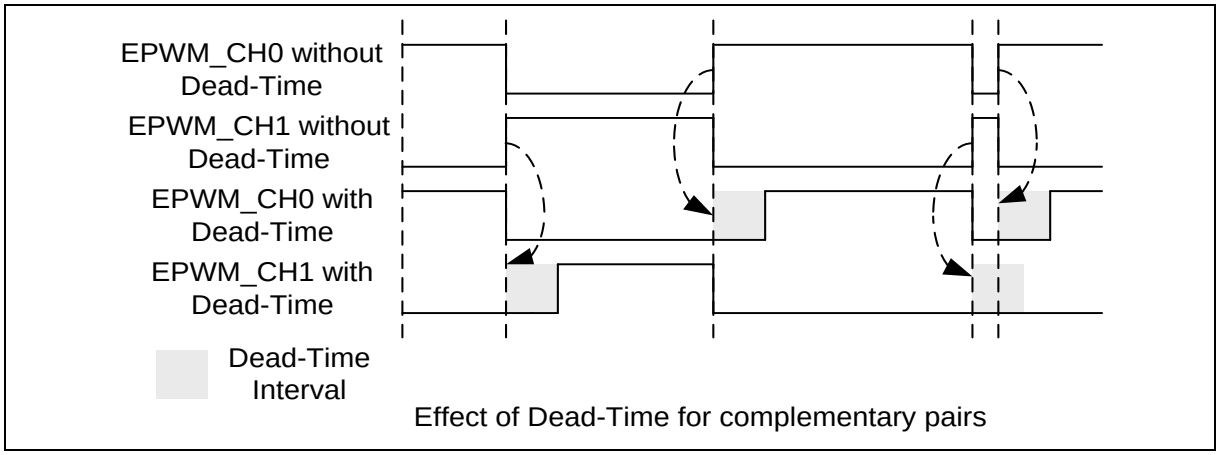


Figure 6.15-28 Dead-Time Insertion

6.15.5.22 EPWM Mask Output Function

Each of the EPWM channel output value can be manually overridden with the settings in the EPWM Mask Enable Control Register (EPWM_MSKEN) and the EPWM Masked Data Register (EPWM_MSK) With these settings, the EPWM channel outputs can be assigned to specified logic states independent of the duty cycle comparison units. The EPWM mask bits are useful when controlling various types of Electrically Commutated Motor (ECM) like a BLDC motor. The EPWM_MSKEN register contains six bits, MSKENn(EPWM_MSKEN[5:0]). If the MASKENn is set to active-high, the EPWM channel n output will be overridden. The EPWM_MSK register contains six bits, MSKDATn(EPWM_MSK[5:0]). The bit value of the MSKDATn determines the state value of the EPWM channel n output when the channel is overridden. Figure 6.15-29 shows an example of how EPWM mask control can be used for the override feature.

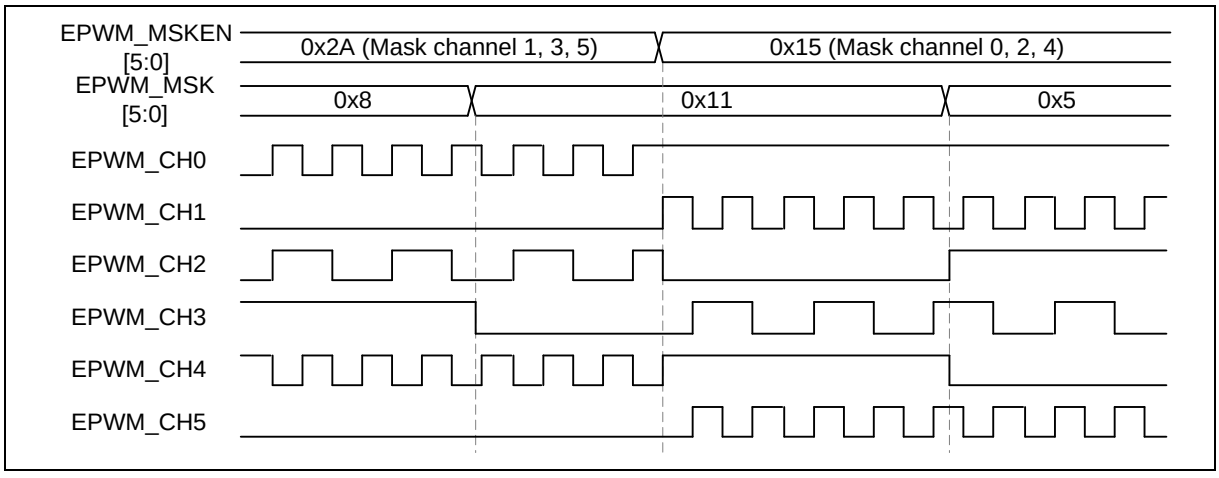


Figure 6.15-29 Illustration of Mask Control Waveform

6.15.5.23 EPWM Brake

Each EPWM module has two external input brake control signals. User can select active brake pin source is from EPWMx_BRAKEy pin by BKxSRC bits of EPWM_BNF register(x=0,1, y=0,1). The external signals will be filtered by a 3-bit noise filter. User can enable the noise filter function by BRKxNFEN bits of EPWM_BNF reigster, and noise filter sampling clock can be selected by setting BRKxNFSEL bits of EPWM_BNF register to fit different noise properties. Moreover, by setting the BRKxFCNT bits, user can define by how many sampling clock cycles a filter will recognize the effective edge of the brake signal.

In addition, it can be inversed by setting the BRKxPINV (x denotes input external pin 0 or 1) bits of EPWM_BNF register to realize the polarity setup for the brake control signals. Set BRKxPINV bit to 0, brake event will occurred when EPWMx_BRAKEy(x=0,1, y=0,1) pin status is from low to high; set BRKxPINV to 1, brake event will occurred when EPWMx_BRAKEy pin status is from high to low.

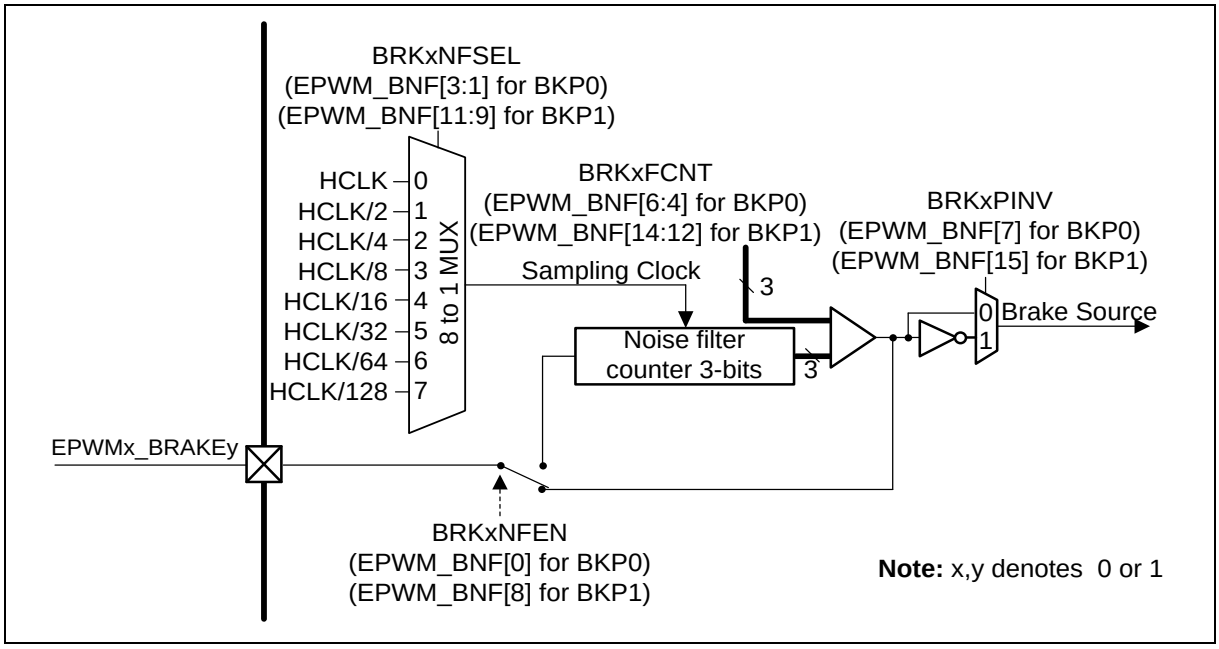


Figure 6.15-30 Brake Noise Filter Block Diagram

For complementary mode, it is often necessary to set a safe output state to the complement output pairs once the brake event occurs.

Each complementary channel pair shares a EPWM brake function, as shown Figure 6.15-31. To control paired channels to output safety state, user can setup BRKAEVEN (EPWM_BRKCTL0_1[17:16]) for even channels and BRKAODD (EPWM_BRKCTL0_1[19:18]) for odd channels when the fault brake event happens. There are two brake detectors: Edge detector and Level detector. When the edge detector detects the brake signal and BRKEIENnm (EPWM_INTEN1[2:0]) is enabled, the brake function generates BRK_INT. This interrupt needs software to clear, and the BRKESTSn (EPWM_INTSTS1[21:16]) brake state will keep until the next EPWM period starts after the interrupt cleared. The brake function can also operate in another way through the level detector. Once the level detector detects the brake signal and the BRKLIENnm (EPWM_INTEN1[10:8]) is also enabled, the brake function will generate BRK_INT, but BRKLSTSn (EPWM_INTSTS1[29:24]) brake state will auto recovery to normal output while level brake source recovery to high level and pass through “Low Level Detection” at the EPWM waveform period when brake condition removed without clear interrupt.

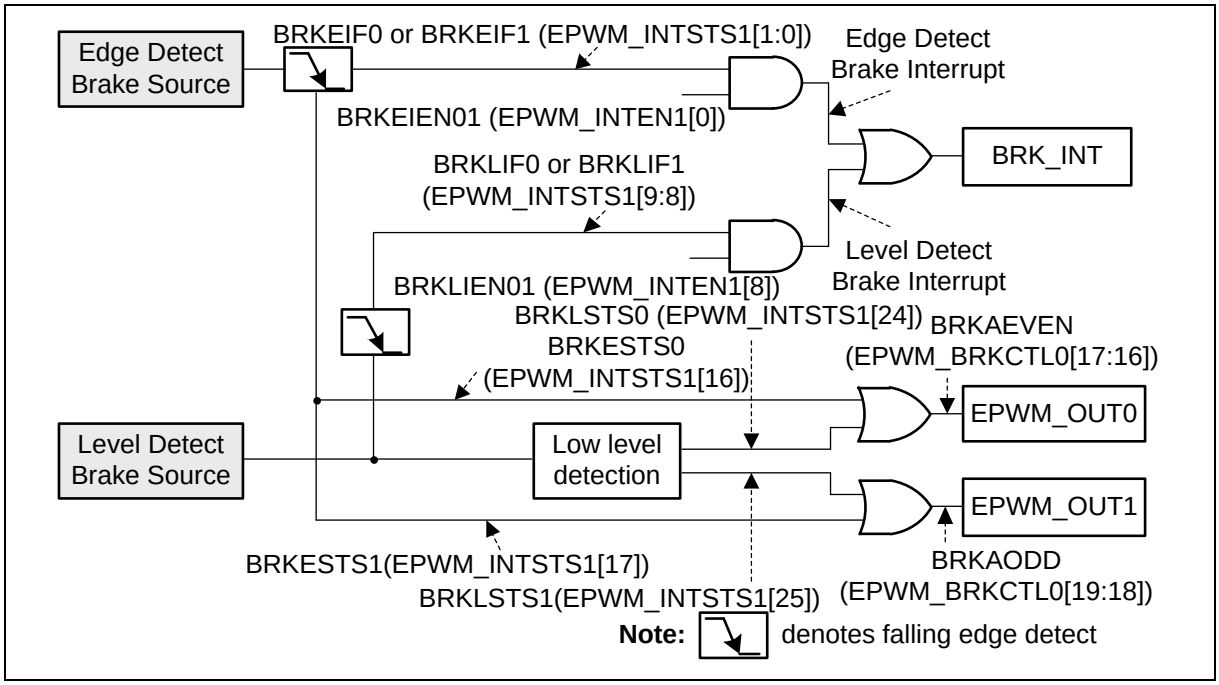


Figure 6.15-31 Brake Block Diagram for EPWMx_CH0 and EPWMx_CH1 Pair

Figure 6.15-32 illustrates the edge detector waveform for EPWMx_CH0 and EPWMx_CH1 pair. In this case, the edge detect brake source has occurred twice for the brake events. When the event occurs, both of the BRKEIF0 and BRKEIF1 flags are set and BRKESTS0 and BRKESTS1 bits are also set to indicate brake state of EPWMx_CH0 and EPWMx_CH1. For the first occurring event, software writes 1 to clear the BRKEIF0 flag. After that, the BRKESTS0 bit is cleared by hardware at the next start of the EPWM period. At the same moment, the EPWMx_CH0 outputs the normal waveform even though the brake event is still occurring. The second event also triggers the same flags, but at this time, software writes 1 to clear the BRKEIF1 flag. Afterward, EPWMx_CH1 outputs normally at the next start of the EPWM period.

As a contrast to the edge detector example, Figure 6.15-33 illustrates the level detector waveform for EPWMx_CH0 and EPWMx_CH1 pair. In this case, the BRKLIF0 and BRKLIF1 flags can only indicate the brake event having occurred. The BRKLSTS0 and BRKLSTS1 brake states will automatically recover at the start of the next EPWM period no matter at what states the BRKLIF0 and BRKLIF1 flags are at that moment.

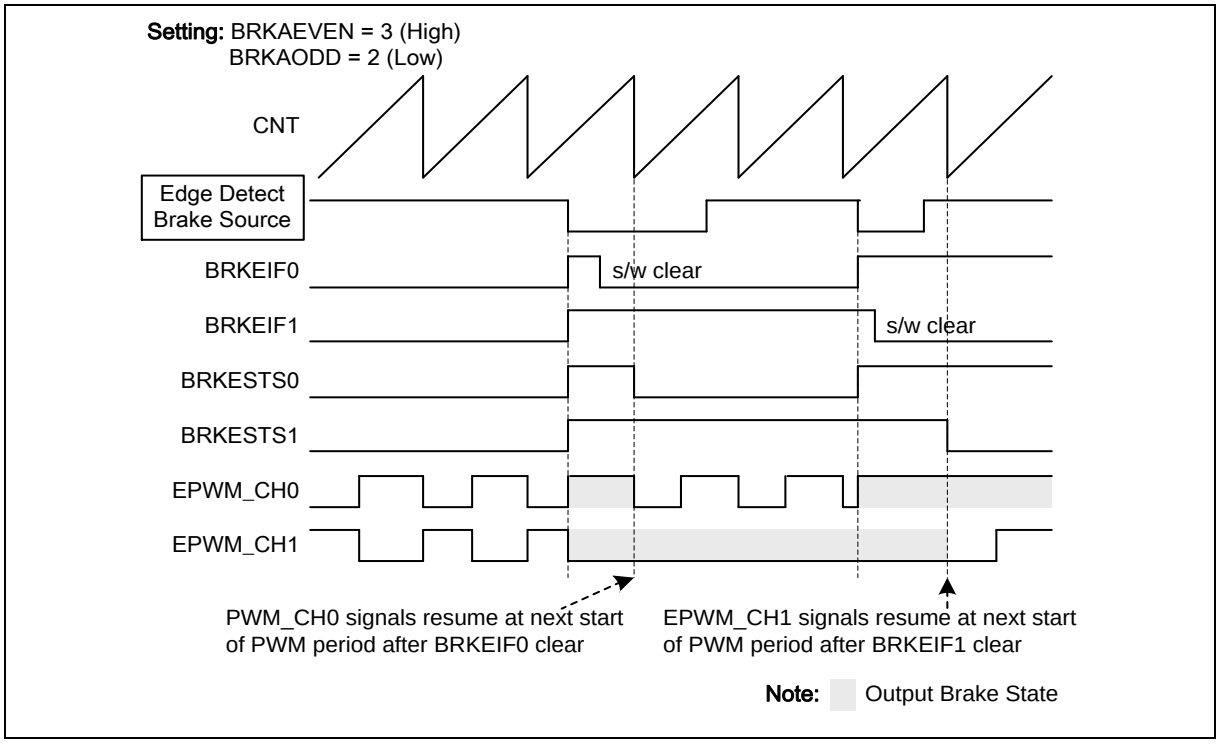


Figure 6.15-32 Edge Detector Waveform for EPWMx_CH0 and EPWMx_CH1 Pair

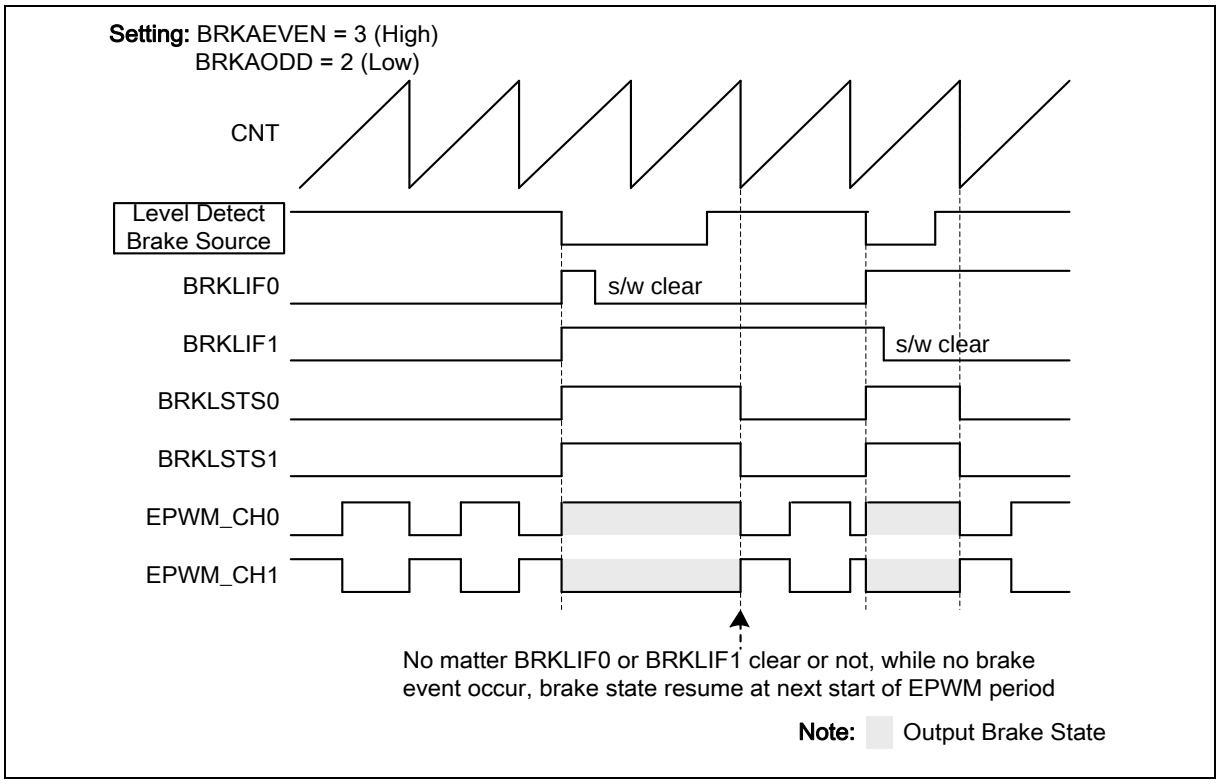


Figure 6.15-33 Level Detector Waveform for EPWMx_CH0 and EPWMx_CH1 Pair

The two kinds of detectors detect the same seven brake sources: two from external input signals, two

from analog comparators(ACMP), one from EADC result monitor (EADCRM), one from system fail and one from software triggered, that are shown in Figure 6.15-34. ACMP brake sources will be detected only when internal ACMP0_O or ACMP1_O signal from low to high.

Among the above described brake sources, the brake source coming from system fail can still be specified to several different system fail conditions. These conditions include clock fail, Brown-out detect, SRAM parity check error and Core lockup. Figure 6.15-35 shows that by setting corresponding enable bits, the enabled system fail condition can be one of the sources to issue the Brake system fail to the EPWM brake.

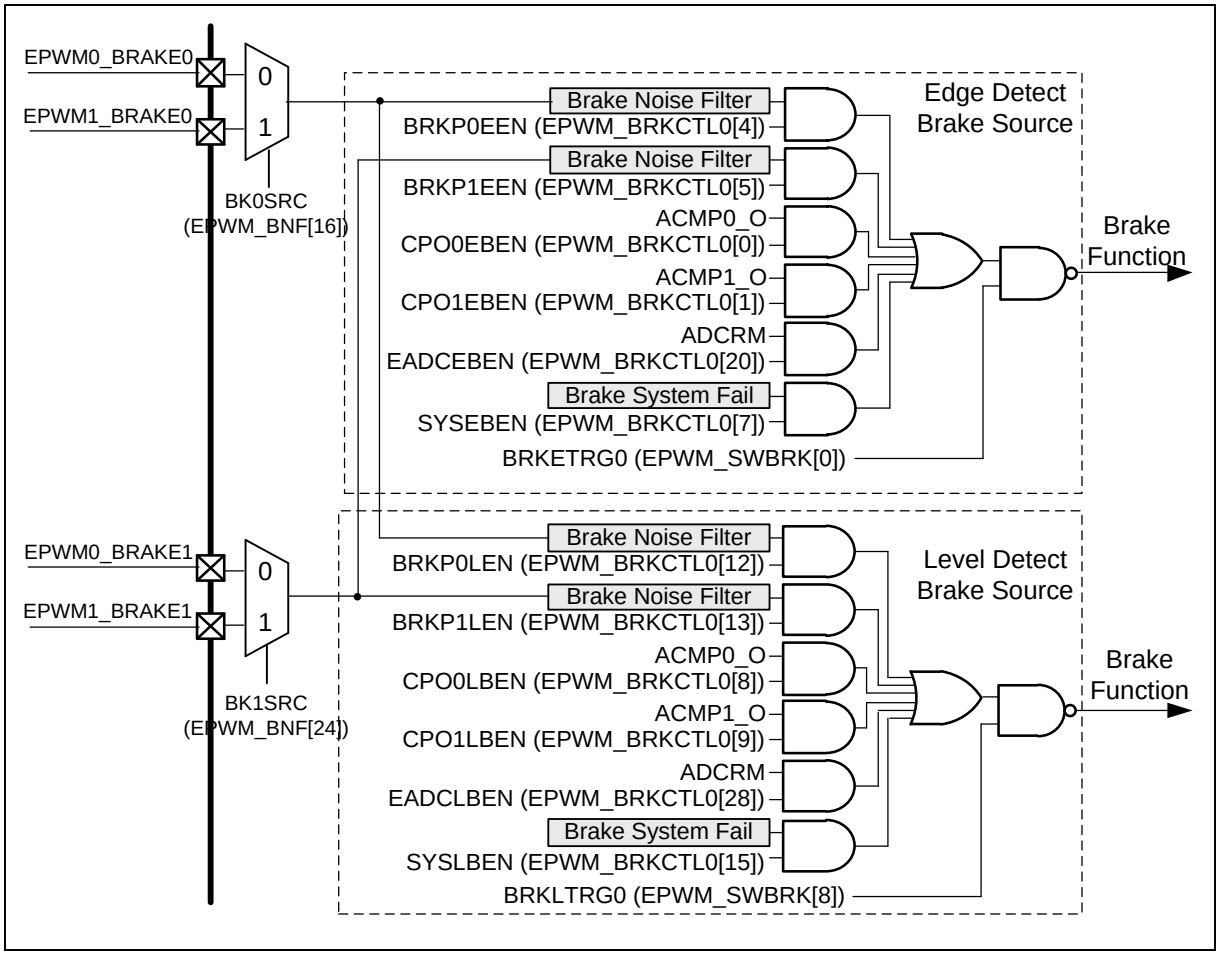


Figure 6.15-34 Brake Source Block Diagram

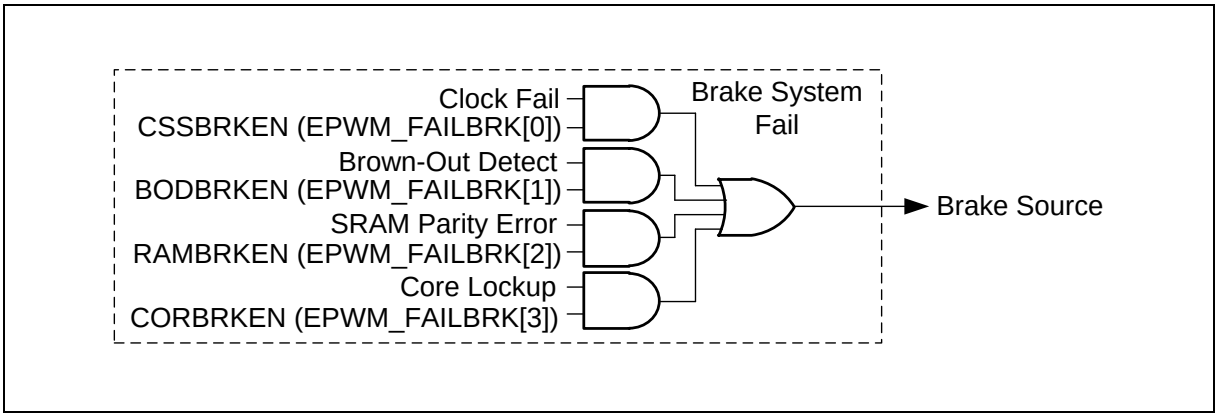


Figure 6.15-35 Brake System Fail Block Diagram

6.15.5.24 LEB Function

Leading edge blanking (LEB) function is used to blank the false trigger from brake source ACMP which may caused by EPWM output transition. Set LEBEN (EPWM_LEBCTL[0]) to enable this function. LEB source comes from EPWM_CH0, EPWM_CH2 and EPWM_CH4, use SRCENn (EPWM_LEBCTL[10:8]) as input source enable. LEB function blanking time is decided by LEBCNT (EPWM_LEBCNT[8:0]), when LEB detected trigger edge, then blanking time will count from LEBCNT+1 to 0, the counter clock base is ECLK. If a new trigger event occurs, blanking counter will reset to LEBCNT and down count again. LEB trigger edge can be rising, falling or both rising and falling edge by setting TRGTYPE (EPWM_LEBCTL[17:16]). Figure 6.15-36 shows that LEB will blanking leading edge caused by EPWM_CH0 and EPWM_CH4.

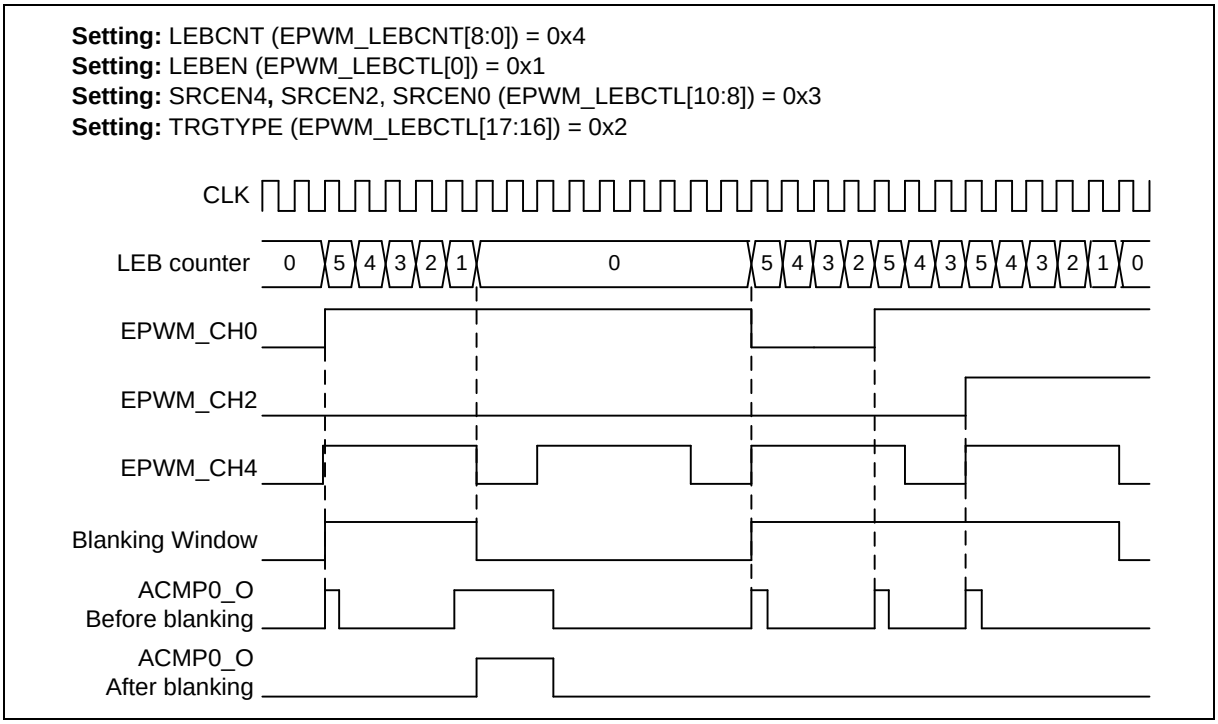


Figure 6.15-36 EPWM LEB Function Waveform

6.15.5.25 Polarity Control

Each EPWM port, from EPWM_CH0 to EPWM_CH5, has an independent polarity control module to configure the polarity of the active state of the EPWM output. By default, the EPWM output is active high. This implies the EPWM OFF state is low and ON state is high. This definition is variable through setting the EPWM Negative Polarity Control Register (EPWM_POLCTL), for each individual EPWM channel. Figure 6.15-37 shows the initial state before EPWM starting with different polarity settings.

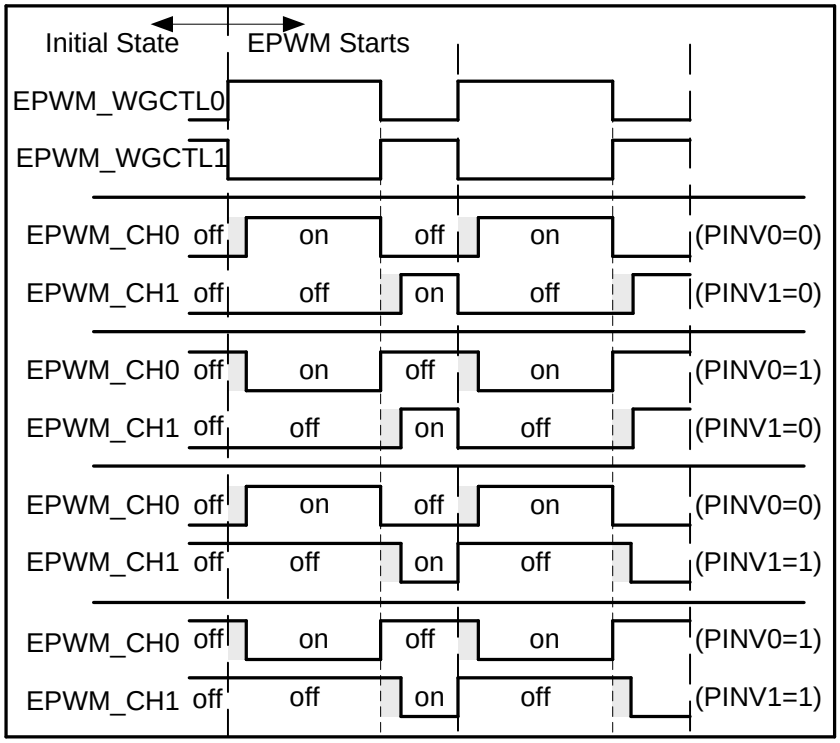


Figure 6.15-37 Initial State and Polarity Control with Rising Edge Dead-Time Insertion

6.15.5.26 EPWM Interrupt Generator

Interrupts for each EPWM are shown in Figure 6.15-39 and Figure 6.15-40.

The 1st EPWM interrupt (EPWM_INT) comes from EPWM complementary pair events. The counter can generate the Zero point Interrupt Flag ZIFn (EPWM_INTSTS0[5:0], n=0,1..5) and the Period point Interrupt Flag PIFn (EPWM_INTSTS0[13:8], n=0,1..5). When EPWM channel n's counter equals to the comparator value stored in EPWM_CMPDATn register, the different interrupt flags will be triggered depending on the counting direction. If the matching occurs at up–count direction, the Up Interrupt Flag CMPUIFn (EPWM_INTSTS0[21:16]) is set and if matching at the opposite direction, the Down Interrupt Flag CMPDIFn (EPWM_INTSTS0[29:24]) is set. If the corresponding interrupt enable bits are set, the trigger events will generates interrupt signals.

EPWM_INT can use the EPWM_IFAn (n=0~5) register to accumulate the number of times that the interrupt flags have been triggered for each channel. By setting one of IFAEN (EPWM_IFAn[31], n=0~5) bit to 1 to enable accumulator, EPWM_INT will switch interrupt source from every event trigger interrupt to trigger interrupt once every accumulate times.

By setting the IFASEL (EPWM_IFAn[29:28], n=0~5) bits, user can select one of the 4 interrupt sources to accumulate interrupt flag times for each channel, and the number of times interrupt flags will compare with IFACNT (EPWM_IFAn[15:0], n=0~5) bits. When interrupt accumulator equals IFACNT then set IFAIFn (EPWM_AINTSTS[n], n=0~5) bits as EPWM_INT signal if user enable IFAIENn (EPWM_AINTEN[n], n=0~5) bits. Accumulator interrupt of each channel can also be as request source of PDMA. Figure 6.15-38 is an example of channel 0 using EPWM_IFA0 register to output EPWM_INT once every IFCNT0+1 times interrupt events occurred.

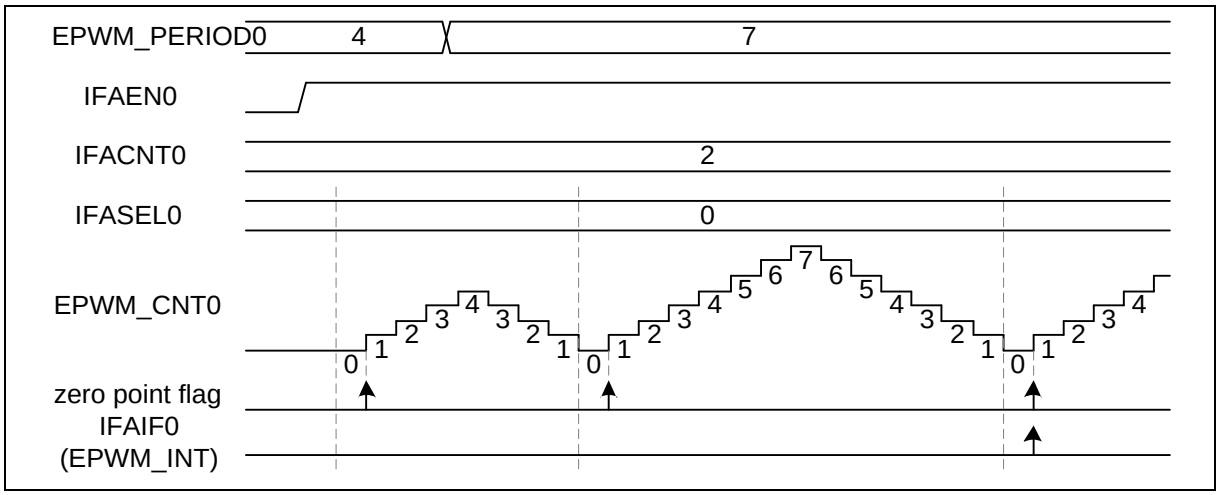


Figure 6.15-38 EPWMx_CH0 Accumulate Interrupt Waveform

The 2nd interrupt is the capture interrupt (CAP_INT). It shares the EPWM_INT vector in NVIC. The CAP_INT can be generated when the CRLIFn (EPWM_CAPIF[5:0]) flag is triggered and the Capture Rising Interrupt Enable bit CAPRIENn (EPWM_CAPIEN[5:0]) is set to 1. Or in the falling edge condition, the CFLIFn (EPWM_CAPIF[13:8]) flag can be triggered when the Capture Falling Interrupt Enable bit CAPFIENn (EPWM_CAPIEN[13:8]) is set to 1.

The 3rd is the brake interrupt (BRK_INT). The details of the BRK_INT is described in the EPWM Brake section. Figure 6.15-39 demonstrates the architecture of the EPWM interrupts.

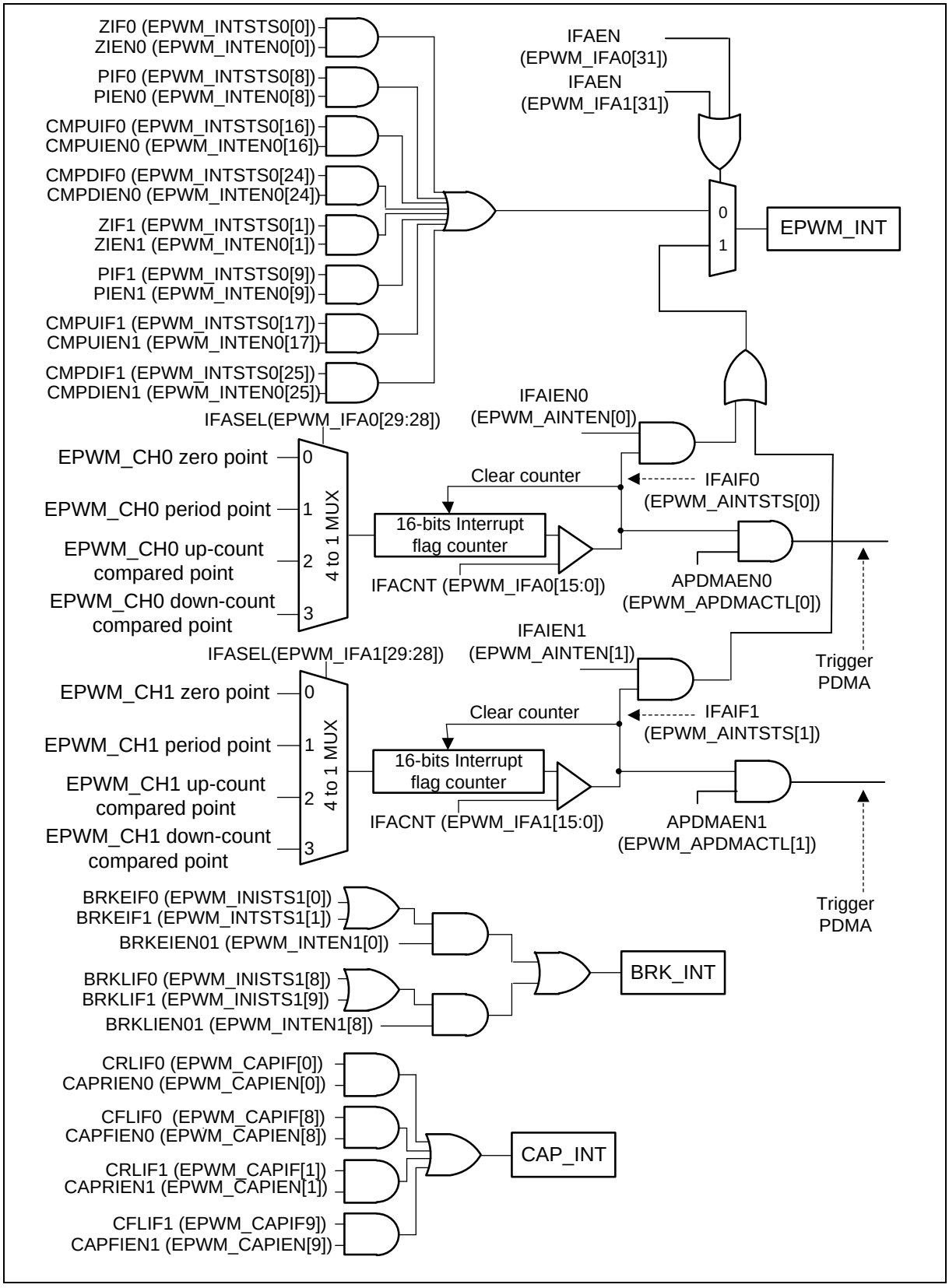


Figure 6.15-39 EPWMx_CH0 and EPWMx_CH1 Pair Interrupt Architecture Diagram

The 4th is Fault Detect Function interrupt (FLT_INT). It shares the EPWM_INT vector in NVIC. Figure 6.15-40 shows the FLT_INT can be generated when FDIENn (EPWM_FDIEN[n], n=0~5) is set to 1 and FDIFn (EPWM_FDSTS[n], n=0~5) is set to 1 by detecting output open short.

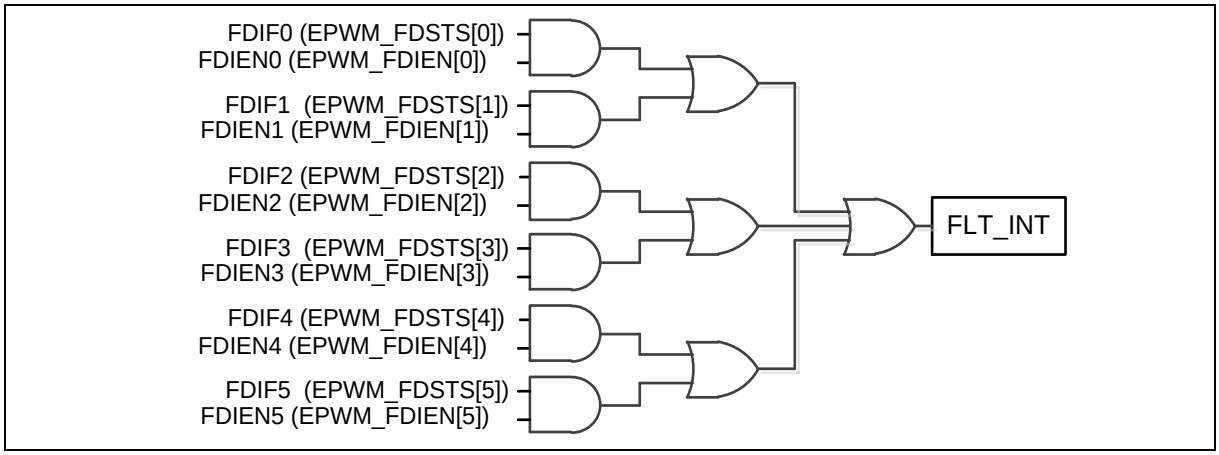


Figure 6.15-40 Fault Detect Function Interrupt Architecture Diagram

6.15.5.27 EPWM Trigger EADC/DAC Generator

EPWM can be one of the EADC conversion trigger source. Each EPWM pair channels share the same trigger source. Setting TRGSELn bit of EPWM_EADCTS0 and EPWM_EADCTS1 registers is to select the trigger sources, where TRGSELn bit is TRGSEL0, TRGSEL1, …, and TRGSEL5, which are located in EPWM_EADCTS0[3:0], EPWM_EADCTS0[11:8], EPWM_EADCTS0[19:16], EPWM_EADCTS0[27:24], EPWM_EADCTS1[3:0] and EPWM_EADCTS1[11:8], respectively. Setting TRGENn bit of EPWM_EADCTS0 and EPWM_EADCTS1 registers is to enable the trigger output to EADC, where TRGENn bit is TRGEN0, TRGEN1, …, TRGEN5, which are located in EPWM_EADCTS0[7], EPWM_EADCTS0[15], EPWM_EADCTS0[23], EPWM_EADCTS0[31], EPWM_EADCTS1[7] and EPWM_EADCTS1[15], respectively. The number n (n = 0,1, ..,5) denotes EPWM channel number.

There are 16 EPWM events can be selected as the trigger source for one pair of channels which shown in Figure 6.15-41. Figure 6.15-42 shows trigger EADC block diagram with prescaler. By setting PSCENn (EPWM_EADCPSCCTL[n], n=0~5) bit to 1, EPWM will trigger EADC when the number of trigger events is equal to EADCPSCn+1 (n=0~5). Reading PSCNTn (n=0~5) can know how many events happened. If PSCENn (n=0~5) is 0, user also can write initial data to PSCNTn (n=0~5) and pre-scale counter will start from this data. If PSCENn (n=0~5) is set from 1 to 0, PSCNTn (n=0~5) will reset to 0. Figure 6.15-43 is the trigger EADC timing waveform in the up-down counter type.

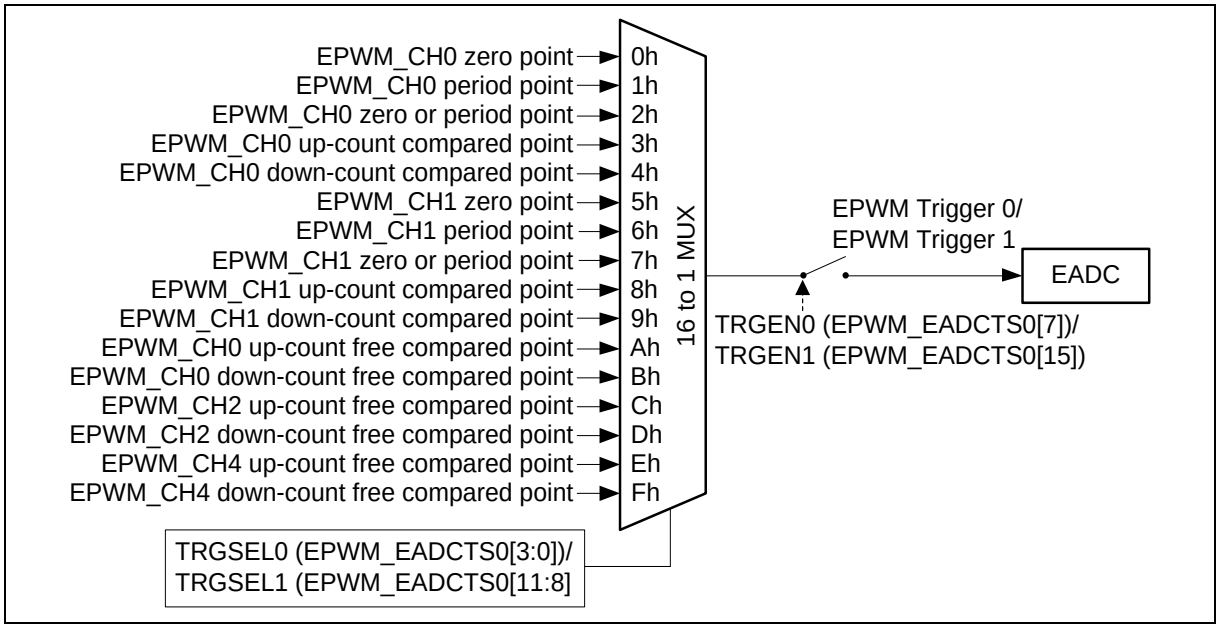


Figure 6.15-41 EPWMx_CH0 and EPWMx_CH1 Pair Trigger EADC Events

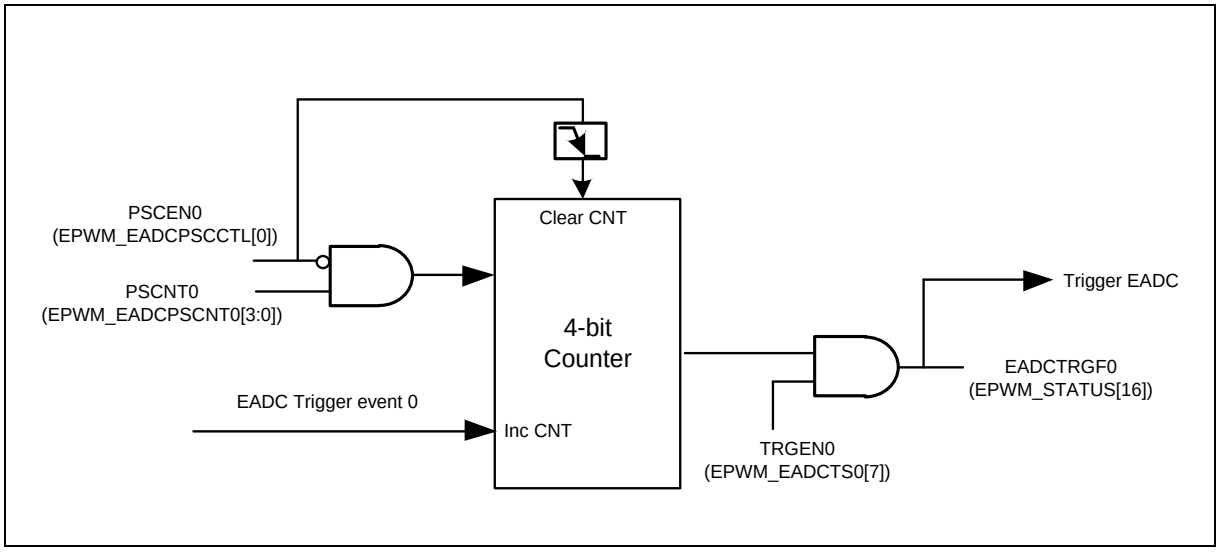


Figure 6.15-42 EPWMx_CH0 Trigger EADC Block Diagram

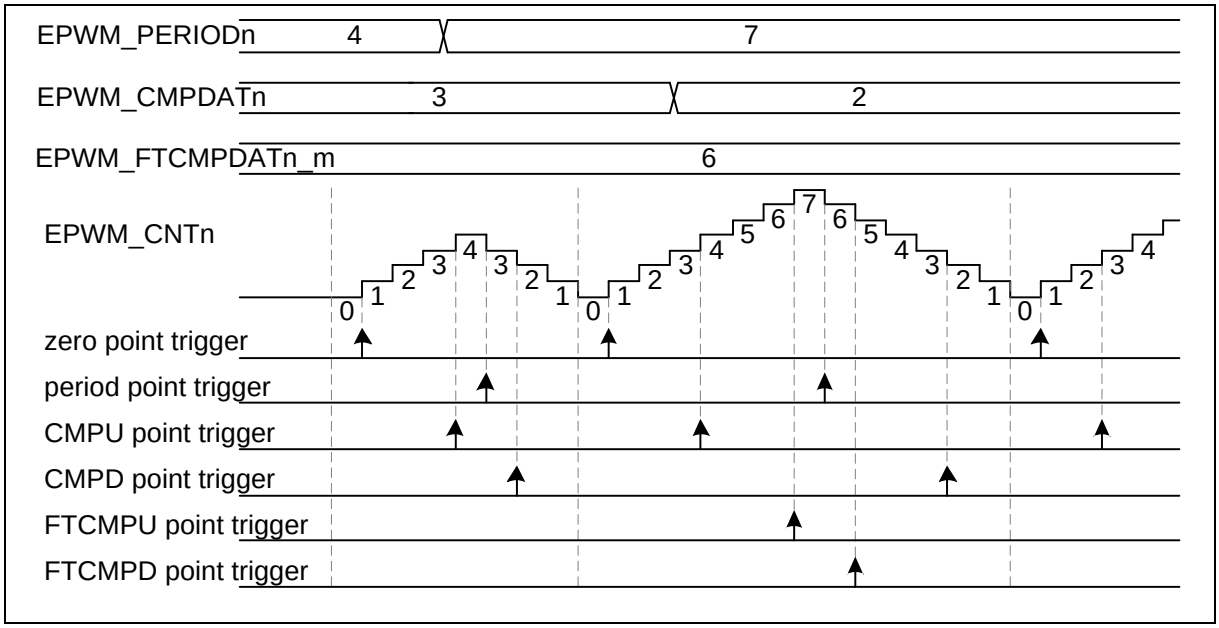


Figure 6.15-43 EPWM Trigger EADC in Up-Down Counter Type Timing Waveform

EPWM can also be used to trigger DAC conversion. Each EPWM pair channel (CH0 and CH1, CH2 and CH3, CH4 and CH5) generates a trigger signal. Using the EPWM Trigger DAC Enable Register (EPWM_DACTRGEN) can decide at which points to trigger DAC. The timing of the EPWM triggering DAC is similar to those for triggering EADC. However, DAC triggering function does not include the triggering events from comparison with FTCMPDAT, that is, there are no trigger points the same as FTCMPU and FTCMPD which are shown in EADC triggering.

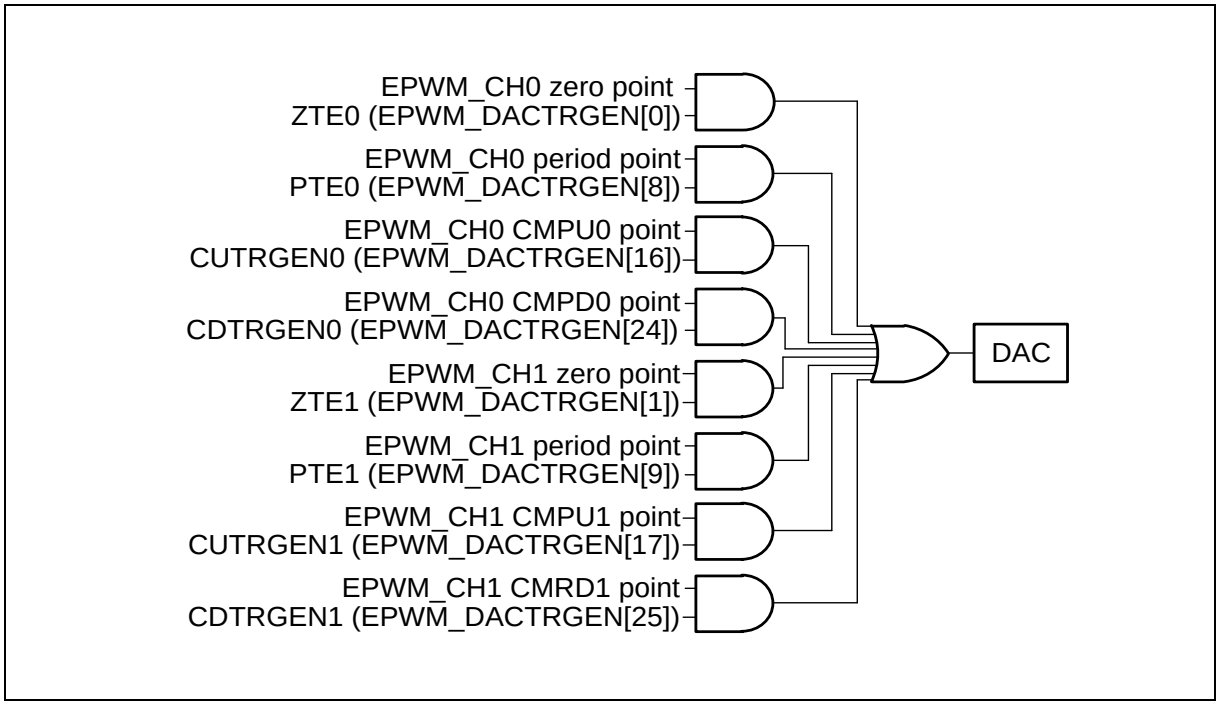


Figure 6.15-44 EPWM_CH0 and EPWM_CH1 Pair Trigger DAC Block Diagram

6.15.5.28 Capture Operation

The channels of the capture input and the EPWM output share the same pin and counter. The counter can operating in up or down counter type. The capture function will always latch the EPWM counter to the RCAPDATn (EPWM_RCAPDATn[15:0]) bits or the FCAPDATn (EPWM_FCAPDATn[15:0]) bits, if the input channel has a rising transition or a falling transition, respectively. The capture function will also generate an interrupt CAP_INT (using EPWM_INT vector) if the rising or falling latch occurs and the corresponding channel n's rising or falling interrupt enable bits are set, where the CAPRIENn (EPWM_CAPIEN[5:0]) bit is for the rising edge and the CAPFIENn (EPWM_CAPIEN[13:8]) bit is for the falling edge. When rising or falling latch occurs, the corresponding EPWM counter may be reloaded with the value of EPWM_PERIODn register, depending on the setting of RCRLDENn or FCRLDENn bits (where RCRLDENn and FCRLDENn are located at EPWM_CAPCTL[21:16] and EPWM_CAPCTL[29:24], respectively). Note that the corresponding GPIO pins must be configured as the capture function by enable the CAPINENn (EPWM_CAPINEN[5:0]) bits for the corresponding capture channel n. Figure 6.15-45 is the capture block diagram of channel 0.

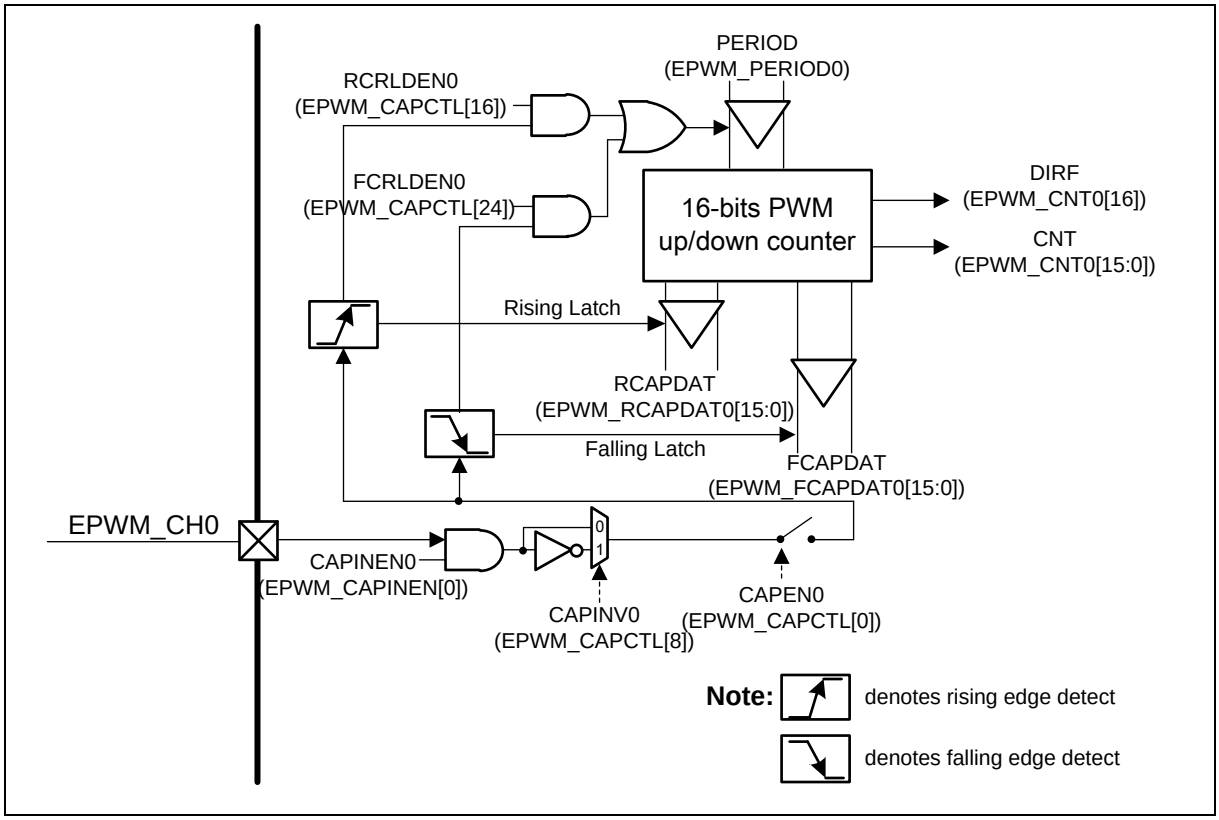


Figure 6.15-45 EPWM_CH0 Capture Block Diagram

Figure 6.15-46 illustrates the capture function timing. In this case, the capture counter is set as EPWM down counter type and the PERIOD is set to 8 so that the counter counts in the down direction, from 8 to 0. When detecting a falling edge at the capture input pin, the capture function latches the counter value to the EPWM_FCAPDATn register. When detecting the rising edge, it latches the counter value to the EPWM_RCAPDATn register. In this timing diagram, when the falling edge is detected at the first time, the capture function will reload the counter value from the PERIOD setting because the FCRLDENn bit is enabled. But at the second time, the falling edge does not result in a reload because of the disabled FCRLDENn bit. In this example, the counter also reloads at the rising edge of the capture input because the RCRLDENn bit is enabled, too.

Moreover, if the case is setup as the up counter type, the counter will reload the value zero and count

up to the value PERIOD.

Figure 6.15-46 also illustrates the timing example for the interrupt and interrupt flag generation. When the rising edge at channel n is detected, the corresponding CRLIFn (EPWM_CAPIF[5:0]) bit is set by hardware. Similarly, a falling edge detection at channel n causes the corresponding CFLIFn (EPWM_CAPIF[13:8]) bit is set by hardware. CRLIFn and CFLIFn bits can be cleared by software by writing '1'. If the CRLIFn bit is set and the CAPRIENn bit is enabled, the capture function generates an interrupt. If the CFLIFn bit is set and the CAPFIENn bit is enabled, the interrupt also happens.

A condition which is not shown in this figure is: if the rising latch happens again when the CRLIFn bit is already set, the Overrun status CRLIFOVn (EPWM_CAPSTS[5:0]) bit will be set to 1 by hardware to indicate the CRLIF flag overrunning. Also, if the falling latch happens again, the same hardware operation occurs for the CFLIF interrupt flag and the Overrun status CFLIFOVn (EPWM_CAPSTS[13:8]).

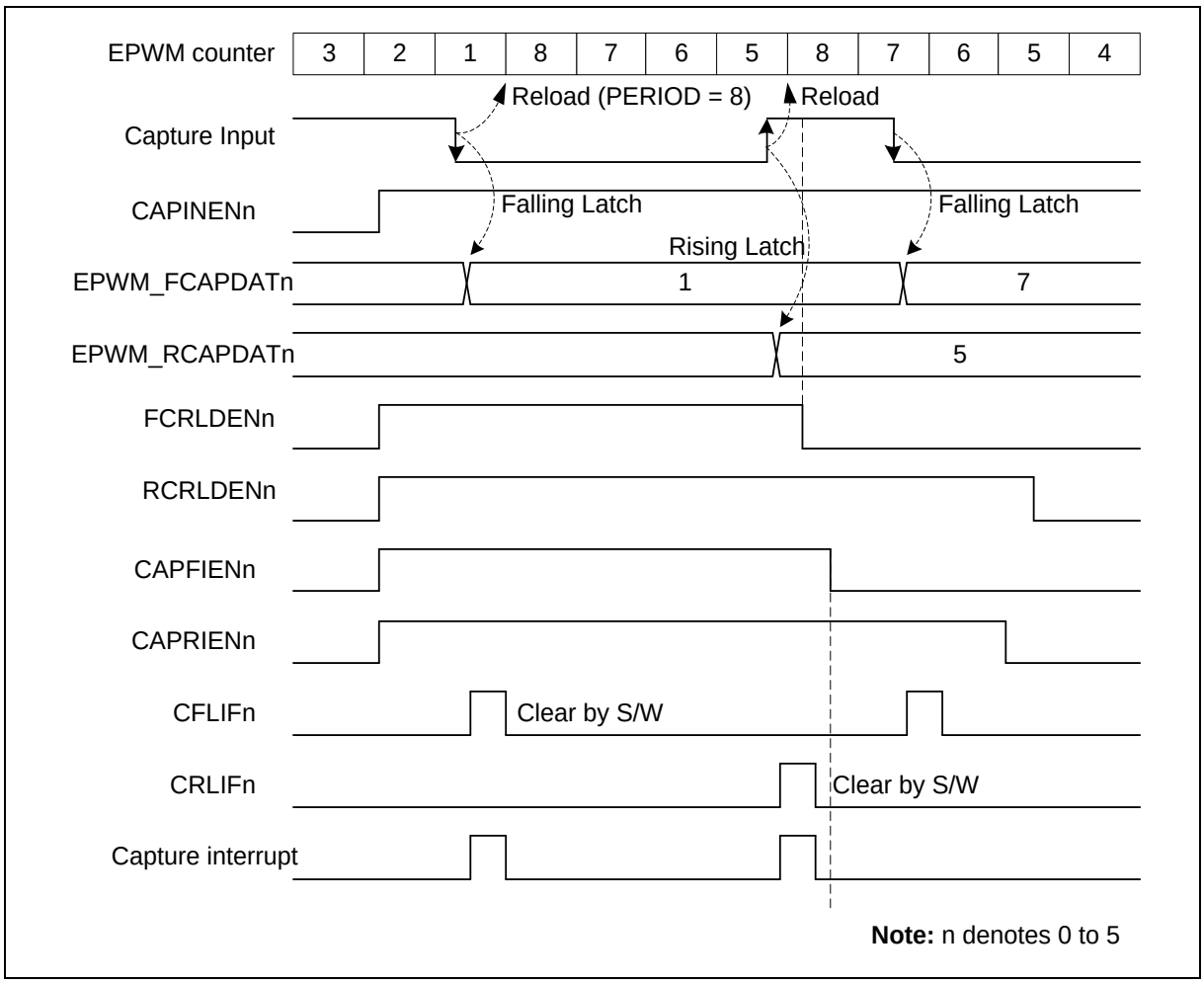


Figure 6.15-46 Capture Operation Waveform

The capture pulse width meeting the following conditions can be calculated according to the formula.

1. The capture positive or negative pulse width is shorter than a counter period.
2. The counter operates in down counter type.
3. The counter can be reloaded by both falling and rising capture events through setting FCRLDENn and RCRLDENn bits of PWM_CAPCTL register to 1.

For the negative pulse case, the channel low pulse width is calculated as (EPWM_PERIODn + 1 - EPWM_RCAPDATn) EPWM counter time, where one EPWM counter time is (CLKPSC+1) * EPWMx_CLK clock time. In Figure 6.15-46, the low pulse width is 8+1-5 = 4 EPWM counter time.

For the positive pulse case, the channel high pulse width is calculated as (EPWM_PERIODn + 1 - EPWM_FCAPDATn) EPWM counter time, where one EPWM counter time is (CLKPSC+1) * EPWMx_CLK clock time. In Figure 6.15-46, the high pulse width is 8+1-7 = 2 EPWM counter time.

6.15.5.29 Capture PDMA Function

The EPWM module supports the PDMA transfer function when operating in the capture mode. When the corresponding PDMA enable bit CHENnm (CHEN01 at EPWM_PDMACTL[0], CHEN23 at EPWM_PDMACTL[8] and CHEN45 at EPWM_PDMACTL[16], where n and m denote complement pair channels) is set, the capture module will issue a request to PDMA controller when the preceding capture event has happened. The PDMA controller will issue an acknowledgement to the capture module after it has read back the CAPBUF (EPWM_PDMACAPn_m[15:0], n, m denotes complement pair channels) register in the capture module and has sent the register value to the memory. By setting CAPMODnm (CAPMOD01 at EPWM_PDMACTL[2:1], CAPMOD23 at EPWM_PDMACTL[10:9] and CAPMOD45 at EPWM_PDMACTL[18:17]) bits, the PDMA can transfer the rising edge captured data or falling edge captured data or both of them to the memory. When using the PDMA to transfer both of the falling and rising edge data, remember to set CAPORDnm (CAPORD01 at EPWM_PDMACTL[3], CAPORD23 at EPWM_PDMACTL[11] and CAPORD45 at EPWM_PDMACTL[19]) bit to decide the order of the transferred data (falling edge captured is first or rising edge captured first). The complement pair channels share a PDMA channel. Therefore, a selection bit CHSELnm (CHSEL01 (EPWM_PDMACTL[4]), CHSEL23 (EPWM_PDMACTL[12]) and CHSEL45 (EPWM_PDMACTL[20])) bit is used to decide either channel n or channel m can be serviced by the PDMA channel.

Figure 6.15-47 is capture PDMA waveform. In this case, the CHSEL01 (EPWM_PDMACTL[4]) bit is set to 0. Hence the PDMA will service channel 0 for the capture data transfer. CAPMOD01 (EPWM_PDMACTL[2:1]) bits are set to 3. That means both of the rising and falling edge captured data will be transferred to the memory. The CAPORD01 (EPWM_PDMACTL[3]) bit is set to 1, so the rising edge data will be the first data to transfer and following is the falling edge data to transfer. As shown in Figure 6.15-47, the last assertions of the CRLIF0 and CFLIF0 signal have some overlap. The value of EPWM_RCAPDAT0 register is 11 will be loaded to EPWM_PDMACAP0_1 register to wait for transfer but not the EPWM_FCAPDAT0 value. The EPWM_PDMACAP0_1 register saves the data which will be transferred to the memory by PDMA. The HWDATA in this figure denotes the data which are being transferred by PDMA.

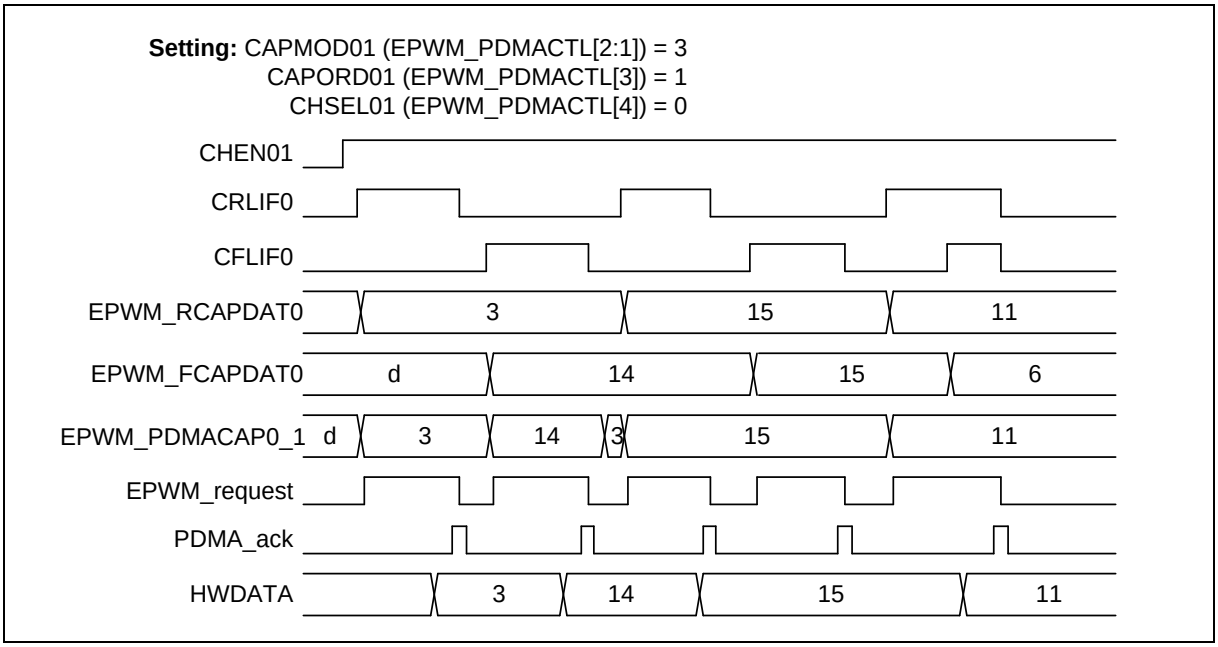


Figure 6.15-47 Capture PDMA Operation Waveform of Channel 0

6.15.5.30 Accumulator PDMA Function

The EPWM module supports the PDMA transfer function when accumulator interrupt happened. Figure 6.15-48 shows accumulator PDMA function architecture. When the corresponding PDMA enable bit APDMAENn (EPWM_APDMACTL[n], n=0~5) is set, accumulator module will send a request to PDMA controller when accumulator interrupt has happened, meaning that IFAIFn (EPWM_AINTSTS[n], n=0~5) is set 1. The PDMA controller will issue an acknowledge to accumulator after it has read memory data and send the data to the particular register (EPWM_PERIODn, etc.). So, user can use this function to change accumulator interrupt frequency.

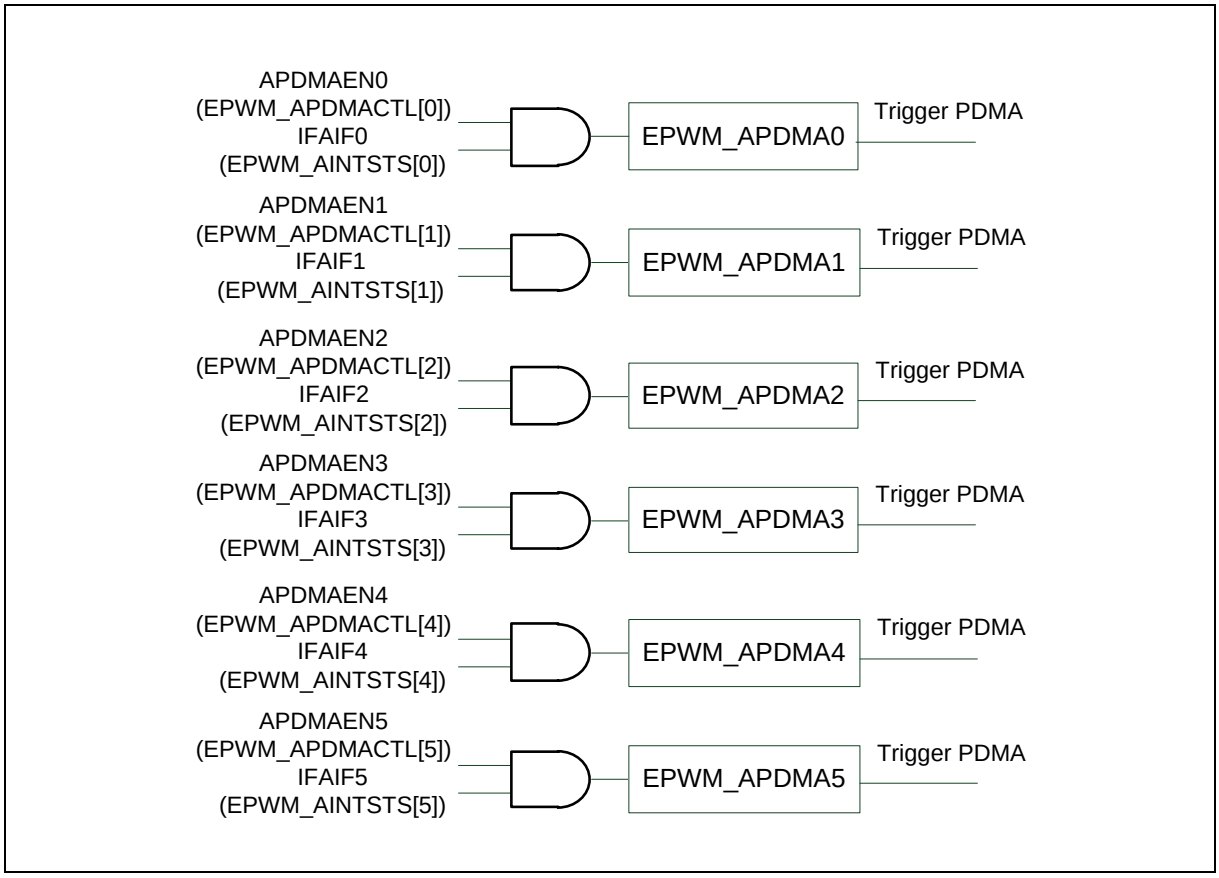


Figure 6.15-48 Accumulator PDMA Function Architecture

6.15.5.31 Accumulator Stop Mode

The EPWM module supports the Accumulator Stop Mode to stop counting when accumulator interrupt happened. Figure 6.15-49 shows Accumulator Stop Mode waveform. When the corresponding STPMOD (EPWM_IFAn[24], n=0~5) is set, accumulator module will disable CNTENn (EPWM_CNTEN[n], n=0~5) after counter finishes whole period.

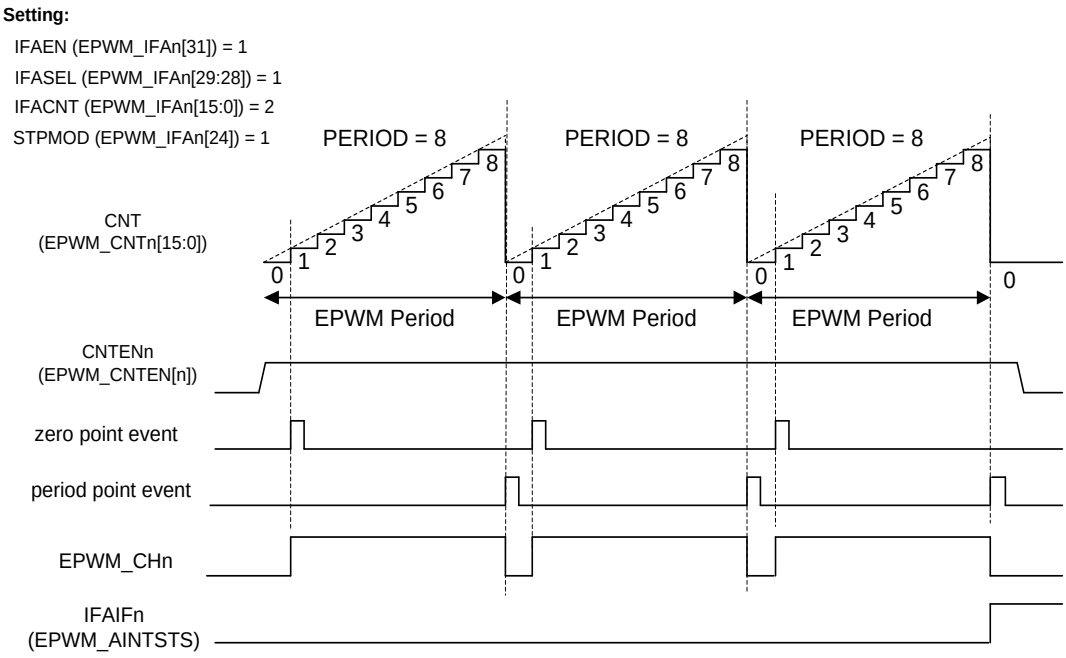


Figure 6.15-49 EPWM Accumulator Stop Mode Waveform

6.15.5.32 Fault Detect Function

The EPWM module supports Fault Detect Function to detect output short condition. Figure 6.15-50 shows fault detect function architecture. When the corresponding channel fault detect enable bit FDENn(EPWM_FDEN[n]) is set to 1, two statement will start detection. One is that EPWM output becomes from low to high or from high to low. The other is that POENn (EPWM_POEN[n], n=0~5) becomes from 0 to 1 or from 1 to 0.

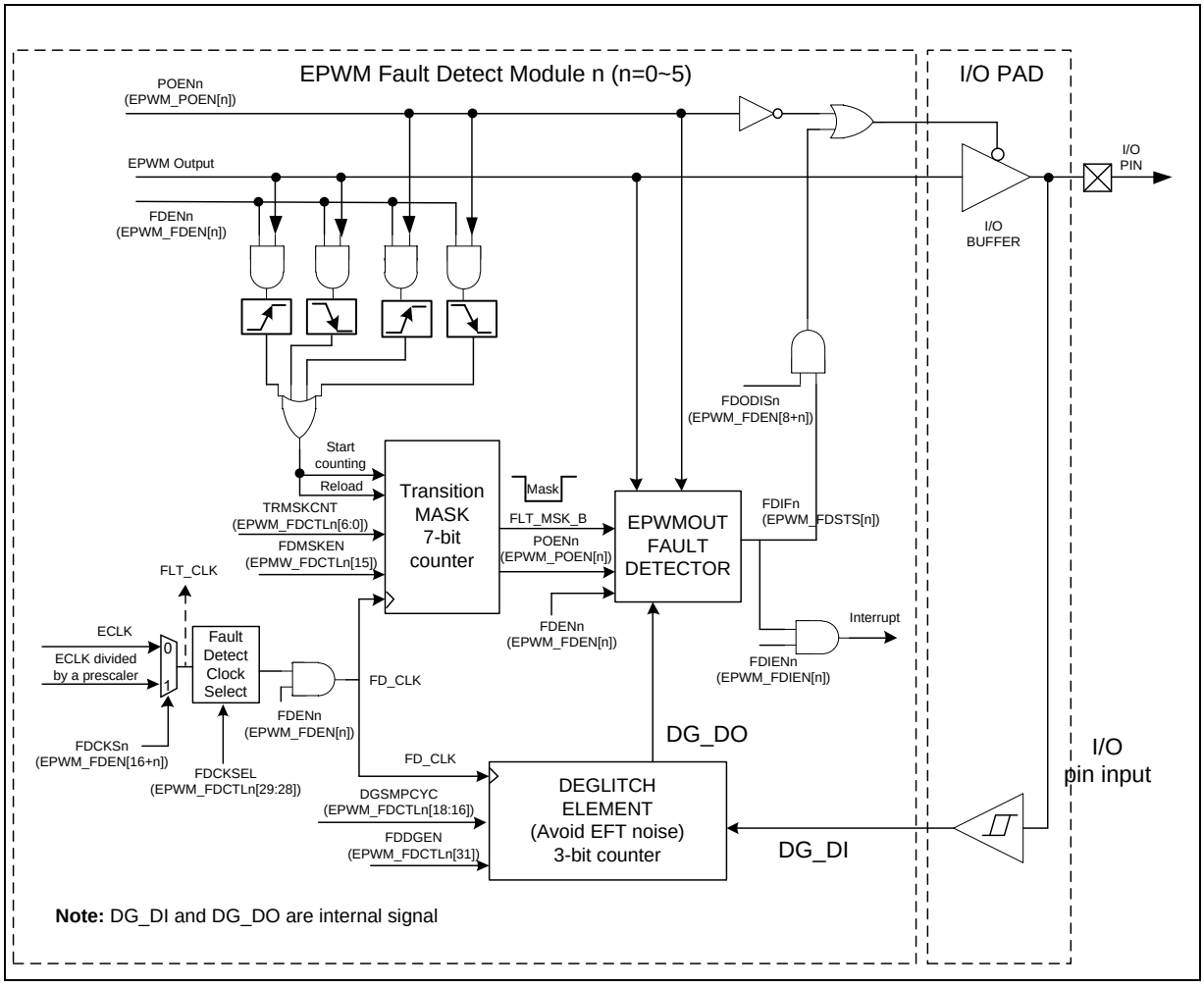


Figure 6.15-50 Fault Detect Function Architecture

Figure 6.15-51 shows Mask Function waveform example. Setting FDMSKEN (EPWM_FDCTLn[15], n=0~5) will enable the MASK Function. The detection will disable in the beginning until mask counter count from 0 to TRMSKCNT (EPWM_FDCTLn[6:0], n=0~5).

Figure 6.15-52 shows Deglitch Function waveform example. Setting FDDGEN (EPWM_FDCTLn[31], n=0~5) will enable Deglitch Function. The feedback signal will be sampled DGSMPCYC (EPWM_FDCTLn[18:16], n=0~5) +1 times and become valid input to be used for detection. FDCKSn (EPWM_FDEN[n+16], n=0~5) and FDCKSEL (EPWM_FDCTLn[29:28]) are clock setting for MASK and Dglitch Function. For more detail, please see the register discription.

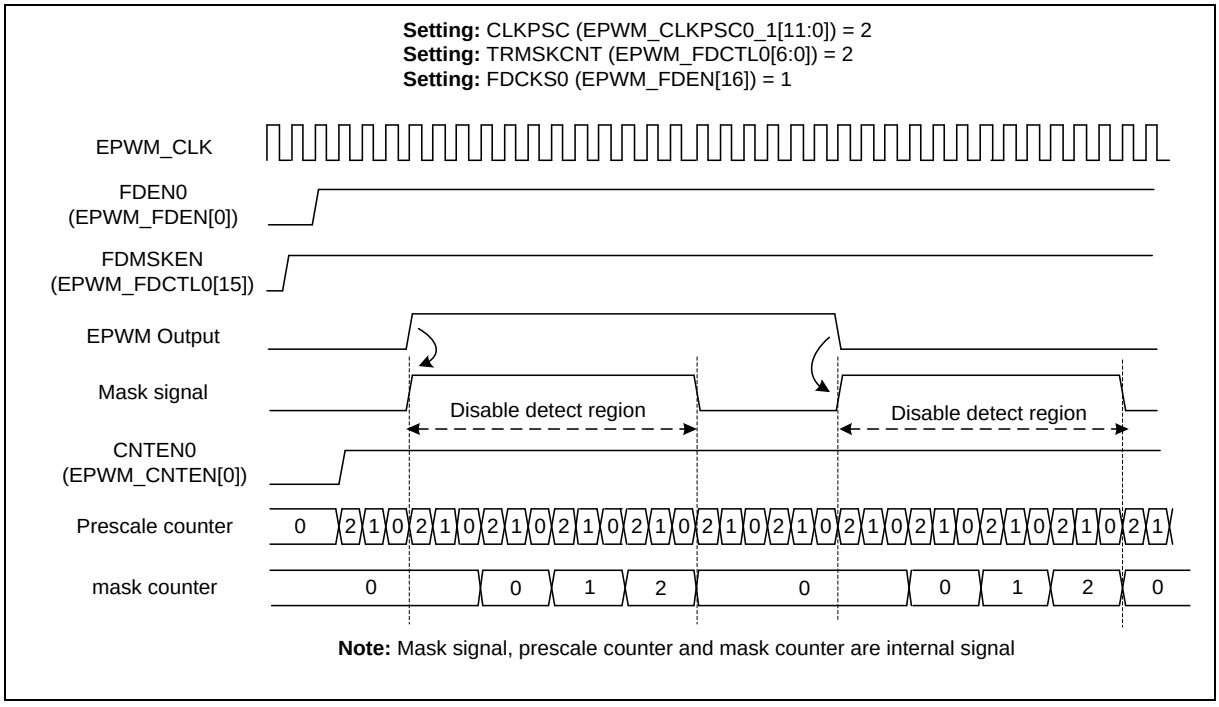


Figure 6.15-51 EPWM Mask Function Waveform

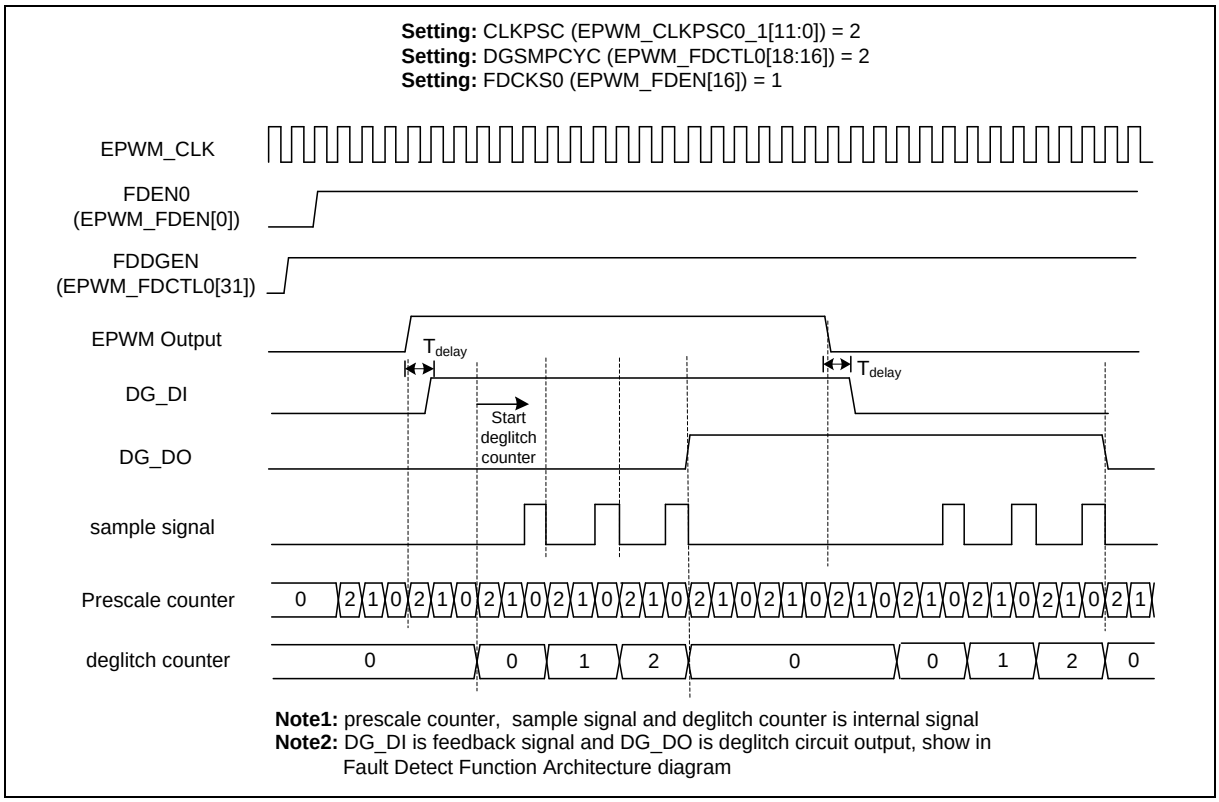


Figure 6.15-52 EPWM Deglitch Function Waveform

6.15.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
EPWM Base Address: **EPWM0_BA = 0x4005_8000** **EPWM1_BA = 0x4005_9000** **EPWM non-secure base address is EPWM0_BA + 0x1000_0000** **EPWM non-secure base address is EPWM1_BA + 0x1000_0000**				
EPWM_CTL0 **x=0,1**	EPWMx_BA+0x00	R/W	EPWM Control Register 0	0x0000_0000
EPWM_CTL1 **x=0,1**	EPWMx_BA+0x04	R/W	EPWM Control Register 1	0x0000_0000
EPWM_SYNC **x=0,1**	EPWMx_BA+0x08	R/W	EPWM Synchronization Register	0x0000_0000
EPWM_SWSYNC **x=0,1**	EPWMx_BA+0x0C	R/W	EPWM Software Control Synchronization Register	0x0000_0000
EPWM_CLKSRC **x=0,1**	EPWMx_BA+0x10	R/W	EPWM Clock Source Register	0x0000_0000
EPWM_CLKPSC0_1 **x=0,1**	EPWMx_BA+0x14	R/W	EPWM Clock Prescale Register 0/1	0x0000_0000
EPWM_CLKPSC2_3 **x=0,1**	EPWMx_BA+0x18	R/W	EPWM Clock Prescale Register 2/3	0x0000_0000
EPWM_CLKPSC4_5 **x=0,1**	EPWMx_BA+0x1C	R/W	EPWM Clock Prescale Register 4/5	0x0000_0000
EPWM_CNTEN **x=0,1**	EPWMx_BA+0x20	R/W	EPWM Counter Enable Register	0x0000_0000
EPWM_CNTCLR **x=0,1**	EPWMx_BA+0x24	R/W	EPWM Clear Counter Register	0x0000_0000
EPWM_LOAD **x=0,1**	EPWMx_BA+0x28	R/W	EPWM Load Register	0x0000_0000
EPWM_PERIOD0 **x=0,1**	EPWMx_BA+0x30	R/W	EPWM Period Register 0	0x0000_0000
EPWM_PERIOD1 **x=0,1**	EPWMx_BA+0x34	R/W	EPWM Period Register 1	0x0000_0000
EPWM_PERIOD2 **x=0,1**	EPWMx_BA+0x38	R/W	EPWM Period Register 2	0x0000_0000
EPWM_PERIOD3 **x=0,1**	EPWMx_BA+0x3C	R/W	EPWM Period Register 3	0x0000_0000

EPWM_PERIOD4 x=0,1	EPWMx_BA+0x40	R/W	EPWM Period Register 4	0x0000_0000
EPWM_PERIOD5 x=0,1	EPWMx_BA+0x44	R/W	EPWM Period Register 5	0x0000_0000
EPWM_CMPDAT0 x=0,1	EPWMx_BA+0x50	R/W	EPWM Comparator Register 0	0x0000_0000
EPWM_CMPDAT1 x=0,1	EPWMx_BA+0x54	R/W	EPWM Comparator Register 1	0x0000_0000
EPWM_CMPDAT2 x=0,1	EPWMx_BA+0x58	R/W	EPWM Comparator Register 2	0x0000_0000
EPWM_CMPDAT3 x=0,1	EPWMx_BA+0x5C	R/W	EPWM Comparator Register 3	0x0000_0000
EPWM_CMPDAT4 x=0,1	EPWMx_BA+0x60	R/W	EPWM Comparator Register 4	0x0000_0000
EPWM_CMPDAT5 x=0,1	EPWMx_BA+0x64	R/W	EPWM Comparator Register 5	0x0000_0000
EPWM_DTCTL0_1 x=0,1	EPWMx_BA+0x70	R/W	EPWM Dead-time Control Register 0/1	0x0000_0000
EPWM_DTCTL2_3 x=0,1	EPWMx_BA+0x74	R/W	EPWM Dead-time Control Register 2/3	0x0000_0000
EPWM_DTCTL4_5 x=0,1	EPWMx_BA+0x78	R/W	EPWM Dead-time Control Register 4/5	0x0000_0000
EPWM_PHS0_1 x=0,1	EPWMx_BA+0x80	R/W	EPWM Counter Phase Register 0/1	0x0000_0000
EPWM_PHS2_3 x=0,1	EPWMx_BA+0x84	R/W	EPWM Counter Phase Register 2/3	0x0000_0000
EPWM_PHS4_5 x=0,1	EPWMx_BA+0x88	R/W	EPWM Counter Phase Register 4/5	0x0000_0000
EPWM_CNT0 x=0,1	EPWMx_BA+0x90	R	EPWM Counter Register 0	0x0000_0000
EPWM_CNT1 x=0,1	EPWMx_BA+0x94	R	EPWM Counter Register 1	0x0000_0000
EPWM_CNT2 x=0,1	EPWMx_BA+0x98	R	EPWM Counter Register 2	0x0000_0000
EPWM_CNT3 x=0,1	EPWMx_BA+0x9C	R	EPWM Counter Register 3	0x0000_0000
EPWM_CNT4 x=0,1	EPWMx_BA+0xA0	R	EPWM Counter Register 4	0x0000_0000
EPWM_CNT5 x=0,1	EPWMx_BA+0xA4	R	EPWM Counter Register 5	0x0000_0000

EPWM_WGCTL0 x=0,1	EPWMx_BA+0xB0	R/W	EPWM Generation Register 0	0x0000_0000
EPWM_WGCTL1 x=0,1	EPWMx_BA+0xB4	R/W	EPWM Generation Register 1	0x0000_0000
EPWM_MSKEN x=0,1	EPWMx_BA+0xB8	R/W	EPWM Mask Enable Register	0x0000_0000
EPWM_MSK x=0,1	EPWMx_BA+0xBC	R/W	EPWM Mask Data Register	0x0000_0000
EPWM_BNF x=0,1	EPWMx_BA+0xC0	R/W	EPWM Brake Noise Filter Register	0x0000_0000
EPWM_FAILBRK x=0,1	EPWMx_BA+0xC4	R/W	EPWM System Fail Brake Control Register	0x0000_0000
EPWM_BRKCTL0_1 x=0,1	EPWMx_BA+0xC8	R/W	EPWM Brake Edge Detect Control Register 0/1	0x0000_0000
EPWM_BRKCTL2_3 x=0,1	EPWMx_BA+0xCC	R/W	EPWM Brake Edge Detect Control Register 2/3	0x0000_0000
EPWM_BRKCTL4_5 x=0,1	EPWMx_BA+0xD0	R/W	EPWM Brake Edge Detect Control Register 4/5	0x0000_0000
EPWM_POLCTL x=0,1	EPWMx_BA+0xD4	R/W	EPWM Pin Polar Inverse Register	0x0000_0000
EPWM_POEN x=0,1	EPWMx_BA+0xD8	R/W	EPWM Output Enable Register	0x0000_0000
EPWM_SWBRK x=0,1	EPWMx_BA+0xDC	W	EPWM Software Brake Control Register	0x0000_0000
EPWM_INTEN0 x=0,1	EPWMx_BA+0xE0	R/W	EPWM Interrupt Enable Register 0	0x0000_0000
EPWM_INTEN1 x=0,1	EPWMx_BA+0xE4	R/W	EPWM Interrupt Enable Register 1	0x0000_0000
EPWM_INTSTS0 x=0,1	EPWMx_BA+0xE8	R/W	EPWM Interrupt Flag Register 0	0x0000_0000
EPWM_INTSTS1 x=0,1	EPWMx_BA+0xEC	R/W	EPWM Interrupt Flag Register 1	0x0000_0000
EPWM_DACTRGEN x=0,1	EPWMx_BA+0xF4	R/W	EPWM Trigger DAC Enable Register	0x0000_0000
EPWM_EADCTS0 x=0,1	EPWMx_BA+0xF8	R/W	EPWM Trigger EADC Source Select Register 0	0x0000_0000
EPWM_EADCTS1 x=0,1	EPWMx_BA+0xFC	R/W	EPWM Trigger EADC Source Select Register 1	0x0000_0000
EPWM_FTCMPDAT0_1 x=0,1	EPWMx_BA+0x100	R/W	EPWM Free Trigger Compare Register 0/1	0x0000_0000

EPWM_FTCMPDAT2_3 x=0,1	EPWMx_BA+0x104	R/W	EPWM Free Trigger Compare Register 2/3	0x0000_0000
EPWM_FTCMPDAT4_5 x=0,1	EPWMx_BA+0x108	R/W	EPWM Free Trigger Compare Register 4/5	0x0000_0000
EPWM_SSCTL x=0,1	EPWMx_BA+0x110	R/W	EPWM Synchronous Start Control Register	0x0000_0000
EPWM_SSTRG x=0,1	EPWMx_BA+0x114	W	EPWM Synchronous Start Trigger Register	0x0000_0000
EPWM_LEBCTL x=0,1	EPWMx_BA+0x118	R/W	EPWM Leading Edge Blanking Control Register	0x0000_0000
EPWM_LEBCNT x=0,1	EPWMx_BA+0x11C	R/W	EPWM Leading Edge Blanking Counter Register	0x0000_0000
EPWM_STATUS x=0,1	EPWMx_BA+0x120	R/W	EPWM Status Register	0x0000_0000
EPWM_IFA0 x=0,1	EPWMx_BA+0x130	R/W	EPWM Interrupt Flag Accumulator Register 0	0x0000_0000
EPWM_IFA1 x=0,1	EPWMx_BA+0x134	R/W	EPWM Interrupt Flag Accumulator Register 1	0x0000_0000
EPWM_IFA2 x=0,1	EPWMx_BA+0x138	R/W	EPWM Interrupt Flag Accumulator Register 2	0x0000_0000
EPWM_IFA3 x=0,1	EPWMx_BA+0x13C	R/W	EPWM Interrupt Flag Accumulator Register 3	0x0000_0000
EPWM_IFA4 x=0,1	EPWMx_BA+0x140	R/W	EPWM Interrupt Flag Accumulator Register 4	0x0000_0000
EPWM_IFA5 x=0,1	EPWMx_BA+0x144	R/W	EPWM Interrupt Flag Accumulator Register 5	0x0000_0000
EPWM_AINTSTS x=0,1	EPWMx_BA+0x150	R/W	EPWM Accumulator Interrupt Flag Register	0x0000_0000
EPWM_AINTEN x=0,1	EPWMx_BA+0x154	R/W	EPWM Accumulator Interrupt Enable Register	0x0000_0000
EPWM_APDMACTL x=0,1	EPWMx_BA+0x158	R/W	EPWM Accumulator PDMA Control Register	0x0000_0000
EPWM_FDEN x=0,1	EPWMx_BA+0x160	R/W	EPWM Fault Detect Enable Register	0x0000_0000
EPWM_FDCTL0 x=0,1	EPWMx_BA+0x164	R/W	EPWM Fault Detect Control Register 0	0x0000_0000
EPWM_FDCTL1 x=0,1	EPWMx_BA+0x168	R/W	EPWM Fault Detect Control Register 1	0x0000_0000
EPWM_FDCTL2 x=0,1	EPWMx_BA+0x16C	R/W	EPWM Fault Detect Control Register 2	0x0000_0000

EPWM_FDCTL3 x=0,1	EPWMx_BA+0x170	R/W	EPWM Fault Detect Control Register 3	0x0000_0000
EPWM_FDCTL4 x=0,1	EPWMx_BA+0x174	R/W	EPWM Fault Detect Control Register 4	0x0000_0000
EPWM_FDCTL5 x=0,1	EPWMx_BA+0x178	R/W	EPWM Fault Detect Control Register 5	0x0000_0000
EPWM_FDIEN x=0,1	EPWMx_BA+0x17C	R/W	EPWM Fault Detect Interrupt Enable Register	0x0000_0000
EPWM_FDSTS x=0,1	EPWMx_BA+0x180	R/W	EPWM Fault Detect Interrupt Flag Register	0x0000_0000
EPWM_EADCPSCCTL x=0,1	EPWMx_BA+0x184	R/W	EPWM Trigger EADC Prescale Control Register	0x0000_0000
EPWM_EADCPSC0 x=0,1	EPWMx_BA+0x188	R/W	EPWM Trigger EADC Prescale Register 0	0x0000_0000
EPWM_EADCPSC1 x=0,1	EPWMx_BA+0x18C	R/W	EPWM Trigger EADC Prescale Register 1	0x0000_0000
EPWM_EADCPSCNT0 x=0,1	EPWMx_BA+0x190	R/W	EPWM Trigger EADC Prescale Counter Register 0	0x0000_0000
EPWM_EADCPSCNT1 x=0,1	EPWMx_BA+0x194	R/W	EPWM Trigger EADC Prescale Counter Register 1	0x0000_0000
EPWM_CAPINEN x=0,1	EPWMx_BA+0x200	R/W	EPWM Capture Input Enable Register	0x0000_0000
EPWM_CAPCTL x=0,1	EPWMx_BA+0x204	R/W	EPWM Capture Control Register	0x0000_0000
EPWM_CAPSTS x=0,1	EPWMx_BA+0x208	R	EPWM Capture Status Register	0x0000_0000
EPWM_RCAPDAT0 x=0,1	EPWMx_BA+0x20C	R	EPWM Rising Capture Data Register 0	0x0000_0000
EPWM_FCAPDAT0 x=0,1	EPWMx_BA+0x210	R	EPWM Falling Capture Data Register 0	0x0000_0000
EPWM_RCAPDAT1 x=0,1	EPWMx_BA+0x214	R	EPWM Rising Capture Data Register 1	0x0000_0000
EPWM_FCAPDAT1 x=0,1	EPWMx_BA+0x218	R	EPWM Falling Capture Data Register 1	0x0000_0000
EPWM_RCAPDAT2 x=0,1	EPWMx_BA+0x21C	R	EPWM Rising Capture Data Register 2	0x0000_0000
EPWM_FCAPDAT2 x=0,1	EPWMx_BA+0x220	R	EPWM Falling Capture Data Register 2	0x0000_0000
EPWM_RCAPDAT3 x=0,1	EPWMx_BA+0x224	R	EPWM Rising Capture Data Register 3	0x0000_0000

EPWM_FCAPDAT3 x=0,1	EPWMx_BA+0x228	R	EPWM Falling Capture Data Register 3	0x0000_0000
EPWM_RCAPDAT4 x=0,1	EPWMx_BA+0x22C	R	EPWM Rising Capture Data Register 4	0x0000_0000
EPWM_FCAPDAT4 x=0,1	EPWMx_BA+0x230	R	EPWM Falling Capture Data Register 4	0x0000_0000
EPWM_RCAPDAT5 x=0,1	EPWMx_BA+0x234	R	EPWM Rising Capture Data Register 5	0x0000_0000
EPWM_FCAPDAT5 x=0,1	EPWMx_BA+0x238	R	EPWM Falling Capture Data Register 5	0x0000_0000
EPWM_PDMACTL x=0,1	EPWMx_BA+0x23C	R/W	EPWM PDMA Control Register	0x0000_0000
EPWM_PDMACAP0_1 x=0,1	EPWMx_BA+0x240	R	EPWM Capture Channel 01 PDMA Register	0x0000_0000
EPWM_PDMACAP2_3 x=0,1	EPWMx_BA+0x244	R	EPWM Capture Channel 23 PDMA Register	0x0000_0000
EPWM_PDMACAP4_5 x=0,1	EPWMx_BA+0x248	R	EPWM Capture Channel 45 PDMA Register	0x0000_0000
EPWM_CAPIEN x=0,1	EPWMx_BA+0x250	R/W	EPWM Capture Interrupt Enable Register	0x0000_0000
EPWM_CAPIF x=0,1	EPWMx_BA+0x254	R/W	EPWM Capture Interrupt Flag Register	0x0000_0000
EPWM_PBUF0 x=0,1	EPWMx_BA+0x304	R	EPWM PERIOD0 Buffer	0x0000_0000
EPWM_PBUF1 x=0,1	EPWMx_BA+0x308	R	EPWM PERIOD1 Buffer	0x0000_0000
EPWM_PBUF2 x=0,1	EPWMx_BA+0x30C	R	EPWM PERIOD2 Buffer	0x0000_0000
EPWM_PBUF3 x=0,1	EPWMx_BA+0x310	R	EPWM PERIOD3 Buffer	0x0000_0000
EPWM_PBUF4 x=0,1	EPWMx_BA+0x314	R	EPWM PERIOD4 Buffer	0x0000_0000
EPWM_PBUF5 x=0,1	EPWMx_BA+0x318	R	EPWM PERIOD5 Buffer	0x0000_0000
EPWM_CMPBUF0 x=0,1	EPWMx_BA+0x31C	R	EPWM CMPDAT0 Buffer	0x0000_0000
EPWM_CMPBUF1 x=0,1	EPWMx_BA+0x320	R	EPWM CMPDAT1 Buffer	0x0000_0000
EPWM_CMPBUF2 x=0,1	EPWMx_BA+0x324	R	EPWM CMPDAT2 Buffer	0x0000_0000

EPWM_CMPBUF3 **x=0,1**	EPWMx_BA+0x328	R	EPWM CMPDAT3 Buffer	0x0000_0000
EPWM_CMPBUF4 **x=0,1**	EPWMx_BA+0x32C	R	EPWM CMPDAT4 Buffer	0x0000_0000
EPWM_CMPBUF5 **x=0,1**	EPWMx_BA+0x330	R	EPWM CMPDAT5 Buffer	0x0000_0000
EPWM_CPSCBUF0_1 **x=0,1**	EPWMx_BA+0x334	R	EPWM CLKPSC0_1 Buffer	0x0000_0000
EPWM_CPSCBUF2_3 **x=0,1**	EPWMx_BA+0x338	R	EPWM CLKPSC2_3 Buffer	0x0000_0000
EPWM_CPSCBUF4_5 **x=0,1**	EPWMx_BA+0x33C	R	EPWM CLKPSC4_5 Buffer	0x0000_0000
EPWM_FTCBUF0_1 **x=0,1**	EPWMx_BA+0x340	R	EPWM FTCMPDAT0_1 Buffer	0x0000_0000
EPWM_FTCBUF2_3 **x=0,1**	EPWMx_BA+0x344	R	EPWM FTCMPDAT2_3 Buffer	0x0000_0000
EPWM_FTCBUF4_5 **x=0,1**	EPWMx_BA+0x348	R	EPWM FTCMPDAT4_5 Buffer	0x0000_0000
EPWM_FTCI **x=0,1**	EPWMx_BA+0x34C	R/W	EPWM FTCMPDAT Indicator Register	0x0000_0000

6.15.7 Register Description

EPWM Control Register 0 (EPWM_CTL0)

Register	Offset	R/W	Description	Reset Value
EPWM_CTL0	EPWMx_BA+0x00	R/W	EPWM Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
DBGTRIOFF	DBGHALT	Reserved					GROUPEN
23	22	21	20	19	18	17	16
Reserved		IMMLDEN5	IMMLDEN4	IMMLDEN3	IMMLDEN2	IMMLDEN1	IMMLDEN0
15	14	13	12	11	10	9	8
Reserved		WINLDEN5	WINLDEN4	WINLDEN3	WINLDEN2	WINLDEN1	WINLDEN0
7	6	5	4	3	2	1	0
Reserved		CTRLD5	CTRLD4	CTRLD3	CTRLD2	CTRLD1	CTRLD0

Bits	Description	
[31]	**DBGTRIOFF**	**ICE Debug Mode Acknowledge Disable Bit (Write Protect)** 0 = ICE debug mode acknowledgement effects EPWM output. EPWM pin will be forced as tri-state while ICE debug mode acknowledged. 1 = ICE debug mode acknowledgement disabled. EPWM pin will keep output no matter ICE debug mode acknowledged or not. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[30]	**DBGHALT**	**ICE Debug Mode Counter Halt (Write Protect)** If counter halt is enabled, EPWM all counters will keep current value until exit ICE debug mode. 0 = ICE debug mode counter halt Disabled. 1 = ICE debug mode counter halt Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[29:25]	**Reserved**	Reserved.
[24]	**GROUPEN**	**Group Function Enable Bit** 0 = The output waveform of each EPWM channel are independent. 1 = Unify the EPWM_CH2 and EPWM_CH4 to output the same waveform as EPWM_CH0 and unify the EPWM_CH3 and EPWM_CH5 to output the same waveform as EPWM_CH1.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**IMMLDENn**	**Immediately Load Enable Bits** 0 = PERIOD will load to PBUF at the end point of each period. CMP will load to CMPBUF at the end point or center point of each period by setting CTRLD bit. 1 = PERIOD/CMP will load to PBUF and CMPBUF immediately when software update PERIOD/CMP. **Note:** If IMMLDENn is enabled, WINLDENn and CTRLDn will be invalid.
[15:14]	**Reserved**	Reserved.
[8+n]	**WINLDENn**	**Window Load Enable Bits**

n=0,1..5		0 = PERIOD will load to PBUF at the end point of each period. CMP will load to CMPBUF at the end point or center point of each period by setting CTRLD bit. 1 = PERIOD will load to PBUF at the end point of each period. CMP will load to CMPBUF at the end point of each period when valid reload window is set. The valid reload window is set by software write 1 to EPWM_LOAD register and cleared by hardware after load success.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CTRLDn**	**Center Re-load** In up-down counter type, PERIOD will load to PBUF at the end point of each period. CMP will load to CMPBUF at the center point of a period.

EPWM Control Register 1 (EPWM_CTL1)

Register	Offset	R/W	Description	Reset Value
EPWM_CTL1	EPWMx_BA+0x04	R/W	EPWM Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved					OUTMODE4	OUTMODE2	OUTMODE0
23	22	21	20	19	18	17	16
Reserved		CNTMODE5	CNTMODE4	CNTMODE3	CNTMODE2	CNTMODE1	CNTMODE0
15	14	13	12	11	10	9	8
Reserved				CNTTYPE5		CNTTYPE4	
7	6	5	4	3	2	1	0
CNTTYPE3		CNTTYPE2		CNTTYPE1		CNTTYPE0	

Bits	Description	
[31:27]	**Reserved**	Reserved.
[24+n/2] n=0,2,4	**OUTMODEn**	**EPWM Output Mode** Each bit n controls the output mode of corresponding EPWM channel n. 0 = EPWM independent mode. 1 = EPWM complementary mode. **Note:** When operating in group function, these bits must all set to the same mode.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**CNTMODEn**	**EPWM Counter Mode** 0 = Auto-reload mode. 1 = One-shot mode.
[15:12]	**Reserved**	Reserved.
[2n+1:2n] n=0,1..5	**CNTTYPEn**	**EPWM Counter Behavior Type** 00 = Up counter type (supported in capture mode). 01 = Down count type (supported in capture mode). 10 = Up-down counter type. 11 = Reserved.

EPWM Synchronization Register (EPWM_SYNC)

Register	Offset	R/W	Description	Reset Value
EPWM_SYNC	EPWMx_BA+0x08	R/W	EPWM Synchronization Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved					PHSDIR4	PHSDIR2	PHSDIR0
23	22	21	20	19	18	17	16
SINPINV	SFLTCNT			SFLTCSEL			SNFLTEN
15	14	13	12	11	10	9	8
Reserved		SINSRC4		SINSRC2		SINSRC0	
7	6	5	4	3	2	1	0
Reserved					PHSEN4	PHSEN2	PHSEN0

Bits	Description	
[31:27]	**Reserved**	Reserved.
[24+n/2] n=0,2,4	**PHSDIRn**	**EPWM Phase Direction Control** 0 = Control EPWM counter count decrement after synchronizing. 1 = Control EPWM counter count increment after synchronizing.
[23]	**SINPINV**	**SYNC Input Pin Inverse** 0 = The state of pin SYNC is passed to the negative edge detector. 1 = The inversed state of pin SYNC is passed to the negative edge detector.
[22:20]	**SFLTCNT**	**SYNC Edge Detector Filter Count** The register bits control the counter number of edge detector.
[19:17]	**SFLTCSEL**	**SYNC Edge Detector Filter Clock Selection** 000 = Filter clock = HCLK. 001 = Filter clock = HCLK/2. 010 = Filter clock = HCLK/4. 011 = Filter clock = HCLK/8. 100 = Filter clock = HCLK/16. 101 = Filter clock = HCLK/32. 110 = Filter clock = HCLK/64. 111 = Filter clock = HCLK/128.
[16]	**SNFLTEN**	**EPWM0_SYNC_IN Noise Filter Enable Bits** 0 = Noise filter of input pin EPWM0_SYNC_IN Disabled. 1 = Noise filter of input pin EPWM0_SYNC_IN Enabled.
[15:14]	**Reserved**	Reserved.
[9+n:8+n] n=0,2,4	**SINSRCn**	**EPWM0_SYNC_IN Source Selection** 00 = Synchronize source from SYNC_IN or SWSYNC.

		01 = Counter equal to 0. 10 = Counter equal to EPWM_CMPDATm, m denotes 1, 3, 5. 11 = SYNC_OUT will not be generated.
[7:3]	**Reserved**	Reserved.
[n/2] n=0,2,4	**PHSENn**	**SYNC Phase Enable Bits** 0 = EPWM counter disabled to load PHS value. 1 = EPWM counter enabled to load PHS value.

EPWM Software Control Synchronization Register (EPWM_SWSYNC)

Register	Offset	R/W	Description	Reset Value
EPWM_SWSYNC	EPWMx_BA+0x0C	R/W	EPWM Software Control Synchronization Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**SWSYNC4**	**SWSYNC2**	**SWSYNC0**

Bits	Description	
[31:3]	**Reserved**	Reserved.
[n/2] n=0,2,4	**SWSYNCn**	**Software SYNC Function** When SINSRCn (EPWM_SYNC[13:8]) is selected to 0, SYNC_OUT source comes from SYNC_IN or this bit.

EPWM Clock Source Register (EPWM_CLKSRC)

Register	Offset	R/W	Description	Reset Value
EPWM_CLKSRC	EPWMx_BA+0x10	R/W	EPWM Clock Source Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved					**ECLKSRC4**		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**ECLKSRC2**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**ECLKSRC0**		

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18:16]	**ECLKSRC4**	**EPWM_CH45 External Clock Source Select** 000 = EPWMx_CLK, x denotes 0 or 1. 001 = TIMER0 overflow. 010 = TIMER1 overflow. 011 = TIMER2 overflow. 100 = TIMER3 overflow. Others = Reserved.
[15:11]	**Reserved**	Reserved.
[10:8]	**ECLKSRC2**	**EPWM_CH23 External Clock Source Select** 000 = EPWMx_CLK, x denotes 0 or 1. 001 = TIMER0 overflow. 010 = TIMER1 overflow. 011 = TIMER2 overflow. 100 = TIMER3 overflow. Others = Reserved.
[7:3]	**Reserved**	Reserved.
[2:0]	**ECLKSRC0**	**EPWM_CH01 External Clock Source Select** 000 = EPWMx_CLK, x denotes 0 or 1. 001 = TIMER0 overflow. 010 = TIMER1 overflow. 011 = TIMER2 overflow. 100 = TIMER3 overflow. Others = Reserved.

EPWM Clock Prescale Register 0_1, 2_3, 4_5 (EPWM_CLKPSC0_1, 2_3, 4_5)

Register	Offset	R/W	Description	Reset Value
EPWM_CLKPSC0_1	EPWMx_BA+0x14	R/W	EPWM Clock Prescale Register 0/1	0x0000_0000
EPWM_CLKPSC2_3	EPWMx_BA+0x18	R/W	EPWM Clock Prescale Register 2/3	0x0000_0000
EPWM_CLKPSC4_5	EPWMx_BA+0x1C	R/W	EPWM Clock Prescale Register 4/5	0x0000_0000

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

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**CLKPSC**	**EPWM Counter Clock Prescale** The clock of EPWM counter is decided by clock prescaler. Each EPWM pair share one EPWM counter clock prescaler. The clock of EPWM counter is divided by (CLKPSC+ 1).

EPWM Counter Enable Register (EPWM_CNTEN)

Register	Offset	R/W	Description	Reset Value
EPWM_CNTEN	EPWMx_BA+0x20	R/W	EPWM Counter Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CNTEN5**	**CNTEN4**	**CNTEN3**	**CNTEN2**	**CNTEN1**	**CNTEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CNTENn**	**EPWM Counter Enable Bits** 0 = EPWM Counter and clock prescaler stop running. 1 = EPWM Counter and clock prescaler start running.

EPWM Clear Counter Register (EPWM_CNTCLR)

Register	Offset	R/W	Description	Reset Value
EPWM_CNTCLR	EPWMx_BA+0x24	R/W	EPWM Clear Counter Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CNTCLR5**	**CNTCLR4**	**CNTCLR3**	**CNTCLR2**	**CNTCLR1**	**CNTCLR0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CNTCLRn**	**Clear EPWM Counter Control Bit** It is automatically cleared by hardware. Each bit n controls the corresponding EPWM channel n. 0 = No effect. 1 = Clear 16-bit EPWM counter to 0000H.

EPWM Load Register (EPWM_LOAD)

Register	Offset	R/W	Description	Reset Value
EPWM_LOAD	EPWMx_BA+0x28	R/W	EPWM Load Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**LOAD5**	**LOAD4**	**LOAD3**	**LOAD2**	**LOAD1**	**LOAD0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**LOADn**	**Re-load EPWM Comparator Register (EPWM_CMPDATn) Control Bit** This bit is software write, hardware clear when current EPWM period end. Write Operation: 0 = No effect. 1 = Set load window of window loading mode. Read Operation: 0 = No load window is set. 1 = Load window is set. **Note:** This bit only use in window loading mode, WINLDENn(EPWM_CTL0[13:8]) = 1.

EPWM Period Register 0~5 (EPWM_PERIOD0~5)

Register	Offset	R/W	Description	Reset Value
EPWM_PERIOD0	EPWMx_BA+0x30	R/W	EPWM Period Register 0	0x0000_0000
EPWM_PERIOD1	EPWMx_BA+0x34	R/W	EPWM Period Register 1	0x0000_0000
EPWM_PERIOD2	EPWMx_BA+0x38	R/W	EPWM Period Register 2	0x0000_0000
EPWM_PERIOD3	EPWMx_BA+0x3C	R/W	EPWM Period Register 3	0x0000_0000
EPWM_PERIOD4	EPWMx_BA+0x40	R/W	EPWM Period Register 4	0x0000_0000
EPWM_PERIOD5	EPWMx_BA+0x44	R/W	EPWM Period Register 5	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PERIOD**	**EPWM Period Register** Up-Count mode: In this mode, EPWM counter counts from 0 to PERIOD, and restarts from 0. EPWM period time = (PERIOD+1) * (CLKPSC+1) * EPWM_CLK . Down-Count mode: In this mode, EPWM counter counts from PERIOD to 0, and restarts from PERIOD. EPWM period time = (PERIOD+1) * (CLKPSC+1) * EPWM_CLK . Up-Down-Count mode: In this mode, EPWM counter counts from 0 to PERIOD, then decrements to 0 and repeats again. EPWM period time = 2 * PERIOD * (CLKPSC+1) * EPWM_CLK.

EPWM Comparator Register 0~5 (EPWM_CMPDAT0~5)

Register	Offset	R/W	Description	Reset Value
EPWM_CMPDAT0	EPWMx_BA+0x50	R/W	EPWM Comparator Register 0	0x0000_0000
EPWM_CMPDAT1	EPWMx_BA+0x54	R/W	EPWM Comparator Register 1	0x0000_0000
EPWM_CMPDAT2	EPWMx_BA+0x58	R/W	EPWM Comparator Register 2	0x0000_0000
EPWM_CMPDAT3	EPWMx_BA+0x5C	R/W	EPWM Comparator Register 3	0x0000_0000
EPWM_CMPDAT4	EPWMx_BA+0x60	R/W	EPWM Comparator Register 4	0x0000_0000
EPWM_CMPDAT5	EPWMx_BA+0x64	R/W	EPWM Comparator Register 5	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CMP**	**EPWM Comparator Register** CMP is used to compare with CNT (EPWM_CNTn[15:0]) bits to generate EPWM waveform, interrupt and trigger EADC/DAC. In independent mode, EPWM_CMPDATn, n=0,1..5 denote as 6 independent EPWM_CH0~5 compared point. In complementary mode, EPWM_CMPDAT0, EPWM_CMPDAT2 and EPWM_CMPDAT4 denote as first compared point, and EPWM_CMPDAT1, EPWM_CMPDAT3 and EPWM_CMPDAT5 denote as second compared point for the corresponding 3 complementary pairs EPWM_CH0 and EPWM_CH1, EPWM_CH2 and EPWM_CH3, EPWM_CH4 and EPWM_CH5.

EPWM Dead-time Control Register 0_1, 2_3, 4_5 (EPWM_DTCTL0_1, 2_3, 4_5)

Register	Offset	R/W	Description	Reset Value
EPWM_DTCTL0_1	EPWMx_BA+0x70	R/W	EPWM Dead-time Control Register 0/1	0x0000_0000
EPWM_DTCTL2_3	EPWMx_BA+0x74	R/W	EPWM Dead-time Control Register 2/3	0x0000_0000
EPWM_DTCTL4_5	EPWMx_BA+0x78	R/W	EPWM Dead-time Control Register 4/5	0x0000_0000

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

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24]	**DTCKSEL**	**Dead-time Clock Select (Write Protect)** 0 = Dead-time clock source from EPWM_CLK. 1 = Dead-time clock source from prescaler output. **Note:** This bit is write protected. Refer to REGWRPROT register.
[23:17]	**Reserved**	Reserved.
[16]	**DTEN**	**Enable Dead-time Insertion for EPWM Pair (EPWM_CH0, EPWM_CH1) (EPWM_CH2, EPWM_CH3) (EPWM_CH4, EPWM_CH5) (Write Protect)** Dead-time insertion is only active when this pair of complementary EPWM is enabled. If dead- time insertion is inactive, the outputs of pin pair are complementary without any delay. 0 = Dead-time insertion Disabled on the pin pair. 1 = Dead-time insertion Enabled on the pin pair. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[15:12]	**Reserved**	Reserved.
[11:0]	**DTCNT**	**Dead-time Counter (Write Protect)** The dead-time can be calculated from the following formula: DTCKSEL=0: Dead-time = (DTCNT[11:0]+1) * EPWM_CLK period. DTCKSEL=1: Dead-time = (DTCNT[11:0]+1) * EPWM_CLK period * (CLKPSC+1). **Note:** This bit is write protected. Refer to SYS_REGLCTL register.

EPWM Counter Phase Register 0_1, 2_3, 4_5 (EPWM_PHS0_1, 2_3, 4_5)

Register	Offset	R/W	Description	Reset Value
EPWM_PHS0_1	EPWMx_BA+0x80	R/W	EPWM Counter Phase Register 0/1	0x0000_0000
EPWM_PHS2_3	EPWMx_BA+0x84	R/W	EPWM Counter Phase Register 2/3	0x0000_0000
EPWM_PHS4_5	EPWMx_BA+0x88	R/W	EPWM Counter Phase Register 4/5	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PHS**	**EPWM Synchronous Start Phase Bits** PHS determines the EPWM synchronous start phase value. These bits only use in synchronous function.

EPWM Counter Register 0~5 (EPWM_CNT0~5)

Register	Offset	R/W	Description	Reset Value
EPWM_CNT0	EPWMx_BA+0x90	R	EPWM Counter Register 0	0x0000_0000
EPWM_CNT1	EPWMx_BA+0x94	R	EPWM Counter Register 1	0x0000_0000
EPWM_CNT2	EPWMx_BA+0x98	R	EPWM Counter Register 2	0x0000_0000
EPWM_CNT3	EPWMx_BA+0x9C	R	EPWM Counter Register 3	0x0000_0000
EPWM_CNT4	EPWMx_BA+0xA0	R	EPWM Counter Register 4	0x0000_0000
EPWM_CNT5	EPWMx_BA+0xA4	R	EPWM Counter Register 5	0x0000_0000

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

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**DIRF**	**EPWM Direction Indicator Flag (Read Only)** 0 = Counter is counting down. 1 = Counter is counting up.
[15:0]	**CNT**	**EPWM Data Register (Read Only)** User can monitor CNT to know the current value in 16-bit period counter.

EPWM Generation Register 0 (EPWM_WGCTL0)

Register	Offset	R/W	Description	Reset Value
EPWM_WGCTL0	EPWMx_BA+0xB0	R/W	EPWM Generation Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				PRDPCTL5		PRDPCTL4	
23	22	21	20	19	18	17	16
PRDPCTL3		PRDPCTL2		PRDPCTL1		PRDPCTL0	
15	14	13	12	11	10	9	8
Reserved				ZPCTL5		ZPCTL4	
7	6	5	4	3	2	1	0
ZPCTL3		ZPCTL2	ZPCTL1			ZPCTL0	

Bits	Description	
[31:28]	**Reserved**	Reserved.
[17+2n:16+2n] n=0,1..5	**PRDPCTLn**	**EPWM Period (Center) Point Control** EPWM can control output level when EPWM counter counts to (PERIODn+1). 00 = Do nothing. 01 = EPWM period (center) point output Low. 10 = EPWM period (center) point output High. 11 = EPWM period (center) point output Toggle. **Note:** This bit is center point control when EPWM counter operating in up-down counter type.
[15:12]	**Reserved**	Reserved.
[1+2n:2n] n=0,1..5	**ZPCTLn**	**EPWM Zero Point Control** EPWM can control output level when EPWM counter counts to 0. 00 = Do nothing. 01 = EPWM zero point output Low. 10 = EPWM zero point output High. 11 = EPWM zero point output Toggle.

EPWM Generation Register 1 (EPWM_WGCTL1)

Register	Offset	R/W	Description	Reset Value
EPWM_WGCTL1	EPWMx_BA+0xB4	R/W	EPWM Generation Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				CMPDCTL5		CMPDCTL4	
23	22	21	20	19	18	17	16
CMPDCTL3		CMPDCTL2		CMPDCTL1		CMPDCTL0	
15	14	13	12	11	10	9	8
Reserved				CMPUCTL5		CMPUCTL4	
7	6	5	4	3	2	1	0
CMPUCTL3		CMPUCTL2		CMPUCTL1		CMPUCTL0	

Bits	Description	
[31:28]	**Reserved**	Reserved.
[17+2n:16+2n] n=0,1..5	**CMPDCTLn**	**EPWM Compare Down Point Control** EPWM can control output level when EPWM counter counts down to CMP. 00 = Do nothing. 01 = EPWM compare down point output Low. 10 = EPWM compare down point output High. 11 = EPWM compare down point output Toggle. **Note:** In complementary mode, CMPDCTL1, 3, 5 is used as another CMPDCTL for channel 0, 2, 4.
[15:12]	**Reserved**	Reserved.
[1+2n:2n] n=0,1..5	**CMPUCTLn**	**EPWM Compare Up Point Control** EPWM can control output level when EPWM counter counts up to CMP. 00 = Do nothing. 01 = EPWM compare up point output Low. 10 = EPWM compare up point output High. 11 = EPWM compare up point output Toggle. **Note:** In complementary mode, CMPUCTL1, 3, 5 is used as another CMPUCTL for channel 0, 2, 4.

EPWM Mask Enable Register (EPWM_MSKEN)

Register	Offset	R/W	Description	Reset Value
EPWM_MSKEN	EPWMx_BA+0xB8	R/W	EPWM Mask Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**MSKEN5**	**MSKEN4**	**MSKEN3**	**MSKEN2**	**MSKEN1**	**MSKEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**MSKENn**	**EPWM Mask Enable Bits** The EPWM output signal will be masked when this bit is enabled. The corresponding EPWM channel n will output MSKDATn (EPWM_MSK[5:0]) data. 0 = EPWM output signal is non-masked. 1 = EPWM output signal is masked and output MSKDATn data.

EPWM Mask DATA Register (EPWM_MSK)

Register	Offset	R/W	Description	Reset Value
EPWM_MSK	EPWMx_BA+0xBC	R/W	EPWM Mask Data Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**MSKDAT5**	**MSKDAT4**	**MSKDAT3**	**MSKDAT2**	**MSKDAT1**	**MSKDAT0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**MSKDATn**	**EPWM Mask Data Bit** This data bit control the state of EPWMn output pin, if corresponding mask function is enabled. 0 = Output logic low to EPWM channel n. 1 = Output logic high to EPWM channel n.

EPWM Brake Noise Filter Register (EPWM_BNF)

Register	Offset	R/W	Description	Reset Value
EPWM_BNF	EPWMx_BA+0xC0	R/W	EPWM Brake Noise Filter Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							BK1SRC
23	22	21	20	19	18	17	16
Reserved							BK0SRC
15	14	13	12	11	10	9	8
BRK1PINV	BRK1FCNT			BRK1NFSEL			BRK1NFEN
7	6	5	4	3	2	1	0
BRK0PINV	BRK0FCNT			BRK0NFSEL			BRK0NFEN

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24]	**BK1SRC**	**Brake 1 Pin Source Select** For EPWM0 setting: 0 = Brake 1 pin source come from EPWM0_BRAKE1. 1 = Brake 1 pin source come from EPWM1_BRAKE1. For EPWM1 setting: 0 = Brake 1 pin source come from EPWM1_BRAKE1. 1 = Brake 1 pin source come from EPWM0_BRAKE1.
[23:17]	**Reserved**	Reserved.
[16]	**BK0SRC**	**Brake 0 Pin Source Select** For EPWM0 setting: 0 = Brake 0 pin source come from EPWM0_BRAKE0. 1 = Brake 0 pin source come from EPWM1_BRAKE0. For EPWM1 setting: 0 = Brake 0 pin source come from EPWM1_BRAKE0. 1 = Brake 0 pin source come from EPWM0_BRAKE0.
[15]	**BRK1PINV**	**Brake 1 Pin Inverse** 0 **=** brake pin event will be detected if EPWMx_BRAKE1 pin status transfer from low to high in edge-detect, or pin status is high in level-detect. 1 = brake pin event will be detected if EPWMx_BRAKE1 pin status transfer from high to low in edge-detect, or pin status is low in level-detect.
[14:12]	**BRK1FCNT**	**Brake 1 Edge Detector Filter Count** The register bits control the Brake1 filter counter to count from 0 to BRK1FCNT.
[11:9]	**BRK1NFSEL**	**Brake 1 Edge Detector Filter Clock Selection** 000 = Filter clock = HCLK.

		001 = Filter clock = HCLK/2. 010 = Filter clock = HCLK/4. 011 = Filter clock = HCLK/8. 100 = Filter clock = HCLK/16. 101 = Filter clock = HCLK/32. 110 = Filter clock = HCLK/64. 111 = Filter clock = HCLK/128.
[8]	**BRK1NFEN**	**EPWM Brake 1 Noise Filter Enable Bit** 0 = Noise filter of EPWM Brake 1 Disabled. 1 = Noise filter of EPWM Brake 1 Enabled.
[7]	**BRK0PINV**	**Brake 0 Pin Inverse** 0 **=** brake pin event will be detected if EPWMx_BRAKE0 pin status transfer from low to high in edge-detect, or pin status is high in level-detect. 1 = brake pin event will be detected if EPWMx_BRAKE0 pin status transfer from high to low in edge-detect, or pin status is low in level-detect.
[6:4]	**BRK0FCNT**	**Brake 0 Edge Detector Filter Count** The register bits control the Brake0 filter counter to count from 0 to BRK0FCNT.
[3:1]	**BRK0NFSEL**	**Brake 0 Edge Detector Filter Clock Selection** 000 = Filter clock = HCLK. 001 = Filter clock = HCLK/2. 010 = Filter clock = HCLK/4. 011 = Filter clock = HCLK/8. 100 = Filter clock = HCLK/16. 101 = Filter clock = HCLK/32. 110 = Filter clock = HCLK/64. 111 = Filter clock = HCLK/128.
[0]	**BRK0NFEN**	**EPWM Brake 0 Noise Filter Enable Bit** 0 = Noise filter of EPWM Brake 0 Disabled. 1 = Noise filter of EPWM Brake 0 Enabled.

EPWM System Fail Brake Control Register (EPWM_FAILBRK)

Register	Offset	R/W	Description	Reset Value
EPWM_FAILBRK	EPWMx_BA+0xC4	R/W	EPWM System Fail Brake Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				CORBRKEN	RAMBRKEN	BODBRKEN	CSSBRKEN

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**CORBRKEN**	**Core Lockup Detection Trigger EPWM Brake Function 0 Enable Bit** 0 = Brake Function triggered by Core lockup detection Disabled. 1 = Brake Function triggered by Core lockup detection Enabled.
[2]	**RAMBRKEN**	**SRAM Parity Error Detection Trigger EPWM Brake Function 0 Enable Bit** 0 = Brake Function triggered by SRAM parity error detection Disabled. 1 = Brake Function triggered by SRAM parity error detection Enabled.
[1]	**BODBRKEN**	**Brown-out Detection Trigger EPWM Brake Function 0 Enable Bit** 0 = Brake Function triggered by BOD Disabled. 1 = Brake Function triggered by BOD Enabled.
[0]	**CSSBRKEN**	**Clock Security System Detection Trigger EPWM Brake Function 0 Enable Bit** 0 = Brake Function triggered by CSS detection Disabled. 1 = Brake Function triggered by CSS detection Enabled.

EPWM Brake Edge Detect Control Register 0_1, 2_3, 4_5 (EPWM_BRKCTL0_1, 2_3, 4_5)

Register	Offset	R/W	Description	Reset Value
EPWM_BRKCTL0_1	EPWMx_BA+0xC8	R/W	EPWM Brake Edge Detect Control Register 0/1	0x0000_0000
EPWM_BRKCTL2_3	EPWMx_BA+0xCC	R/W	EPWM Brake Edge Detect Control Register 2/3	0x0000_0000
EPWM_BRKCTL4_5	EPWMx_BA+0xD0	R/W	EPWM Brake Edge Detect Control Register 4/5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			EADCLBEN	Reserved			
23	22	21	20	19	18	17	16
Reserved			EADCEBEN	BRKAODD		BRKAEVEN	
15	14	13	12	11	10	9	8
SYSLBEN	Reserved	BRKP1LEN	BRKP0LEN	Reserved		CPO1LBEN	CPO0LBEN
7	6	5	4	3	2	1	0
SYSEBEN	Reserved	BRKP1EEN	BRKP0EEN	Reserved		CPO1EBEN	CPO0EBEN

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28]	**EADCLBEN**	**Enable EADC Result Monitor (EADCRM) As Level-detect Brake Source (Write Protect)** 0 = EADCRM as level-detect brake source Disabled. 1 = EADCRM as level-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[27:21]	**Reserved**	Reserved.
[20]	**EADCEBEN**	**Enable EADC Result Monitor (EADCRM) As Edge-detect Brake Source (Write Protect)** 0 = EADCRM as edge-detect brake source Disabled. 1 = EADCRM as edge-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[19:18]	**BRKAODD**	**EPWM Brake Action Select for Odd Channel (Write Protect)** 00 = EPWMx brake event will not affect odd channels output. 01 = EPWM odd channel output tri-state when EPWMx brake event happened. 10 = EPWM odd channel output low level when EPWMx brake event happened. 11 = EPWM odd channel output high level when EPWMx brake event happened. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[17:16]	**BRKAEVEN**	**EPWM Brake Action Select for Even Channel (Write Protect)** 00 = EPWMx brake event will not affect even channels output. 01 = EPWM even channel output tri-state when EPWMx brake event happened. 10 = EPWM even channel output low level when EPWMx brake event happened. 11 = EPWM even channel output high level when EPWMx brake event happened. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.

[15]	**SYSLBEN**	**Enable System Fail As Level-detect Brake Source (Write Protect)** 0 = System Fail condition as level-detect brake source Disabled. 1 = System Fail condition as level-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[14]	**Reserved**	Reserved.
[13]	**BRKP1LEN**	**Enable BKP1 Pin As Level-detect Brake Source (Write Protect)** 0 = EPWMx_BRAKE1 pin as level-detect brake source Disabled. 1 = EPWMx_BRAKE1 pin as level-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[12]	**BRKP0LEN**	**Enable BKP0 Pin As Level-detect Brake Source (Write Protect)** 0 = EPWMx_BRAKE0 pin as level-detect brake source Disabled. 1 = EPWMx_BRAKE0 pin as level-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[11:10]	**Reserved**	Reserved.
[9]	**CPO1LBEN**	**Enable ACMP1_O Digital Output As Level-detect Brake Source (Write Protect)** 0 = ACMP1_O as level-detect brake source Disabled. 1 = ACMP1_O as level-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[8]	**CPO0LBEN**	**Enable ACMP0_O Digital Output As Level-detect Brake Source (Write Protect)** 0 = ACMP0_O as level-detect brake source Disabled. 1 = ACMP0_O as level-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[7]	**SYSEBEN**	**Enable System Fail As Edge-detect Brake Source (Write Protect)** 0 = System Fail condition as edge-detect brake source Disabled. 1 = System Fail condition as edge-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[6]	**Reserved**	Reserved.
[5]	**BRKP1EEN**	**Enable EPWMx_BRAKE1 Pin As Edge-detect Brake Source (Write Protect)** 0 = EPWMx_BRAKE1 pin as edge-detect brake source Disabled. 1 = EPWMx_BRAKE1 pin as edge-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[4]	**BRKP0EEN**	**Enable EPWMx_BRAKE0 Pin As Edge-detect Brake Source (Write Protect)** 0 = EPWMx_BRAKE0 pin as edge-detect brake source Disabled. 1 = EPWMx_BRAKE0 pin as edge-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[3:2]	**Reserved**	Reserved.
[1]	**CPO1EBEN**	**Enable ACMP1_O Digital Output As Edge-detect Brake Source (Write Protect)** 0 = ACMP1_O as edge-detect brake source Disabled. 1 = ACMP1_O as edge-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[0]	**CPO0EBEN**	**Enable ACMP0_O Digital Output As Edge-detect Brake Source (Write Protect)** 0 = ACMP0_O as edge-detect brake source Disabled.

		1 = ACMP0_O as edge-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.

EPWM Pin Polar Inverse Control (EPWM_POLCTL)

Register	Offset	R/W	Description	Reset Value
EPWM_POLCTL	EPWMx_BA+0xD4	R/W	EPWM Pin Polar Inverse Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**PINV5**	**PINV4**	**PINV3**	**PINV2**	**PINV1**	**PINV0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**PINVn**	**EPWM PIN Polar Inverse Control** The register controls polarity state of EPWMx_CHn output pin. 0 = EPWMx_CHn output pin polar inverse Disabled. 1 = EPWMx_CHn output pin polar inverse Enabled.

EPWM Output Enable Register (EPWM_POEN)

Register	Offset	R/W	Description	Reset Value
EPWM_POEN	EPWMx_BA+0xD8	R/W	EPWM Output Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**POEN5**	**POEN4**	**POEN3**	**POEN2**	**POEN1**	**POEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**POENn**	**EPWM Pin Output Enable Bits** 0 = EPWMx_CHn pin at tri-state. 1 = EPWMx_CHn pin in output mode.

EPWM Software Brake Control Register (EPWM_SWBRK)

Register	Offset	R/W	Description	Reset Value
EPWM_SWBRK	EPWMx_BA+0xDC	W	EPWM Software Brake Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**BRKLTRG4**	**BRKLTRG2**	**BRKLTRG0**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**BRKETRG4**	**BRKETRG2**	**BRKETRG0**

Bits	Description	
[31:11]	**Reserved**	Reserved.
[8+n/2] n=0,2,4	**BRKLTRGn**	**EPWM Level Brake Software Trigger (Write Only) (Write Protect)** Write 1 to this bit will trigger level brake, and set BRKLIFn to 1 in EPWM_INTSTS1 register. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[7:3]	**Reserved**	Reserved.
[n/2] n=0,2,4	**BRKETRGn**	**EPWM Edge Brake Software Trigger (Write Only) (Write Protect)** Write 1 to this bit will trigger edge brake, and set BRKEIFn to 1 in EPWM_INTSTS1 register. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.

EPWM Interrupt Enable Register 0 (EPWM_INTEN0)

Register	Offset	R/W	Description	Reset Value
EPWM_INTEN0	EPWMx_BA+0xE0	R/W	EPWM Interrupt Enable Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		CMPDIEN5	CMPDIEN4	CMPDIEN3	CMPDIEN2	CMPDIEN1	CMPDIEN0
23	22	21	20	19	18	17	16
Reserved		CMPUIEN5	CMPUIEN4	CMPUIEN3	CMPUIEN2	CMPUIEN1	CMPUIEN0
15	14	13	12	11	10	9	8
Reserved		PIEN5	PIEN4	PIEN3	PIEN2	PIEN1	PIEN0
7	6	5	4	3	2	1	0
Reserved		ZIEN5	ZIEN4	ZIEN3	ZIEN2	ZIEN1	ZIEN0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[24+n] n=0,1..5	**CMPDIENn**	**EPWM Compare Down Count Interrupt Enable Bits** 0 = Compare down count interrupt Disabled. 1 = Compare down count interrupt Enabled. **Note:** In complementary mode, CMPDIEN1, 3, 5 is used as another CMPDIEN for channel 0, 2, 4.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**CMPUIENn**	**EPWM Compare Up Count Interrupt Enable Bits** 0 = Compare up count interrupt Disabled. 1 = Compare up count interrupt Enabled. **Note:** In complementary mode, CMPUIEN1, 3, 5 is used as another CMPUIEN for channel 0, 2, 4.
[15:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**PIENn**	**EPWM Period Point Interrupt Enable Bits** 0 = Period point interrupt Disabled. 1 = Period point interrupt Enabled. **Note 1:** When up-down counter type period point means center point. **Note 2:** Odd channels will read always 0 at complementary mode.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**ZIENn**	**EPWM Zero Point Interrupt Enable Bits** 0 = Zero point interrupt Disabled. 1 = Zero point interrupt Enabled. **Note:** Odd channels will read always 0 at complementary mode.

EPWM Interrupt Enable Register 1 (EPWM_INTEN1)

Register	Offset	R/W	Description	Reset Value
EPWM_INTEN1	EPWMx_BA+0xE4	R/W	EPWM Interrupt Enable Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**BRKLIEN45**	**BRKLIEN23**	**BRKLIEN01**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**BRKEIEN45**	**BRKEIEN23**	**BRKEIEN01**

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10]	**BRKLIEN45**	**EPWM Level-detect Brake Interrupt Enable for Channel4/5 (Write Protect)** 0 = Level-detect Brake interrupt for channel4/5 Disabled. 1 = Level-detect Brake interrupt for channel4/5 Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[9]	**BRKLIEN23**	**EPWM Level-detect Brake Interrupt Enable for Channel2/3 (Write Protect)** 0 = Level-detect Brake interrupt for channel2/3 Disabled. 1 = Level-detect Brake interrupt for channel2/3 Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[8]	**BRKLIEN01**	**EPWM Level-detect Brake Interrupt Enable for Channel0/1 (Write Protect)** 0 = Level-detect Brake interrupt for channel0/1 Disabled. 1 = Level-detect Brake interrupt for channel0/1 Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[7:3]	**Reserved**	Reserved.
[2]	**BRKEIEN45**	**EPWM Edge-detect Brake Interrupt Enable for Channel4/5 (Write Protect)** 0 = Edge-detect Brake interrupt for channel4/5 Disabled. 1 = Edge-detect Brake interrupt for channel4/5 Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[1]	**BRKEIEN23**	**EPWM Edge-detect Brake Interrupt Enable for Channel2/3 (Write Protect)** 0 = Edge-detect Brake interrupt for channel2/3 Disabled. 1 = Edge-detect Brake interrupt for channel2/3 Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[0]	**BRKEIEN01**	**EPWM Edge-detect Brake Interrupt Enable for Channel0/1 (Write Protect)** 0 = Edge-detect Brake interrupt for channel0/1 Disabled.

		1 = Edge-detect Brake interrupt for channel0/1 Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.

EPWM Interrupt Flag Register 0 (EPWM_INTSTS0)

Register	Offset	R/W	Description	Reset Value
EPWM_INTSTS0	EPWMx_BA+0xE8	R/W	EPWM Interrupt Flag Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		CMPDIF5	CMPDIF4	CMPDIF3	CMPDIF2	CMPDIF1	CMPDIF0
23	22	21	20	19	18	17	16
Reserved		CMPUIF5	CMPUIF4	CMPUIF3	CMPUIF2	CMPUIF1	CMPUIF0
15	14	13	12	11	10	9	8
Reserved		PIF5	PIF4	PIF3	PIF2	PIF1	PIF0
7	6	5	4	3	2	1	0
Reserved		ZIF5	ZIF4	ZIF3	ZIF2	ZIF1	ZIF0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[24+n] n=0,1..5	**CMPDIFn**	**EPWM Compare Down Count Interrupt Flag** Flag is set by hardware when EPWM counter down count and reaches EPWM_CMPDATn, software can clear this bit by writing 1 to it. **Note:** In complementary mode, CMPDIF1, 3, 5 is used as another CMPDIF for channel 0, 2, 4.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**CMPUIFn**	**EPWM Compare Up Count Interrupt Flag** Flag is set by hardware when EPWM counter up count and reaches EPWM_CMPDATn, software can clear this bit by writing 1 to it. **Note:** In complementary mode, CMPUIF1, 3, 5 is used as another CMPUIF for channel 0, 2, 4.
[15:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**PIFn**	**EPWM Period Point Interrupt Flag** This bit is set by hardware when EPWM counter reaches EPWM_PERIODn. **Note:** This bit can be cleared to 0 by software writing 1.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**ZIFn**	**EPWM Zero Point Interrupt Flag** This bit is set by hardware when EPWM counter reaches 0. **Note:** This bit can be cleared to 0 by software writing 1

EPWM Interrupt Flag Register 1 (EPWM_INTSTS1)

Register	Offset	R/W	Description	Reset Value
EPWM_INTSTS1	EPWMx_BA+0xEC	R/W	EPWM Interrupt Flag Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		BRKLSTS5	BRKLSTS4	BRKLSTS3	BRKLSTS2	BRKLSTS1	BRKLSTS0
23	22	21	20	19	18	17	16
Reserved		BRKESTS5	BRKESTS4	BRKESTS3	BRKESTS2	BRKESTS1	BRKESTS0
15	14	13	12	11	10	9	8
Reserved		BRKLIF5	BRKLIF4	BRKLIF3	BRKLIF2	BRKLIF1	BRKLIF0
7	6	5	4	3	2	1	0
Reserved		BRKEIF5	BRKEIF4	BRKEIF3	BRKEIF2	BRKEIF1	BRKEIF0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[24+n] n=0,1..5	**BRKLSTSn**	**EPWM Channel n Level-detect Brake Status (Read Only)** 0 = EPWM channel n level-detect brake state is released. 1 = When EPWM channel n level-detect brake detects a falling edge of any enabled brake source; this flag will be set to indicate the EPWM channel n at brake state. **Note:** This bit is read only and auto cleared by hardware. When enabled brake source return to high level, EPWM will release brake state until current EPWM period finished. The EPWM waveform will start output from next full EPWM period.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**BRKESTSn**	**EPWM Channel n Edge-detect Brake Status (Read Only)** 0 = EPWM channel n edge-detect brake state is released. 1 = When EPWM channel n edge-detect brake detects a falling edge of any enabled brake source; this flag will be set to indicate the EPWM channel n at brake state. **Note:** This bit is read only and auto cleared by hardware. When edge-detect brake interrupt flag is cleared, EPWM will release brake state until current EPWM period finished. The EPWM waveform will start output from next full EPWM period.
[15:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**BRKLIFn**	**EPWM Channel n Level-detect Brake Interrupt Flag (Write Protect)** 0 = EPWM channel n level-detect brake event do not happened. 1 = When EPWM channel n level-detect brake event happened, this bit is set to 1, writing 1 to clear. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**BRKEIFn**	**EPWM Channel n Edge-detect Brake Interrupt Flag (Write Protect)** 0 = EPWM channel n edge-detect brake event do not happened. 1 = When EPWM channel n edge-detect brake event happened, this bit is set to 1, writing 1 to clear. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.

EPWM Trigger DAC Enable Register (EPWM_DACTRGEN)

Register	Offset	R/W	Description	Reset Value
EPWM_DACTRGEN	EPWMx_BA+0xF4	R/W	EPWM Trigger DAC Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		CDTRGEN5	CDTRGEN4	CDTRGEN3	CDTRGEN2	CDTRGEN1	CDTRGEN0
23	22	21	20	19	18	17	16
Reserved		CUTRGEN5	CUTRGEN4	CUTRGEN3	CUTRGEN2	CUTRGEN1	CUTRGEN0
15	14	13	12	11	10	9	8
Reserved		PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
7	6	5	4	3	2	1	0
Reserved		ZTE5	ZTE4	ZTE3	ZTE2	ZTE1	ZTE0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[24+n] n=0,1..5	**CDTRGENn**	**EPWM Compare Down Count Point Trigger DAC Enable Bits** EPWM can trigger DAC to start action when EPWM counter down count to CMP if this bit is set to1. 0 = EPWM Compare Down count point trigger DAC function Disabled. 1 = EPWM Compare Down count point trigger DAC function Enabled. **Note 1:** This bit should keep at 0 when EPWM counter operating in up counter type. **Note 2:** In complementary mode, CDTRGEN1, 3, 5 is used as another CDTRGEN for channel 0, 2, 4.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**CUTRGEn**	**EPWM Compare Up Count Point Trigger DAC Enable Bits** EPWM can trigger DAC to start action when EPWM counter counts up to CMP if this bit is set to1. 0 = EPWM Compare Up point trigger DAC function Disabled. 1 = EPWM Compare Up point trigger DAC function Enabled. **Note 1:** This bit should keep at 0 when EPWM counter operating in down counter type. **Note 2:** In complementary mode, CUTRGEN1, 3, 5 is used as another CUTRGEN for channel 0, 2, 4.
[15:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**PTEn**	**EPWM Period Point Trigger DAC Enable Bits** EPWM can trigger DAC to start action when EPWM counter counts up to (PERIODn+1) if this bit is set to1. 0 = EPWM period point trigger DAC function Disabled. 1 = EPWM period point trigger DAC function Enabled.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**ZTEn**	**EPWM Zero Point Trigger DAC Enable Bits** EPWM can trigger EADC/DAC/DMA to start action when EPWM counter down count to zero if this bit is set to1. 0 = EPWM period point trigger DAC function Disabled.

		1 = EPWM period point trigger DAC function Enabled.

EPWM Trigger EADC Source Select Register 0 (EPWM_EADCTS0)

Register	Offset	R/W	Description	Reset Value
EPWM_EADCTS0	EPWMx_BA+0xF8	R/W	EPWM Trigger EADC Source Select Register 0	0x0000_0000

31	30	29	28	27	26	25	24
TRGEN3	Reserved			TRGSEL3			
23	22	21	20	19	18	17	16
TRGEN2	Reserved			TRGSEL2			
15	14	13	12	11	10	9	8
TRGEN1	Reserved			TRGSEL1			
7	6	5	4	3	2	1	0
TRGEN0	Reserved			TRGSEL0			

Bits	Description	
[31]	**TRGEN3**	**EPWM_CH3 Trigger EADC Enable Bit** 0 = EPWM_CH3 Trigger EADC function Disabled. 1 = EPWM_CH3 Trigger EADC function Enabled.
[30:28]	**Reserved**	Reserved.
[27:24]	**TRGSEL3**	**EPWM_CH3 Trigger EADC Source Select** 0000 = EPWM_CH2 zero point. 0001 = EPWM_CH2 period point. 0010 = EPWM_CH2 zero or period point. 0011 = EPWM_CH2 up-count compared point. 0100 = EPWM_CH2 down-count compared point. 0101 = EPWM_CH3 zero point. 0110 = EPWM_CH3 period point. 0111 = EPWM_CH3 zero or period point. 1000 = EPWM_CH3 up-count compared point. 1001 = EPWM_CH3 down-count compared point. 1010 = EPWM_CH0 up-count free trigger compared point. 1011 = EPWM_CH0 down-count free trigger compared point. 1100 = EPWM_CH2 up-count free trigger compared point. 1101 = EPWM_CH2 down-count free trigger compared point. 1110 = EPWM_CH4 up-count free trigger compared point. 1111 = EPWM_CH4 down-count free trigger compared point.
[23]	**TRGEN2**	**EPWM_CH2 Trigger EADC Enable Bit** 0 = EPWM_CH2 Trigger EADC function Disabled. 1 = EPWM_CH2 Trigger EADC function Enabled.
[22:20]	**Reserved**	Reserved.

[19:16]	**TRGSEL2**	**EPWM_CH2 Trigger EADC Source Select** 0000 = EPWM_CH2 zero point. 0001 = EPWM_CH2 period point. 0010 = EPWM_CH2 zero or period point. 0011 = EPWM_CH2 up-count compared point. 0100 = EPWM_CH2 down-count compared point. 0101 = EPWM_CH3 zero point. 0110 = EPWM_CH3 period point. 0111 = EPWM_CH3 zero or period point. 1000 = EPWM_CH3 up-count compared point. 1001 = EPWM_CH3 down-count compared point. 1010 = EPWM_CH0 up-count free trigger compared point. 1011 = EPWM_CH0 down-count free trigger compared point. 1100 = EPWM_CH2 up-count free trigger compared point. 1101 = EPWM_CH2 down-count free trigger compared point. 1110 = EPWM_CH4 up-count free trigger compared point. 1111 = EPWM_CH4 down-count free trigger compared point.
[15]	**TRGEN1**	**EPWM_CH1 Trigger EADC Enable Bit** 0 = EPWM_CH1 Trigger EADC function Disabled. 1 = EPWM_CH1 Trigger EADC function Enabled.
[14:12]	**Reserved**	Reserved.
[11:8]	**TRGSEL1**	**EPWM_CH1 Trigger EADC Source Select** 0000 = EPWM_CH0 zero point. 0001 = EPWM_CH0 period point. 0010 = EPWM_CH0 zero or period point. 0011 = EPWM_CH0 up-count compared point. 0100 = EPWM_CH0 down-count compared point. 0101 = EPWM_CH1 zero point. 0110 = EPWM_CH1 period point. 0111 = EPWM_CH1 zero or period point. 1000 = EPWM_CH1 up-count compared point. 1001 = EPWM_CH1 down-count compared point. 1010 = EPWM_CH0 up-count free trigger compared point. 1011 = EPWM_CH0 down-count free trigger compared point. 1100 = EPWM_CH2 up-count free trigger compared point. 1101 = EPWM_CH2 down-count free trigger compared point. 1110 = EPWM_CH4 up-count free trigger compared point. 1111 = EPWM_CH4 down-count free trigger compared point.
[7]	**TRGEN0**	**EPWM_CH0 Trigger EADC Enable Bit** 0 = EPWM_CH0 Trigger EADC function Disabled. 1 = EPWM_CH0 Trigger EADC function Enabled.
[6:4]	**Reserved**	Reserved.
[3:0]	**TRGSEL0**	**EPWM_CH0 Trigger EADC Source Select** 0000 = EPWM_CH0 zero point. 0001 = EPWM_CH0 period point.

		0010 = EPWM_CH0 zero or period point. 0011 = EPWM_CH0 up-count compared point. 0100 = EPWM_CH0 down-count compared point. 0101 = EPWM_CH1 zero point. 0110 = EPWM_CH1 period point. 0111 = EPWM_CH1 zero or period point. 1000 = EPWM_CH1 up-count compared point. 1001 = EPWM_CH1 down-count compared point. 1010 = EPWM_CH0 up-count free trigger compared point. 1011 = EPWM_CH0 down-count free trigger compared point. 1100 = EPWM_CH2 up-count free trigger compared point. 1101 = EPWM_CH2 down-count free trigger compared point. 1110 = EPWM_CH4 up-count free trigger compared point. 1111 = EPWM_CH4 down-count free trigger compared point.

EPWM Trigger EADC Source Select Register 1 (EPWM_EADCTS1)

Register	Offset	R/W	Description	Reset Value
EPWM_EADCTS1	EPWMx_BA+0xFC	R/W	EPWM Trigger EADC Source Select Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
TRGEN5	Reserved			TRGSEL5			
7	6	5	4	3	2	1	0
TRGEN4	Reserved			TRGSEL4			

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**TRGEN5**	**EPWM_CH5 Trigger EADC Enable Bit** 0 = EPWM_CH5 Trigger EADC function Disabled. 1 = EPWM_CH5 Trigger EADC function Enabled.
[14:12]	**Reserved**	Reserved.
[11:8]	**TRGSEL5**	**EPWM_CH5 Trigger EADC Source Select** 0000 = EPWM_CH4 zero point. 0001 = EPWM_CH4 period point. 0010 = EPWM_CH4 zero or period point. 0011 = EPWM_CH4 up-count compared point. 0100 = EPWM_CH4 down-count compared point. 0101 = EPWM_CH5 zero point. 0110 = EPWM_CH5 period point. 0111 = EPWM_CH5 zero or period point. 1000 = EPWM_CH5 up-count compared point. 1001 = EPWM_CH5 down-count compared point. 1010 = EPWM_CH0 up-count free trigger compared point. 1011 = EPWM_CH0 down-count free trigger compared point. 1100 = EPWM_CH2 up-count free trigger compared point. 1101 = EPWM_CH2 down-count free trigger compared point. 1110 = EPWM_CH4 up-count free trigger compared point. 1111 = EPWM_CH4 down-count free trigger compared point.
[7]	**TRGEN4**	**EPWM_CH4 Trigger EADC Enable Bit** 0 = EPWM_CH4 Trigger EADC function Disabled. 1 = EPWM_CH4 Trigger EADC function Enabled.

[6:4]	**Reserved**	Reserved.
[3:0]	**TRGSEL4**	**EPWM_CH4 Trigger EADC Source Select** 0000 = EPWM_CH4 zero point. 0001 = EPWM_CH4 period point. 0010 = EPWM_CH4 zero or period point. 0011 = EPWM_CH4 up-count compared point. 0100 = EPWM_CH4 down-count compared point. 0101 = EPWM_CH5 zero point. 0110 = EPWM_CH5 period point. 0111 = EPWM_CH5 zero or period point. 1000 = EPWM_CH5 up-count compared point. 1001 = EPWM_CH5 down-count compared point. 1010 = EPWM_CH0 up-count free trigger compared point. 1011 = EPWM_CH0 down-count free trigger compared point. 1100 = EPWM_CH2 up-count free trigger compared point. 1101 = EPWM_CH2 down-count free trigger compared point. 1110 = EPWM_CH4 up-count free trigger compared point. 1111 = EPWM_CH4 down-count free trigger compared point.

EPWM Free Trigger Compare Register 0_1, 2_3, 4_5 (EPWM_FTCMPDAT0_1, 2_3, 4_5)

Register	Offset	R/W	Description	Reset Value
EPWM_FTCMPDAT0_1	EPWMx_BA+0x100	R/W	EPWM Free Trigger Compare Register 0/1	0x0000_0000
EPWM_FTCMPDAT2_3	EPWMx_BA+0x104	R/W	EPWM Free Trigger Compare Register 2/3	0x0000_0000
EPWM_FTCMPDAT4_5	EPWMx_BA+0x108	R/W	EPWM Free Trigger Compare Register 4/5	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**FTCMP**	**EPWM Free Trigger Compare Register** FTCMP is used to compare with even CNT (EPWM_CNTm[15:0], m=0,2,4) to trigger EADC. EPWM_FTCMPDAT0_1, EPWM_FTCMPDAT2_3, EPWM_FTCMPDAT4_5 corresponding complementary pairs EPWM_CH0and EPWM_CH1, EPWM_CH2 and EPWM_CH3, EPWM_CH4 and EPWM_CH5.

EPWM Synchronous Start Control Register (EPWM_SSCTL)

Register	Offset	R/W	Description	Reset Value
EPWM_SSCTL	EPWMx_BA+0x110	R/W	EPWM Synchronous Start Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						SSRC	
7	6	5	4	3	2	1	0
Reserved		SSEN5	SSEN4	SSEN3	SSEN2	SSEN1	SSEN0

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9:8]	**SSRC**	**EPWM Synchronous Start Source Select Bits** 00 = Synchronous start source come from EPWM0. 01 = Synchronous start source come from EPWM1. 10 = Synchronous start source come from BPWM0. 11 = Synchronous start source come from BPWM1.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**SSENn**	**EPWM Synchronous Start Function Enable Bits** When synchronous start function is enabled, the EPWM counter enable register (EPWM_CNTEN) can be enabled by writing EPWM synchronous start trigger bit (CNTSEN). 0 = EPWM synchronous start function Disabled. 1 = EPWM synchronous start function Enabled.

EPWM Synchronous Start Trigger Register (EPWM_SSTRG)

Register	Offset	R/W	Description	Reset Value
EPWM_SSTRG	EPWMx_BA+0x114	W	EPWM Synchronous Start Trigger Register	0x0000_0000

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

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	CNTSEN	**EPWM Counter Synchronous Start Enable (Write Only)** PMW counter synchronous enable function is used to make selected EPWM channels (include EPWM0_CHx and EPWM1_CHx) start counting at the same time. Writing this bit to 1 will also set the counter enable bit (CNTENn, n denotes channel 0 to 5) if correlated EPWM channel counter synchronous start function is enabled.

EPWM Leading Edge Blanking Control Register (EPWM_LEBCTL)

Register	Offset	R/W	Description	Reset Value
EPWM_LEBCTL	EPWMx_BA+0x118	R/W	EPWM Leading Edge Blanking Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						TRGTYPE	
15	14	13	12	11	10	9	8
Reserved					SRCEN4	SRCEN2	SRCEN0
7	6	5	4	3	2	1	0
Reserved							LEBEN

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17:16]	**TRGTYPE**	**EPWM Leading Edge Blanking Trigger Type** 0 = When detect leading edge blanking source rising edge, blanking counter start counting. 1 = When detect leading edge blanking source falling edge, blanking counter start counting. 2 = When detect leading edge blanking source rising or falling edge, blanking counter start counting. 3 = Reserved.
[15:11]	**Reserved**	Reserved.
[10]	**SRCEN4**	**EPWM Leading Edge Blanking Source From EPWM_CH4 Enable Bit** 0 = EPWM Leading Edge Blanking Source from EPWM_CH4 Disabled. 1 = EPWM Leading Edge Blanking Source from EPWM_CH4 Enabled.
[9]	**SRCEN2**	**EPWM Leading Edge Blanking Source From EPWM_CH2 Enable Bit** 0 = EPWM Leading Edge Blanking Source from EPWM_CH2 Disabled. 1 = EPWM Leading Edge Blanking Source from EPWM_CH2 Enabled.
[8]	**SRCEN0**	**EPWM Leading Edge Blanking Source From EPWM_CH0 Enable Bit** 0 = EPWM Leading Edge Blanking Source from EPWM_CH0 Disabled. 1 = EPWM Leading Edge Blanking Source from EPWM_CH0 Enabled.
[7:1]	**Reserved**	Reserved.
[0]	**LEBEN**	**EPWM Leading Edge Blanking Enable Bit** 0 = EPWM Leading Edge Blanking Disabled. 1 = EPWM Leading Edge Blanking Enabled.

EPWM Leading Edge Blanking Counter Register (EPWM_LEBCNT)

Register	Offset	R/W	Description	Reset Value
EPWM_LEBCNT	EPWMx_BA+0x11C	R/W	EPWM Leading Edge Blanking Counter Register	0x0000_0000

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

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8:0]	**LEBCNT**	**EPWM Leading Edge Blanking Counter** This counter value decides leading edge blanking window size. Blanking window size = LEBCNT+1, and LEB counter clock base is ECLK.

EPWM Status Register (EPWM_STATUS)

Register	Offset	R/W	Description	Reset Value
EPWM_STATUS	EPWMx_BA+0x120	R/W	EPWM Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							DACTRGF
23	22	21	20	19	18	17	16
Reserved		EADCTRGF5	EADCTRGF4	EADCTRGF3	EADCTRGF2	EADCTRGF1	EADCTRGF0
15	14	13	12	11	10	9	8
Reserved					SYNCINF4	SYNCINF2	SYNCINF0
7	6	5	4	3	2	1	0
Reserved		CNTMAXF5	CNTMAXF4	CNTMAXF3	CNTMAXF2	CNTMAXF1	CNTMAXF0

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24]	**DACTRGF**	**DAC Start of Conversion Flag** 0 = No DAC start of conversion trigger event has occurred. 1 = A DAC start of conversion trigger event has occurred. **Note:** This bit can be cleared by software writing 1.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**EADCTRGFn**	**EADC Start of Conversion Flag** 0 = No EADC start of conversion trigger event has occurred. 1 = An EADC start of conversion trigger event has occurred. **Note:** This bit can be cleared by software writing 1.
[15:11]	**Reserved**	Reserved.
[8+n/2] n=0,2,4	**SYNCINFn**	**Input Synchronization Latched Flag** 0 = No SYNC_IN event has occurred. 1 = A SYNC_IN event has occurred. **Note:** This bit can be cleared by software writing 1.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CNTMAXFn**	**Time-base Counter Equal to 0xFFFF Latched Flag** 0 = The time-base counter never reached its maximum value 0xFFFF. 1 = The time-base counter reached its maximum value. **Note:** This bit can be cleared by software writing 1.

EPWM Interrupt Flag Accumulator Register (EPWM_IFAn)

Register	Offset	R/W	Description	Reset Value
EPWM_IFA0	EPWMx_BA+0x130	R/W	EPWM Interrupt Flag Accumulator Register 0	0x0000_0000
EPWM_IFA1	EPWMx_BA+0x134	R/W	EPWM Interrupt Flag Accumulator Register 1	0x0000_0000
EPWM_IFA2	EPWMx_BA+0x138	R/W	EPWM Interrupt Flag Accumulator Register 2	0x0000_0000
EPWM_IFA3	EPWMx_BA+0x13C	R/W	EPWM Interrupt Flag Accumulator Register 3	0x0000_0000
EPWM_IFA4	EPWMx_BA+0x140	R/W	EPWM Interrupt Flag Accumulator Register 4	0x0000_0000
EPWM_IFA5	EPWMx_BA+0x144	R/W	EPWM Interrupt Flag Accumulator Register 5	0x0000_0000

31	30	29	28	27	26	25	24
IFAEN	Reserved	IFASEL		Reserved			STPMOD
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
IFACNT							
7	6	5	4	3	2	1	0
IFACNT							

Bits	Description	
[31]	**IFAEN**	**EPWM_CHn Interrupt Flag Accumulator Enable Bits** 0 = EPWM_CHn interrupt flag accumulator Disabled. 1 = EPWM_CHn interrupt flag accumulator Enabled.
[30]	**Reserved**	Reserved.
[29:28]	**IFASEL**	**EPWM_CHn Interrupt Flag Accumulator Source Select** 00 = EPWM_CHn zero point. 01 = EPWM_CHn period in channel n. 10 = EPWM_CHn up-count compared point. 11 = EPWM_CHn down-count compared point.
[27:25]	**Reserved**	Reserved.
[24]	**STPMOD**	**EPWM_CHn Accumulator Stop Mode Enable Bits** 0 = EPWM_CHn Stop Mode Disabled. 1 = EPWM_CHn Stop Mode Enabled.
[23:16]	**Reserved**	Reserved.
[15:0]	**IFACNT**	**EPWM_CHn Interrupt Flag Counter** The register sets the count number which defines how many times of EPWM_CHn period occurs to set bit IFAIFn to request the EPWM period interrupt.

		EPWM flag will be set in every IFACNT[15:0] times of EPWM period.

EPWM Accumulator Interrupt Flag Register (EPWM_AINTSTS)

Register	Offset	R/W	Description	Reset Value
EPWM_AINTSTS	EPWMx_BA+0x150	R/W	EPWM Accumulator Interrupt Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		IFAIF5	IFAIF4	IFAIF3	IFAIF2	IFAIF1	IFAIF0

Bits	Description	
[31:6]	Reserved	Reserved.
[n] n=0,1..5	IFAIFn	**EPWM_CHn Interrupt Flag Accumulator Interrupt Flag** Flag is set by hardware when condition match IFASEL in EPWM_IFAn register, software can clear this bit by writing 1 to it.

EPWM Accumulator Interrupt Enable Register (EPWM_AINTEN)

Register	Offset	R/W	Description	Reset Value
EPWM_AINTEN	EPWMx_BA+0x154	R/W	EPWM Accumulator Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**IFAIEN5**	**IFAIEN4**	**IFAIEN3**	**IFAIEN2**	**IFAIEN1**	**IFAIEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**IFAIENn**	**EPWM_CHn Interrupt Flag Accumulator Interrupt Enable Bits** 0 = Interrupt Flag accumulator interrupt Disabled. 1 = Interrupt Flag accumulator interrupt Enabled.

EPWM Accumulator PDMA Control Register (EPWM_APDMACTL)

Register	Offset	R/W	Description	Reset Value
EPWM_APDMACTL	EPWMx_BA+0x158	R/W	EPWM Accumulator PDMA Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		APDMAEN5	APDMAEN4	APDMAEN3	APDMAEN2	APDMAEN1	APDMAEN0

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**APDMAENn**	**Channel n Accumulator PDMA Enable Bits** 0 = Channel n PDMA function Disabled. 1 = Channel n PDMA function Enabled for the channel n to trigger PDMA to transfer memory data to register.

EPWM Fault Detect Enable Register (EPWM_FDEN)

Register	Offset	R/W	Description	Reset Value
EPWM_FDEN	EPWMx_BA+0x160	R/W	EPWM Fault Detect Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		FDCKS5	FDCKS4	FDCKS3	FDCKS2	FDCKS1	FDCKS0
15	14	13	12	11	10	9	8
Reserved		FDODIS5	FDODIS4	FDODIS3	FDODIS2	FDODIS1	FDODIS0
7	6	5	4	3	2	1	0
Reserved		FDEN5	FDEN4	FDEN3	FDEN2	FDEN1	FDEN0

Bits	Description	
[31:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**FDCKSn**	**EPWM Channel n Fault Detect Clock Source Select Bit** 0 = EPWMx_CLK, x denotes 0 or 1. 1 = EPWMx_CLK divide by prescaler, x denotes 0 or 1.
[8+n] n=0,1..5	**FDODISn**	**EPWM Channel n Output Fault Detect Disable Bit** 0 = EPWM detect fault and output Enabled. 1 = EPWM detect fault and output Disabled.
[n] n=0,1..5	**FDENn**	**EPWM Fault Detect Function Enable Bit** 0 = Fault detect function Disabled. 1 = Fault detect function Enabled.

EPWM Fault Detect Control Register 0~5 (EPWM_FDCTL0~5)

Register	Offset	R/W	Description	Reset Value
EPWM_FDCTL0	EPWMx_BA+0x164	R/W	EPWM Fault Detect Control Register 0	0x0000_0000
EPWM_FDCTL1	EPWMx_BA+0x168	R/W	EPWM Fault Detect Control Register 1	0x0000_0000
EPWM_FDCTL2	EPWMx_BA+0x16C	R/W	EPWM Fault Detect Control Register 2	0x0000_0000
EPWM_FDCTL3	EPWMx_BA+0x170	R/W	EPWM Fault Detect Control Register 3	0x0000_0000
EPWM_FDCTL4	EPWMx_BA+0x174	R/W	EPWM Fault Detect Control Register 4	0x0000_0000
EPWM_FDCTL5	EPWMx_BA+0x178	R/W	EPWM Fault Detect Control Register 5	0x0000_0000

31	30	29	28	27	26	25	24
FDDGEN	**Reserved**	**FDCKSEL**		**Reserved**			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved					**DGSMPCYC**		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FDMSKEN	**Reserved**						
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**TRMSKCNT**						

Bits	Description	
[31]	**FDDGEN**	**Fault Detect Deglitch Enable Bit** 0 = Fault detect deglitch function Disabled. 1 = Fault detect deglitch function Enabled.
[30]	**Reserved**	Reserved.
[29:28]	**FDCKSEL**	**EPWM Channel Fault Detect Clock Select** 00 = FLT_CLK/1. 01 = FLT_CLK/2. 10 = FLT_CLK/4. 11 = FLT_CLK/8. **Note:** FLT_CLK is FDCKSn (EPWM_FDENn[16+n], n=0,1..5) selected clock.
[27:19]	**Reserved**	Reserved.
[18:16]	**DGSMPCYC**	**Deglitch Sampling Cycle** FDCKS is set to 0: Sampling detect signal each EPWMx_CLK * (2^FDCKSEL) period and detect DGSMPCYC+1 times FDCKS is set to 1: Sampling detect signal each EPWMx_CLK * CLKPSC * (2^FDCKSEL) period and detect DGSMPCYC+1 times **Note:**

		CLKPSC (EPWM_CLKPSCn_m[11:0]) is 0: TRMSKCNT >= DGSMPCYC + 2. FDCKS is 1 and CLKPSC (EPWM_CLKPSCn_m[11:0]) is 1: TRMSKCNT >= DGSMPCYC + 1. FDCKS is 1 and CLKPSC (EPWM_CLKPSCn_m[11:0]) is 2: TRMSKCNT >= DGSMPCYC.
[15]	**FDMSKEN**	**Fault Detect Mask Enable Bit** 0 = Fault detect mask function Disabled. 1 = Fault detect mask function Enabled.
[14:7]	**Reserved**	Reserved.
[6:0]	**TRMSKCNT**	**Transition Mask Counter** The fault detect result will be masked before counter count from 0 to TRMSKCNT. FDCKS is set to 0: Mask time is EPWMx_CLK * (2^FDCKSEL) * (TRMSKCNT+2) FDCKS is set to 1: Mask time EPWMx_CLK * CLKPSC * (2^FDCKSEL) * (TRMSKCNT+2) **Note:** CLKPSC (EPWM_CLKPSCn_m[11:0]) is 0: TRMSKCNT >= DGSMPCYC + 2. FDCKS is 1 and CLKPSC (EPWM_CLKPSCn_m[11:0]) is 1: TRMSKCNT >= DGSMPCYC + 1. FDCKS is 1 and CLKPSC (EPWM_CLKPSCn_m[11:0]) is 2: TRMSKCNT >= DGSMPCYC.

EPWM Fault Detect Interrupt Enable Register (EPWM_FDIEN)

Register	Offset	R/W	Description	Reset Value
EPWM_FDIEN	EPWMx_BA+0x17C	R/W	EPWM Fault Detect Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**FDIEN5**	**FDIEN4**	**FDIEN3**	**FDIEN2**	**FDIEN1**	**FDIEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n]	**FDIENn**	**EPWM Channel n Fault Detect Interrupt Enable Bit** 0 = EPWM Channel n Fault Detect Interrupt Disabled. 1 = EPWM Channel n Fault Detect Interrupt Enabled.

EPWM Fault Detect Interrupt Flag Register (EPWM_FDSTS)

Register	Offset	R/W	Description	Reset Value
EPWM_FDSTS	EPWMx_BA+0x180	R/W	EPWM Fault Detect Interrupt Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		FDIF5	FDIF4	FDIF3	FDIF2	FDIF1	FDIF0

Bits	Description	
[31:6]	Reserved	Reserved.
[5:0]	FDIFn	**EPWM Channel n Fault Detect Interrupt Flag Bit** Fault Detect Interrupt Flag will be set when EPWM output short. Software can clear this bit by writing 1 to it.

EPWM Trigger EADC Prescale Control Register (EPWM_EADCPSCCTL)

Register	Offset	R/W	Description	Reset Value
EPWM_EADCPSCCTL	EPWMx_BA+0x184	R/W	EPWM Trigger EADC Prescale Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**PSCEN5**	**PSCEN4**	**PSCEN3**	**PSCEN2**	**PSCEN1**	**PSCEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**PSCENn**	**EPWM Trigger EADC Pre-scale Function Enable Bits** 0 = EPWM Trigger EADC Pre-scale Function Disabled. 1 = EPWM Trigger EADC Pre-scale Function Enabled.

EPWM Trigger EADC Prescale Register 0 (EPWM_EADCPSC0)

Register	Offset	R/W	Description	Reset Value
EPWM_EADCPSC0	EPWMx_BA+0x188	R/W	EPWM Trigger EADC Prescale Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				EADCPSC3			
23	22	21	20	19	18	17	16
Reserved				EADCPSC2			
15	14	13	12	11	10	9	8
Reserved				EADCPSC1			
7	6	5	4	3	2	1	0
Reserved				EADCPSC0			

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:24]	**EADCPSC3**	**EPWM Channel 3 Trigger EADC Prescale** The register sets the count number which defines (EADCPSC3+1) times of EPWM_CH3 trigger EADC event occurs to trigger EADC and set trigger EADC flag bit EADCTRGF3.
[23:20]	**Reserved**	Reserved.
[19:16]	**EADCPSC2**	**EPWM Channel 2 Trigger EADC Prescale** The register sets the count number which defines (EADCPSC2+1) times of EPWM_CH2 trigger EADC event occurs to trigger EADC and set trigger EADC flag bit EADCTRGF2.
[15:12]	**Reserved**	Reserved.
[11:8]	**EADCPSC1**	**EPWM Channel 1 Trigger EADC Prescale** The register sets the count number which defines (EADCPSC1+1) times of EPWM_CH1 trigger EADC event occurs to trigger EADC and set trigger EADC flag bit EADCTRGF1.
[7:4]	**Reserved**	Reserved.
[3:0]	**EADCPSC0**	**EPWM Channel 0 Trigger EADC Prescale** The register sets the count number which defines (EADCPSC0+1) times of EPWM_CH0 trigger EADC event occurs to trigger EADC and set trigger EADC flag bit EADCTRGF0.

EPWM Trigger EADC Prescale Register 1 (EPWM_EADCPSC1)

Register	Offset	R/W	Description	Reset Value
EPWM_EADCPSC1	EPWMx_BA+0x18C	R/W	EPWM Trigger EADC Prescale Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				EADCPSC5			
7	6	5	4	3	2	1	0
Reserved				EADCPSC4			

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:8]	**EADCPSC5**	**EPWM Channel 5 Trigger EADC Prescale** The register sets the count number which defines (EADCPSC5+1) times of EPWM_CH5 trigger EADC event occurs to trigger EADC and set trigger EADC flag bit EADCTRGF5.
[7:4]	**Reserved**	Reserved.
[3:0]	**EADCPSC4**	**EPWM Channel 4 Trigger EADC Prescale** The register sets the count number which defines (EADCPSC4+1) times of EPWM_CH4 trigger EADC event occurs to trigger EADC and set trigger EADC flag bit EADCTRGF4.

EPWM Trigger EADC Prescale Counter Register 0 (EPWM_EADCPSCNT0)

Register	Offset	R/W	Description	Reset Value
EPWM_EADCPSCNT0	EPWMx_BA+0x190	R/W	EPWM Trigger EADC Prescale Counter Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				PSCNT3			
23	22	21	20	19	18	17	16
Reserved				PSCNT2			
15	14	13	12	11	10	9	8
Reserved				PSCNT1			
7	6	5	4	3	2	1	0
Reserved				PSCNT0			

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:24]	**PSCNT3**	**EPWM Trigger EADC Prescale Counter 3** User can monitor PSCNT3 to know the current value in 4-bit trigger EADC prescale counter. **Note 1:** user can write only when PSCEN3 is 0. **Note 2:** Write data limitation: PSCNT3 < EADCPSC3.
[23:20]	**Reserved**	Reserved.
[19:16]	**PSCNT2**	**EPWM Trigger EADC Prescale Counter 2** User can monitor PSCNT2 to know the current value in 4-bit trigger EADC prescale counter. **Note 1:** user can write only when PSCEN2 is 0. **Note 2:** Write data limitation: PSCNT2 < EADCPSC2.
[15:12]	**Reserved**	Reserved.
[11:8]	**PSCNT1**	**EPWM Trigger EADC Prescale Counter 1** User can monitor PSCNT1 to know the current value in 4-bit trigger EADC prescale counter. **Note 1:** user can write only when PSCEN1 is 0. **Note 2:** Write data limitation: PSCNT1 < EADCPSC1.
[7:4]	**Reserved**	Reserved.
[3:0]	**PSCNT0**	**EPWM Trigger EADC Prescale Counter 0** User can monitor PSCNT0 to know the current value in 4-bit trigger EADC prescale counter. **Note 1:** user can write only when PSCEN0 is 0. **Note 2:** Write data limitation: PSCNT0 < EADCPSC0.

EPWM Trigger EADC Prescale Counter Register 1 (EPWM_EADCPSCNT1)

Register	Offset	R/W	Description	Reset Value
EPWM_EADCPSCNT1	EPWMx_BA+0x194	R/W	EPWM Trigger EADC Prescale Counter Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**PSCNT5**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**PSCNT4**			

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:8]	**PSCNT5**	**EPWM Trigger EADC Prescale Counter 5** User can monitor PSCNT5 to know the current value in 4-bit trigger EADC prescale counter. **Note 1:** user can write only when PSCEN5 is 0. **Note 2:** Write data limitation: PSCNT5 < EADCPSC5.
[7:4]	**Reserved**	Reserved.
[3:0]	**PSCNT4**	**EPWM Trigger EADC Prescale Counter 4** User can monitor PSCNT4 to know the current value in 4-bit trigger EADC prescale counter. **Note 1:** user can write only when PSCEN4 is 0. **Note 2:** Write data limitation: PSCNT4 < EADCPSC4.

EPWM Capture Input Enable Register (EPWM_CAPINEN)

Register	Offset	R/W	Description	Reset Value
EPWM_CAPINEN	EPWMx_BA+0x200	R/W	EPWM Capture Input Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CAPINEN5**	**CAPINEN4**	**CAPINEN3**	**CAPINEN2**	**CAPINEN1**	**CAPINEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CAPINENn**	**Capture Input Enable Bits** 0 = EPWM Channel capture input path Disabled. The input of EPWM channel capture function is always regarded as 0. 1 = EPWM Channel capture input path Enabled. The input of EPWM channel capture function comes from correlative multifunction pin.

EPWM Capture Control Register (EPWM_CAPCTL)

Register	Offset	R/W	Description	Reset Value
EPWM_CAPCTL	EPWMx_BA+0x204	R/W	EPWM Capture Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		FCRLDEN5	FCRLDEN4	FCRLDEN3	FCRLDEN2	FCRLDEN1	FCRLDEN0
23	22	21	20	19	18	17	16
Reserved		RCRLDEN5	RCRLDEN4	RCRLDEN3	RCRLDEN2	RCRLDEN1	RCRLDEN0
15	14	13	12	11	10	9	8
Reserved		CAPINV5	CAPINV4	CAPINV3	CAPINV2	CAPINV1	CAPINV0
7	6	5	4	3	2	1	0
Reserved		CAPEN5	CAPEN4	CAPEN3	CAPEN2	CAPEN1	CAPEN0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[24+n] n=0,1..5	**FCRLDENn**	**Falling Capture Reload Enable Bits** 0 = Falling capture reload counter Disabled. 1 = Falling capture reload counter Enabled.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**RCRLDENn**	**Rising Capture Reload Enable Bits** 0 = Rising capture reload counter Disabled. 1 = Rising capture reload counter Enabled.
[15:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**CAPINVn**	**Capture Inverter Enable Bits** 0 = Capture source inverter Disabled. 1 = Capture source inverter Enabled. Reverse the input signal from GPIO.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CAPENn**	**Capture Function Enable Bits** 0 = Capture function Disabled. EPWM_RCAPDATn/EPWM_FCAPDATn register will not be updated. 1 = Capture function Enabled. Capture latched the EPWM counter value when detected rising or falling edge of input signal and saved to RCAPDAT (Rising latch) and FCAPDAT (Falling latch).

EPWM Capture Status Register (EPWM_CAPSTS)

Register	Offset	R/W	Description	Reset Value
EPWM_CAPSTS	EPWMx_BA+0x208	R	EPWM Capture Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		CFLIFOV5	CFLIFOV4	CFLIFOV3	CFLIFOV2	CFLIFOV1	CFLIFOV0
7	6	5	4	3	2	1	0
Reserved		CRLIFOV5	CRLIFOV4	CRLIFOV3	CRLIFOV2	CRLIFOV1	CRLIFOV0

Bits	Description	
[31:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**CFLIFOVn**	**Capture Falling Latch Interrupt Flag Overrun Status (Read Only)** This flag indicates if falling latch happened when the corresponding CFLIFn(EPWM_CAPIF[8+n]) is 1. **Note:** This bit will be cleared automatically when user clear corresponding CFLIFn(EPWM_CAPIF[n]).
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CRLIFOVn**	**Capture Rising Latch Interrupt Flag Overrun Status (Read Only)** This flag indicates if rising latch happened when the corresponding CRLIFn(EPWM_CAPIF[n]) is 1. **Note:** This bit will be cleared automatically when user clear corresponding CRLIFn(EPWM_CAPIF[n]).

EPWM Rising Capture Data Register 0~5 (EPWM_RCAPDAT 0~5)

Register	Offset	R/W	Description	Reset Value
EPWM_RCAPDAT0	EPWMx_BA+0x20C	R	EPWM Rising Capture Data Register 0	0x0000_0000
EPWM_RCAPDAT1	EPWMx_BA+0x214	R	EPWM Rising Capture Data Register 1	0x0000_0000
EPWM_RCAPDAT2	EPWMx_BA+0x21C	R	EPWM Rising Capture Data Register 2	0x0000_0000
EPWM_RCAPDAT3	EPWMx_BA+0x224	R	EPWM Rising Capture Data Register 3	0x0000_0000
EPWM_RCAPDAT4	EPWMx_BA+0x22C	R	EPWM Rising Capture Data Register 4	0x0000_0000
EPWM_RCAPDAT5	EPWMx_BA+0x234	R	EPWM Rising Capture Data Register 5	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**RCAPDAT**	**EPWM Rising Capture Data Register (Read Only)** When rising capture condition happened, the EPWM counter value will be saved in this register.

EPWM Falling Capture Data Register 0~5 (EPWM_FCAPDAT 0~5)

Register	Offset	R/W	Description	Reset Value
EPWM_FCAPDAT0	EPWMx_BA+0x210	R	EPWM Falling Capture Data Register 0	0x0000_0000
EPWM_FCAPDAT1	EPWMx_BA+0x218	R	EPWM Falling Capture Data Register 1	0x0000_0000
EPWM_FCAPDAT2	EPWMx_BA+0x220	R	EPWM Falling Capture Data Register 2	0x0000_0000
EPWM_FCAPDAT3	EPWMx_BA+0x228	R	EPWM Falling Capture Data Register 3	0x0000_0000
EPWM_FCAPDAT4	EPWMx_BA+0x230	R	EPWM Falling Capture Data Register 4	0x0000_0000
EPWM_FCAPDAT5	EPWMx_BA+0x238	R	EPWM Falling Capture Data Register 5	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**FCAPDAT**	**EPWM Falling Capture Data Register (Read Only)** When falling capture condition happened, the EPWM counter value will be saved in this register.

EPWM PDMA Control Register (EPWM_PDMACTL)

Register	Offset	R/W	Description	Reset Value
EPWM_PDMACTL	EPWMx_BA+0x23C	R/W	EPWM PDMA Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved			CHSEL45	CAPORD45	CAPMOD45		CHEN45
15	14	13	12	11	10	9	8
Reserved			CHSEL23	CAPORD23	CAPMOD23		CHEN23
7	6	5	4	3	2	1	0
Reserved			CHSEL01	CAPORD01	CAPMOD01		CHEN01

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20]	**CHSEL45**	**Select Channel 4/5 to Do PDMA Transfer** 0 = Channel4. 1 = Channel5.
[19]	**CAPORD45**	**Capture Channel 4/5 Rising/Falling Order** Set this bit to determine whether the EPWM_RCAPDAT4/5 or EPWM_FCAPDAT4/5 is the first captured data transferred to memory through PDMA when CAPMOD45 =11. 0 = EPWM_FCAPDAT4/5 is the first captured data to memory. 1 = EPWM_RCAPDAT4/5 is the first captured data to memory.
[18:17]	**CAPMOD45**	**Select EPWM_RCAPDAT4/5 or EPWM_FCAPDAT4/5 to Do PDMA Transfer** 00 = Reserved. 01 = EPWM_RCAPDAT4/5. 10 = EPWM_FCAPDAT4/5. 11 = Both EPWM_RCAPDAT4/5 and EPWM_FCAPDAT4/5.
[16]	**CHEN45**	**Channel 4/5 PDMA Enable Bit** 0 = Channel 4/5 PDMA function Disabled. 1 = Channel 4/5 PDMA function Enabled for the channel 4/5 captured data and transfer to memory.
[15:13]	**Reserved**	Reserved.
[12]	**CHSEL23**	**Select Channel 2/3 to Do PDMA Transfer** 0 = Channel2. 1 = Channel3.
[11]	**CAPORD23**	**Capture Channel 2/3 Rising/Falling Order** Set this bit to determine whether the EPWM_RCAPDAT2/3 or EPWM_FCAPDAT2/3 is the first captured data transferred to memory through PDMA when CAPMOD23 =11. 0 = EPWM_FCAPDAT2/3 is the first captured data to memory.

		1 = EPWM_RCAPDAT2/3 is the first captured data to memory.
[10:9]	**CAPMOD23**	**Select EPWM_RCAPDAT2/3 or EPWM_FCAODAT2/3 to Do PDMA Transfer** 00 = Reserved. 01 = EPWM_RCAPDAT2/3. 10 = EPWM_FCAPDAT2/3. 11 = Both EPWM_RCAPDAT2/3 and EPWM_FCAPDAT2/3.
[8]	**CHEN23**	**Channel 2/3 PDMA Enable Bit** 0 = Channel 2/3 PDMA function Disabled. 1 = Channel 2/3 PDMA function Enabled for the channel 2/3 captured data and transfer to memory.
[7:5]	**Reserved**	Reserved.
[4]	**CHSEL01**	**Select Channel 0/1 to Do PDMA Transfer** 0 = Channel0. 1 = Channel1.
[3]	**CAPORD01**	**Capture Channel 0/1 Rising/Falling Order** Set this bit to determine whether the EPWM_RCAPDAT0/1 or EPWM_FCAPDAT0/1 is the first captured data transferred to memory through PDMA when CAPMOD01 =11. 0 = EPWM_FCAPDAT0/1 is the first captured data to memory. 1 = EPWM_RCAPDAT0/1 is the first captured data to memory.
[2:1]	**CAPMOD01**	**Select EPWM_RCAPDAT0/1 or EPWM_FCAPDAT0/1 to Do PDMA Transfer** 00 = Reserved. 01 = EPWM_RCAPDAT0/1. 10 = EPWM_FCAPDAT0/1. 11 = Both EPWM_RCAPDAT0/1 and EPWM_FCAPDAT0/1.
[0]	**CHEN01**	**Channel 0/1 PDMA Enable Bit** 0 = Channel 0/1 PDMA function Disabled. 1 = Channel 0/1 PDMA function Enabled for the channel 0/1 captured data and transfer to memory.

EPWM Capture Channel 0_1, 2_3, 4_5 PDMA Register (EPWM_PDMACAP 0_1, 2_3, 4_5)

Register	Offset	R/W	Description	Reset Value
EPWM_PDMACAP0_1	EPWMx_BA+0x240	R	EPWM Capture Channel 01 PDMA Register	0x0000_0000
EPWM_PDMACAP2_3	EPWMx_BA+0x244	R	EPWM Capture Channel 23 PDMA Register	0x0000_0000
EPWM_PDMACAP4_5	EPWMx_BA+0x248	R	EPWM Capture Channel 45 PDMA Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CAPBUF**	**EPWM Capture PDMA Register (Read Only)** This register is used as a buffer to transfer EPWM capture rising or falling data to memory by PDMA.

EPWM Capture Interrupt Enable Register (EPWM_CAPIEN)

Register	Offset	R/W	Description	Reset Value
EPWM_CAPIEN	EPWMx_BA+0x250	R/W	EPWM Capture Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		CAPFIEN5	CAPFIEN4	CAPFIEN3	CAPFIEN2	CAPFIEN1	CAPFIEN0
7	6	5	4	3	2	1	0
Reserved		CAPRIEN5	CAPRIEN4	CAPRIEN3	CAPRIEN2	CAPRIEN1	CAPRIEN0

Bits	Description	
[31:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**CAPFIENn**	**EPWM Capture Falling Latch Interrupt Enable Bits** 0 = Capture falling edge latch interrupt Disabled. 1 = Capture falling edge latch interrupt Enabled.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CAPRIENn**	**EPWM Capture Rising Latch Interrupt Enable Bits** 0 = Capture rising edge latch interrupt Disabled. 1 = Capture rising edge latch interrupt Enabled.

EPWM Capture Interrupt Flag Register (EPWM_CAPIF)

Register	Offset	R/W	Description	Reset Value
EPWM_CAPIF	EPWMx_BA+0x254	R/W	EPWM Capture Interrupt Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		CFLIF5	CFLIF4	CFLIF3	CFLIF2	CFLIF1	CFLIF0
7	6	5	4	3	2	1	0
Reserved		CRLIF5	CRLIF4	CRLIF3	CRLIF2	CRLIF1	CRLIF0

Bits	Description	
[31:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**CFLIFn**	**EPWM Capture Falling Latch Interrupt Flag** 0 = No capture falling latch condition happened. 1 = Capture falling latch condition happened, this flag will be set to high. **Note 1:** When Capture with PDMA operating, EPWM_CAPIF corresponding channel CFLIFn will be cleared by hardware after PDMA transfer data. **Note 2:** This bit is cleared by writing 1 to it.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CRLIFn**	**EPWM Capture Rising Latch Interrupt Flag** 0 = No capture rising latch condition happened. 1 = Capture rising latch condition happened, this flag will be set to high. **Note 1:** When Capture with PDMA operating, EPWM_CAPIF corresponding channel CRLIFn will be cleared by hardware after PDMA transfer data. **Note 2:** This bit is cleared by writing 1 to it.

EPWM Period Register Buffer 0~5 (EPWM_PBUF0~5)

Register	Offset	R/W	Description	Reset Value
EPWM_PBUF0	EPWMx_BA+0x304	R	EPWM PERIOD0 Buffer	0x0000_0000
EPWM_PBUF1	EPWMx_BA+0x308	R	EPWM PERIOD1 Buffer	0x0000_0000
EPWM_PBUF2	EPWMx_BA+0x30C	R	EPWM PERIOD2 Buffer	0x0000_0000
EPWM_PBUF3	EPWMx_BA+0x310	R	EPWM PERIOD3 Buffer	0x0000_0000
EPWM_PBUF4	EPWMx_BA+0x314	R	EPWM PERIOD4 Buffer	0x0000_0000
EPWM_PBUF5	EPWMx_BA+0x318	R	EPWM PERIOD5 Buffer	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PBUF**	**EPWM Period Register Buffer (Read Only)** Used as PERIOD active register.

EPWM Comparator Register Buffer 0~5 (EPWM_CMPBUF0~5)

Register	Offset	R/W	Description	Reset Value
EPWM_CMPBUF0	EPWMx_BA+0x31C	R	EPWM CMPDAT0 Buffer	0x0000_0000
EPWM_CMPBUF1	EPWMx_BA+0x320	R	EPWM CMPDAT1 Buffer	0x0000_0000
EPWM_CMPBUF2	EPWMx_BA+0x324	R	EPWM CMPDAT2 Buffer	0x0000_0000
EPWM_CMPBUF3	EPWMx_BA+0x328	R	EPWM CMPDAT3 Buffer	0x0000_0000
EPWM_CMPBUF4	EPWMx_BA+0x32C	R	EPWM CMPDAT4 Buffer	0x0000_0000
EPWM_CMPBUF5	EPWMx_BA+0x330	R	EPWM CMPDAT5 Buffer	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CMPBUF**	**EPWM Comparator Register Buffer (Read Only)** Used as CMP active register.

EPWM CLKPSC Buffer 0_1, 2_3, 4_5 (EPWM_CPSCBUF0_1, 2_3, 4_5)

Register	Offset	R/W	Description	Reset Value
EPWM_CPSCBUF0_1	EPWMx_BA+0x334	R	EPWM CLKPSC0_1 Buffer	0x0000_0000
EPWM_CPSCBUF2_3	EPWMx_BA+0x338	R	EPWM CLKPSC2_3 Buffer	0x0000_0000
EPWM_CPSCBUF4_5	EPWMx_BA+0x33C	R	EPWM CLKPSC4_5 Buffer	0x0000_0000

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

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**CPSCBUF**	**EPWM Counter Clock Prescale Buffer** Used as EPWM counter clock pre-scare active register.

EPWM_FTCMPDAT Buffer (EPWM_FTCBUF0_1,2_3,4_5)

Register	Offset	R/W	Description	Reset Value
EPWM_FTCBUF0_1	EPWMx_BA+0x340	R	EPWM FTCMPDAT0_1 Buffer	0x0000_0000
EPWM_FTCBUF2_3	EPWMx_BA+0x344	R	EPWM FTCMPDAT2_3 Buffer	0x0000_0000
EPWM_FTCBUF4_5	EPWMx_BA+0x348	R	EPWM FTCMPDAT4_5 Buffer	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**FTCMPBUF**	**EPWM FTCMPDAT Buffer (Read Only)** Used as FTCMP active buffer.

EPWM FTCMPDAT Indicator Register (EPWM_FTCI)

Register	Offset	R/W	Description	Reset Value
EPWM_FTCI	EPWMx_BA+0x34C	R/W	EPWM FTCMPDAT Indicator Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**FTCMD4**	**FTCMD2**	**FTCMD0**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**FTCMU4**	**FTCMU2**	**FTCMU0**

Bits	Description	
[31:11]	**Reserved**	Reserved.
[8+n/2] n=0,2,4	**FTCMDn**	**EPWM FTCMPDAT Down Indicator** Indicator is set by hardware when EPWM counter counts down and reaches EPWM_FTCMPDATn. Software can clear this bit by writing 1 to it.
[7:3]	**Reserved**	Reserved.
[n/2] n=0,2,4	**FTCMUn**	**EPWM FTCMPDAT Up Indicator** Indicator is set by hardware when EPWM counter counts up and reaches EPWM_FTCMPDATn. Software can clear this bit by writing 1 to it.

6.16 Basic PWM Generator and Capture Timer (BPWM)

6.16.1 Overview

The chip provides two BPWM generators — BPWM0 and BPWM1 as shown in Figure 6.16-1. Each BPWM supports 6 channels of BPWM output or input capture. There is a 12-bit prescaler to support flexible clock to the 16-bit BPWM counter with 16-bit comparator. The BPWM counter supports up, down and up-down counter types, all 6 channels share one counter. BPWM uses the comparator compared with counter to generate events. These events are used to generate BPWM pulse, interrupt and trigger signal for EADC to start conversion. For BPWM output control unit, it supports polarity output, independent pin mask and tri-state output enable.

The BPWM generator also supports input capture function to latch BPWM counter value to corresponding register when input channel has a rising transition, falling transition or both transition is happened.

6.16.2 Features

6.16.2.1 BPWM Function Features

- Supports maximum clock frequency up to maximum PLL frequency.
- Supports up to two BPWM modules; each module provides 6 output channels
- Supports independent mode for BPWM output/Capture input channel
- Supports 12-bit prescalar from 1 to 4096
- Supports 16-bit resolution BPWM counter; each module provides 1 BPWM counter
  - Up, down and up/down counter operation type
- Supports mask function and tri-state enable for each BPWM pin
- Supports interrupt in the following events:
  - BPWM counter matches 0, period value or compared value
- Supports trigger EADC in the following events:
  - BPWM counter matches 0, period value or compared value

6.16.2.2 Capture Function Features

- Supports up to 12 capture input channels with 16-bit resolution
- Supports rising or falling capture condition
- Supports input rising/falling capture interrupt
- Supports rising/falling capture with counter reload option

6.16.3 Block Diagram

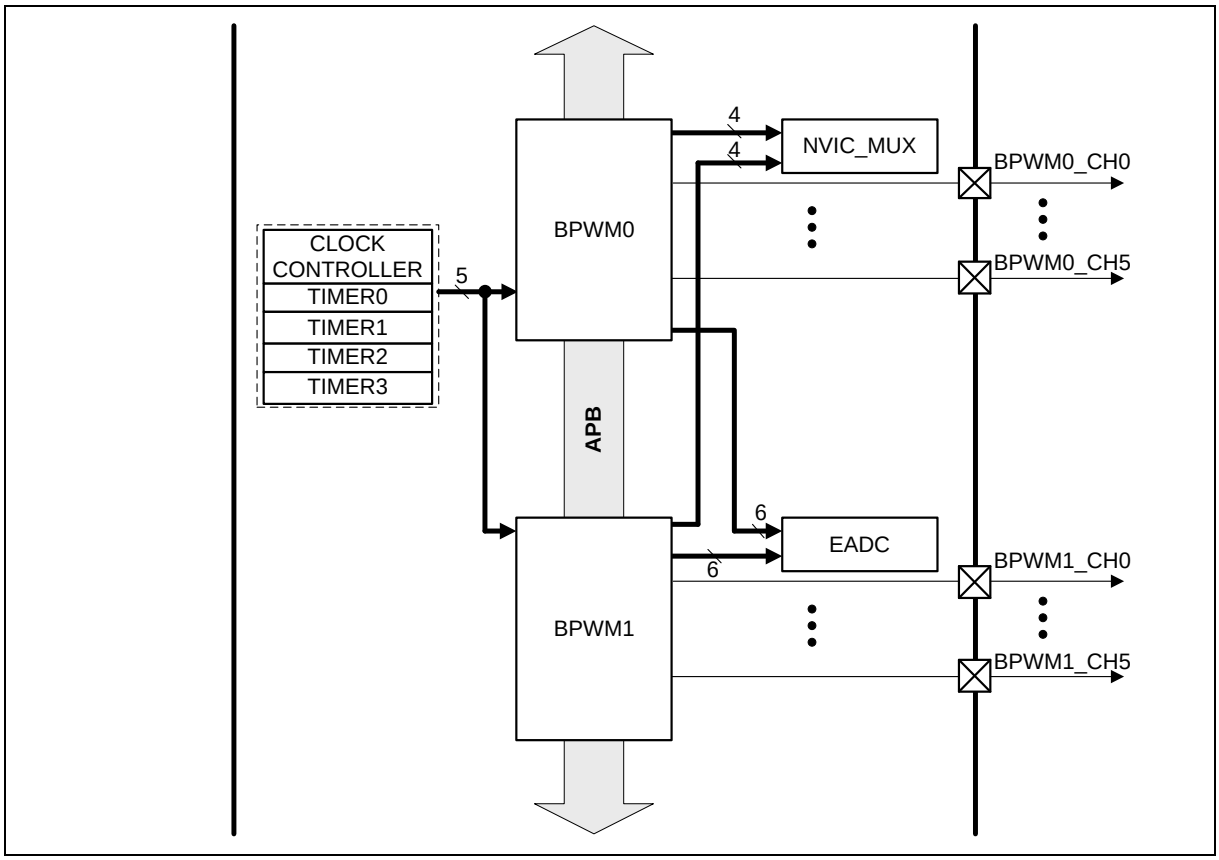


Figure 6.16-1 BPWM Generator Overview Block Diagram

Each BPWM generator has only one clock source inputs and can be selected from BPWM Clock or four TIMER trigger BPWM outputs as shown in Figure 6.16-2 by ECLKSRC0 (BPWM_CLKSRC[2:0]) for BPWM_CLK0. If the clock source of BPWM counter is selected from TIMERn interrupt events, the TRGPWM(TIMERn_TRGCTL[1] , n=0,1..3) bit must be set as 1.

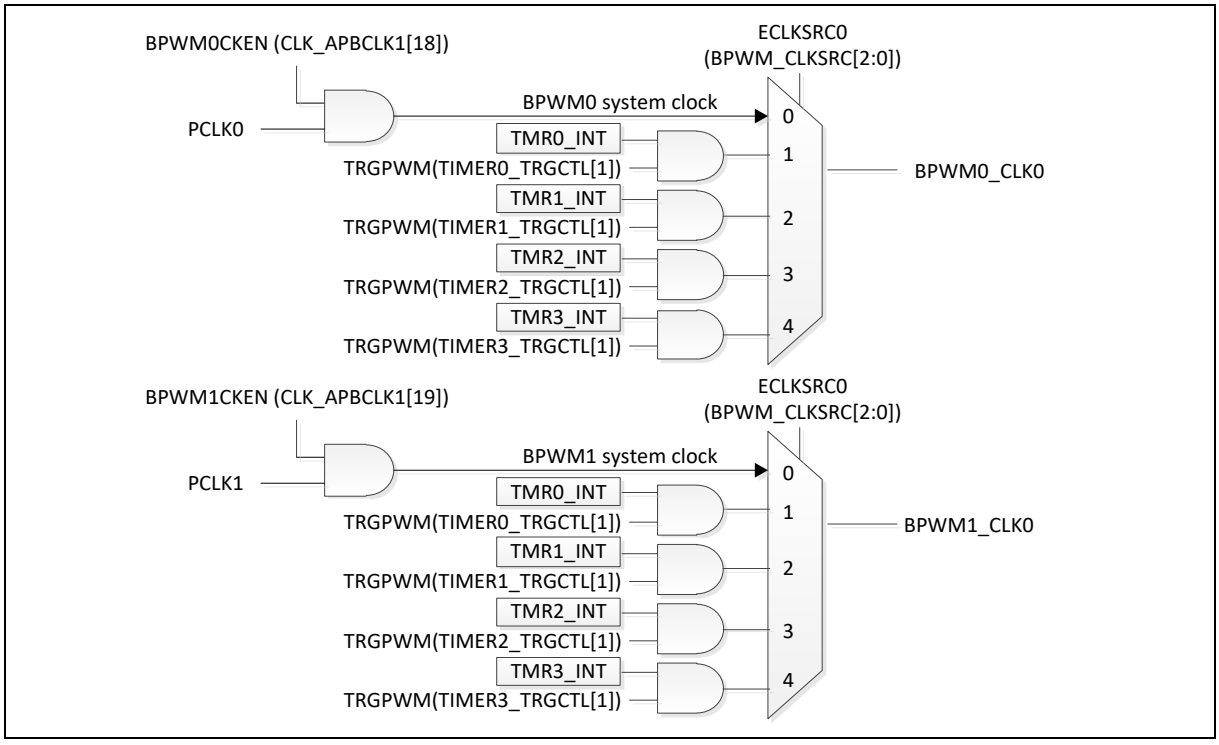


Figure 6.16-2 BPWM Clock Source Control

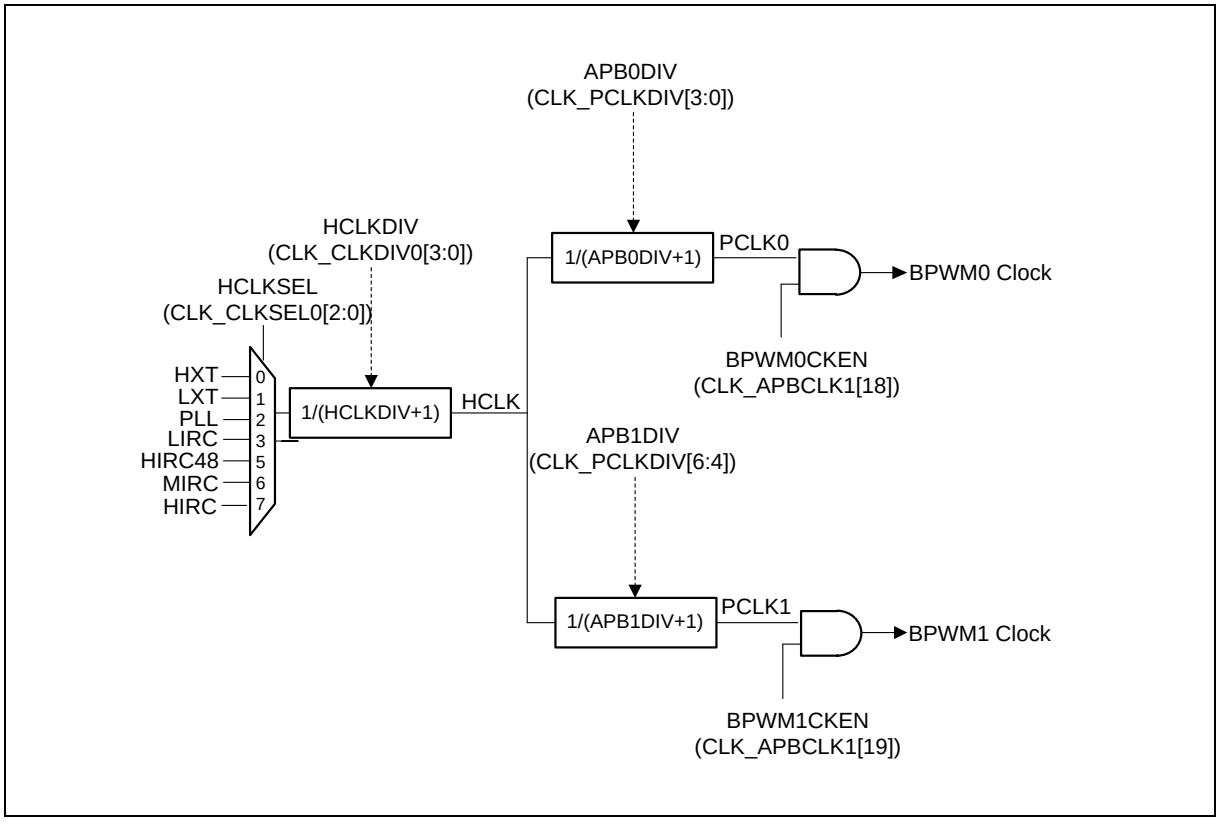


Figure 6.16-3 BPWM Clock Source Control

Figure 6.16-4 illustrates the architecture of BPWM Independent mode. All six channels share the same counter. When the counter counts to 0, PERIOD (BPWM_PERIOD[15:0]), or compared register(BPWM_CMPDATn[15:0], n = 0,1..5), events will be generated. These events are passed to the corresponding generators to generate BPWM pulse, interrupt signal and trigger signal for EADC to start conversion. Output control is used to change the BPWM pulse output state.

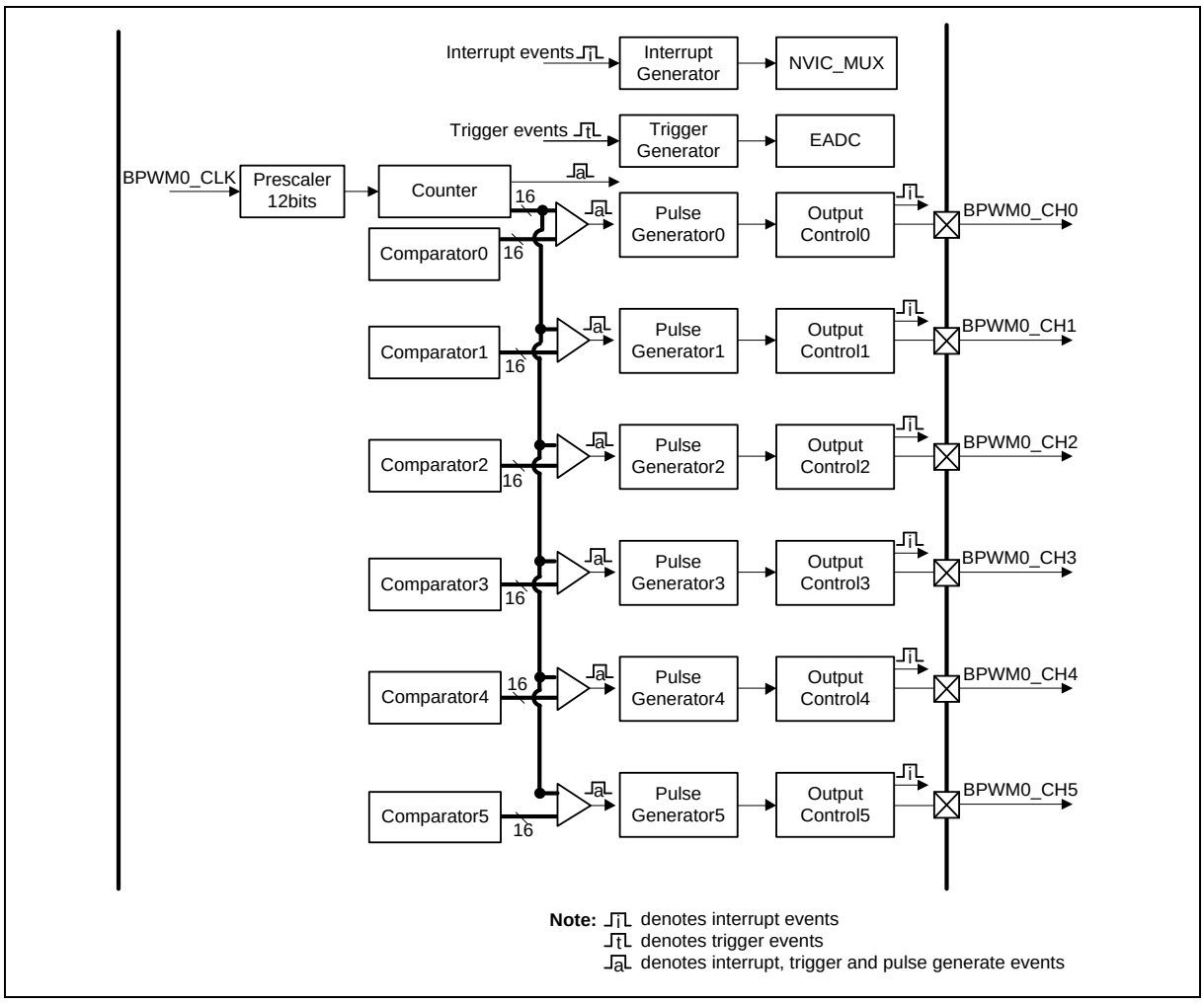


Figure 6.16-4 BPWM Independent Mode Architecture Diagram

6.16.4 Basic Configuration

6.16.4.1 BPWM0 Basic Configuration

- Clock Source Configuration
  - Select the source of BPWM0 peripheral clock on BPWM0SEL (CLK_CLKSEL2[8]).
  - Enable BPWM0 peripheral clock in BPWM0CKEN (CLK_APBCLK1[18]).
- Reset Configuration
  - Reset BPWM0 controller in BPWM0RST (SYS_IPRST2[18]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
BPWM0	BPWM0_CH0	PA.11	MFP9
		PA.0, PG.14	MFP12
		PE.2	MFP13
	BPWM0_CH1	PA.10	MFP9
		PA.1, PG.13	MFP12
		PE.3	MFP13
	BPWM0_CH2	PA.9	MFP9
		PA.2, PG.12	MFP12
		PE.4	MFP13
	BPWM0_CH3	PA.8	MFP9
		PA.3, PG.11	MFP12
		PE.5	MFP13
	BPWM0_CH4	PF.5	MFP8
		PC.13	MFP9
		PA.4, PG.10	MFP12
		PE.6	MFP13
	BPWM0_CH5	PF.4	MFP8
		PD.12	MFP9
		PA.5, PG.9	MFP12
		PE.7	MFP13

6.16.4.2 BPWM1 Basic Configuration

- Clock Source Configuration
  - Select the source of BPWM1 peripheral clock on BPWM1SEL (CLK_CLKSEL2[9]).
  - Enable BPWM1 peripheral clock in BPWM1CKEN (CLK_APBCLK1[19]).
- Reset Configuration
  - Reset BPWM1 controller in BPWM1RST (SYS_IPRST2[19]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
BPWM1	BPWM1_CH0	PB.11	MFP10
		PF.3	MFP11
		PC.7, PF.0	MFP12
	BPWM1_CH1	PB.10	MFP10
		PF.2	MFP11

		PC.6, PF.1	MFP12
	BPWM1_CH2	PB.9	MFP10
		PA.12	MFP11
		PA.7	MFP12
	BPWM1_CH3	PB.8	MFP10
		PA.13	MFP11
		PA.6	MFP12
	BPWM1_CH4	PB.7	MFP10
		PA.14	MFP11
		PC.8	MFP12
	BPWM1_CH5	PB.6	MFP10
		PA.15	MFP11
		PE.13	MFP12

6.16.5 Functional Description

6.16.5.1 BPWM Prescaler

The BPWM prescaler is used to divide clock source, prescaler counting CLKPSC +1 times, and the BPWM counter only counts once. The prescale is set by CLKPSC (BPWM_CLKPSC[11:0]). Figure 6.16-5 shows an example of BPWM channel 0 CLKPSC waveform. The prescale counter will reload CLKPSC at the beginning of the next prescale counter down-count.

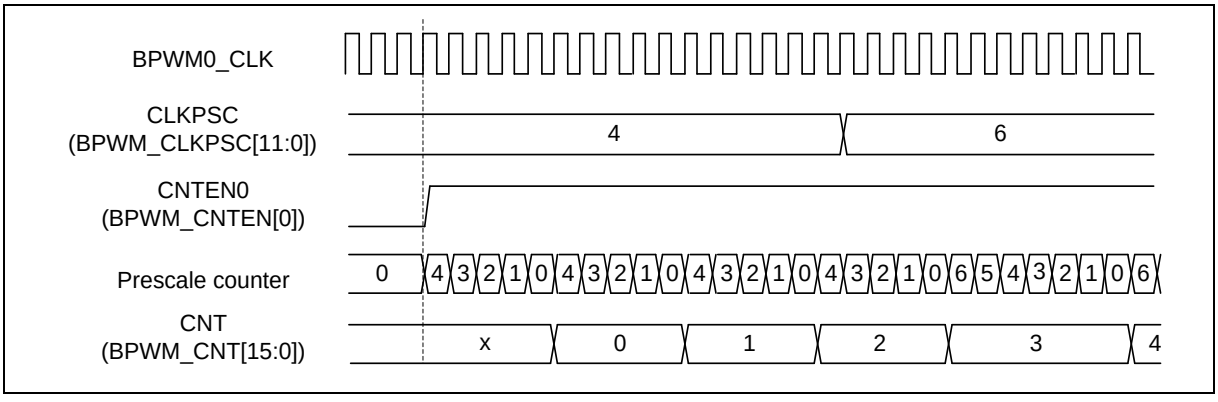


Figure 6.16-5 BPWM_CH0 CLKPSC Waveform

6.16.5.2 BPWM Counter

BPWM has one counter, and supports 3 counter types operation: Up Counter, Down Counter and Up-Down Counter types.

For BPWM channel0, CNT(BPWM_CNT[15:0]) can clear to 0x00 by CNTCLR0 (BPWM_CNTCLR[0]) when the prescale counter counts down to 0, and CNTCLR0(BPWM_CNTCLR[0]) will be set as 0 by hardware automatically.

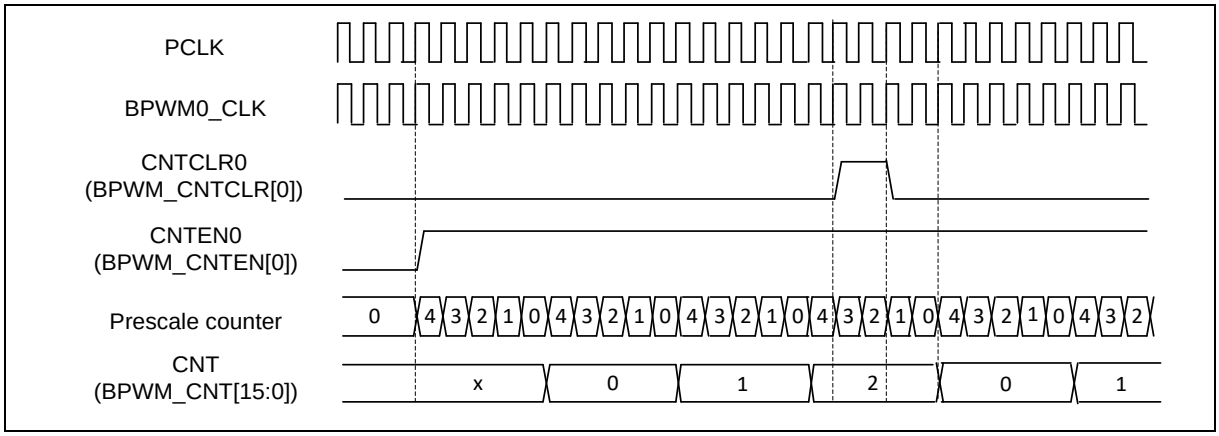


Figure 6.16-6 BPWM Counter Clear Waveform

6.16.5.3 Up Counter Type

In the up counter operation, CNTTYPE0 (BPWM_CTL1[1:0]) is 0x0, the 16 bits BPWM counter is an up counter and starts up-counting from 0 to PERIOD (BPWM_PERIOD[15:0]) to finish a BPWM period. The current counter value can be found by reading the CNT (BPWM_CNT[15:0]). BPWM generates zero point event when the counter counts to 0 and generates period point event when counting to PERIOD. An example of the period time in up counter type, the BPWM period time = (PERIOD+1) * (CLKPSC+1) * BPWMx_CLK clock time, is shown in Figure 6.16-7.

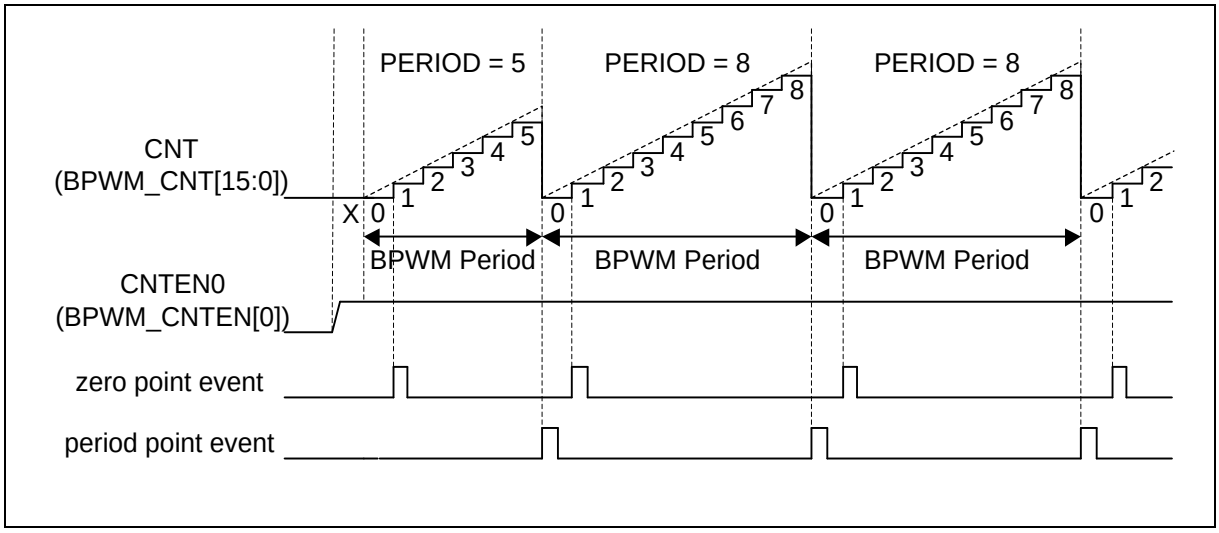


Figure 6.16-7 BPWM Up Counter Type

6.16.5.4 Down Counter Type

In the down counter operation, CNTTYPE0 (BPWM_CTL1[1:0]) is 0x1, the 16 bits BPWM counter is a down counter and starts down-counting from PERIOD to 0 to finish a BPWM period. The current counter value can be found by reading the CNT. BPWM generates zero point event when the counter counts to 0 and generates period point event when counting to PERIOD. An example of the period time in down counter type, the BPWM period time = (PERIOD+1) * (CLKPSC+1) * BPWMx_CLK clock time, is shown in Figure 6.16-8.

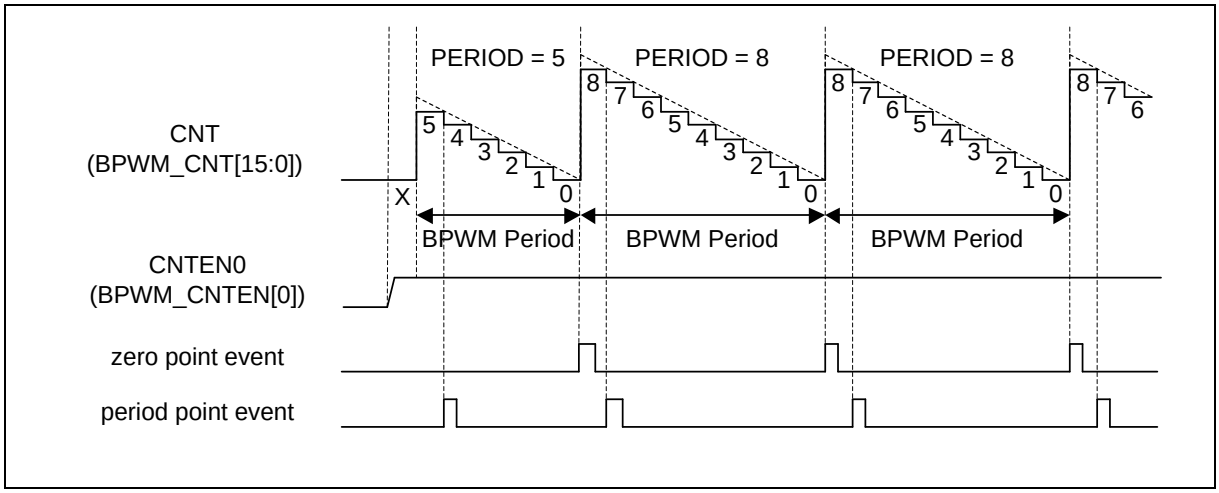
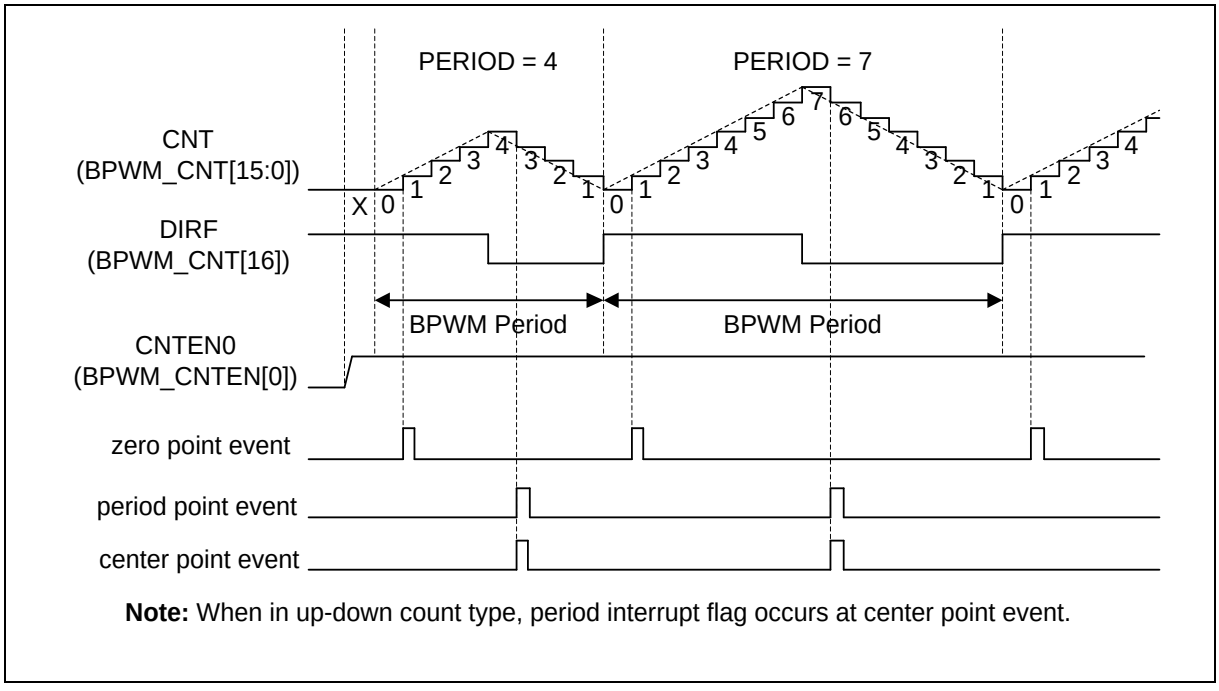


Figure 6.16-8 BPWM Down Counter Type

6.16.5.5 Up-Down Counter Type

In the up-down counter operation, CNTTYPE0 (BPWM_CTL1[1:0]) is 0x2, the 16 bits BPWM counter is an up-down counter and starts counting-up from 0 to PERIOD and then starts counting down to 0 to finish a BPWM period. The current counter value can be found by reading the CNT. BPWM generates zero point event when counter counts to 0 and generates center point event when counting to PERIOD. An example of the period time in up-down counter type, the BPWM period time = (2xPERIOD) * (CLKPSC+1) * BPWMx_CLK clock time, is shown in Figure 6.16-9. The DIRF (BPWM_CNT[16]) is a counter direction indicator flag, where high is up counting, and low is down counting.



Note: When in up-down count type, period interrupt flag occurs at center point event.

Figure 6.16-9 BPWM Up-Down Counter Type

6.16.5.6 BPWM Comparator

The CMPDAT (BPWM_CMPDATn[15:0]) is a basic comparator register of BPWM channel n; each channel only has one CMPDAT. The CMPDAT's value is continuously compared to the counter value. When the counter is equal to the compared register, BPWM generates an event and uses the event to generate BPWM pulse, interrupt or use to trigger EADC. In up-down counter type, two events will be generated in a BPWM period as shown in Figure 6.16-10.

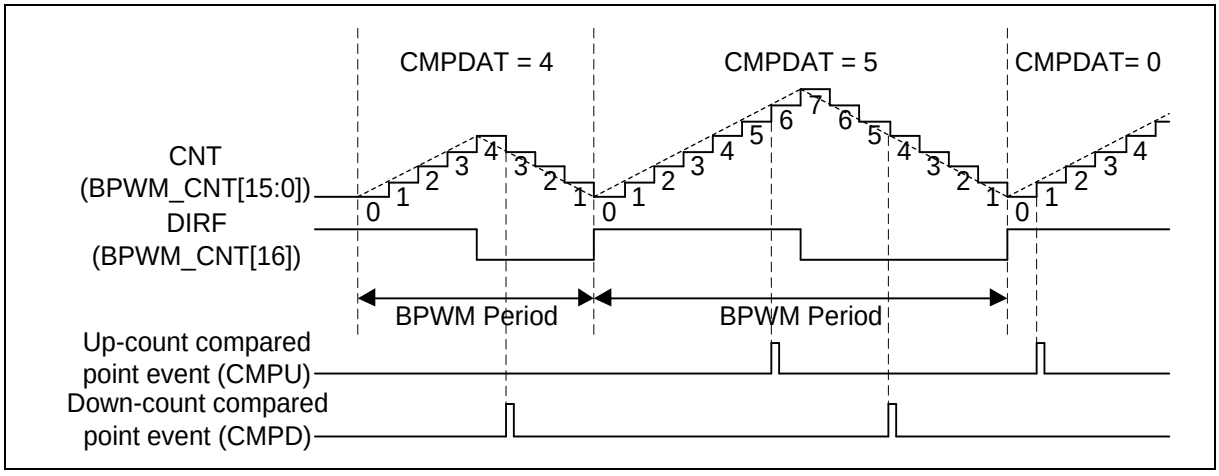


Figure 6.16-10 BPWM CMPDAT Events in Up-Down Counter Type

6.16.5.7 Period Loading Mode

When immediately loading mode and center loading mode are disabled after IMMLDENn bits and CTRLDn bits of BPWM_CTL0 register are set to 0, BPWM enters period Loading mode. In period Loading mode, PERIOD (BPWM_PERIOD[15:0]) and CMPDAT (BPWM_CMPDATn[15:0]) will all load to their active PBUF(BPWM_PBUF[15:0]) and CMPBUF(BPWM_CMPBUFn[15:0]) buffers while each period is completed. For example, after the BPWM counter counts up from 0 to PERIOD in up-counter operation or counts down from PERIOD to 0 in the down-counter operation or counts up from 0 to PERIOD and then counts down to 0 in up-down counter operation.

Figure 6.16-11 shows period loading timing of up-count operation, where PERIOD DATA0 denotes the initial data of PERIOD, PERIOD DATA1 denotes the first updated PERIOD data by software and so on, CMPDAT also follows this rule. The following describes steps sequence of Figure 6.16-11. User can know the PERIOD and CMPDAT update condition, by watching BPWM period and CMPU event.

1. Software writes CMPDAT DATA1 to CMPDAT at point 1.
2. Hardware loads CMPDAT DATA1 to CMPBUF at the end of PWM period at point 2.
3. Software writes PERIOD DATA1 to PERIOD at point 3.
4. Hardware loads PERIOD DATA1 to PBUF at the end of PWM period at point 4.
5. Software writes PERIOD DATA2 to PERIOD at point 5.
6. Hardware loads PERIOD DATA2 to PBUF at the end of PWM period at point 6.

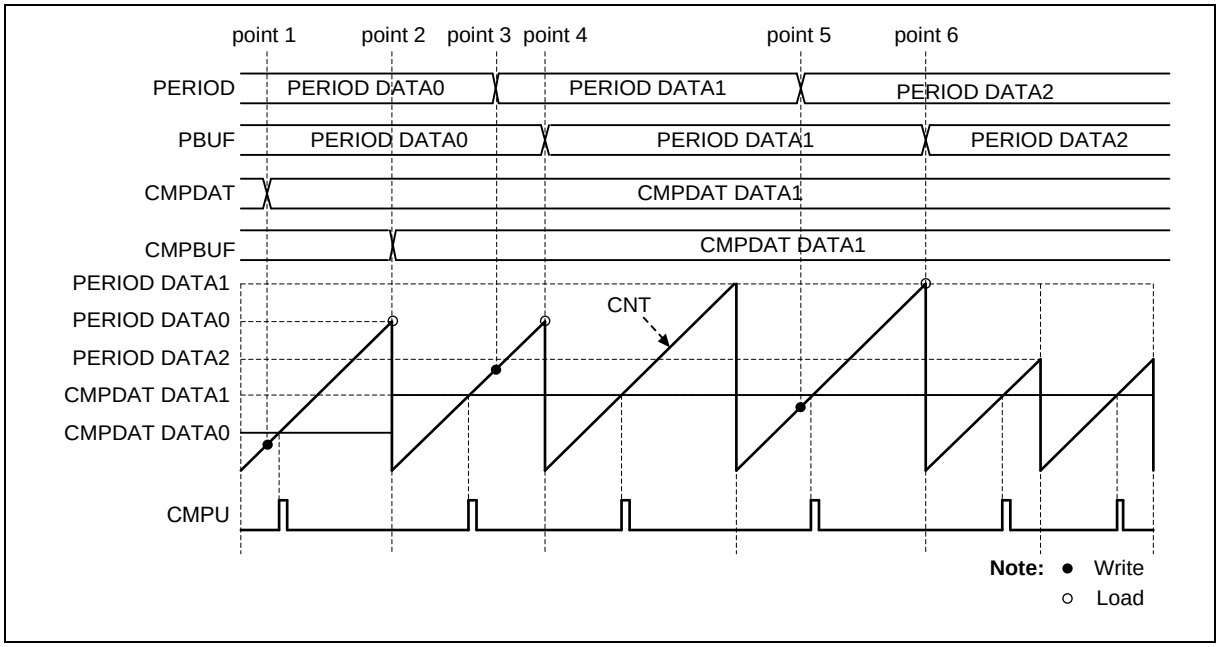


Figure 6.16-11 Period Loading Mode with Up-Counter Type

6.16.5.8 Immediately Loading Mode

If the IMMLDENn (BPWM_CTL0[21:16]) bit is set to 1, the BPWM enters immediately loading mode. In immediately loading mode, when PERIOD(BPWM_PERIOD[15:0]) or CMPDAT(BPWM_CMPDATn[15:0]) is updated, PERIOD or CMPDAT will be loaded to active PBUF (BPWM_PBUF[15:0]) or CMPBUF (BPWM_CMPBUFn[15:0]) after current counting is completed. If the update PERIOD value is less than the current counter value, counter will count to 0xFFFF, when the counter counts to 0xFFFF and prescale count to 0, the flag CNTMAX0(BPWM_STATUS[0]) will raise, and then counter will count wraparound. Immediately loading mode has the highest priority. If IMMLDENn has been set, other loading mode for channel n will become invalid. Figure 6.16-12 shows an example and its steps sequence is described below.

1. Software writes CMPDAT DATA1 and hardware immediately loads CMPDAT DATA1 to CMPBUF at point 1.
2. Software writes PERIOD DATA1 which is greater than the current counter value at point 2; counter will continue counting until it is equal to PERIOD DATA1 to finish a period loading.
3. Software writes PERIOD DATA2 which is less than the current counter value at point 3; counter will continue counting to its maximum value 0xFFFF and count wraparound from 0 to PERIOD DATA2 to finish this period loading.

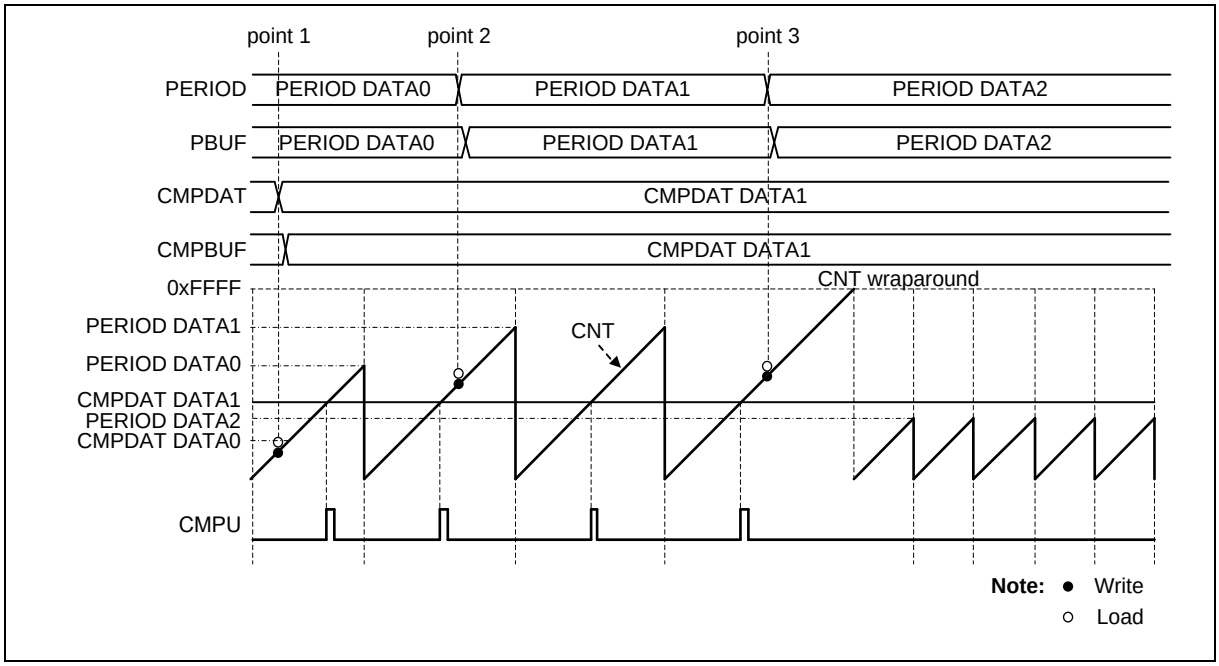


Figure 6.16-12 Immediately Loading Mode with Up-Counter Type

6.16.5.9 Center Loading Mode

When the CTRLDn (BPWM_CTL0[5:0]) bit is set to 1 and BPWM counter is set to up-down count type, CNTTYPE0 (BPWM_CTL1[1:0]) is 0x2, BPWM enters center loading mode. In center loading mode, CMPDAT(BPWM_CMPDATn[15:0]) will be loaded to active CMPBUF(BPWM_CMPBUFn[15:0]) buffer in center of each period, that is, the counter counts to PERIOD. PERIOD(BPWM_PERIOD[15:0]) will be loaded to their active PBUF registers while each period is completed. Figure 6.16-13 shows an example and its steps sequence is described below.

1. Software writes CMPDAT DATA1 at point 1.
2. Hardware loads CMPDAT DATA1 to CMPBUF at center of PWM period at point 2.
3. Software writes PERIOD DATA1 at point 3.
4. Hardware loads PERIOD DATA1 to PBUF at the end of PWM period at point 4.
5. Software writes CMPDAT DATA2 at point 5.
6. Hardware loads CMPDAT DATA2 to CMPBUF at center of PWM period at point 6.
7. Software writes PERIOD DATA2 at point 7.
8. Hardware loads PERIOD DATA2 to PBUF at the end of PWM period at point 8.

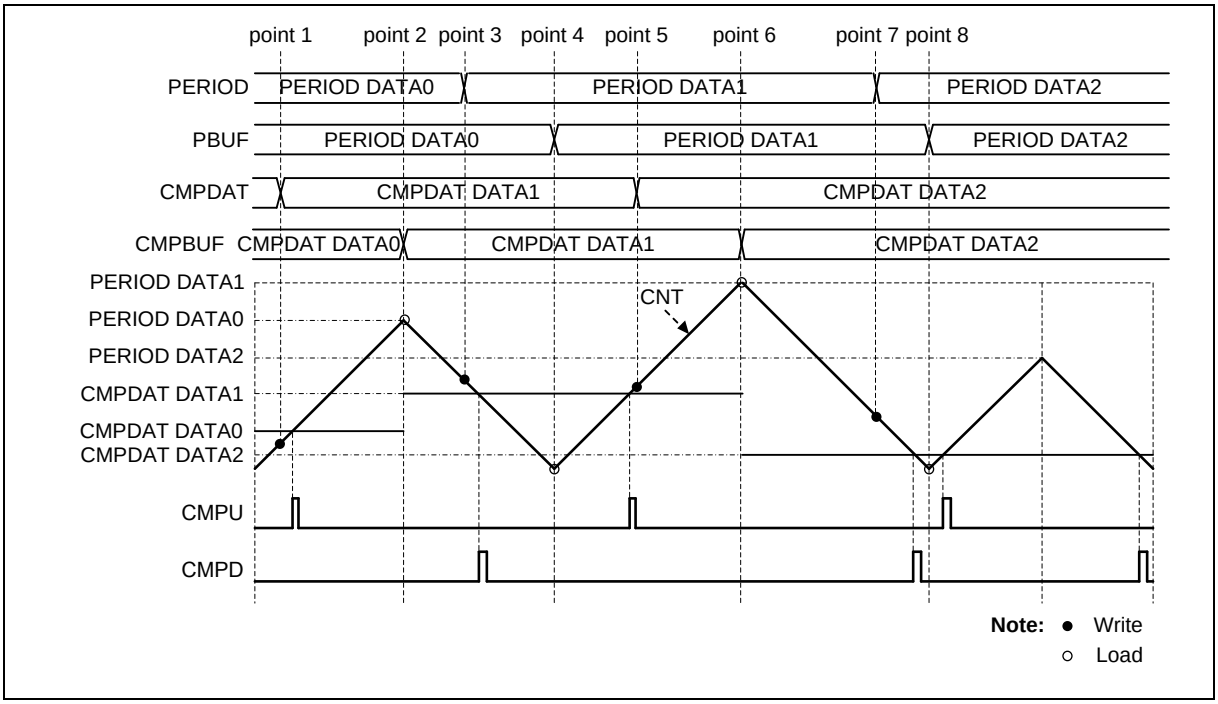


Figure 6.16-13 Center Loading Mode with Up-Down-Counter Type

6.16.5.10 BPWM Pulse Generator

The BPWM pulse generator uses counter and comparator events to generate BPWM pulse. The events are: zero point, period point in up counter type and down counter type, center point in up-down counter type and counter equal to comparator point in three types. As to up-down counter type, there are two counter equal comparator points, one at up count another at down count.

Each event point can decide BPWM waveform to do nothing (X), set Low (L), set High (H) or toggle (T) by setting BPWM_WGCTL0 and BPWM_WGCTL1 registers. Using these points can easily generate asymmetric BPWM pulse or variant waveform as shown in Figure 6.16-14. In the figure, there is a comparator n to generate BPWM pulse, where n denotes channel number 0 to 5. CMPU denotes CNT is equal to CMPDAT when counting up, and CMPD denotes CNT is equal to CMPDAT when counting down.

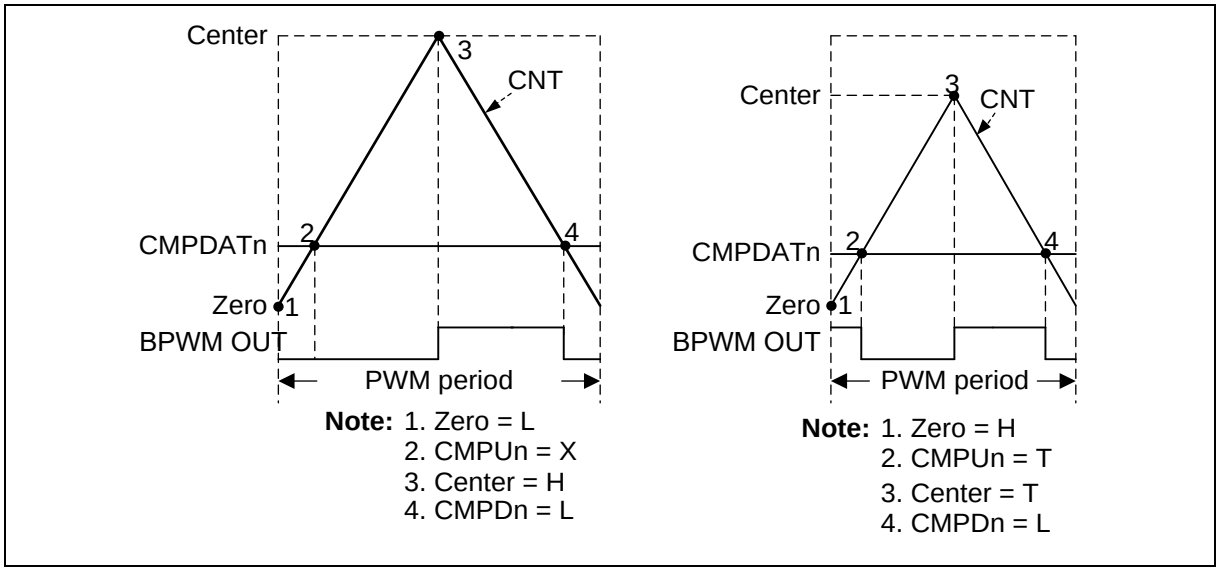


Figure 6.16-14 BPWM Pulse Generation (Left: Asymmetric Pulse, Right: Variety Pulse)

The generation events may be sometimes set to the same value, as the reason, events priority between different counter types are list below, up counter type (Table 6.16-1) down counter type (Table 6.16-2) and up-down counter type (Table 6.16-3). By using event priority, user can easily generate 0% to 100% duty pulse as shown in Figure 6.16-15.

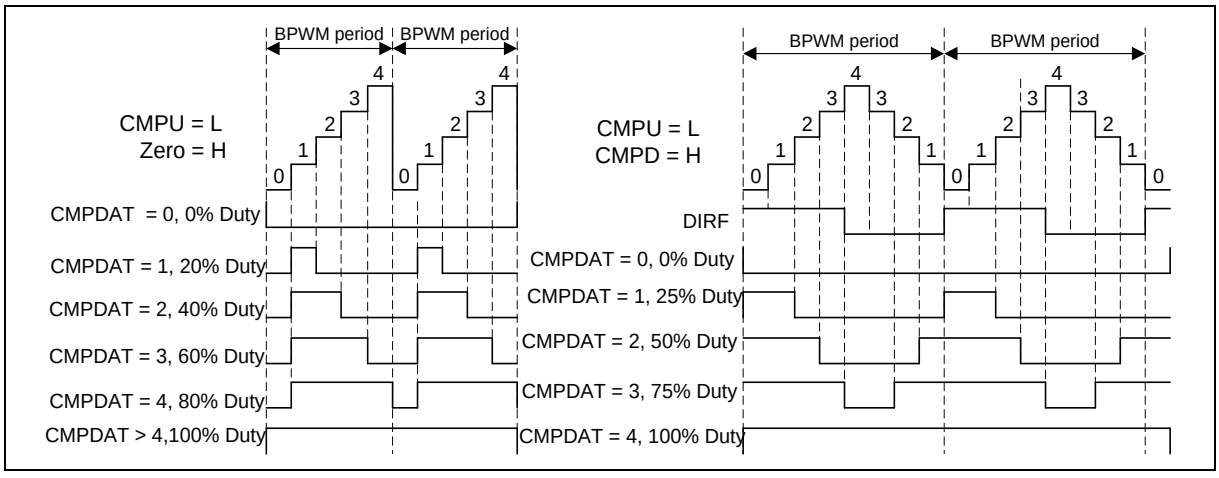


Figure 6.16-15 BPWM 0% to 100% Pulse Generation (Left: Up Counter Type, Right: Up-down Counter Type)

Priority	Up Event
1 (Highest)	Period event (CNT = PERIOD)
2	Compare up event(CNT = CMPUn)
3 (Lowest)	Zero event (CNT = 0)

Table 6.16-1 BPWM Pulse Generation Event Priority for Up-Counter

Priority	Down Event
1 (Highest)	Zero event (CNT = 0)
2	Compare down event (CNT = CMPDn)
3 (Lowest)	Period event (CNT = PERIOD)

Table 6.16-2 BPWM Pulse Generation Event Priority for Down-Counter

Priority	Up Event	Down Event
1 (Highest)	Compare up event (CNT = CMPUn)	Compare down event (CNT = CMPDn)
2 (Lowest)	Zero event (CNT = 0)	Period (center) event (CNT =PERIOD)

Table 6.16-3 BPWM Pulse Generation Event Priority for Up-Down-Counter

6.16.5.11 Synchronous function

To start BPWM and PWM counters in the same time, user has to set the BPWM Synchronous Start Control Register (BPWM_SSCTL[0]) to enable the channel counters which are planned to start counting together, and select the SSRC(BPWM_SSCTL[9:8]) to choose the Synchronous Start source, followed by setting the BPWM Synchronous Start Trigger Register CNTSEN (BPWM_SSTRG[0]).

6.16.5.12 BPWM Output Control

After BPWM pulse generation, there are three steps to control the output of BPWM channels. There are Mask, Pin Polarity and Output Enable three steps as shown in Figure 6.16-16.

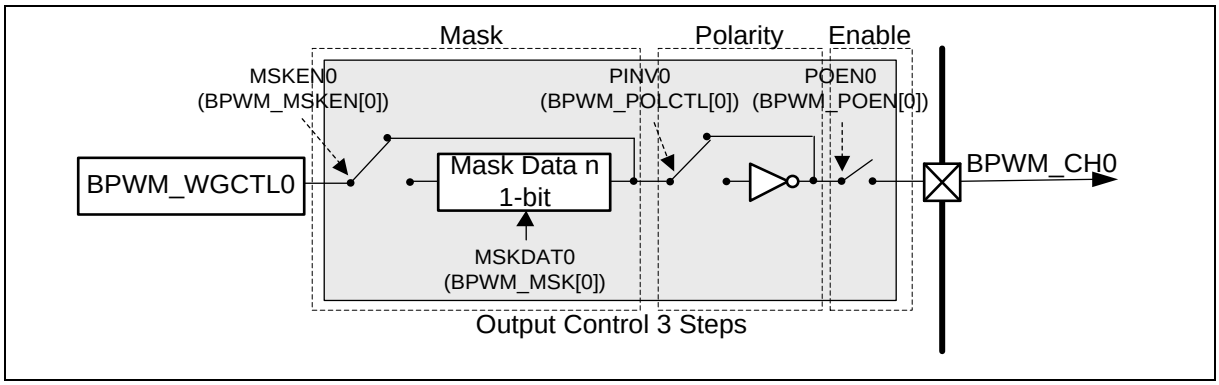


Figure 6.16-16 BPWM_CH0 Output Control 3 Steps

6.16.5.13 BPWM Mask Output Function

Each of the BPWM output channels can be manually overridden by using the appropriate bits in the BPWM Mask Enable Control Register (BPWM_MSKEN) and BPWM Masked Data Register (BPWM_MSK) to drive the BPWM channel outputs to specified logic states independent of the duty cycle comparison units. The BPWM mask bits are useful when controlling various types of Electrically Commutated Motor (ECM) like a BLDC motor. The BPWM_MSKEN register contains six bits, MSKENn(BPWM_MSKEN[5:0]) determine which BPWM channel output will be overridden, MSKENn(BPWM_MSKEN[5:0]) bits are active-high. The BPWM_MSK register contains six bits, MSKDATn(BPWM_MSK[5:0]), which determine the state of the BPWM channel output when the channel is masked via the MSKDAT bits. Figure 6.16-17 shows an example of how BPWM mask control can be used for the override feature.

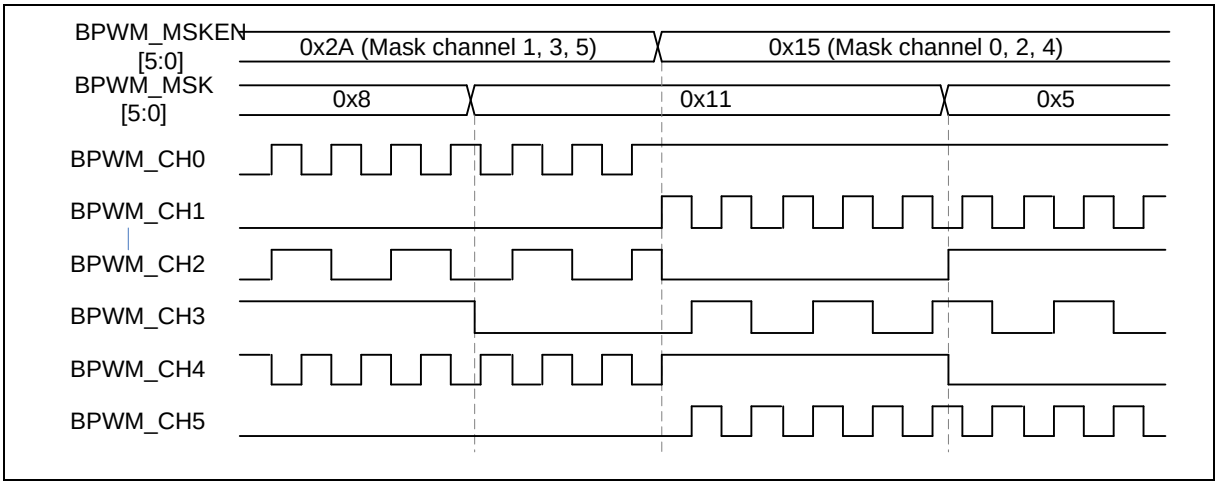
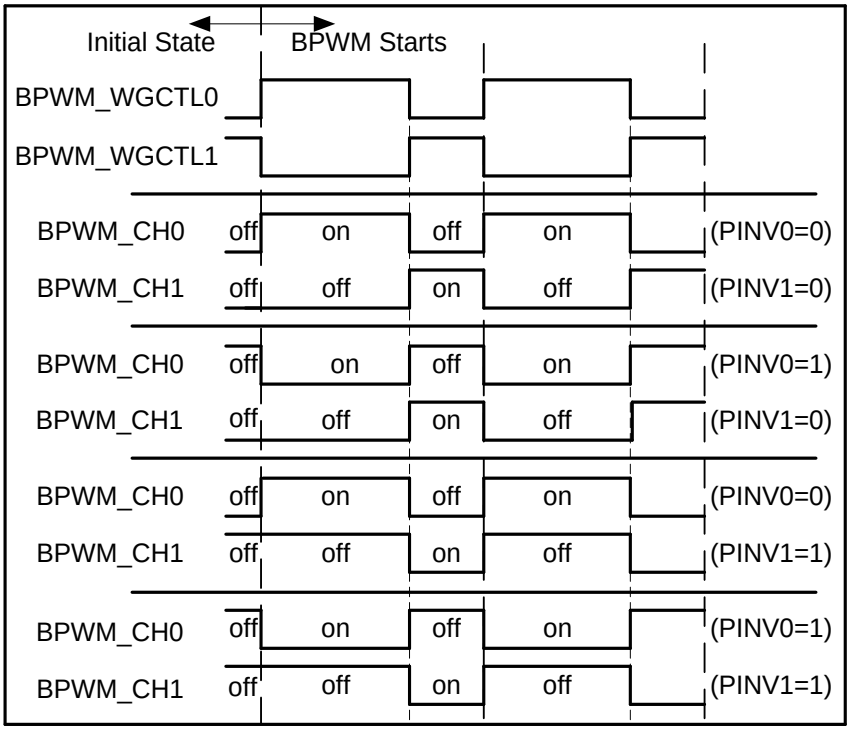


Figure 6.16-17 Mask Control Waveform Illustration

6.16.5.14 Polarity Control

Each BPWM port from BPWM_CH0 to BPWM_CH5 has an independent polarity control module to configure the polarity of the active state of BPWM output. By default, the BPWM output is active high. This implies the BPWM OFF state is low and ON state is high. This definition is variable through setting BPWM Negative Polarity Control Register (BPWM_POLCTL), for each individual BPWM channel. Figure 6.16-18 shows the initial state before BPWM starts with different polarity settings.



Note: PINVx: Negative Polarity control bits; It controls the PWM output initial state and polarity, x denotes 0 or 1.

Figure 6.16-18 Initial State and Polarity Control

6.16.5.15 BPWM Interrupt Generator

There are two independent interrupts for each BPWM as shown in Figure 6.16-19.

BPWM interrupt (BPWM_INT) comes from BPWM complementary pair events. The counter can generate the Zero point Interrupt Flag ZIF0 (BPWM_INTSTS[0]) and the Period point Interrupt Flag PIF0 (BPWM_INTSTS[8]). When BPWM channel n's counter equals to the comparator value stored in BPWM_CMPDATn, the different interrupt flags will be triggered depending on the counting direction. If the matching occurs at up-count direction, the Up Interrupt Flag CMPUIFn (BPWM_INTSTS[21:16]) is set and if matching at the opposite direction, the Down Interrupt Flag CMPDIFn (BPWM_INTSTS[29:24]) is set. If the correspond interrupt enable bits are set, the trigger events will generates interrupt signals.

Another interrupt is the capture interrupt (CAP_INT). It shares the BPWM_INT vector in NVIC, CAP_INT can be generated when the CAPRIFn (BPWM_CAPIF[5:0]) is triggered and the Capture Rising Interrupt Enable bit CAPRIENn (BPWM_CAPIEN[5:0]) is set to 1. Or in the falling edge condition, the CAPFIFn (BPWM_CAPIF[13:8]) can be triggered when the Capture Falling Interrupt Enable bit CAPFIENn (BPWM_CAPIEN[13:8]) is set to 1.

Figure 6.16-19 demonstrates the architecture of the BPWM interrupts.

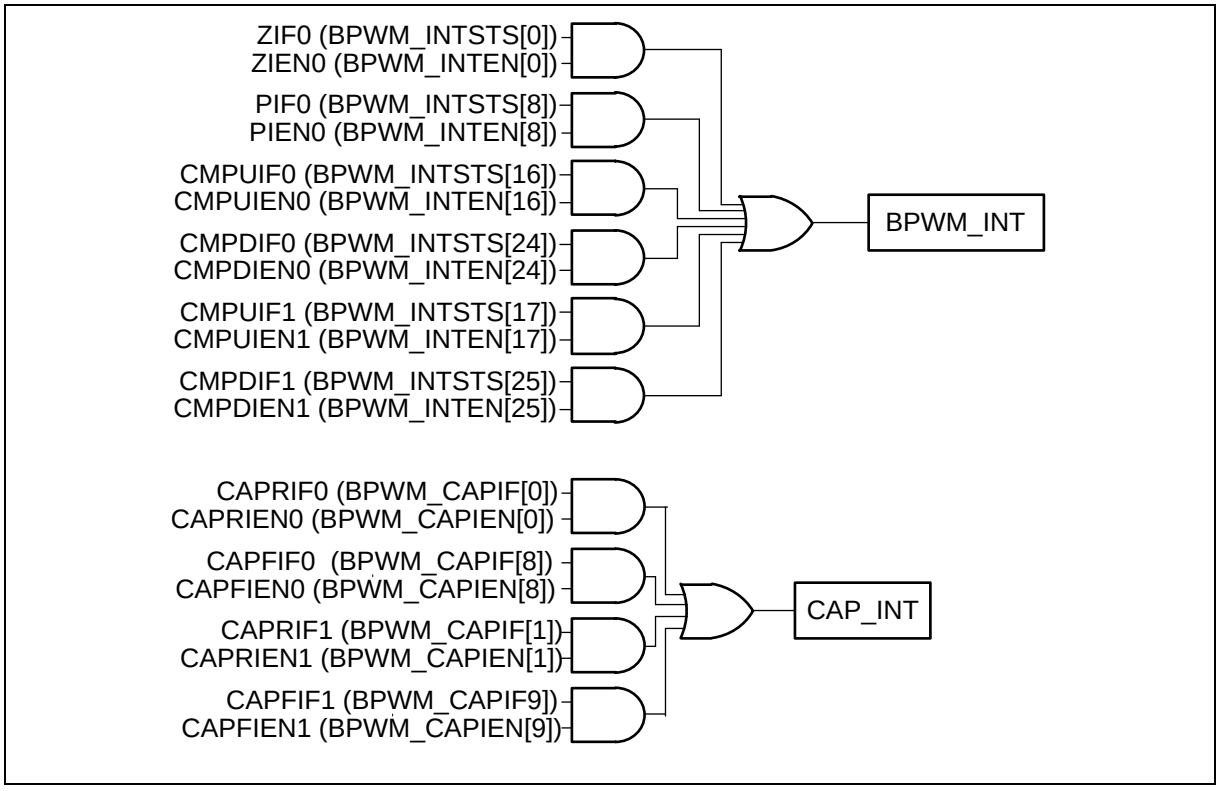


Figure 6.16-19 BPWM_CH0 and BPWM_CH1 Pair Interrupt Architecture Diagram

6.16.5.16 BPWM Trigger EADC Generator

BPWM can be one of the EADC conversion trigger source. Each BPWM pair channels share the same trigger source. Setting TRGSELn is to select the trigger sources, where TRGSELn is TRGSEL0, TRGSEL1, …, and TRGSEL5, which are located in BPWM_EADCTS0[3:0], BPWM_EADCTS0[11:8], BPWM_EADCTS0[19:16], BPWM_EADCTS0[27:24], BPWM_EADCTS1[3:0] and BPWM_EADTS1[11:8], respectively. Setting TRGENn is to enable the trigger output to EADC, where TRGENn is TRGEN0, TRGEN1, …, TRGEN5, which are located in BPWM_EADCTS0[7], BPWM_EADCTS0[15], BPWM_EADCTS0[23], BPWM_EADCTS0[31], BPWM_EADCTS1[7] and

BPWM_EADCTS1[15], respectively. The number n (n = 0,1, ..,5) denotes BPWM channel number.

There are 7 BPWM events can be selected as the trigger source for one pair of channels. Figure 6.16-20 is an example of BPWM_CH0 and BPWM_CH1. BPWM can trigger EADC to start conversion in different timings by setting PERIOD and CMPDAT. Figure 6.16-22 is the trigger EADC timing waveform in the up-down counter type.

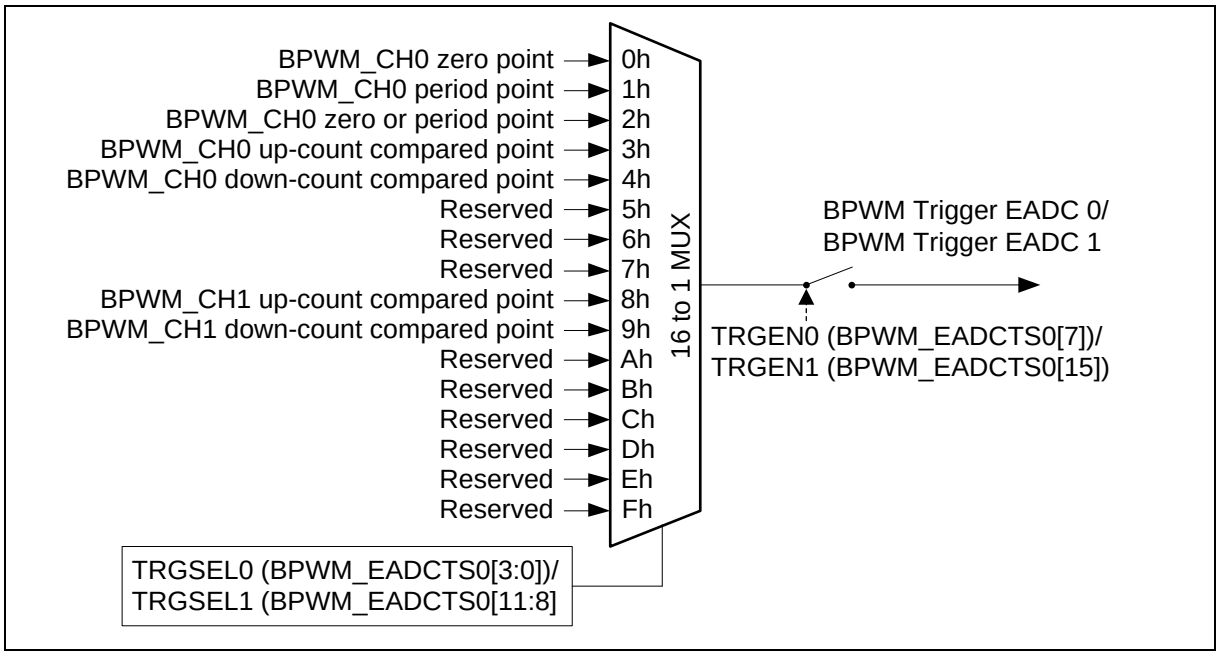


Figure 6.16-20 BPWM_CH0 and BPWM_CH1 Pair Trigger EADC Source Block Diagram

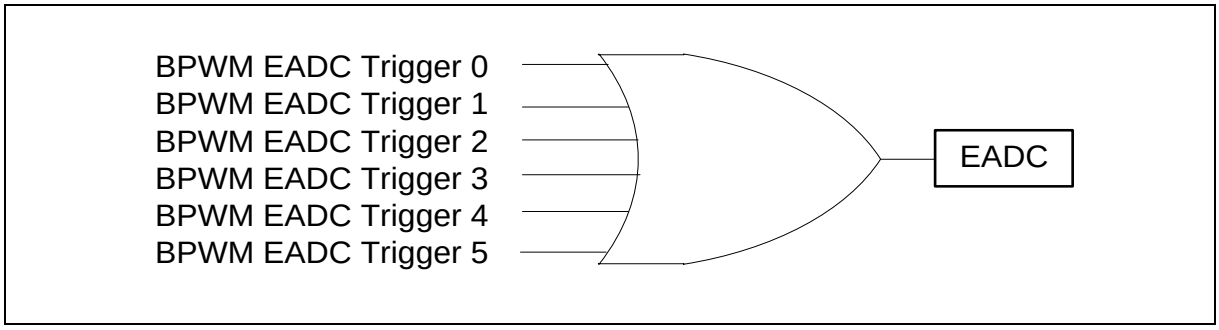


Figure 6.16-21 BPWM CH0~ CH5 Trigger EADC Block Diagram

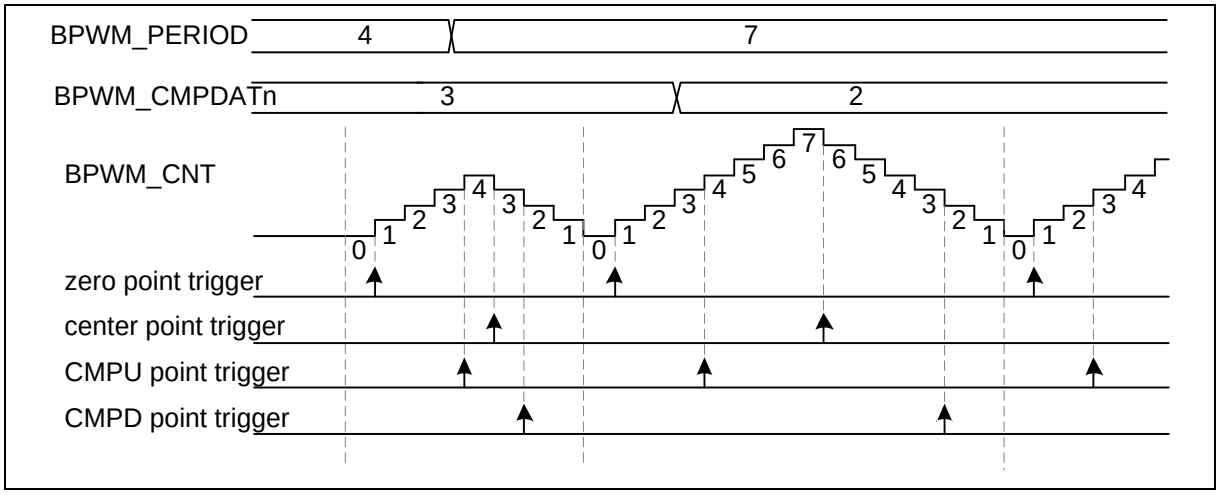


Figure 6.16-22 BPWM Trigger EADC in Up-Down Counter Type Timing Waveform

6.16.5.17 Capture Operation

The channels of the capture input and the BPWM output share the same pin and counter. The counter can operate in up or down counter type. The capture function will always latch the BPWM counter to the register RCAPDATn (BPWM_RCAPDATn[15:0]) or the register FCAPDATn (BPWM_FCAPDATn[15:0]) if the input channel has a rising transition or a falling transition, respectively. The capture function will also generate an interrupt CAP_INT (using BPWM_INT vector) if the rising or falling latch occurs and the corresponding channel n's rising or falling interrupt enable bits are set, where the CAPRIENn (BPWM_CAPIEN[5:0]) is for the rising edge and the CAPFIENn (BPWM_CAPIEN[13:8]) is for the falling edge. When rising or falling latch occurs, the corresponding BPWM counter may be reloaded with the value of BPWM_PERIOD, depending on the setting of RCRLDENn or FCRLDENn bits (where RCRLDENn and FCRLDENn are located at BPWM_CAPCTL[21:16] and BPWM_CAPCTL[29:24], respectively). Note that the corresponding GPIO pins must be configured as the capture function by enable the CAPINENn (BPWM_CAPINEN[5:0]) for the corresponding capture channel n. Figure 6.16-23 is the capture block diagram of channel 0.

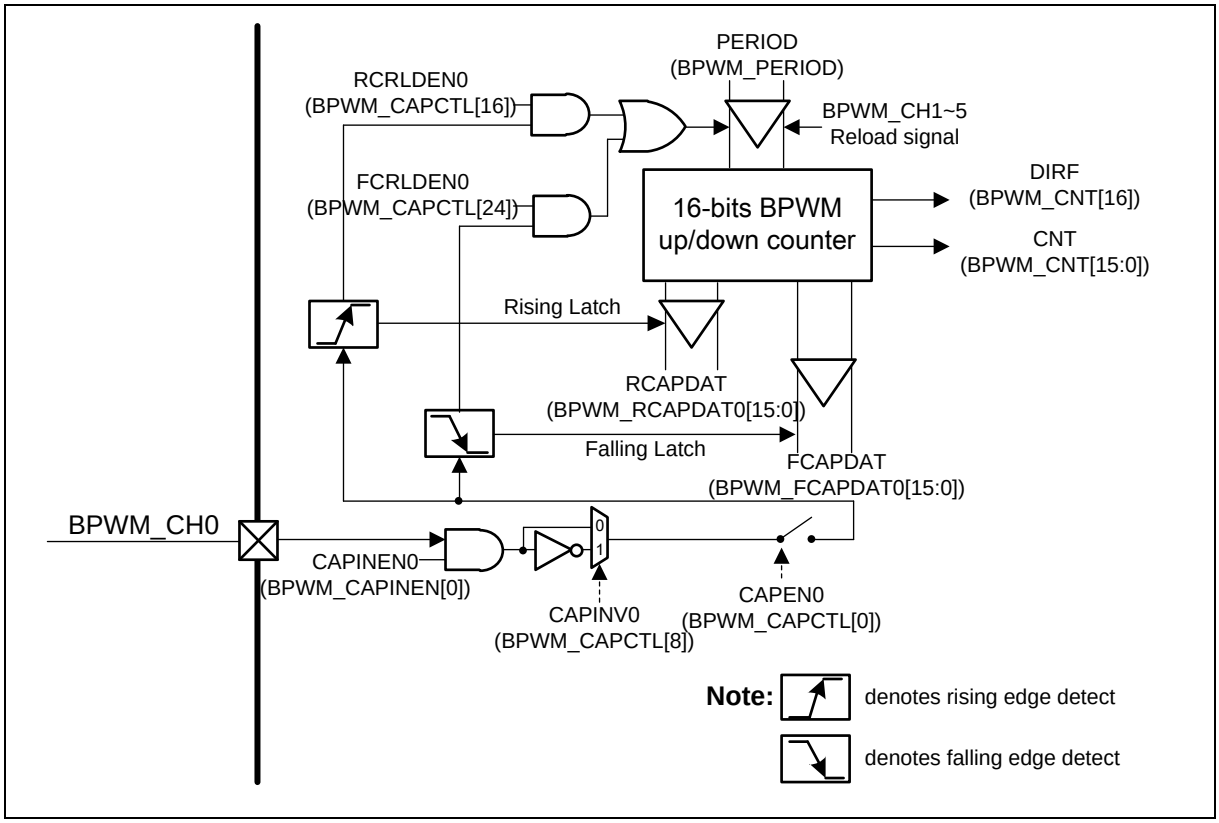


Figure 6.16-23 BPWM_CH0 Capture Block Diagram

Figure 6.16-24 illustrates the capture function timing. In this case, the capture counter is set as BPWM down counter type and the PERIOD is set to 8 so that the counter counts in the down direction, from 8 to 0. When detecting a falling edge at the capture input pin, the capture function latches counter value to the BPWM_FCAPDATn. When detecting the rising edge, it latches the counter value to the BPWM_RCAPDATn. In this timing diagram, when the falling edge is detected at the first time, the capture function will reload the counter value from the PERIOD setting because the FCRLDENn is enabled. But at the second time, the falling edge does not result in a reload because of the disabled FCRLDENn. In this example, the counter also reloads at the rising edge of the capture input because the RCRLDENn is enabled, too.

Moreover, if the case is setup as the up counter type, the counter will reload the value zero and count up to the value PERIOD. It is important that the counter is shared by all channels, so the counter reloads time also controlled by all channels' reload signals.

Figure 6.16-24 also illustrates the timing example for the interrupt and interrupt flag generation. When the rising edge at channel n is detected, the corresponding bit CAPRIFn (BPWM_CAPIF[5:0]) is set by hardware. Similarly, a falling edge detection at channel n causes the corresponding bit CAPFIFn (BPWM_CAPIF[13:8]) set by hardware. CAPRIFn (BPWM_CAPIF[5:0]) and CAPFIFn (BPWM_CAPIF[13:8]) can be cleared by software by writing '1'. If the CAPRIFn (BPWM_CAPIF[5:0]) is set and the CAPRIENn is enabled, the capture function generates an interrupt. If the CAPFIFn (BPWM_CAPIF[13:8]) is set and the CAPFIENn is enabled, the interrupt also happens.

A condition which is not shown in this figure is: if the rising latch happens again when the CAPRIFn(BPWM_CAPIF[5:0]) is already set, the Overrun status CRIFOVn (BPWM_CAPSTS[5:0]) will be set to 1 by hardware to indicate the CAPRIFn (BPWM_CAPIF[5:0]) overrunning. Also, if the falling latch happens again, the same hardware operation occurs for the interrupt flag CAPFIFn (BPWM_CAPIF[13:8]) and the Overrun status CFIFOVn (BPWM_CAPSTS[13:8]).

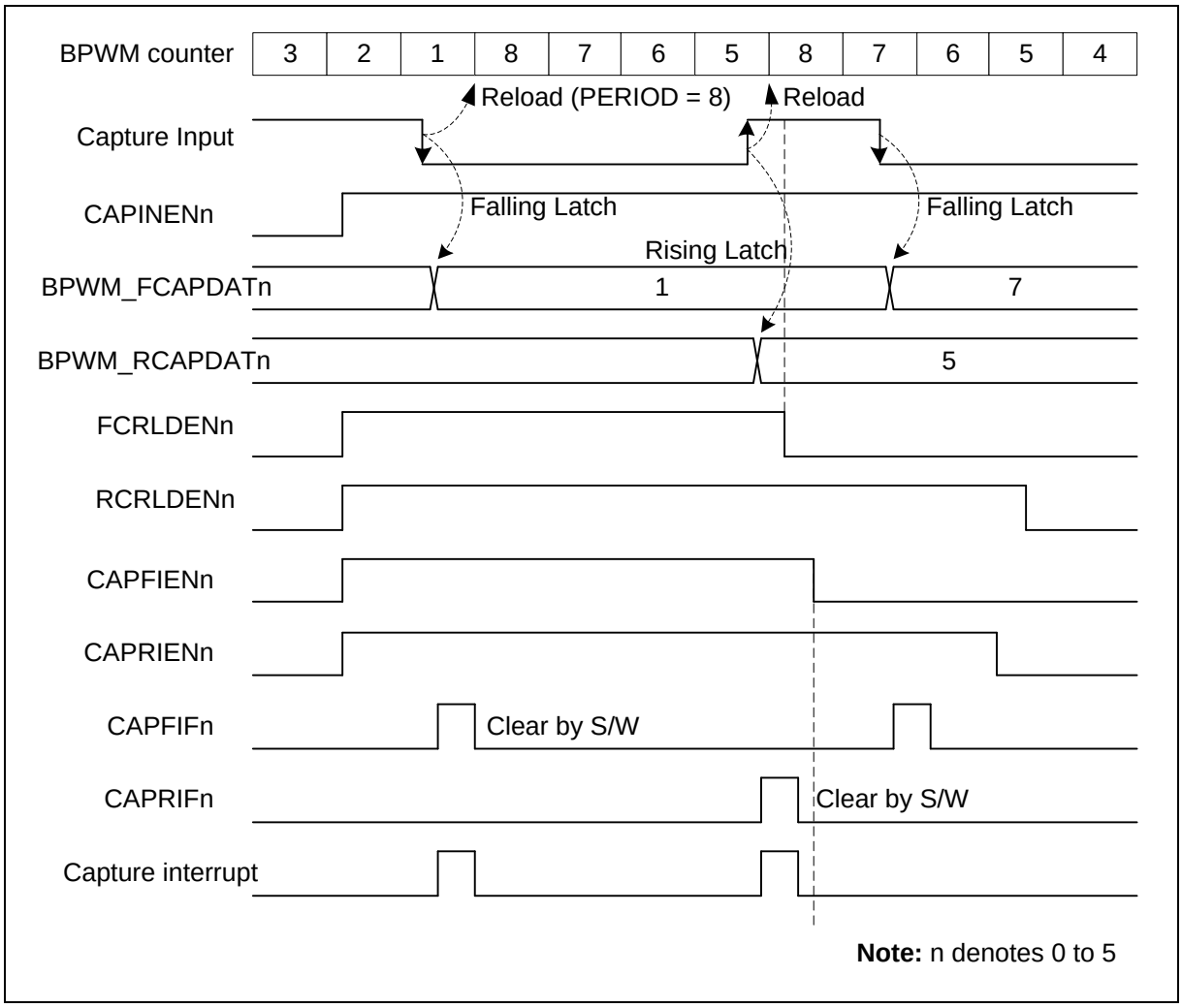


Figure 6.16-24 Capture Operation Waveform

The capture pulse width meeting the following conditions can be calculated according to the formula.

4. The capture positive or negative pulse width is shorter than a counter period.
5. The counter operates in down counter type.
6. The counter can be reloaded by both falling and rising capture events through setting FCRLDENn and RCRLDENn bits of BPWM_CAPCTL register to 1.

For the negative pulse case, the channel low pulse width is calculated as (BPWM_PERIOD + 1 - BPWM_RCAPDATn) BPWM counter time, where one BPWM counter time is (CLKPSC+1) * BPWMx_CLK clock time. In the case shown in Figure 6.16-24, low pulse width is 8+1-5 = 4 BPWM counter time.

For the positive pulse case, the channel high pulse width is calculated as (BPWM_PERIOD + 1 - BPWM_FCAPDATn) BPWM counter time, where one BPWM counter time is (CLKPSC+1) * BPWMx_CLK clock time. In the case shown in Figure 6.16-24, high pulse width is 8+1-7 = 2 BPWM counter time.

6.16.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
BPWM Base Address: **BPWM0_BA = 0x4005_A000** **BPWM1_BA = 0x4005_B000** **BPWM non-secure base address is BPWM0_BA + 0x1000_0000** **BPWM non-secure base address is BPWM1_BA + 0x1000_0000**				
BPWM_CTL0 **x=0, 1**	BPWMx_BA+0x00	R/W	BPWM Control Register 0	0x0000_0000
BPWM_CTL1 **x=0, 1**	BPWMx_BA+0x04	R/W	BPWM Control Register 1	0x0000_0000
BPWM_CLKSRC **x=0, 1**	BPWMx_BA+0x10	R/W	BPWM Clock Source Register	0x0000_0000
BPWM_CLKPSC **x=0, 1**	BPWMx_BA+0x14	R/W	BPWM Clock Prescale Register	0x0000_0000
BPWM_CNTEN **x=0, 1**	BPWMx_BA+0x20	R/W	BPWM Counter Enable Register	0x0000_0000
BPWM_CNTCLR **x=0, 1**	BPWMx_BA+0x24	R/W	BPWM Clear Counter Register	0x0000_0000
BPWM_PERIOD **x=0, 1**	BPWMx_BA+0x30	R/W	BPWM Period Register	0x0000_0000
BPWM_CMPDAT0 **x=0, 1**	BPWMx_BA+0x50	R/W	BPWM Comparator Register 0	0x0000_0000
BPWM_CMPDAT1 **x=0, 1**	BPWMx_BA+0x54	R/W	BPWM Comparator Register 1	0x0000_0000
BPWM_CMPDAT2 **x=0, 1**	BPWMx_BA+0x58	R/W	BPWM Comparator Register 2	0x0000_0000
BPWM_CMPDAT3 **x=0, 1**	BPWMx_BA+0x5C	R/W	BPWM Comparator Register 3	0x0000_0000
BPWM_CMPDAT4 **x=0, 1**	BPWMx_BA+0x60	R/W	BPWM Comparator Register 4	0x0000_0000
BPWM_CMPDAT5 **x=0, 1**	BPWMx_BA+0x64	R/W	BPWM Comparator Register 5	0x0000_0000
BPWM_CNT **x=0, 1**	BPWMx_BA+0x90	R	BPWM Counter Register	0x0000_0000
BPWM_WGCTL0 **x=0, 1**	BPWMx_BA+0xB0	R/W	BPWM Generation Register 0	0x0000_0000
BPWM_WGCTL1 **x=0, 1**	BPWMx_BA+0xB4	R/W	BPWM Generation Register 1	0x0000_0000

BPWM_MSKEN x=0, 1	BPWMx_BA+0xB8	R/W	BPWM Mask Enable Register	0x0000_0000
BPWM_MSK x=0, 1	BPWMx_BA+0xBC	R/W	BPWM Mask Data Register	0x0000_0000
BPWM_POLCTL x=0, 1	BPWMx_BA+0xD4	R/W	BPWM Pin Polar Inverse Register	0x0000_0000
BPWM_POEN x=0, 1	BPWMx_BA+0xD8	R/W	BPWM Output Enable Register	0x0000_0000
BPWM_INTEN x=0, 1	BPWMx_BA+0xE0	R/W	BPWM Interrupt Enable Register	0x0000_0000
BPWM_INTSTS x=0, 1	BPWMx_BA+0xE8	R/W	BPWM Interrupt Flag Register	0x0000_0000
BPWM_EADCTS0 x=0, 1	BPWMx_BA+0xF8	R/W	BPWM Trigger EADC Source Select Register 0	0x0000_0000
BPWM_EADCTS1 x=0, 1	BPWMx_BA+0xFC	R/W	BPWM Trigger EADC Source Select Register 1	0x0000_0000
BPWM_SSCTL x=0, 1	BPWMx_BA+0x110	R/W	BPWM Synchronous Start Control Register	0x0000_0000
BPWM_SSTRG x=0, 1	BPWMx_BA+0x114	W	BPWM Synchronous Start Trigger Register	0x0000_0000
BPWM_STATUS x=0, 1	BPWMx_BA+0x120	R/W	BPWM Status Register	0x0000_0000
BPWM_CAPINEN x=0, 1	BPWMx_BA+0x200	R/W	BPWM Capture Input Enable Register	0x0000_0000
BPWM_CAPCTL x=0, 1	BPWMx_BA+0x204	R/W	BPWM Capture Control Register	0x0000_0000
BPWM_CAPSTS x=0, 1	BPWMx_BA+0x208	R	BPWM Capture Status Register	0x0000_0000
BPWM_RCAPDAT0 x=0, 1	BPWMx_BA+0x20C	R	BPWM Rising Capture Data Register 0	0x0000_0000
BPWM_FCAPDAT0 x=0, 1	BPWMx_BA+0x210	R	BPWM Falling Capture Data Register 0	0x0000_0000
BPWM_RCAPDAT1 x=0, 1	BPWMx_BA+0x214	R	BPWM Rising Capture Data Register 1	0x0000_0000
BPWM_FCAPDAT1 x=0, 1	BPWMx_BA+0x218	R	BPWM Falling Capture Data Register 1	0x0000_0000
BPWM_RCAPDAT2 x=0, 1	BPWMx_BA+0x21C	R	BPWM Rising Capture Data Register 2	0x0000_0000
BPWM_FCAPDAT2 x=0, 1	BPWMx_BA+0x220	R	BPWM Falling Capture Data Register 2	0x0000_0000

BPWM_RCAPDAT3 **x=0, 1**	BPWMx_BA+0x224	R	BPWM Rising Capture Data Register 3	0x0000_0000
BPWM_FCAPDAT3 **x=0, 1**	BPWMx_BA+0x228	R	BPWM Falling Capture Data Register 3	0x0000_0000
BPWM_RCAPDAT4 **x=0, 1**	BPWMx_BA+0x22C	R	BPWM Rising Capture Data Register 4	0x0000_0000
BPWM_FCAPDAT4 **x=0, 1**	BPWMx_BA+0x230	R	BPWM Falling Capture Data Register 4	0x0000_0000
BPWM_RCAPDAT5 **x=0, 1**	BPWMx_BA+0x234	R	BPWM Rising Capture Data Register 5	0x0000_0000
BPWM_FCAPDAT5 **x=0, 1**	BPWMx_BA+0x238	R	BPWM Falling Capture Data Register 5	0x0000_0000
BPWM_CAPIEN **x=0, 1**	BPWMx_BA+0x250	R/W	BPWM Capture Interrupt Enable Register	0x0000_0000
BPWM_CAPIF **x=0, 1**	BPWMx_BA+0x254	R/W	BPWM Capture Interrupt Flag Register	0x0000_0000
BPWM_PBUF **x=0, 1**	BPWMx_BA+0x304	R	BPWM PERIOD Buffer	0x0000_0000
BPWM_CMPBUF0 **x=0, 1**	BPWMx_BA+0x31C	R	BPWM CMPDAT 0 Buffer	0x0000_0000
BPWM_CMPBUF1 **x=0, 1**	BPWMx_BA+0x320	R	BPWM CMPDAT 1 Buffer	0x0000_0000
BPWM_CMPBUF2 **x=0, 1**	BPWMx_BA+0x324	R	BPWM CMPDAT 2 Buffer	0x0000_0000
BPWM_CMPBUF3 **x=0, 1**	BPWMx_BA+0x328	R	BPWM CMPDAT 3 Buffer	0x0000_0000
BPWM_CMPBUF4 **x=0, 1**	BPWMx_BA+0x32C	R	BPWM CMPDAT 4 Buffer	0x0000_0000
BPWM_CMPBUF5 **x=0, 1**	BPWMx_BA+0x330	R	BPWM CMPDAT 5 Buffer	0x0000_0000

6.16.7 Register Description

BPWM Control Register 0 (BPWM_CTL0)

Register	Offset	R/W	Description	Reset Value
BPWM_CTL0	BPWMx_BA+0x00	R/W	BPWM Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
DBGTRIOFF	**DBGHALT**	**Reserved**					
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved		**IMMLDEN5**	**IMMLDEN4**	**IMMLDEN3**	**IMMLDEN2**	**IMMLDEN1**	**IMMLDEN0**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CTRLD5**	**CTRLD4**	**CTRLD3**	**CTRLD2**	**CTRLD1**	**CTRLD0**

Bits	Description	
[31]	**DBGTRIOFF**	**ICE Debug Mode Acknowledge Disable (Write Protect)** 0 = ICE debug mode acknowledgement effects BPWM output. BPWM pin will be forced as tri-state while ICE debug mode acknowledged. 1 = ICE debug mode acknowledgement Disabled. BPWM pin will keep output no matter ICE debug mode acknowledged or not. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[30]	**DBGHALT**	**ICE Debug Mode Counter Halt (Write Protect)** If counter halt is enabled, BPWM all counters will keep current value until exit ICE debug mode. 0 = ICE debug mode counter halt Disabled. 1 = ICE debug mode counter halt Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[29:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**IMMLDENn**	**Immediately Load Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = PERIOD will load to PBUF at the end point of each period. CMPDAT will load to CMPBUF at the end point or center point of each period by setting CTRLD bit. 1 = PERIOD/CMPDAT will load to PBUF and CMPBUF immediately when software update PERIOD/CMPDAT. **Note:** If IMMLDENn is Enabled, CTRLDn will be invalid.
[15:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CTRLDn**	**Center Re-load Enable Bits** Each bit n controls the corresponding BPWM channel n. In up-down counter type, PERIOD will load to PBUF at the end point of each period. CMPDAT will load to CMPBUF at the center point of a period.

BPWM Control Register 1 (BPWM_CTL1)

Register	Offset	R/W	Description	Reset Value
BPWM_CTL1	BPWMx_BA+0x04	R/W	BPWM Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**CNTTYPE0**	

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1:0]	**CNTTYPE0**	**BPWM Counter Behavior Type 0** 00 = Up counter type (supports in capture mode). 01 = Down count type (supports in capture mode). 10 = Up-down counter type. 11 = Reserved.

BPWM Clock Source Register (BPWM_CLKSRC)

Register	Offset	R/W	Description	Reset Value
BPWM_CLKSRC	BPWMx_BA+0x10	R/W	BPWM Clock Source Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**ECLKSRC0**		

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2:0]	**ECLKSRC0**	**BPWM External Clock Source Select** 000 = BPWMx_CLK, x denotes 0 or 1. 001 = TIMER0 overflow. 010 = TIMER1 overflow. 011 = TIMER2 overflow. 100 = TIMER3 overflow. Others = Reserved.

BPWM Clock Prescale Register (BPWM_CLKPSC)

Register	Offset	R/W	Description	Reset Value
BPWM_CLKPSC	BPWMx_BA+0x14	R/W	BPWM Clock Prescale Register	0x0000_0000

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

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**CLKPSC**	**BPWM Counter Clock Prescale** The clock of BPWM counter is decided by clock prescaler. The clock of BPWM counter is divided by (CLKPSC+ 1).

BPWM Counter Enable Register (BPWM_CNTEN)

Register	Offset	R/W	Description	Reset Value
BPWM_CNTEN	BPWMx_BA+0x20	R/W	BPWM Counter Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**CNTEN0**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**CNTEN0**	**BPWM Counter 0 Enable Bit** 0 = BPWM Counter and clock prescaler stop running. 1 = BPWM Counter and clock prescaler start running.

BPWM Clear Counter Register (BPWM_CNTCLR)

Register	Offset	R/W	Description	Reset Value
BPWM_CNTCLR	BPWMx_BA+0x24	R/W	BPWM Clear Counter Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**CNTCLR0**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**CNTCLR0**	**Clear BPWM Counter Control Bit 0** 0 = No effect. 1 = Clear 16-bit BPWM counter to 0000H. **Note:** It is automatically cleared by hardware.

BPWM Period Register (BPWM_PERIOD)

Register	Offset	R/W	Description	Reset Value
BPWM_PERIOD	BPWMx_BA+0x30	R/W	BPWM Period Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PERIOD**	**BPWM Period Register** Up-Count mode: In this mode, BPWM counter counts from 0 to PERIOD, and restarts from 0. BPWM period time = (PERIOD+1) *(CLKPSC+1)* BPWMx_CLK. Down-Count mode: In this mode, BPWM counter counts from PERIOD to 0, and restarts from PERIOD. BPWM period time = (PERIOD+1) *(CLKPSC+1)* BPWMx_CLK. Up-Down-Count mode: In this mode, BPWM counter counts from 0 to PERIOD, then decrements to 0 and repeats again. BPWM period time = (2*PERIOD) *(CLKPSC+1)* BPWMx_CLK.

BPWM Comparator Register 0~5 (BPWM_CMPDAT0~5)

Register	Offset	R/W	Description	Reset Value
BPWM_CMPDAT0	BPWMx_BA+0x50	R/W	BPWM Comparator Register 0	0x0000_0000
BPWM_CMPDAT1	BPWMx_BA+0x54	R/W	BPWM Comparator Register 1	0x0000_0000
BPWM_CMPDAT2	BPWMx_BA+0x58	R/W	BPWM Comparator Register 2	0x0000_0000
BPWM_CMPDAT3	BPWMx_BA+0x5C	R/W	BPWM Comparator Register 3	0x0000_0000
BPWM_CMPDAT4	BPWMx_BA+0x60	R/W	BPWM Comparator Register 4	0x0000_0000
BPWM_CMPDAT5	BPWMx_BA+0x64	R/W	BPWM Comparator Register 5	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CMPDAT**	**BPWM Comparator Register** CMPDAT use to compare with CNT to generate BPWM waveform, interrupt and trigger EADC. In independent mode, BPWM_CMPDATn, n=0,1,…,5 denote as 6 independent BPWM_CH0~5 compared point.

BPWM Counter Register (BPWM_CNT)

Register	Offset	R/W	Description	Reset Value
BPWM_CNT	BPWMx_BA+0x90	R	BPWM Counter Register	0x0000_0000

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

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**DIRF**	**BPWM Direction Indicator Flag (Read Only)** 0 = Counter is down counting. 1 = Counter is up counting.
[15:0]	**CNT**	**BPWM Data Register (Read Only)** Monitor CNT to know the current value in 16-bit period counter.

BPWM Generation Register 0 (BPWM_WGCTL0)

Register	Offset	R/W	Description	Reset Value
BPWM_WGCTL0	BPWMx_BA+0xB0	R/W	BPWM Generation Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				**PRDPCTL5**		**PRDPCTL4**	
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PRDPCTL3		**PRDPCTL2**		**PRDPCTL1**		**PRDPCTL0**	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**ZPCTL5**		**ZPCTL4**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ZPCTL3		**ZPCTL2**		**ZPCTL1**		**ZPCTL0**	

Bits	Description	
[31:28]	**Reserved**	Reserved.
[16+2n+1:16+2n] n=0,1..5	**PRDPCTLn**	**BPWM Period (Center) Point Control** Each bit n controls the corresponding BPWM channel n. 00 = Do nothing. 01 = BPWM period (center) point output Low. 10 = BPWM period (center) point output High. 11 = BPWM period (center) point output Toggle. BPWM can control output level when BPWM counter counts to (PERIOD+1). **Note:** This bit is center point control when BPWM counter operating in up-down counter type.
[15:12]	**Reserved**	Reserved.
[2n+1:2n] n=0,1..5	**ZPCTLn**	**BPWM Zero Point Control** Each bit n controls the corresponding BPWM channel n. 00 = Do nothing. 01 = BPWM zero point output Low. 10 = BPWM zero point output High. 11 = BPWM zero point output Toggle. **Note:** BPWM can control output level when BPWM counter counts to zero.

BPWM Generation Register 1 (BPWM_WGCTL1)

Register	Offset	R/W	Description	Reset Value
BPWM_WGCTL1	BPWMx_BA+0xB4	R/W	BPWM Generation Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				**CMPDCTL5**		**CMPDCTL4**	
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CMPDCTL3		**CMPDCTL2**		**CMPDCTL1**		**CMPDCTL0**	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**CMPUCTL5**		**CMPUCTL4**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CMPUCTL3		**CMPUCTL2**		**CMPUCTL1**		**CMPUCTL0**	

Bits	Description	
[31:28]	**Reserved**	Reserved.
[16+2n+1:16+2n] n=0,1..5	**CMPDCTLn**	**BPWM Compare Down Point Control** Each bit n controls the corresponding BPWM channel n. 00 = Do nothing. 01 = BPWM compare down point output Low. 10 = BPWM compare down point output High. 11 = BPWM compare down point output Toggle. **Note:** BPWM can control output level when BPWM counter down counts to CMPDAT.
[15:12]	**Reserved**	Reserved.
[2n+1:2n] n=0,1..5	**CMPUCTLn**	**BPWM Compare Up Point Control** Each bit n controls the corresponding BPWM channel n. 00 = Do nothing. 01 = BPWM compare up point output Low. 10 = BPWM compare up point output High. 11 = BPWM compare up point output Toggle. **Note:** BPWM can control output level when BPWM counter up counts to CMPDAT.

BPWM Mask Enable Register (BPWM_MSKEN)

Register	Offset	R/W	Description	Reset Value
BPWM_MSKEN	BPWMx_BA+0xB8	R/W	BPWM Mask Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**MSKEN5**	**MSKEN4**	**MSKEN3**	**MSKEN2**	**MSKEN1**	**MSKEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**MSKENn**	**BPWM Mask Enable Bits** Each bit n controls the corresponding BPWM channel n. The BPWM output signal will be masked when this bit is enabled. The corresponding BPWM channel n will output MSKDATn (BPWM_MSK[5:0]) data. 0 = BPWM output signal is non-masked. 1 = BPWM output signal is masked and output MSKDATn data.

BPWM Mask DATA Register (BPWM_MSK)

Register	Offset	R/W	Description	Reset Value
BPWM_MSK	BPWMx_BA+0xBC	R/W	BPWM Mask Data Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**MSKDAT5**	**MSKDAT4**	**MSKDAT3**	**MSKDAT2**	**MSKDAT1**	**MSKDAT0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**MSKDATn**	**BPWM Mask Data Bit** This data bit control the state of BPWMn output pin, if corresponding mask function is enabled. Each bit n controls the corresponding BPWM channel n. 0 = Output logic low to BPWMn. 1 = Output logic high to BPWMn.

BPWM Pin Polar Inverse Control (BPWM_POLCTL)

Register	Offset	R/W	Description	Reset Value
BPWM_POLCTL	BPWMx_BA+0xD4	R/W	BPWM Pin Polar Inverse Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**PINV5**	**PINV4**	**PINV3**	**PINV2**	**PINV1**	**PINV0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**PINVn**	**BPWM PIN Polar Inverse Control** The register controls polarity state of BPWMx_CHn output pin. Each bit n controls the corresponding BPWM channel n. 0 = BPWMx_CHn output pin polar inverse Disabled. 1 = BPWMx_CHn output pin polar inverse Enabled.

BPWM Output Enable Register (BPWM_POEN)

Register	Offset	R/W	Description	Reset Value
BPWM_POEN	BPWMx_BA+0xD8	R/W	BPWM Output Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**POEN5**	**POEN4**	**POEN3**	**POEN2**	**POEN1**	**POEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**POENn**	**BPWM Pin Output Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = BPWMx_CHn pin at tri-state. 1 = BPWMx_CHn pin in output mode.

BPWM Interrupt Enable Register (BPWM_INTEN)

Register	Offset	R/W	Description	Reset Value
BPWM_INTEN	BPWMx_BA+0xE0	R/W	BPWM Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		CMPDIEN5	CMPDIEN4	CMPDIEN3	CMPDIEN2	CMPDIEN1	CMPDIEN0
23	22	21	20	19	18	17	16
Reserved		CMPUIEN5	CMPUIEN4	CMPUIEN3	CMPUIEN2	CMPUIEN1	CMPUIEN0
15	14	13	12	11	10	9	8
Reserved							PIEN0
7	6	5	4	3	2	1	0
Reserved							ZIEN0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[24+n] n=0,1..5	**CMPDIENn**	**BPWM Compare Down Count Interrupt Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = Compare down count interrupt Disabled. 1 = Compare down count interrupt Enabled.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**CMPUIENn**	**BPWM Compare Up Count Interrupt Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = Compare up count interrupt Disabled. 1 = Compare up count interrupt Enabled.
[15:9]	**Reserved**	Reserved.
[8]	**PIEN0**	**BPWM Period Point Interrupt 0 Enable Bit** 0 = Period point interrupt Disabled. 1 = Period point interrupt Enabled. **Note:** When up-down counter type period point means center point.
[7:1]	**Reserved**	Reserved.
[0]	**ZIEN0**	**BPWM Zero Point Interrupt 0 Enable Bit** 0 = Zero point interrupt Disabled. 1 = Zero point interrupt Enabled.

BPWM Interrupt Flag Register (BPWM_INTSTS)

Register	Offset	R/W	Description	Reset Value
BPWM_INTSTS	BPWMx_BA+0xE8	R/W	BPWM Interrupt Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		CMPDIF5	CMPDIF4	CMPDIF3	CMPDIF2	CMPDIF1	CMPDIF0
23	22	21	20	19	18	17	16
Reserved		CMPUIF5	CMPUIF4	CMPUIF3	CMPUIF2	CMPUIF1	CMPUIF0
15	14	13	12	11	10	9	8
Reserved							PIF0
7	6	5	4	3	2	1	0
Reserved							ZIF0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[24+n] n=0,1..5	**CMPDIFn**	**BPWM Compare Down Count Interrupt Flag** Each bit n controls the corresponding BPWM channel n. Flag is set by hardware when BPWM counter down count and reaches BPWM_CMPDATn, software can clear this bit by writing 1 to it. **Note:** If CMPDAT equal to PERIOD, this flag is not working in down counter type selection.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**CMPUIFn**	**BPWM Compare Up Count Interrupt Flag** Flag is set by hardware when BPWM counter up count and reaches BPWM_CMPDATn, software can clear this bit by writing 1 to it. Each bit n controls the corresponding BPWM channel n. **Note:** If CMPDAT equal to PERIOD, this flag is not working in up counter type selection.
[15:9]	**Reserved**	Reserved.
[8]	**PIF0**	**BPWM Period Point Interrupt Flag 0** This bit is set by hardware when BPWM_CH0 counter reaches BPWM_PERIOD, software can write 1 to clear this bit to 0.
[7:1]	**Reserved**	Reserved.
[0]	**ZIF0**	**BPWM Zero Point Interrupt Flag 0** This bit is set by hardware when BPWM_CH0 counter reaches 0, software can write 1 to clear this bit to 0.

BPWM Trigger EADC Source Select Register 0 (BPWM_EADCTS0)

Register	Offset	R/W	Description	Reset Value
BPWM_EADCTS0	BPWMx_BA+0xF8	R/W	BPWM Trigger EADC Source Select Register 0	0x0000_0000

31	30	29	28	27	26	25	24
TRGEN3	Reserved			TRGSEL3			
23	22	21	20	19	18	17	16
TRGEN2	Reserved			TRGSEL2			
15	14	13	12	11	10	9	8
TRGEN1	Reserved			TRGSEL1			
7	6	5	4	3	2	1	0
TRGEN0	Reserved			TRGSEL0			

Bits	Description	
[31]	**TRGEN3**	**BPWM_CH3 Trigger EADC Enable Bit** 0 = BPWM_CH3 Trigger EADC function Disabled. 1 = BPWM_CH3 Trigger EADC function Enabled.
[30:28]	**Reserved**	Reserved.
[27:24]	**TRGSEL3**	**BPWM_CH3 Trigger EADC Source Select** 0000 = BPWM_CH2 zero point. 0001 = BPWM_CH2 period point. 0010 = BPWM_CH2 zero or period point. 0011 = BPWM_CH2 up-count compared point. 0100 = BPWM_CH2 down-count compared point. 1000 = BPWM_CH3 up-count compared point. 1001 = BPWM_CH3 down-count compared point. Others reserved.
[23]	**TRGEN2**	**BPWM_CH2 Trigger EADC Enable Bit** 0 = BPWM_CH2 Trigger EADC function Disabled. 1 = BPWM_CH2 Trigger EADC function Enabled.
[22:20]	**Reserved**	Reserved.
[19:16]	**TRGSEL2**	**BPWM_CH2 Trigger EADC Source Select** 0000 = BPWM_CH2 zero point. 0001 = BPWM_CH2 period point. 0010 = BPWM_CH2 zero or period point. 0011 = BPWM_CH2 up-count compared point. 0100 = BPWM_CH2 down-count compared point. 1000 = BPWM_CH3 up-count compared point. 1001 = BPWM_CH3 down-count compared point.

		Others reserved
[15]	**TRGEN1**	**BPWM_CH1 Trigger EADC Enable Bit** 0 = BPWM_CH1 Trigger EADC function Disabled. 1 = BPWM_CH1 Trigger EADC function Enabled.
[14:12]	**Reserved**	Reserved.
[11:8]	**TRGSEL1**	**BPWM_CH1 Trigger EADC Source Select** 0000 = BPWM_CH0 zero point. 0001 = BPWM_CH0 period point. 0010 = BPWM_CH0 zero or period point. 0011 = BPWM_CH0 up-count compared point. 0100 = BPWM_CH0 down-count compared point. 1000 = BPWM_CH1 up-count compared point. 1001 = BPWM_CH1 down-count compared point. Others reserved
[7]	**TRGEN0**	**BPWM_CH0 Trigger EADC Enable Bit** 0 = BPWM_CH0 Trigger EADC function Disabled. 1 = BPWM_CH0 Trigger EADC function Enabled.
[6:4]	**Reserved**	Reserved.
[3:0]	**TRGSEL0**	**BPWM_CH0 Trigger EADC Source Select** 0000 = BPWM_CH0 zero point. 0001 = BPWM_CH0 period point. 0010 = BPWM_CH0 zero or period point. 0011 = BPWM_CH0 up-count compared point. 0100 = BPWM_CH0 down-count compared point. 1000 = BPWM_CH1 up-count compared point. 1001 = BPWM_CH1 down-count compared point. Others reserved

BPWM Trigger EADC Source Select Register 1 (BPWM_EADCTS1)

Register	Offset	R/W	Description	Reset Value
BPWM_EADCTS1	BPWMx_BA+0xFC	R/W	BPWM Trigger EADC Source Select Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
TRGEN5	Reserved			TRGSEL5			
7	6	5	4	3	2	1	0
TRGEN4	Reserved			TRGSEL4			

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**TRGEN5**	**BPWM_CH5 Trigger EADC Enable Bit** 0 = BPWM_CH5 Trigger EADC function Disabled. 1 = BPWM_CH5 Trigger EADC function Enabled.
[14:12]	**Reserved**	Reserved.
[11:8]	**TRGSEL5**	**BPWM_CH5 Trigger EADC Source Select** 0000 = BPWM_CH4 zero point. 0001 = BPWM_CH4 period point. 0010 = BPWM_CH4 zero or period point. 0011 = BPWM_CH4 up-count compared point. 0100 = BPWM_CH4 down-count compared point. 1000 = BPWM_CH5 up-count compared point. 1001 = BPWM_CH5 down-count compared point. Others reserved
[7]	**TRGEN4**	**BPWM_CH4 Trigger EADC Enable Bit** 0 = BPWM_CH4 Trigger EADC function Disabled. 1 = BPWM_CH4 Trigger EADC function Enabled.
[6:4]	**Reserved**	Reserved.
[3:0]	**TRGSEL4**	**BPWM_CH4 Trigger EADC Source Select** 0000 = BPWM_CH4 zero point. 0001 = BPWM_CH4 period point. 0010 = BPWM_CH4 zero or period point. 0011 = BPWM_CH4 up-count compared point. 0100 = BPWM_CH4 down-count compared point.

		1000 = BPWM_CH5 up-count compared point. 1001 = BPWM_CH5 down-count compared point. Others reserved

BPWM Synchronous Start Control Register (BPWM_SSCTL)

Register	Offset	R/W	Description	Reset Value
BPWM_SSCTL	BPWMx_BA+0x110	R/W	BPWM Synchronous Start Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						SSRC	
7	6	5	4	3	2	1	0
Reserved							SSEN0

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9:8]	**SSRC**	**BPWM Synchronous Start Source Select** 00 = Synchronous start source come from PWM0. 01 = Synchronous start source come from PWM1. 10 = Synchronous start source come from BPWM0. 11 = Synchronous start source come from BPWM1.
[7:1]	**Reserved**	Reserved.
[0]	**SSEN0**	**BPWM Synchronous Start Function 0 Enable Bit** When synchronous start function is enabled, the BPWM_CH0 counter enable bit (CNTEN0) can be enabled by writing BPWM synchronous start trigger bit (CNTSEN). 0 = BPWM synchronous start function Disabled. 1 = BPWM synchronous start function Enabled.

BPWM Synchronous Start Trigger Register (BPWM_SSTRG)

Register	Offset	R/W	Description	Reset Value
BPWM_SSTRG	BPWMx_BA+0x114	W	BPWM Synchronous Start Trigger Register	0x0000_0000

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

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	CNTSEN	**BPWM Counter Synchronous Start Enable Bit (Write Only)** BPMW counter synchronous enable function is used to make PWM or BPWM channels start counting at the same time. Writing this bit to 1 will also set the counter enable bit if correlated BPWM channel counter synchronous start function is enabled.

BPWM Status Register (BPWM_STATUS)

Register	Offset	R/W	Description	Reset Value
BPWM_STATUS	BPWMx_BA+0x120	R/W	BPWM Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		EADCTRG5	EADCTRG4	EADCTRG3	EADCTRG2	EADCTRG1	EADCTRG0
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							CNTMAX0

Bits	Description	
[31:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**EADCTRGn**	**EADC Start of Conversion Status** Each bit n controls the corresponding BPWM channel n. 0 = No EADC start of conversion trigger event has occurred. 1 = An EADC start of conversion trigger event has occurred. **Note:** This bit can be cleared by software write 1.
[15:1]	**Reserved**	Reserved.
[0]	**CNTMAX0**	**Time-base Counter 0 Equal to 0xFFFF Latched Status** 0 = The time-base counter never reached its maximum value 0xFFFF. 1 = The time-base counter reached its maximum value. **Note:** This bit can be cleared by software write 1.

BPWM Capture Input Enable Register (BPWM_CAPINEN)

Register	Offset	R/W	Description	Reset Value
BPWM_CAPINEN	BPWMx_BA+0x200	R/W	BPWM Capture Input Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CAPINEN5**	**CAPINEN4**	**CAPINEN3**	**CAPINEN2**	**CAPINEN1**	**CAPINEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CAPINENn**	**Capture Input Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = BPWM Channel capture input path Disabled. The input of BPWM channel capture function is always regarded as 0. 1 = BPWM Channel capture input path Enabled. The input of BPWM channel capture function comes from correlative multifunction pin.

BPWM Capture Control Register (BPWM_CAPCTL)

Register	Offset	R/W	Description	Reset Value
BPWM_CAPCTL	BPWMx_BA+0x204	R/W	BPWM Capture Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		FCRLDEN5	FCRLDEN4	FCRLDEN3	FCRLDEN2	FCRLDEN1	FCRLDEN0
23	22	21	20	19	18	17	16
Reserved		RCRLDEN5	RCRLDEN4	RCRLDEN3	RCRLDEN2	RCRLDEN1	RCRLDEN0
15	14	13	12	11	10	9	8
Reserved		CAPINV5	CAPINV4	CAPINV3	CAPINV2	CAPINV1	CAPINV0
7	6	5	4	3	2	1	0
Reserved		CAPEN5	CAPEN4	CAPEN3	CAPEN2	CAPEN1	CAPEN0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[24+n] n=0,1..5	**FCRLDENn**	**Falling Capture Reload Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = Falling capture reload counter Disabled. 1 = Falling capture reload counter Enabled.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**RCRLDENn**	**Rising Capture Reload Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = Rising capture reload counter Disabled. 1 = Rising capture reload counter Enabled.
[15:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**CAPINVn**	**Capture Inverter Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = Capture source inverter Disabled. 1 = Capture source inverter Enabled. Reverse the input signal from GPIO.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CAPENn**	**Capture Function Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = Capture function Disabled. BPWM_RCAPDATn/BPWM_FCAPDATn register will not be updated. 1 = Capture function Enabled. Capture latched the BPWM counter value when detected rising or falling edge of input signal and saved to RCAPDAT (Rising latch) and FCAPDAT (Falling latch).

BPWM Capture Status Register (BPWM_CAPSTS)

Register	Offset	R/W	Description	Reset Value
BPWM_CAPSTS	BPWMx_BA+0x208	R	BPWM Capture Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		CFIFOV5	CFIFOV4	CFIFOV3	CFIFOV2	CFIFOV1	CFIFOV0
7	6	5	4	3	2	1	0
Reserved		CRIFOV5	CRIFOV4	CRIFOV3	CRIFOV2	CRIFOV1	CRIFOV0

Bits	Description	
[31:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**CFIFOVn**	**Capture Falling Interrupt Flag Overrun Status (Read Only)** This flag indicates if falling latch happened when the corresponding CAPFIFn is 1. Each bit n controls the corresponding BPWM channel n. **Note:** This bit will be cleared automatically when the corresponding CAPFIFn is cleared.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CRIFOVn**	**Capture Rising Interrupt Flag Overrun Status (Read Only)** This flag indicates if rising latch happened when the corresponding CAPRIFn is 1. Each bit n controls the corresponding BPWM channel n. **Note:** This bit will be cleared automatically when the corresponding CAPRIFn is cleared.

BPWM Rising Capture Data Register 0~5 (BPWM_RCAPDAT 0~5)

Register	Offset	R/W	Description	Reset Value
BPWM_RCAPDAT0	BPWMx_BA+0x20C	R	BPWM Rising Capture Data Register 0	0x0000_0000
BPWM_RCAPDAT1	BPWMx_BA+0x214	R	BPWM Rising Capture Data Register 1	0x0000_0000
BPWM_RCAPDAT2	BPWMx_BA+0x21C	R	BPWM Rising Capture Data Register 2	0x0000_0000
BPWM_RCAPDAT3	BPWMx_BA+0x224	R	BPWM Rising Capture Data Register 3	0x0000_0000
BPWM_RCAPDAT4	BPWMx_BA+0x22C	R	BPWM Rising Capture Data Register 4	0x0000_0000
BPWM_RCAPDAT5	BPWMx_BA+0x234	R	BPWM Rising Capture Data Register 5	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**RCAPDAT**	**BPWM Rising Capture Data (Read Only)** When rising capture condition happened, the BPWM counter value will be saved in this register.

BPWM Falling Capture Data Register 0~5 (BPWM_FCAPDAT 0~5)

Register	Offset	R/W	Description	Reset Value
BPWM_FCAPDAT0	BPWMx_BA+0x210	R	BPWM Falling Capture Data Register 0	0x0000_0000
BPWM_FCAPDAT1	BPWMx_BA+0x218	R	BPWM Falling Capture Data Register 1	0x0000_0000
BPWM_FCAPDAT2	BPWMx_BA+0x220	R	BPWM Falling Capture Data Register 2	0x0000_0000
BPWM_FCAPDAT3	BPWMx_BA+0x228	R	BPWM Falling Capture Data Register 3	0x0000_0000
BPWM_FCAPDAT4	BPWMx_BA+0x230	R	BPWM Falling Capture Data Register 4	0x0000_0000
BPWM_FCAPDAT5	BPWMx_BA+0x238	R	BPWM Falling Capture Data Register 5	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**FCAPDAT**	**BPWM Falling Capture Data (Read Only)** When falling capture condition happened, the BPWM counter value will be saved in this register.

BPWM Capture Interrupt Enable Register (BPWM_CAPIEN)

Register	Offset	R/W	Description	Reset Value
BPWM_CAPIEN	BPWMx_BA+0x250	R/W	BPWM Capture Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**CAPFIEN5**	**CAPFIEN4**	**CAPFIEN3**	**CAPFIEN2**	**CAPFIEN1**	**CAPFIEN0**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CAPRIEN5**	**CAPRIEN4**	**CAPRIEN3**	**CAPRIEN2**	**CAPRIEN1**	**CAPRIEN0**

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13:8]	**CAPFIENn**	**BPWM Capture Falling Latch Interrupt Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = Capture falling edge latch interrupt Disabled. 1 = Capture falling edge latch interrupt Enabled.
[7:6]	**Reserved**	Reserved.
[5:0]	**CAPRIENn**	**BPWM Capture Rising Latch Interrupt Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = Capture rising edge latch interrupt Disabled. 1 = Capture rising edge latch interrupt Enabled.

BPWM Capture Interrupt Flag Register (BPWM_CAPIF)

Register	Offset	R/W	Description	Reset Value
BPWM_CAPIF	BPWMx_BA+0x254	R/W	BPWM Capture Interrupt Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**CAPFIF5**	**CAPFIF4**	**CAPFIF3**	**CAPFIF2**	**CAPFIF1**	**CAPFIF0**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CAPRIF5**	**CAPRIF4**	**CAPRIF3**	**CAPRIF2**	**CAPRIF1**	**CAPRIF0**

Bits	Description	
[31:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**CAPFIFn**	**BPWM Capture Falling Latch Interrupt Flag** Each bit n controls the corresponding BPWM channel n. 0 = No capture falling latch condition happened. 1 = Capture falling latch condition happened, this flag will be set to high. **Note:** This bit is cleared by writing 1 to it.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CAPRIFn**	**BPWM Capture Rising Latch Interrupt Flag** Each bit n controls the corresponding BPWM channel n. 0 = No capture rising latch condition happened. 1 = Capture rising latch condition happened, this flag will be set to high. **Note:** This bit is cleared by writing 1 to it.

BPWM Period Register Buffer (BPWM_PBUF)

Register	Offset	R/W	Description	Reset Value
BPWM_PBUF	BPWMx_BA+0x304	R	BPWM PERIOD Buffer	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PBUF**	**BPWM Period Buffer (Read Only)** Used as PERIOD active register.

BPWM Comparator Register Buffer 0~5 (BPWM_CMPBUF0~5)

Register	Offset	R/W	Description	Reset Value
BPWM_CMPBUF0	BPWMx_BA+0x31C	R	BPWM CMPDAT 0 Buffer	0x0000_0000
BPWM_CMPBUF1	BPWMx_BA+0x320	R	BPWM CMPDAT 1 Buffer	0x0000_0000
BPWM_CMPBUF2	BPWMx_BA+0x324	R	BPWM CMPDAT 2 Buffer	0x0000_0000
BPWM_CMPBUF3	BPWMx_BA+0x328	R	BPWM CMPDAT 3 Buffer	0x0000_0000
BPWM_CMPBUF4	BPWMx_BA+0x32C	R	BPWM CMPDAT 4 Buffer	0x0000_0000
BPWM_CMPBUF5	BPWMx_BA+0x330	R	BPWM CMPDAT 5 Buffer	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CMPBUF**	**BPWM Comparator Buffer (Read Only)** Used as CMPDAT active register.

6.17 Quadrature Encoder Interface (QEI)

6.17.1 Overview

There are two Quadrature Encoder Interfaces (QEI) controllers in this device. The QEI decodes speed of rotation and motion sensor information and can be used in any application that uses a quadrature encoder for feedback.

6.17.2 Features

- Up to two QEI controllers, QEI0 and QEI1.
- Two QEI phase inputs, QEA and QEB; One Index input.
- A 32-bit up/down Quadrature Encoder Pulse Counter (QEI_CNT)
- A 32-bit software-latch Quadrature Encoder Pulse Counter Hold Register (QEI_CNTHOLD)
- A 32-bit Quadrature Encoder Pulse Counter Index Latch Register (QEI_CNTLATCH)
- A 32-bit Quadrature Encoder Pulse Counter Compare Register (QEI_CNTCMP) with a Pre-set Maximum Count Register (QEI_CNTMAX)
- One QEI control register (QEI_CTL) and one QEI Status Register (QEI_STATUS)
- Four Quadrature encoder pulse counter operation modes
  - Support x4 free-counting mode
  - Support x2 free-counting mode
  - Support x4 compare-counting mode
  - Support x2 compare-counting mode
- Encoder Pulse Width measurement mode
- Input frequency of QEA/QEB/IDX without noise filter must be lower than PCLK/4
- Input frequency of QEA/QEB/IDX with noise filter must be lower than Noise Filter Clk/8

6.17.3 Block Diagram

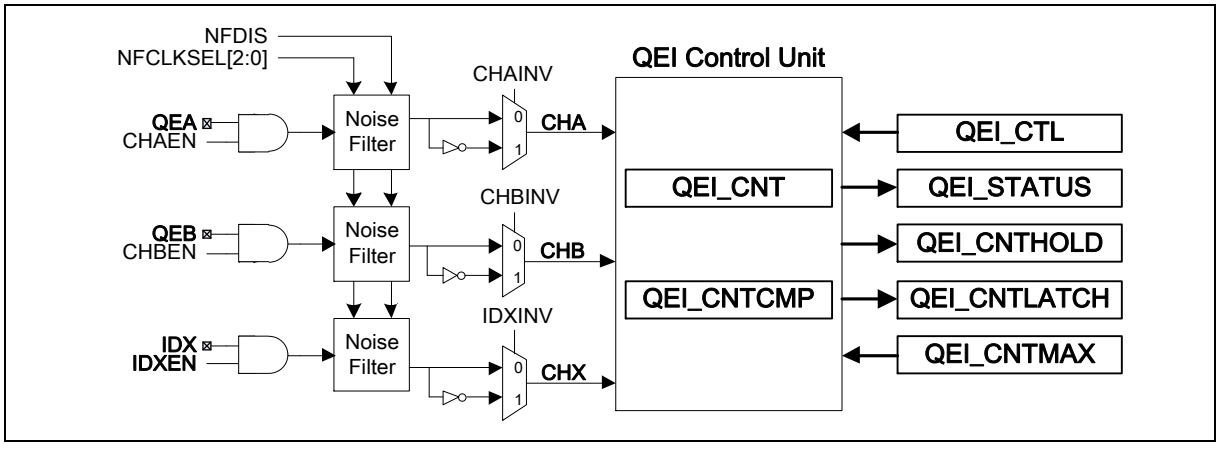


Figure 6.17-1 QEI Block Diagram

The QEI controller inputs, QEA and QEB, accept the outputs from a quadrature encoded source, such

as incremental optical shaft encoder. Two channels, A and B, nominally 90 degrees out of phase, are required. A quadrature encoder usually provides an index signal (to pin IDX) which can be used to indicate an absolute position. There is a noise filter and polarity control for each signal before QEI control unit.

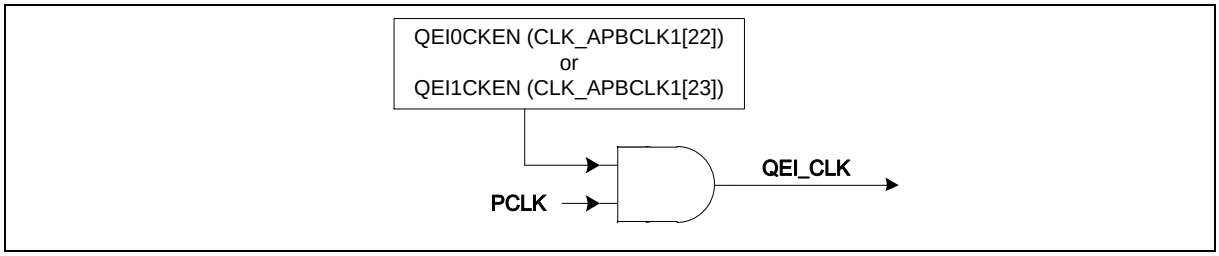


Figure 6.17-2 QEI Clock Source Control

6.17.4 Basic Configuration

6.17.4.1 QEI0 Basic Configuration

- Clock Source Configuration
  - Enable QEI0 peripheral clock in QEI0CKEN (CLK_APBCLK1[22]).
- Reset Configuration
  - Reset QEI0 controller in QEI0RST (SYS_IPRST2[22]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
QEI0	QEI0_A	PD.11	MFP10
		PE.3	MFP11
		PA.4	MFP14
	QEI0_B	PD.10	MFP10
		PE.2	MFP11
		PA.3	MFP14
	QEI0_INDEX	PD.12	MFP10
		PE.4	MFP11
		PA.5	MFP14

6.17.4.2 QEI1 Basic Configuration

- Clock Source Configuration
  - Enable QEI1 peripheral clock in QEI1CKEN (CLK_APBCLK1[23]).
- Reset Configuration
  - Reset QEI1 controller in QEI1RST (SYS_IPRST2[23]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
QEI1	QEI1_A	PA.9	MFP10

		PE.6	MFP11
		PA.13	MFP12
	QEI1_B	PA.8	MFP10
		PE.5	MFP11
		PA.14	MFP12
	QEI1_INDEX	PA.10	MFP10
		PE.7	MFP11
		PA.12	MFP12

6.17.5 Functional Description

The QEI control logic detects the relation of phase lead/lag between the filtered signals CHA and CHB and CHX to produce direction indication bit DIRF(QEI_STATUS[8]) to control the pulse counter. The comparator/reload logic compares the pulse counter and maximum count and control the function of reloading pulse counter in compare-counting mode. In Free-counting mode the pulse counter CNT(QEI_CNT[31:0]) will count until the 0xFFFF_FFFF value; while in Compare-counting mode the pulse counter will counts until the CNTMAX (QEI_CNTMAX[31:0]) and the pulse counter will be reset to 0 to restart the next cyclic counting.

6.17.5.1 Input Noise Filter

Each pin of QEI inputs is equipped with a noise filter which can filter the unwanted noise from GPIO. The QEA, QEB and IDX noise filters can be disabled through bits NFDIS (QEI_CTL[3]). If enabled, the capture logic is required to sample 4 consecutive same capture input value in order to recognize an edge as a capture event. In Figure 6.17-5, it is a possible implementation of digital noise filter; the interval between pulses requirement for input capture is 4 QEI_CLK clocks width. Any pulse width less than or equal to 3 QEI_CLK clocks will not have any trigger. The timing requirement through Noise Filter is also shown in Figure 6.17-5. CHA, CHB and CHX are the outputs of QEA, QEB and IDX respectively after going through noise filter and polarity control. Refer to Figure 6.17-3. If the noise filter is disabled the input signals QEA, QEB and IDX are passed to the internal signals CHA, CHB and CHX respectively without any delay.

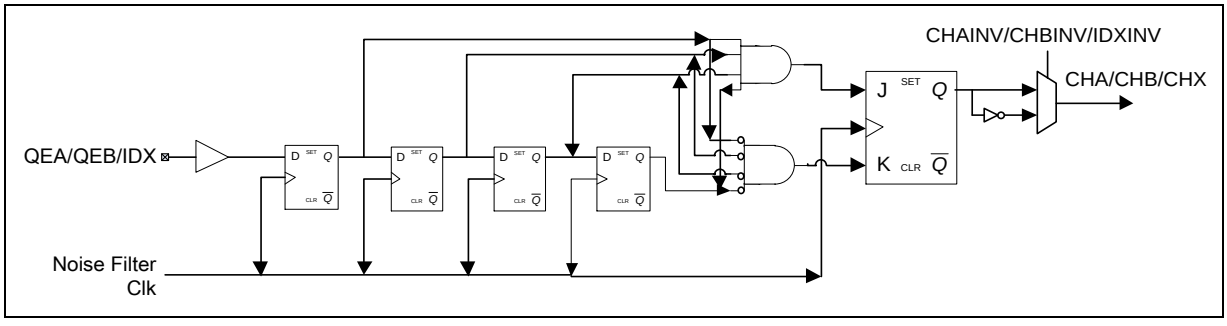


Figure 6.17-3 Noise Filter

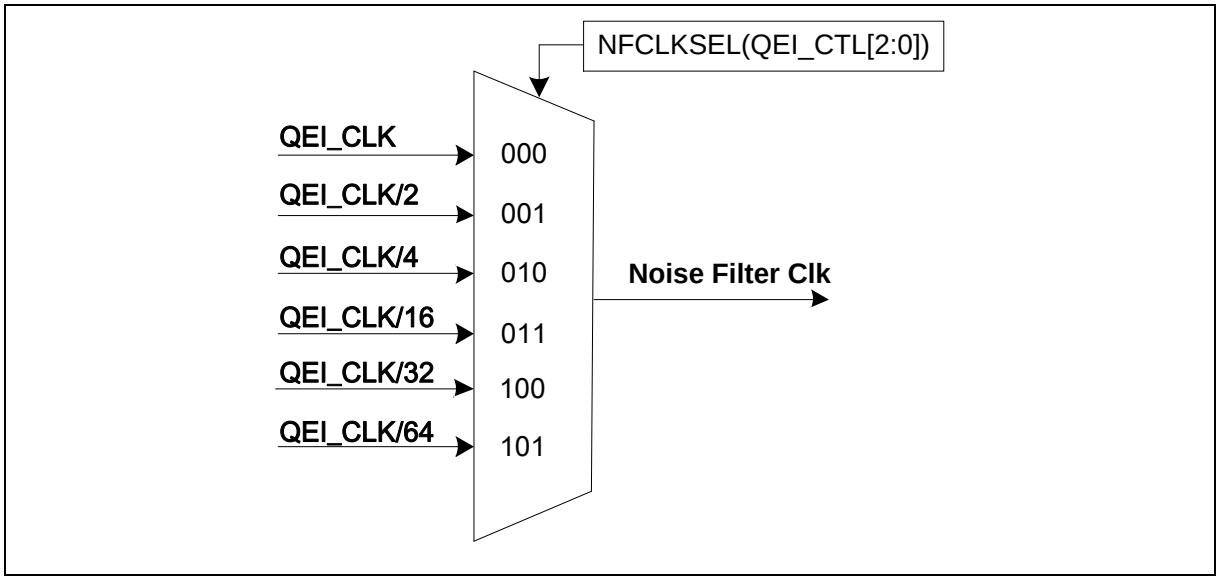


Figure 6.17-4 Noise Filter Sampling Clock Selection

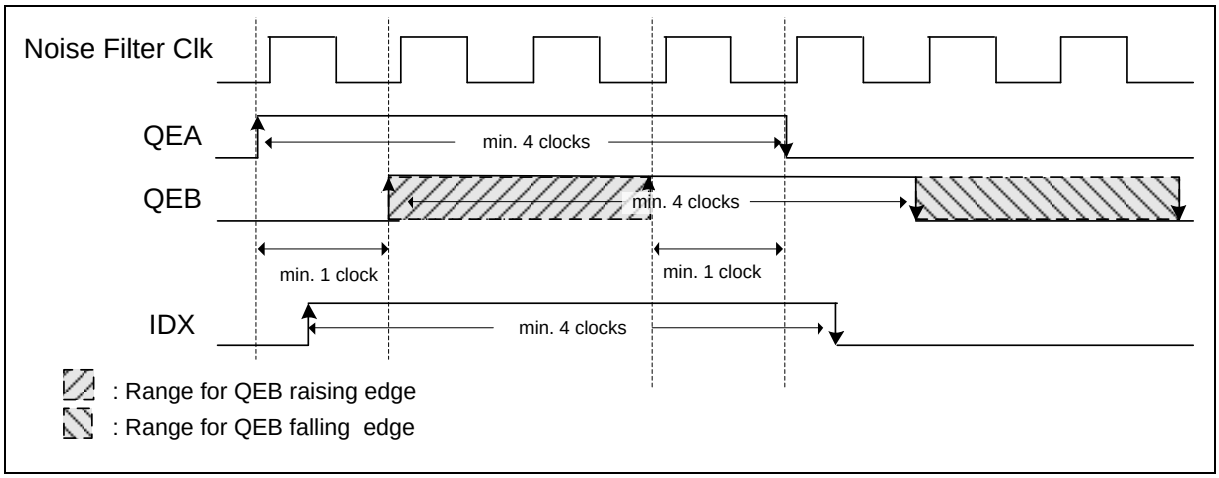


Figure 6.17-5 QEA/QEB/IDX Timing Requirement through Noise Filter

6.17.5.2 Operation of Quadrature Encoder Interface

There are four Quadrature encoder pulse counter operation modes:

- Mode0: x4 free-counting mode
- Mode1: x2 free-counting mode
- Mode2: x4 compare-counting mode
- Mode3: x2 compare-counting mode

6.17.5.3 Free-counting Mode

The quadrature encoder pulse counter CNT(QEI_CNT[31:0]) up or down counts according to the direction indication bit DIRF (QEI_STATUS[8]). When overflow or underflow occurs, it sets flag OVUNF (QEI_STATUS[2]). Refer to Figure 6.17-6 and Figure 6.17-7 for detailed timing.

6.17.5.4 Compare-counting Mode

The pulse counter up or down counts according to the direction indication bit DIRF (QEI_STATUS[8]). On up counting, flag OVUNF (QEI_STATUS[2]) will be asserted when CNT(QEI_CNT[31:0]) overflows from CNTMAX (QEI_CNTMAX[31:0]) to 0 on the next CHA edge for x2 counting mode, and on CHA/CHB edge for x4 counting mode. On down counting, flag OVUNF (QEI_STATUS[2]) will be asserted when CNT(QEI_CNT[31:0]) underflows from 0 to CNTMAX (QEI_CNTMAX[31:0]) on the next CHA edge for x2 counting mode, and on CHA/CHB edge for x4 counting mode. This mode provides the position of a rotor to user. If a quadrature encoder outputs 1024 pulses to CHA per round, user can write QEI_MAXCNT and CNTCMP(QEI_CNTCMP[31:0]) with 4095 in x4 mode or 2047 in x2 mode and reset CNT(QEI_CNT[31:0]) at initial before compare-counting mode is active. When the CNT(QEI_CNT[31:0]) overflows from CNTCMP(QEI_CNTCMP[31:0]), here CNTCMP(QEI_CNTCMP[31:0]) should be preset the same value as CNTMAX (QEI_CNTMAX[31:0]), it means that rotor runs one round on next CHA/CHB edge. Refer to Figure 6.17-6 and Figure 6.17-7 for detailed timing.

6.17.5.5 X4/X2 Counting Modes

In **X4 Counting mode**, the pulse counter increases or decreases one on every CHA and CHB edge based on the phase relationship of CHA and CHB signals.

QEI X4 Counting mode provides a finer resolution of the rotor position, since the counter increments or decrements more frequently for each QEA/QEB input pulse pair than in QEI x2 mode. This mode is selected by setting the QEI Counting Mode Selection bits MODE(QEI_CTL[9:8]) to 00b or 01b. In this mode, the QEI logic detects every edge on every QEA and QEB input edges.

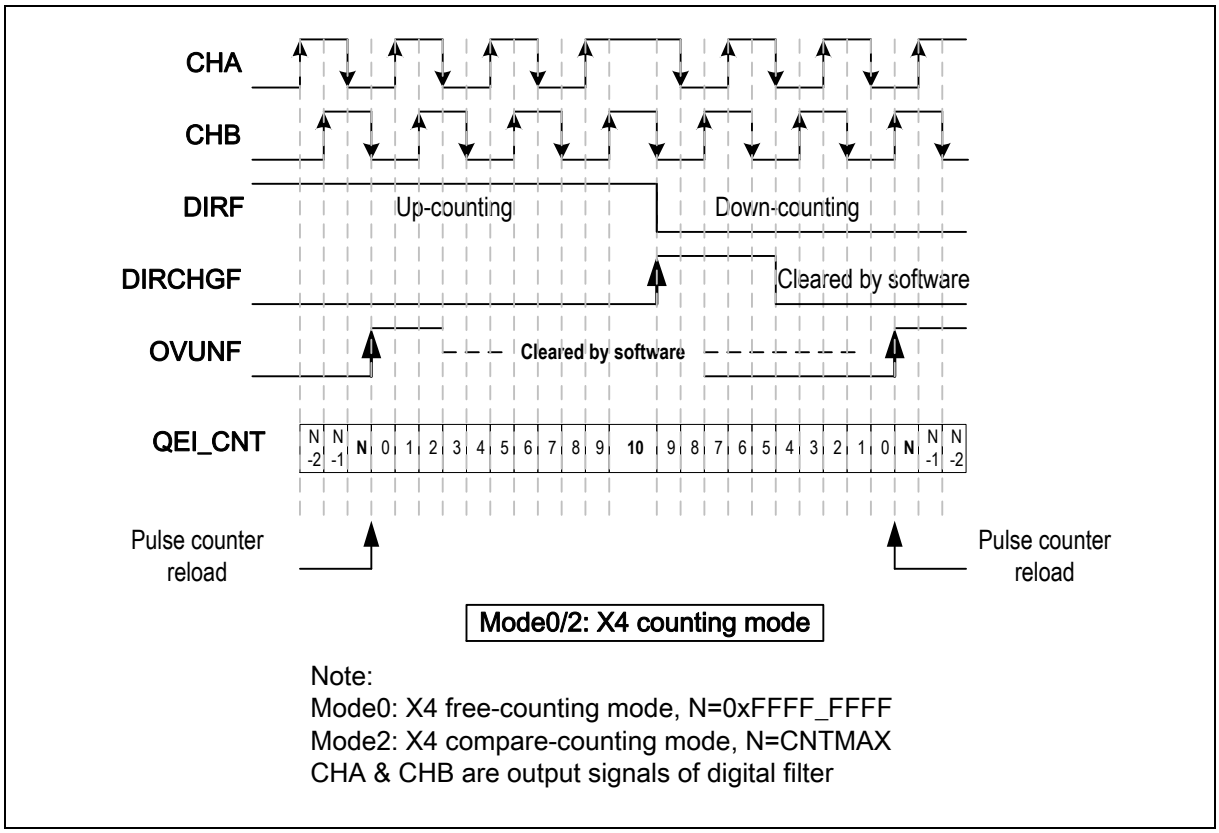


Figure 6.17-6 X4 Counting Mode

In **X2 Counting mode**, the pulse counter increases or decreases one on every CHA edge based on the phase relationship of CHA and CHB signals.

QEI X2 Counting mode is selected by setting the QEI Counting Mode Selection bits (QEI_CTL[9:8]) to 01b or 11b. In this mode, the QEI logic detects every edge on the QEA input only. Every rising and falling edge on the QEA signal clocks the pulse counter.

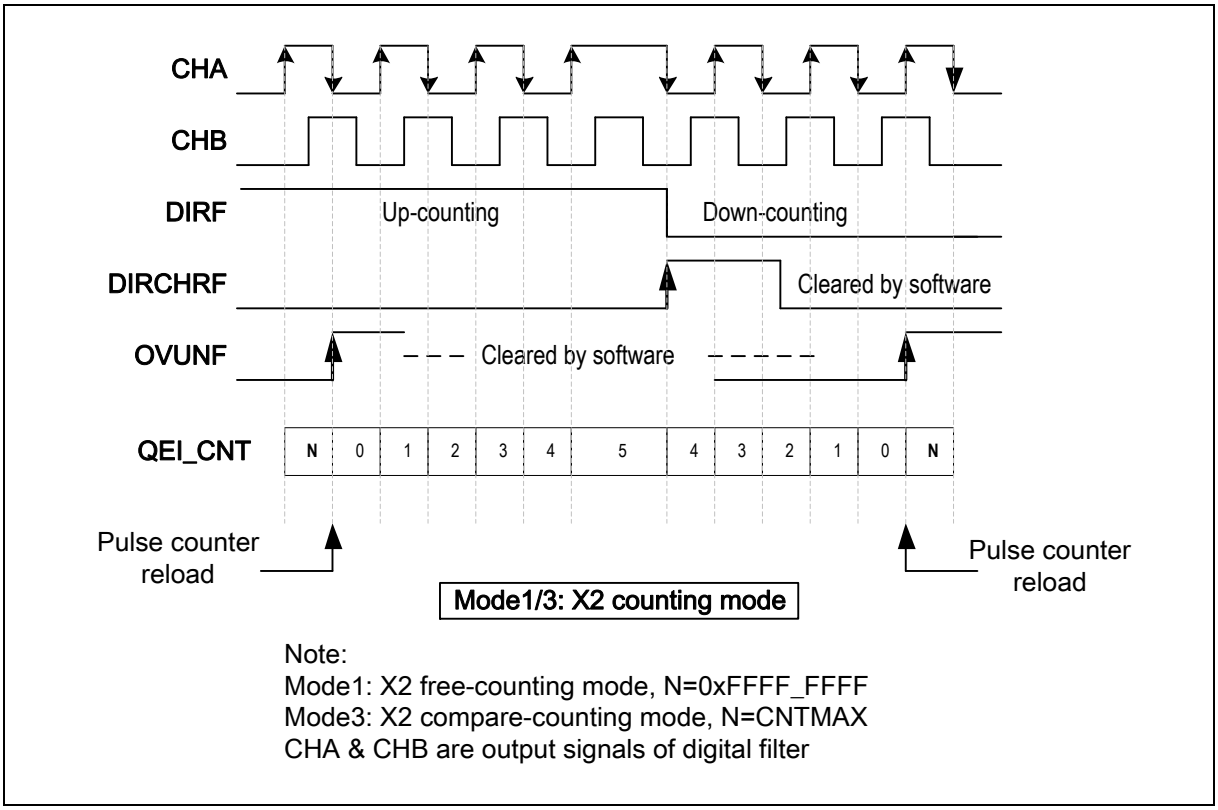


Figure 6.17-7 X2 Counting Mode

6.17.5.6 Direction of Counting

Refer to Table 6.17-1. If CHA leads CHB, the pulse counter is increased by 1. If CHA lags CHB, the pulse counter is decreased by 1. The QEI control logic generates a signal that sets the DIRF (QEI_STATUS[8]); this in turn determines the direction of the count. When CHA leads CHB, DIRF(QEI_STATUS[8]) is set as 1, and the position counter increments on every active edge. When CHA lags CHB, DIRF(QEI_STATUS[8]) is cleared, and the position counter decrements on every active edge.

Current Signal Detected	Previous Signal Detected				DIR (Counting Direction)
	Rising		Falling		
	CHA	CHB	CHA	CHB	
CHA rising				√	1 (Increment)
		√			0 (Decrement)
			√		Toggle (direction change)
CHA falling				√	0 (Decrement)
		√			1 (Increment)
	√				Toggle (direction change)

CHB rising	√				1 (Increment)
			√		0 (Decrement)
				√	Toggle (direction change)
CHB falling			√		1 (Increment)
	√				0 (Decrement)
		√			Toggle (direction change)

Table 6.17-1 Direction of Counting

6.17.5.7 Up-Counting

Under the forward direction the DIRF(QEI_STATUS[8]) bit is 1 when up-counting. Software needs to clear the OVUNF (QEI_STATUS[2]) flag. For the free-counting mode the CNT(QEI_CNT[31:0]) counter will count until it matches 0xFFFF_FFFF and next edges on the forward direction will set the bit OVUNF (QEI_STATUS[2]) high and reset CNT(QEI_CNT[31:0]) to 0. For compare-counting mode the CNT(QEI_CNT[31:0]) counter counts until the CNTMAX (QEI_CNTMAX[31:0]) value and next edges on the forward direction will set the bit OVUNF (QEI_STATUS[2]) high and reset CNT(QEI_CNT[31:0]) to 0. Changes of direction trigger a down-count and CNT(QEI_CNT[31:0]) decreasing in counter value. For X2 mode, only CHA edge will set OVUNF (QEI_STATUS[2]) while for X4 mode both CHA and CHB edges will set OVUNF (QEI_STATUS[2]).

6.17.5.8 Down-Counting

A change of direction will cause the counter to down count for X2/X4 counting mode. It is indicated with the DIRF(QEI_STATUS[8]) bit as 0 and DIRCHGF (QEI_STATUS[3]) flag is set to 1. At this stage the CNT(QEI_CNT[31:0]) will start to down-count. In free-counting mode the pulse counter will reload with 0xFFFF_FFFF when it down counts to 0 and sets OVUNF (QEI_STATUS[2]) to high in the next edge. The pulse counter will reload with CNTMAX (QEI_CNTMAX[31:0]) when it down counts to 0 in compare-counting mode and sets OVUNF (QEI_STATUS[2]) to high in the next edge. For X2 mode, only CHA edge will set OVUNF (QEI_STATUS[2]) while for X4 mode both CHA and CHB edges will set OVUNF (QEI_STATUS[2]).

6.17.5.9 Compare Function

The compare function in QEI controller is used to compare the dynamic counting CNT(QEI_CNT[31:0]) with the compare register CNTCMP(QEI_CNTCMP[31:0]). When CNT(QEI_CNT[31:0]) up or down counts and reaches CNTCMP(QEI_CNTCMP[31:0]), the flag CMPF(QEI_STATUS[1]) will be set. Set bit CMP_EN (QEI_CTL[28]) to one to enable the compare function otherwise disable it.

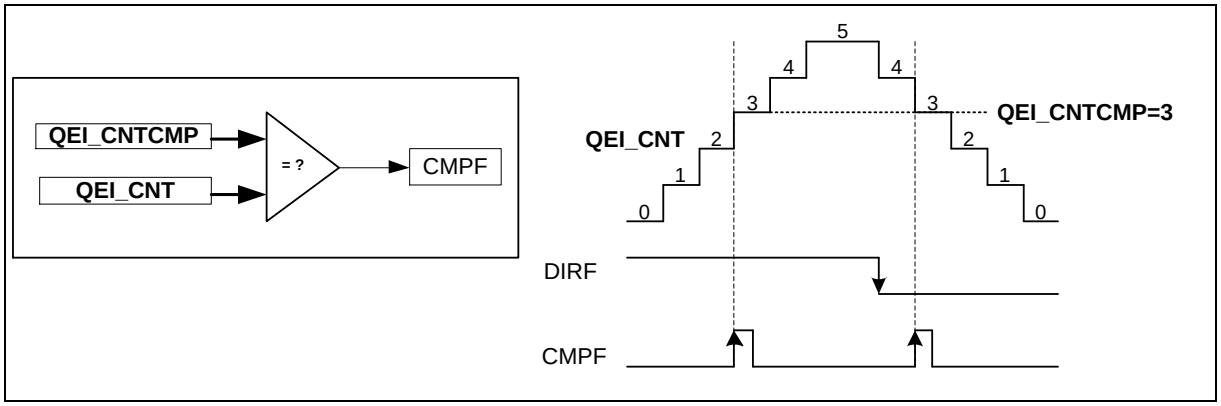


Figure 6.17-8 Compare Operation

6.17.5.10 Reload Counter by Pin IDX

The CNT(QEI_CNT[31:0]) counter can be reset to 0 or reload with the content of CNTMAX (QEI_CNTMAX[31:0]) by the signal CHX (the filtered and polarity-set output of pin IDX) trigger. When the IDX Reload bit IDXRLD_EN(QEI_CTL[27]) is set, a rising edge of CHX causes QEI controller to reset the CNT(QEI_CNT[31:0]) to 0 if the counter is in up-counting; if the counter is in down-counting the rising edge of CHX causes the QEI controller to reload the CNT(QEI_CNT[31:0]) with the content of CNTMAX (QEI_CNTMAX[31:0]). Refer to Figure 6.17-9 for details.

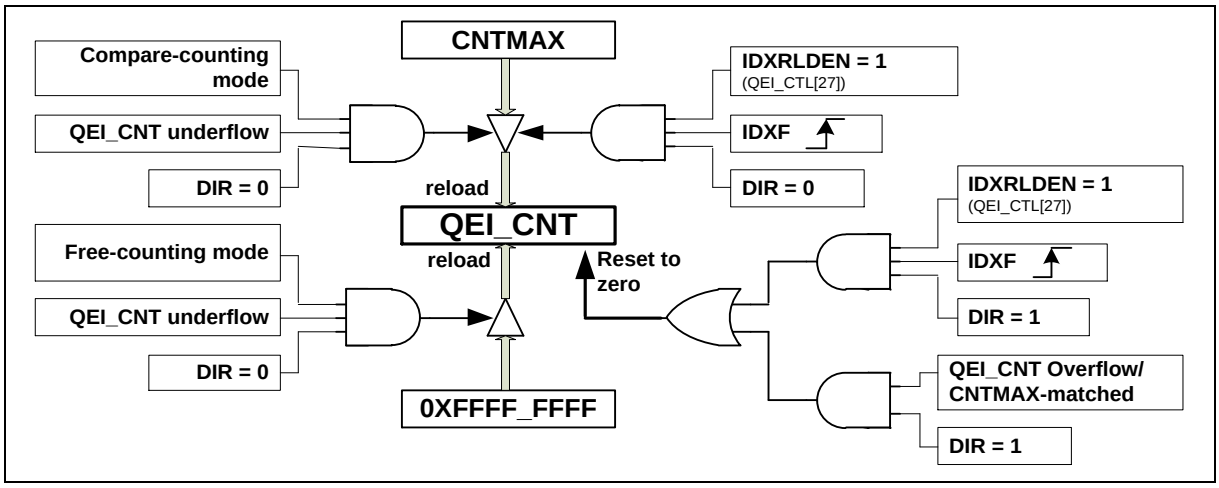


Figure 6.17-9 QEI_CNT Reload/Reset Control

6.17.5.11 Capture QEI Counter

If the bit HOLDCNT(QEI_CTL[24]) is set, the CNT(QEI_CNT[31:0]) content will be captured into QEI Counter Hold Register CNTHOLD(QEI_CNTHOLD[31:0]), the data will be held until the next HOLDCNT (QEI_CTL[24]) trigger comes. The bit HOLDCNT can be set by writing 1 to it or the rising edge of timers interrupt flags TIF (TIMERx_INTSTS[0]).

Note: The bit HOLDCNT is automatically cleared by hardware after CNTHOLD(QEI_CNTHOLD[31:0]) captures the content of QEI counter.

If the bit IDXLATEN (QEI_CTL[25]) is set, the CNT(QEI_CNT[31:0]) content will be latched into QEI Counter Index Latch Register CNTLATCH (QEI_CNTLATCH[31:0]) at every rising edge of CHX signal.

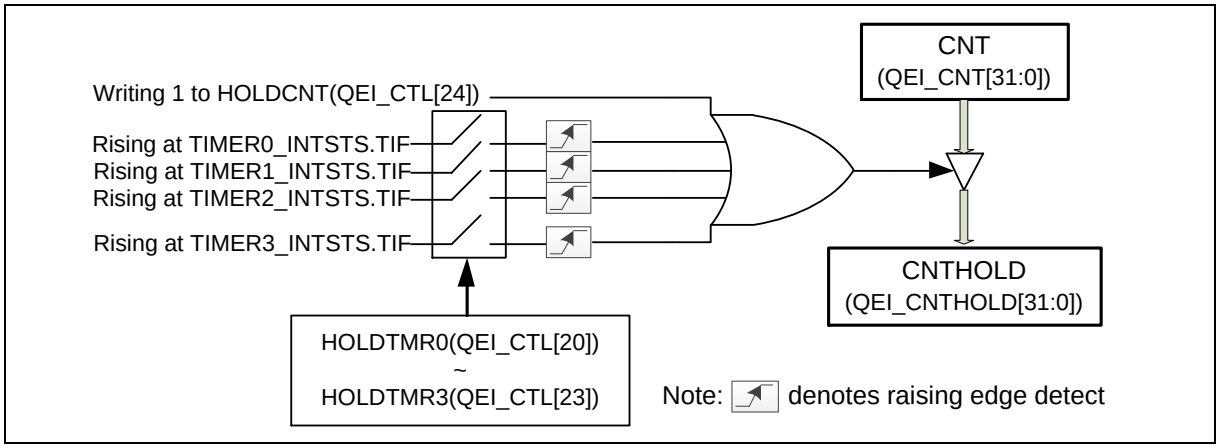


Figure 6.17-10 Trigger Control of Capturing QEI Counter

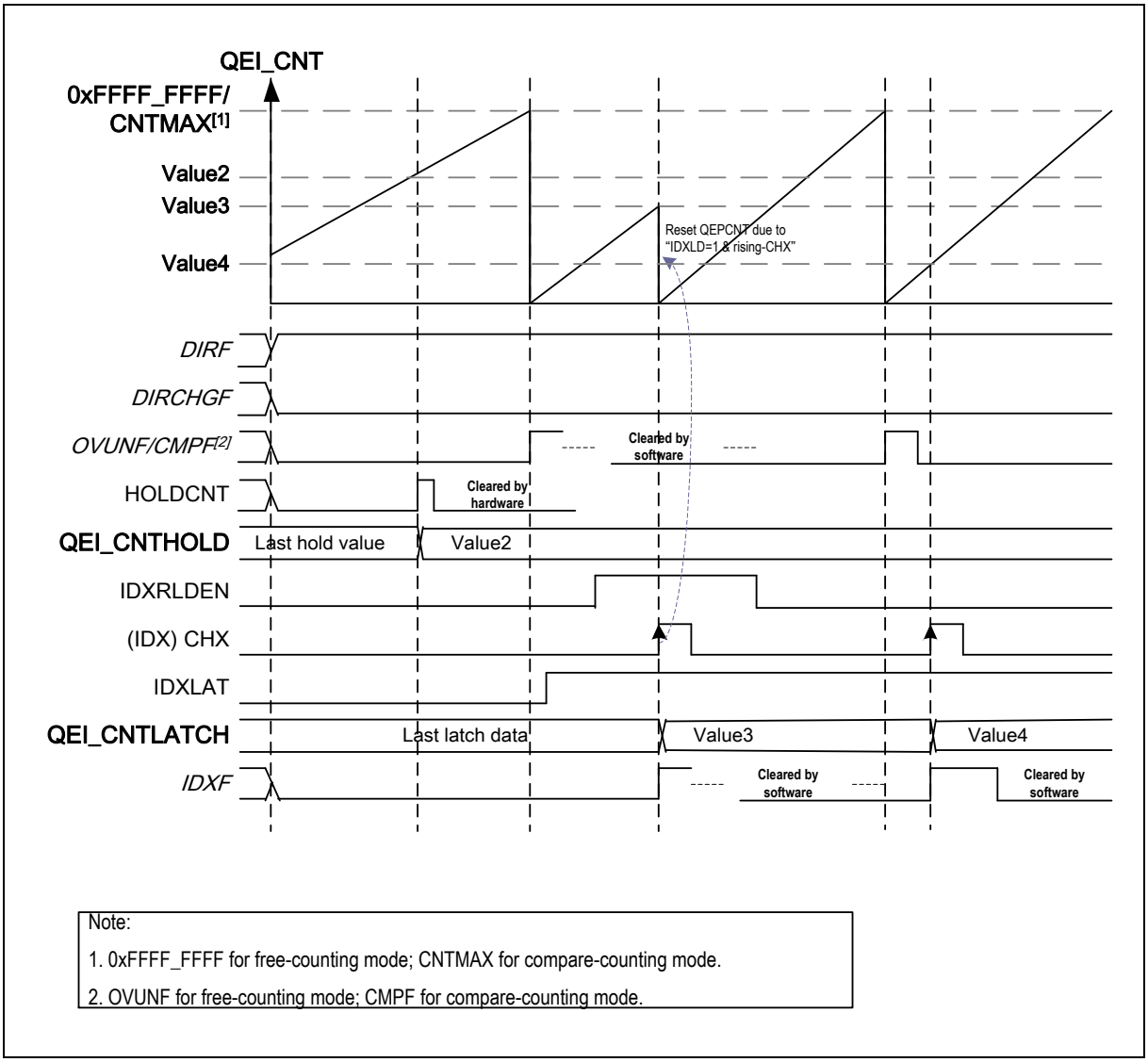


Figure 6.17-11 Capture and Latch QEI Counter

6.17.5.12 QEI Interrupt Architecture

There are four interrupt sources, in which each one has an interrupt flag and enable control bit to trigger QEI Interrupt. When the QEI counter is up-counting and CNT(QEI_CNT[31:0]) overflows or down-counting and underflows, the Overflow/Underflow flag OVUNF (QEI_STATUS[2]) will be set by hardware and it will trigger QEI Interrupt request if bit OVUNIEN (QEI_CTL[16]) is high. When QEI controller detects the encoder rotation change, it toggles the direction indication bit DIRF (QEI_STATUS[8]) and the direction change flag DIRCHGF (QEI_STATUS[3]) will be set by hardware that requests the QEI interrupt if bit DIRIEN (QEI_CTL[17]) is set. When the QEI counter counting value is equal to the value of QEI Counter Compare Register (CNTCMP(QEI_CNTCMP[31:0])), the flag CMPF (QEI_STATUS[1]) will be set by hardware and the QEI Interrupt will be requested if bit CMPIEN (QEI_CTL[18]) is high. When QEI controller detects a rising edge at signal CHX (the filtered and polarity-set output of pin IDX), the flag IDXF(QEI_STATUS[1]) will set by hardware and the QEI interrupt will be requested if bit IDXIEN (QEI_CTL[19]) is set. Note that the four flags, OVUNF(QEI_STATUS[2]), DIRCHGF(QEI_STATUS[3]), CMPF(QEI_STATUS[1]) and IDXF(QEI_STATUS[0]) are set by hardware and must be cleared by software. Figure 6.17-12

demonstrates the architecture of Quadrature Encoder Interface Controller interrupts.

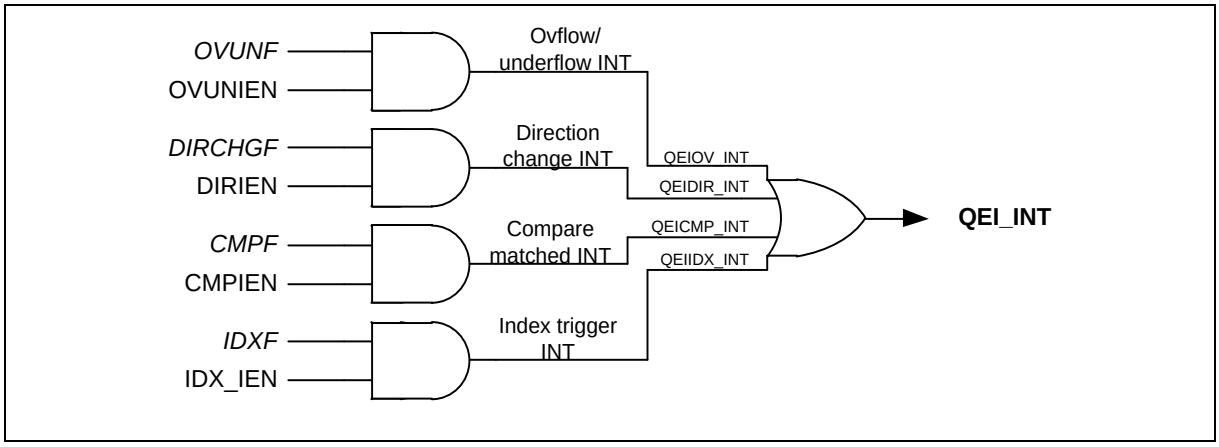


Figure 6.17-12 Quadrature Encoder Interface Interrupt Architecture Diagram

6.17.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
QEI Base Address: **QEI0_BA = 0x400B_0000** **QEI1_BA = 0x400B_1000**				
QEI_CNT **x=0, 1**	QEIx_BA+0x00	R/W	QEI Counter Register	0x0000_0000
QEI_CNTHOLD **x=0, 1**	QEIx_BA+0x04	R/W	QEI Counter Hold Register	0x0000_0000
QEI_CNTLATCH **x=0,1**	QEIx_BA+0x08	R/W	QEI Counter Index Latch Register	0x0000_0000
QEI_CNTCMP **x=0, 1**	QEIx_BA+0x0C	R/W	QEI Counter Compare Register	0x0000_0000
QEI_CNTMAX **x=0, 1**	QEIx_BA+0x14	R/W	QEI Pre-set Maximum Count Register	0x0000_0000
QEI_CTL **x=0, 1**	QEIx_BA+0x18	R/W	QEI Controller Control Register	0x0000_0000
QEI_STATUS **x=0, 1**	QEIx_BA+0x2C	R/W	QEI Controller Status Register	0x0000_0000

6.17.7 Register Description

QEI Counter Register (QEI_CNT)

Register	Offset	R/W	Description	Reset Value
QEI_CNT	QEIx_BA+0x00	R/W	QEI Counter Register	0x0000_0000

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

Bits	Description	
[31:0]	**CNT**	**Quadrature Encoder Interface Counter** A 32-bit up/down counter. When an effective phase pulse is detected, this counter is increased by one if the bit DIRF (QEI_STATUS[8]) is one or decreased by one if the bit DIRF(QEI_STATUS[8]) is 0. This register performs an integrator which count value is proportional to the encoder position. The pulse counter may be initialized to a predetermined value by one of three events occurs: ● Software is written if QEIEN (QEI_CTL[29]) = 0. ● Compare-match event if QEIEN(QEI_CTL[29])=1 and QEI is in compare-counting mode. ● Index signal change if QEIEN(QEI_CTL[29])=1 and IDXRLDEN (QEI_CTL[27])=1.

QEI Counter Hold Register (QEI_CNTHOLD)

Register	Offset	R/W	Description	Reset Value
QEI_CNTHOLD	QEIx_BA+0x04	R/W	QEI Counter Hold Register	0x0000_0000

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

Bits	Description	
[31:0]	CNTHOLD	**Quadrature Encoder Interface Counter Hold** When the bit HOLDCNT (QEI_CTL[24]) goes from low to high, the CNT(QEI_CNT[31:0]) is copied into CNTHOLD (QEI_CNTHOLD[31:0]) register.

QEI Counter Index Latch Register (QEI_CNTLATCH)

Register	Offset	R/W	Description	Reset Value
QEI_CNTLATCH	QEIx_BA+0x08	R/W	QEI Counter Index Latch Register	0x0000_0000

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

Bits	Description	
[31:0]	CNTLATCH	**Quadrature Encoder Interface Counter Index Latch** When the IDXF (QEI_STATUS[0]) bit is set, the CNT(QEI_CNT[31:0]) is copied into CNTLATCH (QEI_CNTLATCH[31:0]) register.

QEI Counter Compare Register (QEI_CNTCMP)

Register	Offset	R/W	Description	Reset Value
QEI_CNTCMP	QEIx_BA+0x0C	R/W	QEI Counter Compare Register	0x0000_0000

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

Bits	Description	
[31:0]	**CNTCMP**	**Quadrature Encoder Interface Counter Compare** If the QEI controller is in the compare-counting mode CMPEN (QEI_CTL[28]) =1, when the value of CNT(QEI_CNT[31:0]) matches CNTCMP(QEI_CNTCMP[31:0]), CMPF will be set. This register is software writable.

QEI Pre-set Maximum Count Register (QEI_CNTMAX)

Register	Offset	R/W	Description	Reset Value
QEI_CNTMAX	QEIx_BA+0x14	R/W	QEI Pre-set Maximum Count Register	0x0000_0000

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

Bits	Description	
[31:0]	**CNTMAX**	**Quadrature Encoder Interface Preset Maximum Count** This register value determined by user stores the maximum value which may be the number of the QEI counter for the QEI controller compare-counting mode.

QEI Controller Control Register (QEI_CTL)

Register	Offset	R/W	Description	Reset Value
QEI_CTL	QEIx_BA+0x18	R/W	QEI Controller Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		QEIEN	CMPEN	IDXRLDEN	Reserved	IDXLATEN	HOLDCNT
23	22	21	20	19	18	17	16
HOLDTMR3	HOLDTMR2	HOLDTMR1	HOLDTMR0	IDXIEN	CMPIEN	DIRIEN	OVUNIEN
15	14	13	12	11	10	9	8
Reserved	IDXINV	CHBINV	CHAINV	Reserved		MODE	
7	6	5	4	3	2	1	0
Reserved	IDXEN	CHBEN	CHAEN	NFDIS	NFCLKSEL		

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29]	**QEIEN**	**Quadrature Encoder Interface Controller Enable Bit** 0 = QEI controller function Disabled. 1 = QEI controller function Enabled.
[28]	**CMPEN**	**The Compare Function Enable Bit** The compare function in QEI controller is to compare the dynamic counting QEI_CNT with the compare register CNTCMP(QEI_CNTCMP[31:0]), if CNT(QEI_CNT[31:0]) reaches CNTCMP(QEI_CNTCMP[31:0]), the flag CMPF will be set. 0 = Compare function Disabled. 1 = Compare function Enabled.
[27]	**IDXRLDEN**	**Index Trigger QEI_CNT Reload Enable Bit** When this bit is high and a rising edge comes on signal CHX, the CNT(QEI_CNT[31:0]) will be reset to 0 if the counter is in up-counting type (DIRF(QEI_STATUS[8]) = 1); while the CNT(QEI_CNT[31:0]) will be reloaded with CNTMAX (QEI_CNTMAX[31:0]) content if the counter is in down-counting type (DIRF(QEI_STATUS[8]) = 0). 0 = Reload function Disabled. 1 = QEI_CNT re-initialized by Index signal Enabled.
[26]	**Reserved**	Reserved.
[25]	**IDXLATEN**	**Index Latch QEI_CNT Enable Bit** If this bit is set to high, the CNT(QEI_CNT[31:0]) content will be latched into CNTLATCH (QEI_CNTLATCH[31:0]) at every rising on signal CHX. 0 = The index signal latch QEI counter function Disabled. 1 = The index signal latch QEI counter function Enabled.

[24]	**HOLDCNT**	**Hold QEI_CNT Control** When this bit is set from low to high, the CNT(QEI_CNT[31:0]) is copied into CNTHOLD(QEI_CNTHOLD[31:0]). This bit may be set by writing 1 to it or Timer0~Timer3 interrupt flag TIF (TIMERx_INTSTS[0]). 0 = No operation. 1 = QEI_CNT content is captured and stored in CNTHOLD(QEI_CNTHOLD[31:0]). **Note:** This bit is automatically cleared after QEI_CNTHOLD holds QEI_CNT value.
[23]	**HOLDTMR3**	**Hold QEI_CNT by Timer 3** 0 = TIF (TIMER3_INTSTS[0]) has no effect on HOLDCNT. 1 = A rising edge of bit TIF(TIMER3_INTSTS[0]) in timer 3 sets HOLDCNT to 1.
[22]	**HOLDTMR2**	**Hold QEI_CNT by Timer 2** 0 = TIF(TIMER2_INTSTS[0]) has no effect on HOLDCNT. 1 = A rising edge of bit TIF(TIMER2_INTSTS[0]) in timer 2 sets HOLDCNT to 1.
[21]	**HOLDTMR1**	**Hold QEI_CNT by Timer 1** 0 = TIF(TIMER1_INTSTS[0]) has no effect on HOLDCNT. 1 = A rising edge of bit TIF (TIMER1_INTSTS[0]) in timer 1 sets HOLDCNT to 1.
[20]	**HOLDTMR0**	**Hold QEI_CNT by Timer 0** 0 = TIF (TIMER0_INTSTS[0]) has no effect on HOLDCNT. 1 = A rising edge of bit TIF(TIMER0_INTSTS[0]) in timer 0 sets HOLDCNT to 1.
[19]	**IDXIEN**	**IDXF Trigger QEI Interrupt Enable Bit** 0 = The IDXF can trigger QEI interrupt Disabled. 1 = The IDXF can trigger QEI interrupt Enabled.
[18]	**CMPIEN**	**CMPF Trigger QEI Interrupt Enable Bit** 0 = CMPF can trigger QEI controller interrupt Disabled. 1 = CMPF can trigger QEI controller interrupt Enabled.
[17]	**DIRIEN**	**DIRCHGF Trigger QEI Interrupt Enable Bit** 0 = DIRCHGF can trigger QEI controller interrupt Disabled. 1 = DIRCHGF can trigger QEI controller interrupt Enabled.
[16]	**OVUNIEN**	**OVUNF Trigger QEI Interrupt Enable Bit** 0 = OVUNF can trigger QEI controller interrupt Disabled. 1 = OVUNF can trigger QEI controller interrupt Enabled.
[15]	**Reserved**	Reserved.
[14]	**IDXINV**	**Inverse IDX Input Polarity** 0 = Not inverse IDX input polarity. 1 = IDX input polarity is inversed to QEI controller.
[13]	**CHBINV**	**Inverse QEB Input Polarity** 0 = Not inverse QEB input polarity. 1 = QEB input polarity is inversed to QEI controller.
[12]	**CHAINV**	**Inverse QEA Input Polarity** 0 = Not inverse QEA input polarity. 1 = QEA input polarity is inversed to QEI controller.
[11:10]	**Reserved**	Reserved.

[9:8]	**MODE**	**QEI Counting Mode Selection** There are four quadrature encoder pulse counter operation modes. 00 = X4 Free-counting Mode. 01 = X2 Free-counting Mode. 10 = X4 Compare-counting Mode. 11 = X2 Compare-counting Mode.
[7]	**Reserved**	Reserved.
[6]	**IDXEN**	**IDX Input to QEI Controller Enable Bit** 0 = IDX input to QEI Controller Disabled. 1 = IDX input to QEI Controller Enabled.
[5]	**CHBEN**	**QEB Input to QEI Controller Enable Bit** 0 = QEB input to QEI Controller Disabled. 1 = QEB input to QEI Controller Enabled.
[4]	**CHAEN**	**QEA Input to QEI Controller Enable Bit** 0 = QEA input to QEI Controller Disabled. 1 = QEA input to QEI Controller Enabled.
[3]	**NFDIS**	**QEI Controller Input Noise Filter Disable Bit** 0 = The noise filter of QEI controller Enabled. 1 = The noise filter of QEI controller Disabled.
[2:0]	**NFCLKSEL**	**Noise Filter Clock Pre-divide Selection** To determine the sampling frequency of the Noise Filter clock . 000 = QEI_CLK. 001 = QEI_CLK/2. 010 = QEI_CLK/4. 011 = QEI_CLK/16. 100 = QEI_CLK/32. 101 = QEI_CLK/64.

QEI Controller Status Register (QEI_STATUS)

Register	Offset	R/W	Description	Reset Value
QEI_STATUS	QEIx_BA+0x2C	R/W	QEI Controller Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							DIRF
7	6	5	4	3	2	1	0
Reserved				DIRCHGF	OVUNF	CMPF	IDXF

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**DIRF**	**QEI Counter Counting Direction Indication** 0 = QEI Counter is in down-counting. 1 = QEI Counter is in up-counting. **Note:** This bit is set/reset by hardware according to the phase detection between CHA and CHB.
[7:4]	**Reserved**	Reserved.
[3]	**DIRCHGF**	**Direction Change Flag** Flag is set by hardware while QEI counter counting direction is changed. Software can clear this bit by writing 1 to it. 0 = No change in QEI counter counting direction. 1 = QEI counter counting direction is changed. **Note:** This bit is only cleared by writing 1 to it.
[2]	**OVUNF**	**QEI Counter Overflow or Underflow Flag** Flag is set by hardware while CNT(QEI_CNT[31:0]) overflows from 0xFFFF_FFFF to 0 in free-counting mode or from the CNTMAX (QEI_CNTMAX[31:0]) to 0 in compare-counting mode. Similarly, the flag is set while QEI counter underflows from 0 to 0xFFFF_FFFF or CNTMAX (QEI_CNTMAX[31:0]). 0 = No overflow or underflow occurs in QEI counter. 1 = QEI counter occurs counting overflow or underflow. **Note:** This bit is only cleared by writing 1 to it.
[1]	**CMPF**	**Compare-match Flag** If the QEI compare function is enabled, the flag is set by hardware while QEI counter up or down counts and reach to the CNTCMP(QEI_CNTCMP[31:0]). 0 = QEI counter does not match with CNTCMP(QEI_CNTCMP[31:0]). 1 = QEI counter counts to the same as CNTCMP(QEI_CNTCMP[31:0]). **Note:** This bit is only cleared by writing 1 to it.

[0]	**IDXF**	**IDX Detected Flag** When the QEI controller detects a rising edge on signal CHX it will set flag IDXF to high. 0 = No rising edge detected on signal CHX. 1 = A rising edge occurs on signal CHX. **Note:** This bit is only cleared by writing 1 to it.

6.18 Enhanced Input Capture Timer (ECAP)

6.18.1 Overview

This device provides up to two units of Input Capture Timer/Counter whose capture function can detect the digital edge-changed signal at channel inputs. Each unit has three input capture channels. The timer/counter is equipped with up counting, reload and compare-match capabilities.

6.18.2 Features

- Up to two Input Capture Timer/Counter units, CAP0 and CAP1.
- Each unit has 3 input channels.
- Each unit has its own interrupt vector.
- Each input channel has its own capture counter hold register.
- 24-bit Input Capture up-counting timer/counter.
- With noise filter in front end of input ports.
- Edge detector with three options:
  - Rising edge detection
  - Falling edge detection
  - Both edge detection
- Captured events reset and/or reload capture counter.
- Supports compare-match function.

6.18.3 Block Diagram

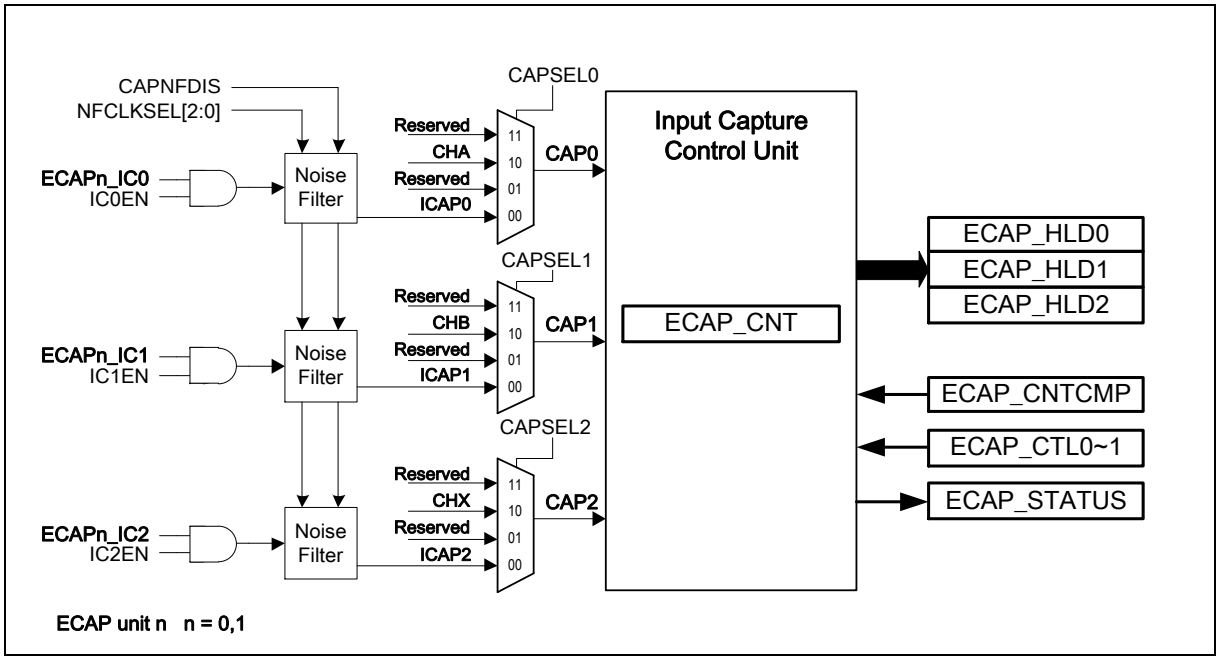


Figure 6.18-1 Input Capture Timer/Counter Architecture

6.18.4 Basic Configuration

6.18.4.1 ECAP0 Basic Configuration

- Clock Source Configuration
  - Enable ECAP0 peripheral clock in ECAP0CKEN(CLK_APBCLK1[26]).
- Reset Configuration
  - Reset ECAP0 peripheral in ECAP0RST (SYS_IPRST2[26]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
ECAP0	ECAP0_IC0	PA.10	MFP11
		PE.8	MFP12
	ECAP0_IC1	PA.9	MFP11
		PE.9	MFP12
	ECAP0_IC2	PA.8	MFP11
		PE.10	MFP12

6.18.4.2 ECAP1 Basic Configuration

- Clock Source Configuration
  - Enable ECAP1 peripheral clock in ECAP1CKEN (CLK_APBCLK1[27]).
- Reset Configuration
  - Reset ECAP1 peripheral in ECAP1RST (SYS_IPRST2[27]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
ECAP1	ECAP1_IC0	PC.10	MFP11
		PE.13	MFP13
	ECAP1_IC1	PC.11	MFP11
		PE.12	MFP13
	ECAP1_IC2	PC.12	MFP11
		PE.11	MFP13

6.18.5 Functional Description

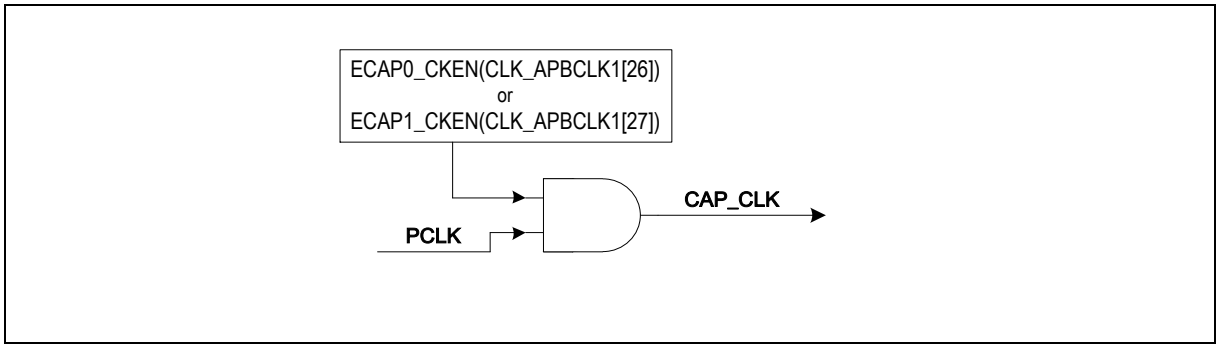


Figure 6.18-2 Input Capture Timer/Counter Clock Source Control

Figure 6.18-1 illustrates the architecture of the Input Capture. Each input capture timer/counter unit supports 3 input channels with programmable input signal sources. The port pins ECAP_IC0 to ECAP_IC2 can be fed to the inputs of capture unit through noise filter or bypass it (CAPNFDIS = 1), and the QEI controller input signals (CHA, CHB and CHX) can also be internally routed to the capture inputs by setting the register ECAP_CTL0 (CAPSEL0~ CAPSEL 2).

6.18.5.1 Input Noise Filter

The architecture of input noise filter is similar to that one used for QEI. Refer to the figure - Noise Filter - in QEI section. With 6 sampling-rate options, it supports a wide range of filtering noise whose duration is from 55 ns (NFCLKSEL = 000) to 3.5 us (NFCLKSEL = 101) while PCLK is running at the frequency of 55 MHz. Table 6.18-1 lists the relation between the setting of NFCLKSEL, the duration of filtered noise and the duration of the signal that is guaranteed to be sampled.

The maximum duration of the noise that is filtered out should not be greater than 3 clock cycles, while the minimum duration of the signal that is guaranteed to be sampled should be greater than 4 clock cycles.

NFCLKSEL	Maximum Duration Of Noise	Minimum Duration Of Signal
000	55 ns	73 ns
001	109 ns	145 ns
010	218 ns	291 ns
011	873 ns	1164 ns
100	1745 ns	2327 ns
101	3491 ns	4655 ns
PCLK= 55 MHz		

Table 6.18-1 Typical Case of Noise Filter Settings

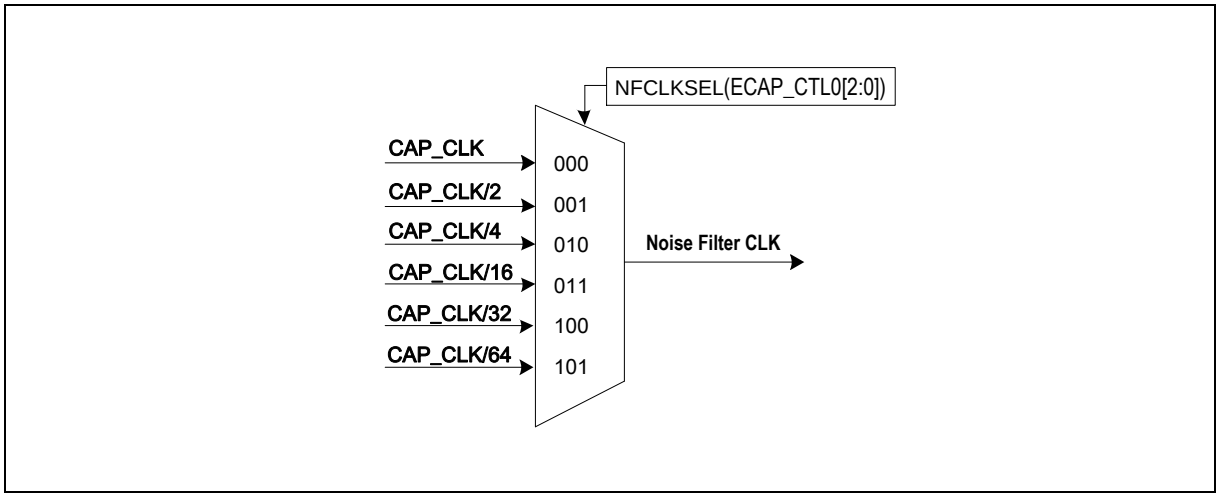


Figure 6.18-3 Noise Filter Sampling Clock Selection

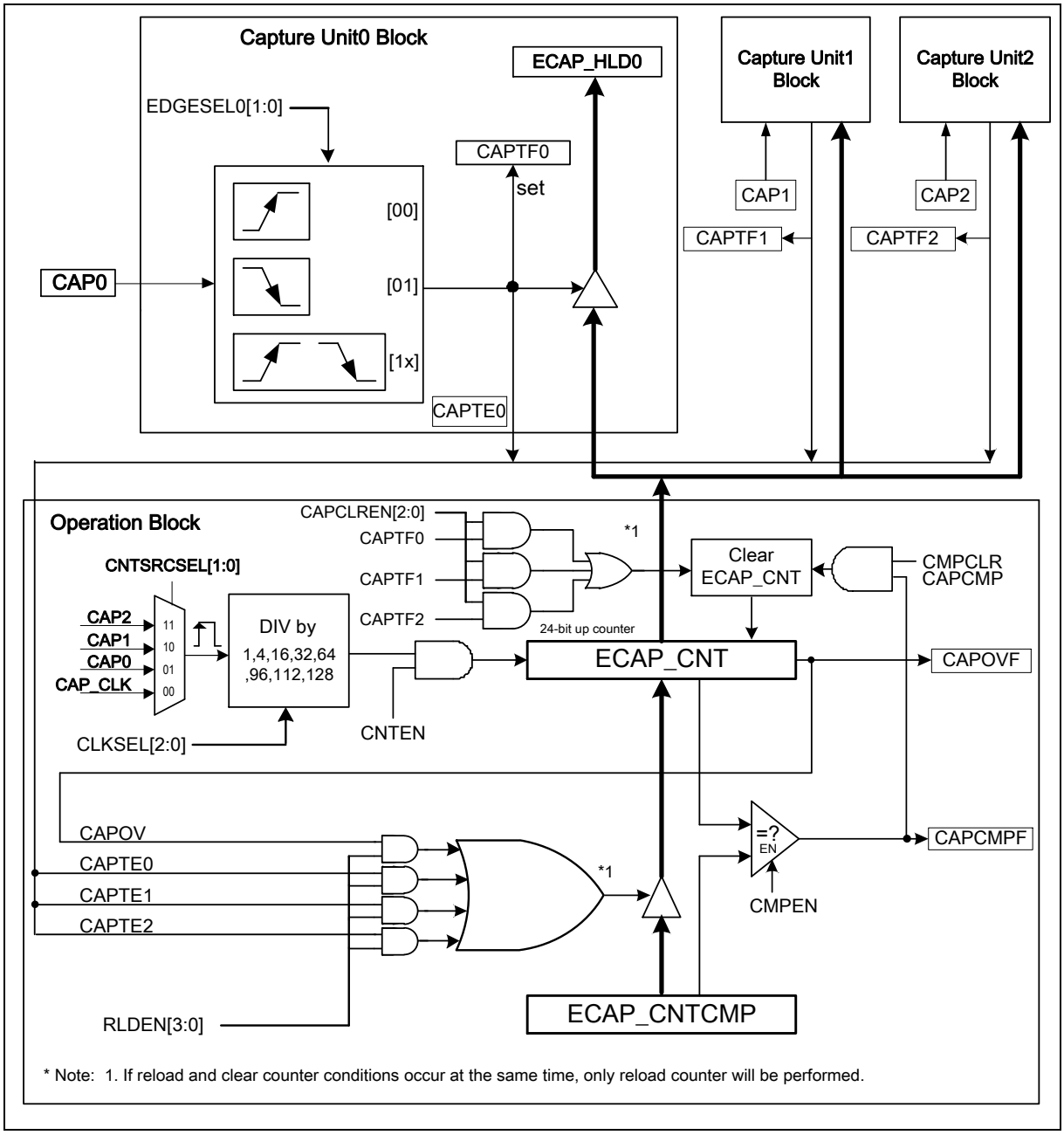


Figure 6.18-4 Input Capture Timer/Counter Function Block

6.18.5.2 Input Capture Timer/Counter Operation

An Input Capture Timer/Counter unit consists of 2 main functional blocks, Capture block and Operation block. There are 3 Input Capture units in Capture block for 3 input channel.

The capture units function as detecting and measuring the pulse width and the period of a square wave. The input channel 0 to 2 have their own edge detectors, which are in Input Capture block but share with one capture timer/counter, ECAP_CNT, which is in Operation block. The edge trigger option is programmable through EDGESEL (ECAP_CTL1[5,4], [3,2], [1,0]) register supporting positive edge, negative edge and both edge triggers. Each capture unit consists of an enable control bit,

IC0EN ~ IC2EN (ECAP_CTL0[6:4]) to enable/disable each input channel and a status bit CAP0 ~ CAP2 (ECAP_STATUS[10:8]) to let software monitor the current status of each channel.

The Input Capture supports reload mode and compare mode. For both mode, the capture counter (ECAP_CNT) serves as a 24-bit up-counting counter whose clock comes from the output of the clock divider and is gated with CNTEN, and the clock source of the clock divider, which can be set by CLKSEL[2:0] to divide clock by 1,4,16,32,64,96,112 and 128, is programmable (by setting CNTSRCSEL[1:0]) to be from system clock source, CAP_CLK or input channel CAP0 ~ CAP2. In reload mode, ECAP_CNTCMP serves as a reload register while in compare mode ECAP_CNTCMP serves as a compare register. The Input Capture Timer/Counter Enable bit (CAPEN) must be set to enable Input Capture Timer/Counter functions. More details of operation are described in the following.

Capture Function

Each time when the capture input detect a valid edge change, it triggers a valid capture event (CAPTE0~2) so that the content of the free running 24-bit capture counter ECAP_CNT will be captured/transferred into the capture hold registers, ECAP_HLD0~2 depending on which channel is triggered. This event also causes the corresponding flag CAPTFx (ECAP_STATUS[2:0]) to be set, which will generate an interrupt if the corresponding interrupt enable bit CAPIENx (ECAP_CTL0[18:16]) is set. Triggered Flags are set by hardware and should be cleared by software. Software can read the register ECAP_STATUS to get the status of flags and has to write 1 to the corresponding bit(s) of ECAP_STATUS to clear flag(s).

In addition, setting the CAPxCLREN(ECAP_CTL1[22:20]) will allow hardware to reset capture counter (ECAP_CNT) automatically whenever the event happens.

Compare Mode

The compare function is enabled by setting the CMPEN (ECAP_CTL0[28]) bit to 1, and ECAP_CNTCMP will serve as a compare register. As ECAP_CNT counting up, upon matching ECAP_CNTCMP value, the flag, CAPCMPF (ECAP_STATUS[4]), will be set, which will generate an interrupt, CMP_INT, if compare interrupt enable bit, CMPIEN (ECAP_CTL0[21]), is set.

Besides, setting the CMPCLR (ECAP_CTL0[25]) will allow hardware to make capture counter cleared to zero automatically after a compare-match event occurs.

Reload Mode

The Input Capture Timer/Counter can also be configured for reload mode. The reload function is enabled by setting the RLDEN[3:0] bits (ECAP_CTL1[11:8]) – OVRLDEN, CAPxRLDEN – to 1, and each bit enables a reload source.

In this mode, ECAP_CNTCMP serves as a reload register. If OVRLDEN is set, a reload event is generated and causes the content of the ECAP_CNTCMP register to be loaded into the ECAP_CNT register when ECAP_CNT overflows. Furthermore, CAPxRLDEN, x=0~2, are enable bits of making CAPTE2 ~ CAPTE0 as reload sources.

One thing should be noted is that if CAPxCLREN as well as CAPxRLDEN are set, when a valid trigger event (CAPTEx) occurs, only RELAOD function will be executed.

6.18.5.3 Input Capture Timer/Counter Interrupt Architecture

Figure 6.18-5 demonstrates the architecture of Input Capture Timer/Counter interrupt module. There are 5 interrupt sources (OVF_INT, CMP_INT, CAPTF0_INT~CAPTF2_INT), which are logical ‘OR’ together, in a input capture unit, and each one has an interrupt flag (CAPOVF, CAPCMPF, CAPTF0~CAPTF2), which can trigger Interrupt (ECAP_INT), as well as an the enable control bit (OVIEN, CMPIEN, CAPIEN0~CAPIEN2) to enable/disable the flag.

Note that all the interrupt flags are set by hardware and must be cleared by software by writing 1 to the bit (ECAP_STATUS[5:4],[2:0]) corresponding to the flag.

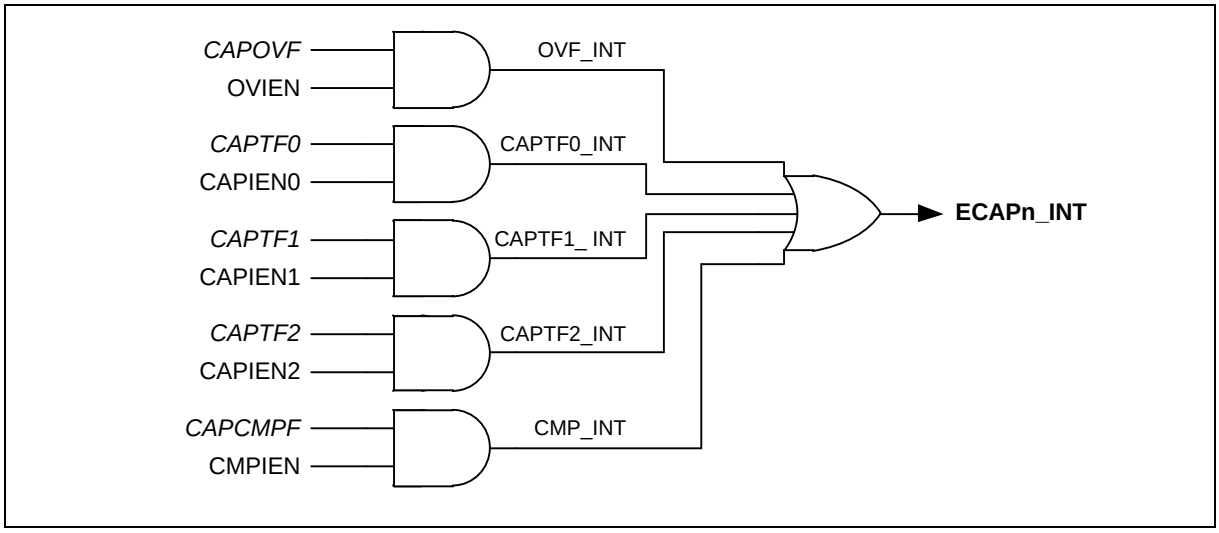


Figure 6.18-5 Input Capture Timer/Counter Interrupt Architecture Diagram

6.18.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
ECAP Base Address: **ECAPn_BA = 0x400B_4000 + (0x0000_1000*n)** **n=0, 1** **ECAP non-secure base address is ECAPn_BA + 0x1000_0000**				
ECAP_CNT	ECAPn_BA+0x00	R/W	Input Capture Counter (24-bit Up Counter)	0x0000_0000
ECAP_HLD0	ECAPn_BA+0x04	R/W	Input Capture Hold Register 0	0x0000_0000
ECAP_HLD1	ECAPn_BA+0x08	R/W	Input Capture Hold Register 1	0x0000_0000
ECAP_HLD2	ECAPn_BA+0x0C	R/W	Input Capture Hold Register 2	0x0000_0000
ECAP_CNTCMP	ECAPn_BA+0x10	R/W	Input Capture Compare Register	0x0000_0000
ECAP_CTL0	ECAPn_BA+0x14	R/W	Input Capture Control Register 0	0x0000_0000
ECAP_CTL1	ECAPn_BA+0x18	R/W	Input Capture Control Register 1	0x0000_0000
ECAP_STATUS	ECAPn_BA+0x1C	R/W	Input Capture Status Register	0x0000_0000

6.18.7 Register Description

Input Capture Counter (ECAP_CNT)

Register	Offset	R/W	Description	Reset Value
ECAP_CNT	ECAPn_BA+0x00	R/W	Input Capture Counter (24-bit Up Counter)	0x0000_0000

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

Bits	Description	
[31:24]	Reserved	Reserved.
[23:0]	CNT	**Input Capture Timer/Counter** The input Capture Timer/Counter is a 24-bit up-counting counter. The clock source for the counter is from the clock divider.

Input Capture Counter Hold Register (ECAP_HLD0~2)

Register	Offset	R/W	Description	Reset Value
ECAP_HLD0	ECAPn_BA+0x04	R/W	Input Capture Hold Register 0	0x0000_0000
ECAP_HLD1	ECAPn_BA+0x08	R/W	Input Capture Hold Register 1	0x0000_0000
ECAP_HLD2	ECAPn_BA+0x0C	R/W	Input Capture Hold Register 2	0x0000_0000

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

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**HOLD**	**Input Capture Counter Hold Register** When an active input capture channel detects a valid edge signal change, the ECAPCNT value is latched into the corresponding holding register. Each input channel has its own holding register named by ECAP_HLDx where x is from 0 to 2 to indicate inputs from IC0 to IC2, respectively.

Input Capture Counter Compare Register (ECAP_CNTCMP)

Register	Offset	R/W	Description	Reset Value
ECAP_CNTCMP	ECAPn_BA+0x10	R/W	Input Capture Compare Register	0x0000_0000

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

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**CNTCMP**	**Input Capture Counter Compare Register** If the compare function is enabled (CMPEN = 1), this register (ECAP_CNTCMP) is used to compare with the capture counter (ECAP_CNT). If the reload control is enabled (RLDEN[n] = 1, n=0~3), an overflow event or capture events will trigger the hardware to load the value of this register (ECAP_CNTCMP) into ECAP_CNT.

Input Capture Timer/Counter Control Register (ECAP_CTL0)

Register	Offset	R/W	Description	Reset Value
ECAP_CTL0	ECAPn_BA+0x14	R/W	Input Capture Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		CAPEN	CMPEN	Reserved		CMPCLREN	CNTEN
23	22	21	20	19	18	17	16
Reserved		CMPIEN	OVIEN	Reserved	CAPIEN2	CAPIEN1	CAPIEN0
15	14	13	12	11	10	9	8
Reserved		CAPSEL2		CAPSEL1		CAPSEL0	
7	6	5	4	3	2	1	0
Reserved	IC2EN	IC1EN	IC0EN	CAPNFDIS	NFCLKSEL		

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29]	**CAPEN**	**Input Capture Timer/Counter Enable Bit** 0 = Input Capture function Disabled. 1 = Input Capture function Enabled.
[28]	**CMPEN**	**Compare Function Enable Bit** The compare function in input capture timer/counter is to compare the dynamic counting ECAP_CNT with the compare register ECAP_CNTCMP, if ECAP_CNT value reaches ECAP_CNTCMP, the flag CAPCMPF will be set. 0 = The compare function Disabled. 1 = The compare function Enabled.
[27:26]	**Reserved**	Reserved.
[25]	**CMPCLREN**	**Input Capture Counter Cleared by Compare-match Control** If this bit is set to 1, the capture counter (ECAP_CNT) will be cleared to 0 when the compare-match event (CAPCMPF = 1) occurs. 0 = Compare-match event (CAPCMPF) can clear capture counter (ECAP_CNT) Disabled. 1 = Compare-match event (CAPCMPF) can clear capture counter (ECAP_CNT) Enabled.
[24]	**CNTEN**	**Input Capture Counter Start Counting Control** Setting this bit to 1, the capture counter (ECAP_CNT) starts up-counting the clock. 0 = ECAP_CNT stops counting. 1 = ECAP_CNT starts up-counting.
[23:22]	**Reserved**	Reserved.
[21]	**CMPIEN**	**CAPCMPF Trigger Input Capture Interrupt Enable Bit** 0 = The flag CAPCMPF can trigger Input Capture interrupt Disabled. 1 = The flag CAPCMPF can trigger Input Capture interrupt Enabled.

Bits	Description	
[20]	**OVIEN**	**CAPOVF Trigger Input Capture Interrupt Enable Bit** 0 = The flag CAPOVF can trigger Input Capture interrupt Disabled. 1 = The flag CAPOVF can trigger Input Capture interrupt Enabled.
[19]	**Reserved**	Reserved.
[18]	**CAPIEN2**	**Input Capture Channel 2 Interrupt Enable Control** 0 = The flag CAPTF2 can trigger Input Capture interrupt Disabled. 1 = The flag CAPTF2 can trigger Input Capture interrupt Enabled.
[17]	**CAPIEN1**	**Input Capture Channel 1 Interrupt Enable Control** 0 = The flag CAPTF1 can trigger Input Capture interrupt Disabled. 1 = The flag CAPTF1 can trigger Input Capture interrupt Enabled.
[16]	**CAPIEN0**	**Input Capture Channel 0 Interrupt Enable Control** 0 = The flag CAPTF0 can trigger Input Capture interrupt Disabled. 1 = The flag CAPTF0 can trigger Input Capture interrupt Enabled.
[15:14]	**Reserved**	Reserved.
[13:12]	**CAPSEL2**	**CAP2 Input Source Selection** 00 = CAP2 input is from port pin ICAP2. 01 = Reserved. 10 = CAP2 input is from signal CHX of QEI controller unit n. 11 = Reserved. **Note:** Input capture unit n matches QEIn, where n = 0~1.
[11:10]	**CAPSEL1**	**CAP1 Input Source Selection** 00 = CAP1 input is from port pin ICAP1. 01 = Reserved. 10 = CAP1 input is from signal CHB of QEI controller unit n. 11 = Reserved. **Note:** Input capture unit n matches QEIn, where n = 0~1.
[9:8]	**CAPSEL0**	**CAP0 Input Source Selection** 00 = CAP0 input is from port pin ICAP0. 01 = Reserved. 10 = CAP0 input is from signal CHA of QEI controller unit n. 11 = Reserved. **Note:** Input capture unit n matches QEIn, where n = 0~1.
[7]	**Reserved**	Reserved.
[6]	**IC2EN**	**Port Pin IC2 Input to Input Capture Unit Enable Control** 0 = IC2 input to Input Capture Unit Disabled. 1 = IC2 input to Input Capture Unit Enabled.
[5]	**IC1EN**	**Port Pin IC1 Input to Input Capture Unit Enable Control** 0 = IC1 input to Input Capture Unit Disabled. 1 = IC1 input to Input Capture Unit Enabled.

Bits	Description	
[4]	**IC0EN**	**Port Pin IC0 Input to Input Capture Unit Enable Control** 0 = IC0 input to Input Capture Unit Disabled. 1 = IC0 input to Input Capture Unit Enabled.
[3]	**CAPNFDIS**	**Input Capture Noise Filter Disable Control** 0 = Noise filter of Input Capture Enabled. 1 = Noise filter of Input Capture Disabled (Bypass).
[2:0]	**NFCLKSEL**	**Noise Filter Clock Pre-divide Selection** To determine the sampling frequency of the Noise Filter clock 000 = CAP_CLK. 001 = CAP_CLK/2. 010 = CAP_CLK/4. 011 = CAP_CLK/16. 100 = CAP_CLK/32. 101 = CAP_CLK/64.

Input Capture Timer/Counter Control Register (ECAP_CTL1)

Register	Offset	R/W	Description	Reset Value
ECAP_CTL1	ECAPn_BA+0x18	R/W	Input Capture Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved	CAP2CLREN	CAP1CLREN	CAP0CLREN	Reserved		CNTSRCSEL	
15	14	13	12	11	10	9	8
Reserved	CLKSEL			OVRLDEN	CAP2RLDEN	CAP1RLDEN	CAP0RLDEN
7	6	5	4	3	2	1	0
Reserved		EDGESEL2		EDGESEL1		EDGESEL0	

Bits	Description	
[31:23]	**Reserved**	Reserved.
[22]	**CAP2CLREN**	**Capture Counter Cleared by Capture Event2 Control** 0 = Event CAPTE2 can clear capture counter (ECAP_CNT) Disabled. 1 = Event CAPTE2 can clear capture counter (ECAP_CNT) Enabled.
[21]	**CAP1CLREN**	**Capture Counter Cleared by Capture Event1 Control** 0 = Event CAPTE1 can clear capture counter (ECAP_CNT) Disabled. 1 = Event CAPTE1 can clear capture counter (ECAP_CNT) Enabled.
[20]	**CAP0CLREN**	**Capture Counter Cleared by Capture Event0 Control** 0 = Event CAPTE0 can clear capture counter (ECAP_CNT) Disabled. 1 = Event CAPTE0 can clear capture counter (ECAP_CNT) Enabled.
[19:18]	**Reserved**	Reserved.
[17:16]	**CNTSRCSEL**	**Capture Timer/Counter Clock Source Selection** Select the capture timer/counter clock source. 00 = CAP_CLK (default). 01 = CAP0. 10 = CAP1. 11 = CAP2.
[15]	**Reserved**	Reserved.

Bits	Description	
[14:12]	**CLKSEL**	**Capture Timer Clock Divide Selection** The capture timer clock has a pre-divider with eight divided options controlled by CLKSEL[2:0]. 000 = CAP_CLK/1. 001 = CAP_CLK/4. 010 = CAP_CLK/16. 011 = CAP_CLK/32. 100 = CAP_CLK/64. 101 = CAP_CLK/96. 110 = CAP_CLK/112. 111 = CAP_CLK/128.
[11]	**OVRLDEN**	**Capture Counter's Reload Function Triggered by Overflow Enable Bit** 0 = The reload triggered by CAPOV Disabled. 1 = The reload triggered by CAPOV Enabled.
[10]	**CAP2RLDEN**	**Capture Counter's Reload Function Triggered by Event CAPTE2 Enable Bit** 0 = The reload triggered by Event CAPTE2 Disabled. 1 = The reload triggered by Event CAPTE2 Enabled.
[9]	**CAP1RLDEN**	**Capture Counter's Reload Function Triggered by Event CAPTE1 Enable Bit** 0 = The reload triggered by Event CAPTE1 Disabled. 1 = The reload triggered by Event CAPTE1 Enabled.
[8]	**CAP0RLDEN**	**Capture Counter's Reload Function Triggered by Event CAPTE0 Enable Bit** 0 = The reload triggered by Event CAPTE0 Disabled. 1 = The reload triggered by Event CAPTE0 Enabled.
[7:6]	**Reserved**	Reserved.
[5:4]	**EDGESEL2**	**Channel 2 Captured Edge Selection** Input capture2 can detect falling edge change only, rising edge change only or both edge changes 00 = Detect rising edge only. 01 = Detect falling edge only. 1x = Detect both rising and falling edge.
[3:2]	**EDGESEL1**	**Channel 1 Captured Edge Selection** Input capture1 can detect falling edge change only, rising edge change only or both edge change 00 = Detect rising edge only. 01 = Detect falling edge only. 1x = Detect both rising and falling edge.
[1:0]	**EDGESEL0**	**Channel 0 Captured Edge Selection** Input capture0 can detect falling edge change only, rising edge change only or both edge change 00 = Detect rising edge only. 01 = Detect falling edge only. 1x = Detect both rising and falling edge.

Input Capture Timer/Counter Status Register (ECAP_STATUS)

Register	Offset	R/W	Description	Reset Value
ECAP_STATUS	ECAPn_BA+0x1C	R/W	Input Capture Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					CAP2	CAP1	CAP0
7	6	5	4	3	2	1	0
Reserved		CAPOVF	CAPCMPF	Reserved	CAPTF2	CAPTF1	CAPTF0

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10]	**CAP2**	**Value of Input Channel 2, CAP2 (Read Only)** Reflecting the value of input channel 2, CAP2. **Note:** The bit is read only and write is ignored.
[9]	**CAP1**	**Value of Input Channel 1, CAP1 (Read Only)** Reflecting the value of input channel 1, CAP1 **Note:** The bit is read only and write is ignored.
[8]	**CAP0**	**Value of Input Channel 0, CAP0 (Read Only)** Reflecting the value of input channel 0, CAP0 **Note:** The bit is read only and write is ignored.
[7:6]	**Reserved**	Reserved.
[5]	**CAPOVF**	**Input Capture Counter Overflow Flag** Flag is set by hardware when counter (ECAP_CNT) overflows from 0x00FF_FFFF to zero. 0 = No overflow event has occurred since last clear. 1 = Overflow event(s) has/have occurred since last clear. **Note:** This bit is only cleared by writing 1 to it.
[4]	**CAPCMPF**	**Input Capture Compare-match Flag** If the input capture compare function is enabled, the flag is set by hardware when capture counter (ECAP_CNT) up counts and reaches the ECAP_CNTCMP value. 0 = ECAP_CNT has not matched ECAP_CNTCMP value since last clear. 1 = ECAP_CNT has matched ECAP_CNTCMP value at least once since last clear. **Note:** This bit is only cleared by writing 1 to it.
[3]	**Reserved**	Reserved.

Bits	Description	
[2]	**CAPTF2**	**Input Capture Channel 2 Triggered Flag** When the input capture channel 2 detects a valid edge change at CAP2 input, it will set flag CAPTF2 to high. 0 = No valid edge change has been detected at CAP2 input since last clear. 1 = At least a valid edge change has been detected at CAP2 input since last clear. **Note:** This bit is only cleared by writing 1 to it.
[1]	**CAPTF1**	**Input Capture Channel 1 Triggered Flag** When the input capture channel 1 detects a valid edge change at CAP1 input, it will set flag CAPTF1 to high. 0 = No valid edge change has been detected at CAP1 input since last clear. 1 = At least a valid edge change has been detected at CAP1 input since last clear. **Note:** This bit is only cleared by writing 1 to it.
[0]	**CAPTF0**	**Input Capture Channel 0 Triggered Flag** When the input capture channel 0 detects a valid edge change at CAP0 input, it will set flag CAPTF0 to high. 0 = No valid edge change has been detected at CAP0 input since last clear. 1 = At least a valid edge change has been detected at CAP0 input since last clear. **Note:** This bit is only cleared by writing 1 to it.

6.19 UART Interface Controller (UART)

6.19.1 Overview

The chip provides six channels of Universal Asynchronous Receiver/Transmitters (UART). The UART controller performs Normal Speed UART and supports flow control function. The UART controller performs a serial-to-parallel conversion on data received from the peripheral and a parallel-to-serial conversion on data transmitted from the CPU. Each UART controller channel supports ten types of interrupts. The UART controller also supports IrDA SIR, LIN and RS-485 function modes and auto-baud rate measuring function.

6.19.2 Features

- Full-duplex asynchronous communications
- Separates receive and transmit 16/16 bytes entry FIFO for data payloads
- Supports hardware auto-flow control
- Programmable receiver buffer trigger level
- Supports programmable baud rate generator for each channel individually
- Supports nCTS, incoming data, Received Data FIFO reached threshold and RS-485 Address Match (AAD mode) wake-up function
- Supports 8-bit receiver buffer time-out detection function
- Programmable transmitting data delay time between the last stop and the next start bit by setting DLY (UART_TOUT [15:8])
- Supports Auto-Baud Rate measurement and baud rate compensation function
  - Support 9600 bps for UART_CLK is selected LXT.
- Supports break error, frame error, parity error and receive/transmit buffer overflow detection function
- Fully programmable serial-interface characteristics
  - Programmable number of data bit, 5-, 6-, 7-, 8- bit character
  - Programmable parity bit, even, odd, no parity or stick parity bit generation and detection
  - Programmable stop bit, 1, 1.5, or 2 stop bit generation
- Supports IrDA SIR function mode
  - Supports for 3/16 bit duration for normal mode
- Supports LIN function mode (Only UART0 /UART1 with LIN function)
  - Supports LIN master/slave mode
  - Supports programmable break generation function for transmitter
  - Supports break detection function for receiver
- Supports RS-485 function mode
  - Supports RS-485 9-bit mode
  - Supports hardware or software enables to program nRTS pin to control RS-485 transmission direction
- Supports PDMA transfer function

- Support Single-wire function mode.

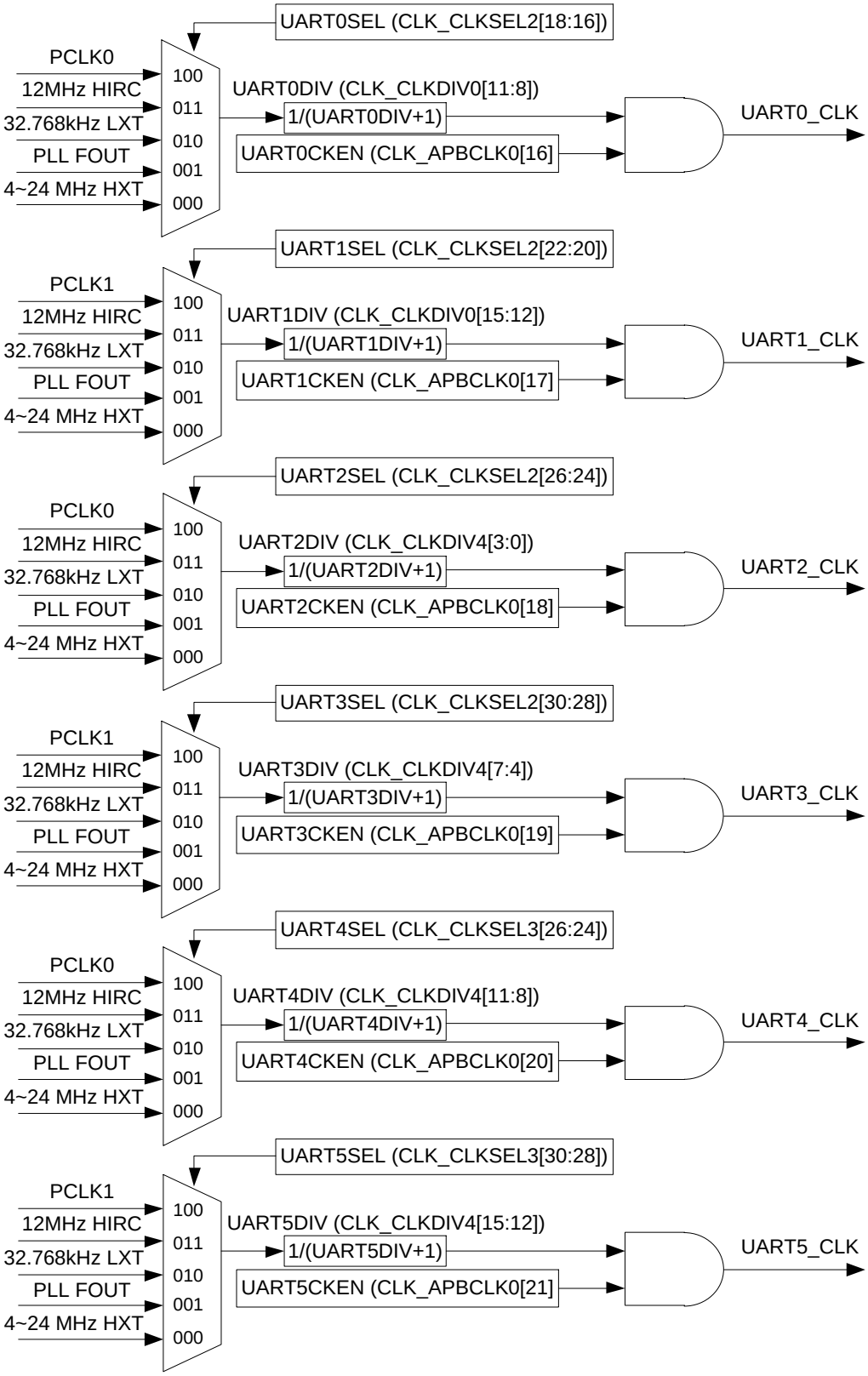
UART Feature	UART0/ UART1	UART2/UART3/ UART4/ UART5	SC_UART	USCI-UART
FIFO	16 Bytes	16 Bytes	4 Bytes	TX: 1byte RX: 2byte
Auto Flow Control (CTS/RTS)	√	√	-	√
IrDA	√	√	-	-
LIN	√	-	-	-
RS-485 Function Mode	√	√	-	√
nCTS Wake-up	√	√	-	√
Incoming Data Wake-up	√	√	-	√
Received Data FIFO reached threshold Wake-up	√	√	-	-
RS-485 Address Match (AAD mode) Wake-up	√	√	-	-
Auto-Baud Rate Measurement	√	√	-	√
STOP Bit Length	1, 1.5, 2 bit	1, 1.5, 2 bit	1, 2 bit	1, 2 bit
Word Length	5, 6, 7, 8 bits	5, 6, 7, 8 bits	5, 6, 7, 8 bits	6~13 bits
Even / Odd Parity	√	√	√	√
Stick Bit	√	√	-	-
Note: √= Supported				

Table 6.19-1 NuMicro® M2354 Series UART Features

6.19.3 Block Diagram

The UART clock control and block diagram are shown in Figure 6.19-1 and Figure 6.19-2 respectively.

Note: The frequency of UARTx_CLK should not be greater than 30 times HCLK.



Note: Before clock switching, both the pre-selected and newly selected clock sources must be turned on and stable.

Figure 6.19-1 UART Clock Control Diagram

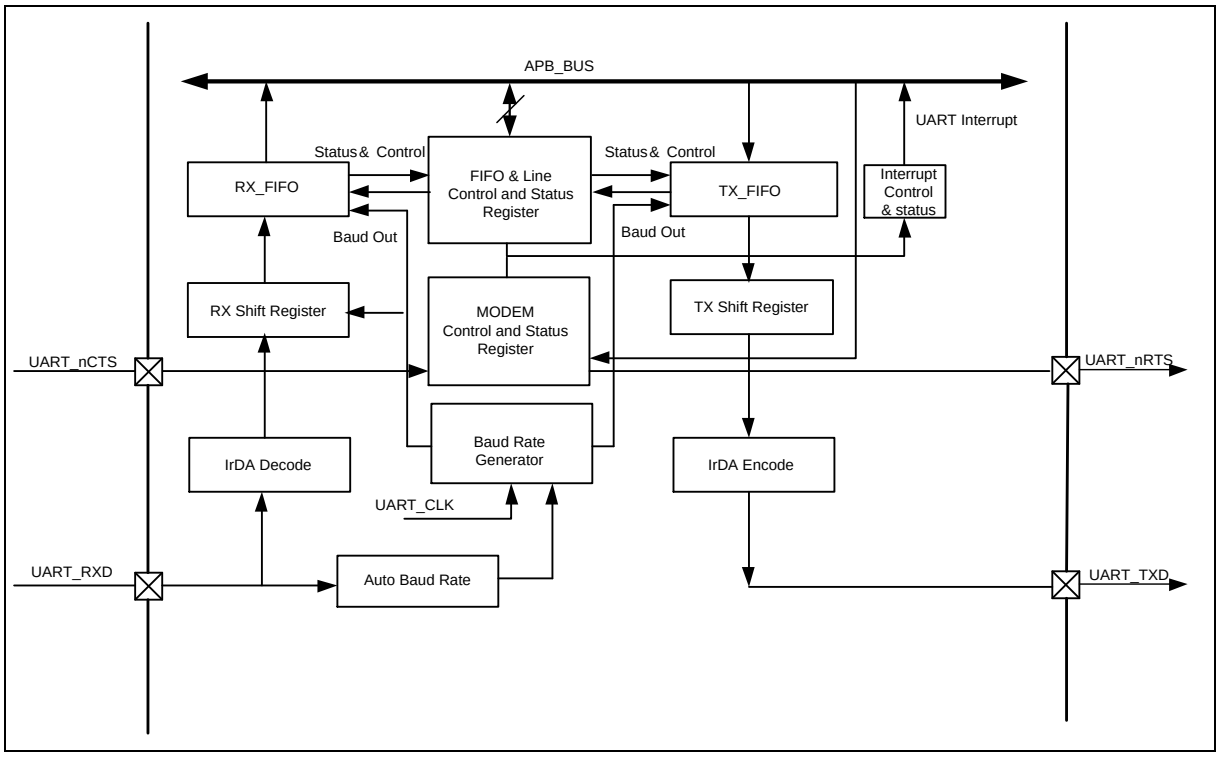


Figure 6.19-2 UART Block Diagram

Each block is described in detail as follows:

TX_FIFO

The transmitter is buffered with a 16 bytes FIFO to reduce the number of interrupts presented to the CPU.

RX_FIFO

The receiver is buffered with a 16 bytes FIFO (plus three error bits, BIF (UART_FIFOSTS[6]), FEF (UART_FIFOSTS[5]), PEF (UART_FIFOSTS[4])) to reduce the number of interrupts presented to the CPU.

TX Shift Register

This block is responsible for shifting out the transmitting data serially.

RX Shift Register

This block is responsible for shifting in the receiving data serially.

Modem Control and Status Register

This register controls the interface to the MODEM or data set (or a peripheral device emulating a MODEM).

Baud Rate Generator

Divide the external clock by the divisor to get the desired baud rate clock. Refer to baud rate equation.

IrDA Encode

This block is IrDA encoding control block.

IrDA Decode

This block is IrDA decoding control block.

FIFO & Line Control and Status Register

This field is register set that including the FIFO control register (UART_FIFO), FIFO status register (UART_FIFOSTS), and line control register (UART_LINE) for transmitter and receiver. The time-out register (UART_TOUT) identifies the condition of time-out interrupt.

Auto-Baud Rate Measurement

This block is responsible for auto-baud rate measurement.

Interrupt Control and Status Register

There are ten types of interrupts, Receive Data Available Interrupt (RDAINT), Transmit Holding Register Empty Interrupt (THERINT), Transmitter Empty Interrupt (TXENDINT), Receive Line Status Interrupt (parity error or framing error or break interrupt) (RLSINT), MODEM Status Interrupt (MODEMINT), Receiver Buffer Time-out Interrupt (RXTOINT), Buffer Error Interrupt (BUFERRINT), LIN Bus Interrupt (LININT), Wake-up Interrupt (WKINT) and Auto-Baud Rate Interrupt (ABRINT). Interrupt enable register (UART_INTEN) enable or disable the responding interrupt and interrupt status register (UART_INTSTS) identifying the occurrence of the responding interrupt.

Interrupt	Description
RDAINT	Receive Data Available Interrupt.
THERINT	Transmit Holding Register Empty Interrupt.
TXENDINT	Transmitter Empty Interrupt.
RLSINT	Receive Line Status Interrupt (parity error or frame error or break error).
MODEMINT	MODEM Status Interrupt.
RXTOINT	Receiver Buffer Time-out Interrupt.
BUFERRINT	Buffer Error Interrupt.
LININT	LIN Bus Interrupt.
WKINT	Wake-up Interrupt.
ABRINT	Auto-Baud Rate Interrupt.
SWBEINT	Single-wire Bit Error Detect Interrupt.

Table 6.19-2 UART Interrupt

6.19.4 Basic Configuration

The basic configurations of UART0 are as follows:

- Clock Source Configuration
  - Select the source of UART0 peripheral clock on UART0SEL (CLK_CLKSEL2[18:16]).
  - Select the clock divider number of UART0 peripheral clock on UART0DIV (CLK_CLKDIV0[11:8]).
  - Enable UART0 peripheral clock in UART0CKEN (CLK_APBCLK0[16]).
- Reset UART0 controller in UART0RST (SYS_IPRST1[16]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
UART0	UART0_RXD	PA.15, PC.11, PF.2	MFP3
		PF.1	MFP4
		PB.8	MFP5
		PB.12	MFP6
		PA.0, PA.6	MFP7
		PH.11	MFP8
		PD.2	MFP9
		PA.4	MFP11
	UART0_TXD	PA.14, PC.12, PF.3	MFP3
		PF.0	MFP4
		PB.9	MFP5
		PB.13	MFP6
		PA.1, PA.7	MFP7
		PH.10	MFP8
		PD.3	MFP9
		PA.5	MFP11
	UART0_nCTS	PB.11	MFP5
		PB.15	MFP6
		PA.5, PC.7	MFP7
	UART0_nRTS	PB.10	MFP5
		PB.14	MFP6
		PA.4, PC.6	MFP7

The basic configurations of UART1 are as follows:

- Clock Source Configuration
  - Select the source of UART1 peripheral clock on UART1SEL (CLK_CLKSEL2[22:20]).
  - Select the clock divider number of UART1 peripheral clock on UART1DIV (CLK_CLKDIV0[15:12]).
  - Enable UART1 peripheral clock in UART1CKEN (CLK_APBCLK0[17]).
- Reset UART1 controller in UART1RST (SYS_IPRST1[17]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
UART1	UART1_RXD	PF.1	MFP2
		PD.6, PD.10	MFP3

		PB.2, PB.6	MFP6
		PA.8	MFP7
		PA.2, PC.8	MFP8
		PH.9	MFP10
	UART1_TXD	PF.0	MFP2
		PD.7, PD.11	MFP3
		PB.3, PB.7	MFP6
		PA.9	MFP7
		PA.3, PE.13	MFP8
		PH.8	MFP10
	UART1_nCTS	PB.9	MFP6
		PA.1, PE.11	MFP8
	UART1_nRTS	PB.8	MFP6
		PA.0, PE.12	MFP8

The basic configurations of UART2 are as follows:

- Clock Source Configuration
  - Select the source of UART2 peripheral clock on UART2SEL (CLK_CLKSEL2[26:24]).
  - Select clock divider number of UART2 peripheral clock on UART2DIV (CLK_CLKDIV4[3:0]).
  - Enable UART2 peripheral clock in UART2CKEN (CLK_APBCLK0[18]).
- Reset UART2 controller in UART2RST (SYS_IPRST1[18]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
UART2	UART2_RXD	PF.5	MFP2
		PE.15	MFP3
		PB.0, PD.12, PE.9	MFP7
		PC.0, PC.4	MFP8
		PB.4	MFP12
	UART2_TXD	PF.4	MFP2
		PE.14	MFP3
		PB.1, PC.13, PE.8	MFP7
		PC.1, PC.5	MFP8
		PB.5	MFP12
	UART2_nCTS	PD.9, PF.5	MFP4

		PC.2	MFP8
	UART2_nRTS	PD.8, PF.4	MFP4
		PC.3	MFP8

The basic configurations of UART3 are as follows:

- Clock Source Configuration
  - Select the source of UART3 peripheral clock on UART3SEL (CLK_CLKSEL2[30:28]).
  - Select the clock divider number of UART3 peripheral clock on UART3DIV (CLK_CLKDIV4[7:4]).
  - Enable UART3 peripheral clock in UART3CKEN (CLK_APBCLK0[19]).
- Reset UART3 controller in UART3RST (SYS_IPRST1[19]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
UART3	UART3_RXD	PD.0	MFP5
		PB.14, PC.9, PE.0, PE.11	MFP7
		PC.2	MFP11
	UART3_TXD	PD.1	MFP5
		PB.15, PC.10, PE.1, PE.10	MFP7
		PC.3	MFP11
	UART3_nCTS	PD.2	MFP5
		PB.12, PH.9	MFP7
	UART3_nRTS	PD.3	MFP5
		PB.13, PH.8	MFP7

The basic configurations of UART4 are as follows:

- Clock Source Configuration
  - Select the source of UART4 peripheral clock on UART4SEL (CLK_CLKSEL3[26:24]).
  - Select the clock divider number of UART4 peripheral clock on UART4DIV (CLK_CLKDIV4[11:8]).
  - Enable UART4 peripheral clock in UART4CKEN (CLK_APBCLK0[20]).
- Reset UART4 controller in UART4RST (SYS_IPRST1[20]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
UART4	UART4_RXD	PA.13	MFP3
		PC.6	MFP5

		PB.10, PF.6	MFP6
		PA.2, PH.11	MFP7
		PC.4	MFP11
	UART4_TXD	PA.12	MFP3
		PC.7	MFP5
		PB.11, PF.7	MFP6
		PA.3, PH.10	MFP7
		PC.5	MFP11
	UART4_nCTS	PC.8	MFP5
		PE.1	MFP9
	UART4_nRTS	PE.13	MFP5
		PE.0	MFP9

The basic configurations of UART5 are as follows:

- Clock Source Configuration
  - Select the source of UART5 peripheral clock on UART5SEL (CLK_CLKSEL3[30:28]).
  - Select the clock divider number of UART5 peripheral clock on UART5DIV (CLK_CLKDIV4[15:12]).
  - Enable UART5 peripheral clock in UART5CKEN (CLK_ APBCLK0[21]).
- Reset UART5 controller in UART5RST (SYS_IPRST1[21]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
UART5	UART5_RXD	PF.10	MPF6
		PB.4	MFP7
		PA.4, PE.6	MFP8
	UART5_TXD	PF.11	MFP6
		PB.5	MFP7
		PA.5, PE.7	MFP8
	UART5_nCTS	PF.8	MFP6
		PB.2	MFP7
	UART5_nRTS	PF.9	MFP6
		PB.3	MFP7

UART Interface Controller Pin description is shown in Table 6.19-3:

Pin	Type	Description
UARTx_TXD	Output	UARTx transmit

UARTx_RXD	Input	UARTx receive
UARTx_nCTS	Input	UARTx modem clear to send
UARTx_nRTS	Output	UARTx modem request to send

Table 6.19-3 UART Interface Controller Pin

6.19.5 Functional Description

The UART controller supports four function modes including UART, IrDA, LIN and RS-485 mode. User can select a function by setting the UART_FUNCSEL register. The four function modes will be described in following section.

6.19.5.1 UART Controller Baud Rate Generator

The UART controller includes a programmable baud rate generator capable of dividing clock input by divisors to produce the serial clock that transmitter and receiver need. Table 6.19-4 list the UART baud rate equations in the various conditions. Table 6.19-5 and Table 6.19-6 list the UART baud rate parameter and register setting example. In IrDA function mode, the baud rate generator must be set in mode 0. More detail register description is shown in UART_BAUD register. There are three setting mode. Mode 0 is set by UART_BAUD[29:28] with 00. Mode 1 is set by UART_BAUD[29:28] with 10. Mode 2 is set by UART_BAUD[29:28] with 11.

Mode	**BAUDM1**	**BAUDM0**	**Baud Rate Equation**
Mode 0	0	0	UART_CLK / [16 * (BRD+2)].
Mode 1	1	0	UART_CLK / [(EDIVM1+1) * (BRD+2)], EDIVM1 must >= 8.
Mode 2	1	1	UART_CLK / (BRD+2). If UART_CLK <= 3*HCLK, BRD must >= 8. If UART_CLK > 3*HCLK, BRD must >= 3*N – 1. N is the smallest integer larger than or equal to the ratio of UART_CLK /HCLK. For example, if 3*HCLK < UART_CLK =< 4*HCLK, BRD must >=11. if 4*HCLK < UART_CLK =< 5*HCLK, BRD must >=14. (If the UART_CLK is selected from LXT, BRD can be greater than or equal to 1)

Table 6.19-4 UART controller Baud Rate Equation Table

UART Peripheral Clock = 12 MHz			
Baud Rate	Mode 0	Mode 1	Mode 2
921600	Not support	Not recommended	BRD=11
460800	Not recommended	BRD=0, EDIVM1 =12	BRD=24
230400	Not recommended	BRD =2, EDIVM1 =12	BRD =50
115200	Not recommended	BRD =6, EDIVM1 =12	BRD =102
57600	BRD =11	BRD =14, EDIVM1 =12	BRD =206
38400	BRD =18	BRD =22, EDIVM1 =12	BRD =311
19200	BRD =37	BRD =61, EDIVM1 =9	BRD =623
9600	BRD =76	BRD =123, EDIVM1 =9	BRD =1248

4800	BRD =154	BRD =248, EDIVM1 =9	BRD =2498

Table 6.19-5 UART controller Baud Rate Parameter Setting Example Table

UART Peripheral Clock = 12 MHz			
Baud Rate	UART_BAUD Value		
	Mode 0	Mode 1	Mode 2
921600	Not support	Not recommended	0x3000_000B
460800	Not recommended	0x2C00_0000	0x3000_0018
230400	Not recommended	0x2C00_0002	0x3000_0032
115200	Not recommended	0x2C00_0006	0x3000_0066
57600	0x0000_000B	0x2C00_000E	0x3000_00CE
38400	0x0000_0012	0x2C00_0016	0x3000_0137
19200	0x0000_0025	0x2900_003D	0x3000_026F
9600	0x0000_004C	0x2900_007B	0x3000_04E0
4800	0x0000_009A	0x2900_00F8	0x3000_09C2

Table 6.19-6 UART controller Baud Rate Register Setting Example Table

6.19.5.2 UART Controller Baud Rate Compensation

The UART controller supports baud rate compensation function. It is used to optimize the precision in each bit. The precision of the compensation is half of UART module clock because there is BRCOMDEC bit (UART_BRCOMP[31]) to define the positive or negative compensation in each bit. If the BRCOMPDEC (UART_BRCOMP[31]) = 0, it is positive compensation for each bit, one more module clock will be append in the compensated bit. If the BRCOMPDEC (UART_BRCOMP[31]) = 1, it is negative compensation for each bit, decrease one module clock in the compensated bit.

There is 9-bits location, BRCOMP[8:0] (UART_BRCOMP[8:0]), can be configured by user to define the relative bit is compensated or not. BRCOMP[7:0] is used to define the compensation of UART_DAT[7:0] and BRCOMP[8] is used to define the parity bit.

Example:

1. UART's peripheral clock = 32.768 kHz and baud rate is 9600 bps

Baud rate is 9600 bps, UART peripheral clock is 32.768 kHz → 3.413 peripheral clock/bit

if the baud divider is set 1 (3 peripheral clock/bit), the inaccuracy of each bit is -0.413 peripheral clock and BRCOMPDEC =0,

Bit	**Name**	**Total INACCURACY**	**BRCOMP Compensated**	**Final Inaccuracy**
0	Start	-0.413	x	-0.413
1	UART_DAT[0]	-0.826(-0.413-0.413)	1	0.174
2	UART_DAT[1]	-0.239(0.174-0.413)	0	-0.239
3	UART_DAT[2]	-0.652(-0.239-0.413)	1	0.348
4	UART_DAT[3]	-0.065(0.348-0.413)	0	-0.065

5	UART_DAT[4]	-0.478(-0.065-0.413)	0	-0.478
6	UART_DAT[5]	-0.891(-0.478-0.413)	1	0.109
7	UART_DAT[6]	-0.304(0.109-0.413)	0	-0.304
8	UART_DAT[7]	-0.717(-0.304-0.413)	1	0.283
9	Parity	-0.130(0.283-0.413)	0	-0.13

Table 6.19-7 Baud Rate Compensation Example Table 1

So that the BRCOMP (UART_BRCOMP[8:0]) can be set as 9'b010100101 = 0xa5.

2. UART's peripheral clock = 32.768 kHz and baud rate is 4800 bps

Baud rate is 4800 bps, UART peripheral clock is 32.768 kHz → 6.827 peripheral clock/bit

if the baud divider is set 5 (7 peripheral clock/bit), the inaccuracy of each bit is 0.173 peripheral clock and BRCOMPDEC =1,

Bit	Name	Total INACCURACY	BRCOMP Compensated	Final Inaccuracy
0	Start	0.173	x	0.173
1	UART_DAT[0]	0.346(0.173+0.173)	0	0.346
2	UART_DAT[1]	0.519(0.346+0.173)	1	-0.481
3	UART_DAT[2]	-0.308(-0.481+0.173)	0	-0.308
4	UART_DAT[3]	-0.135(-0.308+0.173)	0	-0.135
5	UART_DAT[4]	-0.038(-0.135+0.173)	0	0.038
6	UART_DAT[5]	0.211(0.038+0.173)	0	0.211
7	UART_DAT[6]	0.384(0.211+0.173)	0	0.384
8	UART_DAT[7]	0.557(0.384+0.173)	1	-0.443
9	Parity	-0.270(-0.443+0.173)	0	-0.270

Table 6.19-8 Baud Rate Compensation Example Table 2

So that the BRCOMP (UART_BRCOMP[8:0]) can be set as 9'b010000010 = 0x82.

UART Controller Auto-Baud Rate Function Mode

Auto-Baud Rate function can measure baud rate of receiving data from UART RX pin automatically. When the Auto-Baud Rate measurement is finished, the measuring baud rate is loaded to BRD (UART_BAUD[15:0]). Both of the BAUDM1 (UART_BAUD[29]) and BAUDM0 (UART_BAUD[28]) are set to 1 automatically. UART RX data from Start bit to 1st rising edge time is set by 2 ^{ABRDBITS} bit time in Auto-Baud Rate function detection frame.

2 ^{ABRDBITS} bit time from Start bit to the 1st rising edge is calculated by setting ABRDBITS (UART_ALTCTL[20:19]). Setting ABRDEN (UART_ALTCTL[18]) is to enable auto-baud rate function. In beginning stage, the UART RX is kept at 1. Once falling edge is detected, START bit is received. The auto-baud rate counter is reset and starts counting. The auto-baud rate counter will be stop when the 1st rising edge is detected. Then, auto-baud rate counter value divided by ABRDBITS (UART_ALTCTL[20:19]) is loaded to BRD (UART_BAUD[15:0]) automatically. ABRDEN (UART_ALTCTL[18]) is cleared. The Auto-Baud is shown in Figure 6.19-3. Once the auto-baud rate measurement is finished, the ABRDIF (UART_FIFOSTS[1]) is set. When auto-baud rate counter is

overflow, ABRDTOIF (UART_FIFOSTS[2]) is set. ABRDIF (UART_FIFOSTS[1]) or ABRDTOIF (UART_FIFOSTS[2]) cause the auto-baud rate flag ABRIF(UART_ALTCTL[17]) is generated. If the ABRIEN (UART_INTEN[18]) is enabled, ABRIF(UART_ALTCTL[17]) cause the auto-baud rate interrupt ABRINT (UART_INTSTS[31]) is generated.

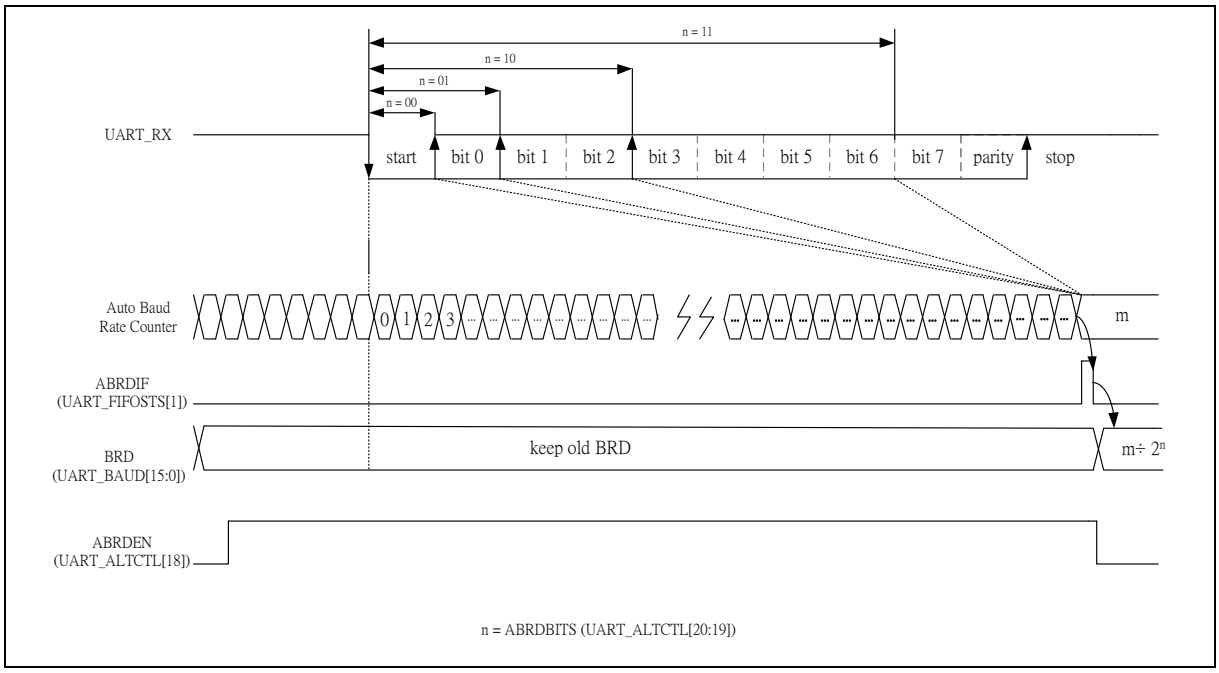


Figure 6.19-3 Auto-Baud Rate Measurement

6.19.5.3 Programming Sequence Example:

1. Program ABRDBITS (UART_ALTCTL[20:19]) to determines UART RX data 1st rising edge time from Start by 2 ABRDBITS bit time.
2. Set ABRIEN (UART_INTEN[18]) to enable auto-baud rate function interrupt.
3. Set ABRDEN (UART_ALTCTL[18]) to enable auto-baud rate function.
4. ABRDIF (UART_FIFOSTS[1]) is set, the auto-baud rate measurement is finished.
5. Operate UART transmit and receive action.
6. ABRDTOIF (UART_FIFOSTS[2]) is set, if auto-baud rate counter is overflow.
7. Go to Step 3.

6.19.5.4 UART Controller Transmit Delay Time Value

The UART controller programs DLY (UART_TOUT [15:8]) to control the transfer delay time between the last stop bit and next start bit in transmission. The unit is baud. The operation is shown in Figure 6.19-4.

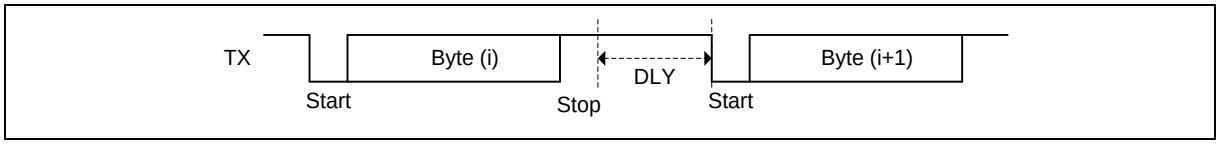


Figure 6.19-4 Transmit Delay Time Operation

6.19.5.5 UART Controller FIFO Control and Status

The UART controller is built-in with a 16 bytes transmitter FIFO (TX_FIFO) and a 16 bytes receiver FIFO (RX_FIFO) that reduces the number of interrupts presented to the CPU. The CPU can read the status of the UART at any time during operation. The reported status information includes condition of the transfer operations being performed by the UART, as well as 3 error conditions (parity error, framing error, break interrupt) occur if receiving data has parity, frame or break error. UART, IrDA, LIN and RS-485 mode support FIFO control and status function.

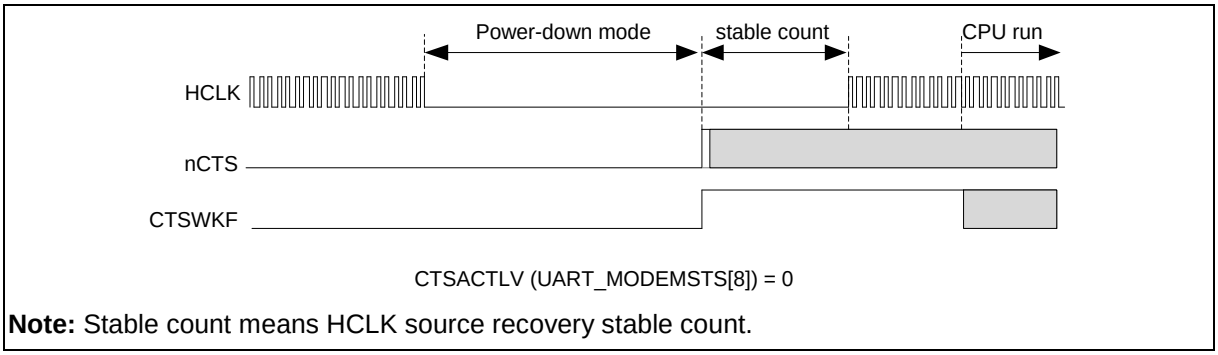
6.19.5.6 UART Controller Wake-up Function

The UART controller supports wake-up system function. The wake-up function includes nCTS pin, incoming data wake-up, Received Data FIFO reached threshold wake-up, RS-485 Address Match (AAD mode) wake-up and Received Data FIFO threshold time-out wake-up function. CTSWKF (UART_WKSTS[0]), DATWKF (UART_WKSTS[1]), RFRTWKF (UART_WKSTS[2]), RS485WKF (UART_WKSTS[3]) or TOUTWKF (UART_WKSTS[4]) cause the wake-up interrupt flag WKIF(UART_INTSTS[6]) is generated. If the WKIEN (UART_INTEN[6]) is enabled, the wake-up interrupt flag WKIF(UART_INTSTS[6]) cause the wake-up interrupt WKINT (UART_INTSTS[14]) is generated.

nCTS pin wake-up :

When the system is in Power-down mode and WKCTSEN (UART_WKCTL[0]) is set, the toggle of nCTS pin can wake-up system. If the WKCTSEN (UART_WKCTL[0]) is enabled, the toggle of nCTS pin cause the nCTS wake-up flag CTSWKF (UART_WKSTS[0]) is generated. The nCTS wake-up is shown in Figure 6.19-5 and Figure 6.19-6.

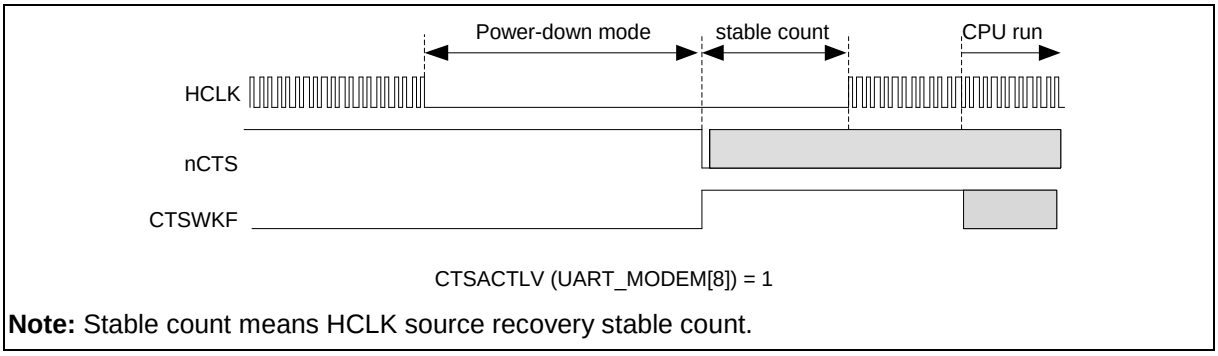
nCTS Wake-up Case 1 (nCTS transition from low to high)



Note: Stable count means HCLK source recovery stable count.

Figure 6.19-5 UART nCTS Wake-up Case1

nCTS Wake-up Case 2 (nCTS transition from high to low)



Note: Stable count means HCLK source recovery stable count.

Figure 6.19-6 UART nCTS Wake-up Case2

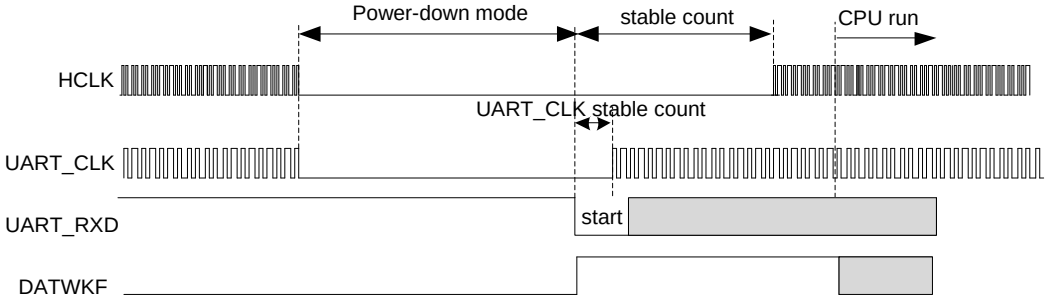
Incoming Data Wake-up

When system is in Power-down mode and the WKDATEN (UART_WKCTL [1]) is set, the toggle of incoming data (UART_RXD) pin can wake-up the system. In order to receive the incoming data after the system wake-up, the STCOMP (UART_DWKCOMP[15:0]) shall be set. These bits field of STCOMP indicate how many clock cycle selected by UART_CLK do the UART controller can get the 1st bit (start bit) when the system is wakeup from Power-down mode.

When incoming data wakes system up, the incoming data will be received and stored in FIFO. If the WKDATEN (UART_WKCTL[1]) is enabled, the toggle of incoming data (UART_RXD) pin cause the incoming data wake-up flag DATWKF (UART_WKSTS[1]) is generated. The incoming data wake-up is shown in Figure 6.19-7.

Note1: The UART controller clock source should be selected as HIRC, and the compensation time for START bit that refer to Datasheet for detailed information about wakeup time electrical characteristics. The STCOMP (UART_DWKCOMP[15:0]) = (wake-up stable time) * (HIRC frequency).

Note2: The value of BRD(UART_BAUD[15:0]) should be greater than STCOMP (UART_DWKCOMP[15:0]).



Note 1: Stable count means HCLK source recovery stable count.

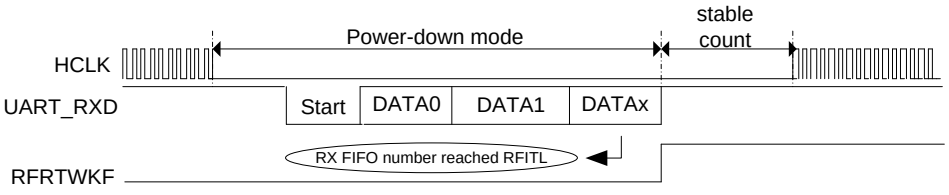
Note 2: UART_CLK stable count means UART clock source recovery stable count.

Figure 6.19-7 UART Data Wake-up

Received Data FIFO Reached Threshold Wake-up

The received data FIFO threshold reached wake-up function is enabled by setting WKRFRTEN (UART_WKCTL[2]). In Power-down mode, when the number of received data in RX FIFO reaches the threshold value RFITL (UART_FIFO[7:4]), it can wake-up the system. If the WKRFRTEN (UART_WKCTL[2]) is enabled, the number of received data in RX FIFO reaches the threshold value RFITL (UART_FIFO[7:4]) cause the received data FIFO reached threshold wake-up flag RFRTWKF (UART_WKSTS[2]) is generated. The Received Data FIFO reached threshold wake-up is shown in Figure 6.19-8.

Note: The UART controller clock source should be selected as LXT in Power-down mode to receive data.



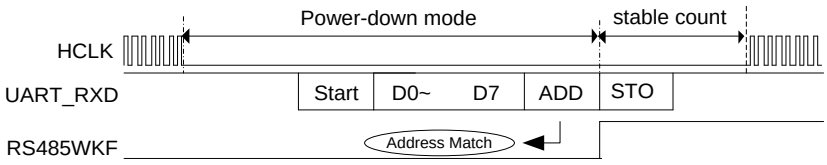
Note: Stable count means HCLK source recovery stable count.

Figure 6.19-8 UART Received Data FIFO reached threshold wake-up

RS-485 Address Match (AAD Mode) Wake-up

The RS-485 address match wake-up function is enabled by setting WKRFRTEN (UART_WKCTL[2]) and WKRS485EN (UART_WKCTL[3]). This function is used for RS-485 Auto Address Detection (AAD) mode in RS-485 function mode and ADDRDEN (UART_ALTCTL[15]) is set to 1. In Power-down mode, when an address byte is detected and matches the ADDRMV (UART_ALTCTL[31:24]) or the number of received data in RX FIFO reaches the threshold value RFITL (UART_FIFO[7:4]), it can wake-up the system. If the WKRS485EN (UART_WKCTL[3]) is enabled, when an address byte is detected and matches the ADDRMV (UART_ALTCTL[31:24]) that cause the RS485 address match (AAD mode) wake-up flag RS485WKF (UART_WKSTS[3]) is generated. The RS-485 Address Match (AAD mode) wake-up is shown in Figure 6.19-9.

Note: The UART controller clock source should be selected as LXT in Power-down mode to receive data.



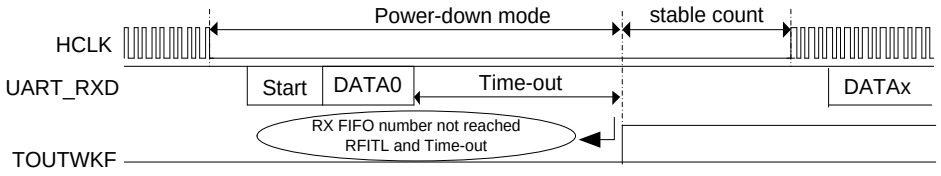
Note: Stable count means HCLK source recovery stable count.

Figure 6.19-9 UART RS-485 AAD Mode Address Match Wake-up

Received Data FIFO Threshold Time-out Wake-up

The received data FIFO threshold time-out wake-up function is enabled by setting WKRFRTEN (UART_WKCTL[2]) and WKTOUTEN (UART_WKCTL[4]). Setting TOCNTEN (UART_INTEN[11]) to enable receiver buffer time-out counter. In Power-down mode, when the number of received data in RX FIFO does not reach the threshold value RFITL (UART_FIFO[7:4]) and the time-out counter equals to the time-out value TOIC (UART_TOUT[7:0]), it can wake-up the system. If the WKTOUTEN (UART_WKCTL[4]) is enabled, when the time-out counter equals to the time-out value TOIC (UART_TOUT[7:0]) that cause the Received Data FIFO threshold time-out wake-up flag TOUTWKF (UART_WKSTS[4]) is generated. The Received Data FIFO threshold time-out wake-up is shown in Figure 6.19-10.

Note: The UART controller clock source should be selected as LXT in Power-down mode to receive data.



Note: Stable count means HCLK source recovery stable count.

Figure 6.19-10 UART Received Data FIFO threshold time-out wake-up

6.19.5.7 UART Controller Interrupt and Status

Each UART controller supports ten types of interrupts including:

- Receive Data Available Interrupt (RDAINT)
- Transmit Holding Register Empty Interrupt (THERINT)

- Transmitter Empty Interrupt (TXENDIF)
- Receive Line Status Interrupt (RLSINT)
  - Break Interrupt Flag (BIF)
  - Framing Error Flag (FEF)
  - Parity Error Flag (PEF)
  - RS-485 Address Byte Detect Flag (ADDRDETF)
- MODEM Status Interrupt (MODEMINT)
  - Detect nCTS State Change Flag (CTSDETF)
- Receiver Buffer Time-out Interrupt (RXTOINT)
- Buffer Error Interrupt (BUFERRINT)
  - TX Overflow Error Interrupt Flag (TXOVIF)
  - RX Overflow Error Interrupt Flag (RXOVIF)
- LIN Bus Interrupt (LININT)
  - LIN Break Detection Flag (BRKDETF)
  - Bit Error Detect Status Flag (BITEF)
  - LIN Slave ID Parity Error Flag (SLVIDPEF)
  - LIN Slave Header Error Flag (SLVHEF)
  - LIN Slave Header Detection Flag (SLVHDETF)
- Wake-up Interrupt (WKINT)
  - nCTS Wake-up Flag (CTSWKF)
  - Incoming Data Wake-up Flag (DATWKF)
  - Received Data FIFO Reached Threshold Wake-up Flag (RFRTWKF)
  - RS-485 Address Match (AAD mode) Wake-up Flag (RS485WKF)
  - Received Data FIFO Threshold Time-out Wake-up Flag (TOUTWKF)
- Auto-Baud Rate Interrupt (ABRINT)
  - Auto-baud Rate Detect Interrupt Flag (ABRDIF)
  - Auto-baud Rate Detect Time-out Interrupt Flag (ABRDTOIF)
- Single-wire Bit Error Detect Interrupt (SWBEINT)

Table 6.19-9 describes the interrupt sources and flags. The interrupt is generated when the interrupt flag is generated and the interrupt enable bit is set. User must clear the interrupt flag after the interrupt is generated.

Interrupt Source	Interrupt Indicator	Interrupt Enable Bit	Interrupt Flag	Flag Caused By	Flag Cleared By
Receive Data Available Interrupt	RDAINT	RDAIEN	RDAIF	N/A	Read UART_DAT
Transmit Holding Register Empty Interrupt	THERINT	THREIEN	THREIF	N/A	Write UART_DAT

Transmitter Empty Interrupt	TXENDINT	TXENDIEN	TXENDIF	N/A	Write UART_DAT
Receive Line Status Interrupt	RLSINT	RLSIEN	RLSIF	RLSIF = BIF	Write ‘1’ to BIF
				RLSIF = FEF	Write ‘1’ to FEF
				RLSIF = PEF	Write ‘1’ to PEF
				RLSIF = ADDRDETF	Write ‘1’ to ADDRDETF
Modem Status Interrupt	MODEMINT	MODEMIEN	MODEMIF	MODEMIF = CTSDETF	Write ‘1’ to CTSDETF
Receiver Buffer Time-out Interrupt	RXTOINT	RXTOIEN	RXTOIF	N/A	Read UART_DAT
Buffer Error Interrupt	BUFERRINT	BUFERRIEN	BUFERRIF	BUFERRIF = TXOVIF	Write ‘1’ to TXOVIF
				BUFERRIF = RXOVIF	Write ‘1’ to RXOVIF
LIN Bus Interrupt	LININT	LINIEN	LINIF	LINIF = BRKDETF	Write ‘1’ to LINIF and Write ‘1’ to BRKDETF
				LINIF = BITEF	Write ‘1’ to LINIF and Write ‘1’ to BITEF
				LINIF = SLVIDPEF	Write ‘1’ to LINIF and Write ‘1’ to SLVIDPEF
				LINIF = SLVHEF	Write ‘1’ to LINIF and Write ‘1’ to SLVHEF
				LINIF = SLVHDETF	Write ‘1’ to LINIF and Write ‘1’ to SLVHDETF
Wake-up Interrupt	WKINT	WKIEN	WKIF	WKIF = CTSWKF	Write ‘1’ to CTSWKF
				WKIF = DATWKF	Write ‘1’ to DATWKF
				WKIF = RFRTWKF	Write ‘1’ to RFRTWKF
				WKIF = RS485WKF	Write ‘1’ to RS485WKF
				WKIF = TOUTWKF	Write ‘1’ to TOUTWKF
Auto-Baud Rate Interrupt	ABRINT	ABRIEN	ABRIF	ABRIF = ABRDIF	Write ‘1’ to ABRDIF
				ABRIF = ABRDTOIF	Write ‘1’ to ABRDTOIF
Single-wire Bit Error Detect Interrupt	SWBEINT	SWBEIEN	SWBEIF	N/A	Writing ‘1’ to SWBEIF

Table 6.19-9 UART controller Interrupt Source and Flag List

6.19.5.8 UART Function Mode

The UART controller provides UART function (Setting FUNCSEL (UART_FUNCSEL [2:0]) to '000' to enable UART function mode). The UART baud rate is up to 1 Mbps.

The UART provides full-duplex and asynchronous communications. The transmitter and receiver contain 16 bytes FIFO for payloads. User can program receiver buffer trigger level and receiver buffer time-out detection for receiver. The transmitting data delay time between the last stop and the next start bit can be programed by setting DLY (UART_TOUT [15:8]) register. The UART supports hardware auto-flow control that provides programmable nRTS flow control trigger level. The number of data bytes in RX FIFO is equal to or greater than RTSTRGLV (UART_FIFO[19:16]), the nRTS is de-asserted.

UART Line Control Function

The UART controller supports fully programmable serial-interface characteristics by setting the UART_LINE register. User can program UART_LINE register for the word length, stop bit and parity bit setting. Table 6.19-10 and Table 6.19-11 list the UART word, stop bit length and the parity bit settings.

NSB (UART_LINE[2])	WLS (UART_LINE[1:0])	Word Length (Bit)	Stop Length (Bit)
0	00	5	1
0	01	6	1
0	10	7	1
0	11	8	1
1	00	5	1.5
1	01	6	2
1	10	7	2
1	11	8	2

Table 6.19-10 UART Line Control of Word and Stop Length Setting

Parity Type	SPE (UART_LINE[5])	EPE (UART_LINE[4])	PSS (UART_LINE[7])	PBE (UART_LINE[3])	Description
No Parity	x	x	x	0	No parity bit output.
Parity source from UART_DAT	x	x	1	1	Parity bit is generated and checked by software.
Odd Parity	0	0	0	1	Odd Parity is calculated by adding all the "1's" in a data stream and adding a parity bit to the total bits, to make the total count an odd number.
Even Parity	0	1	0	1	Even Parity is calculated by adding all the "1's" in a data stream and adding a parity bit to the total bits, to make the count an even number.
Forced Mask	1	0	0	1	Parity bit always logic 1. Parity bit on the serial byte is set to "1" regardless

Parity					of total number of “1’s” (even or odd counts).
Forced Space Parity	1	1	0	1	Parity bit always logic 0. Parity bit on the serial byte is set to “0” regardless of total number of “1’s” (even or odd counts).

Table 6.19-11 UART Line Control of Parity Bit Setting

UART Auto-Flow Control Function

The UART supports auto-flow control function that uses two signals, nCTS (clear-to-send) and nRTS (request-to-send), to control the flow of data transfer between the UART and external devices (e.g. Modem). When auto-flow is enabled, the UART is not allowed to receive data until the UART asserts nRTS to external device. When the number of bytes stored in the RX FIFO equals the value of RTSTRGLV (UART_FIFO [19:16]), the nRTS is de-asserted. The UART sends data out when UART detects nCTS is asserted from external device. If the valid asserted nCTS is not detected, the UART will not send data out. The auto flow control block diagram is shown in Figure 6.19-11.

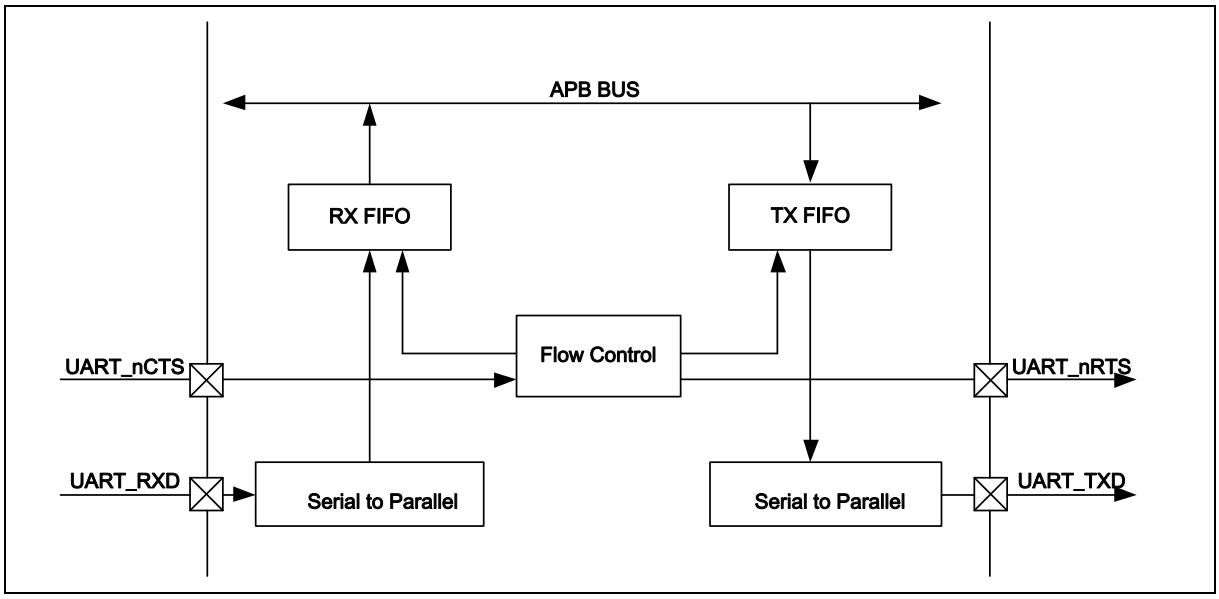


Figure 6.19-11 Auto-Flow Control Block Diagram

Figure 6.19-12 demonstrates the nCTS auto-flow control of UART function mode. User must set ATOCTSEN (UART_INTEN [13]) to enable nCTS auto-flow control function. The CTSACTLV (UART_MODEMSTS [8]) can set nCTS pin input active state. The CTSDETF (UART_MODEMSTS[0]) is set when any state change of nCTS pin input has occurred, and then TX data will be automatically transmitted from TX FIFO.

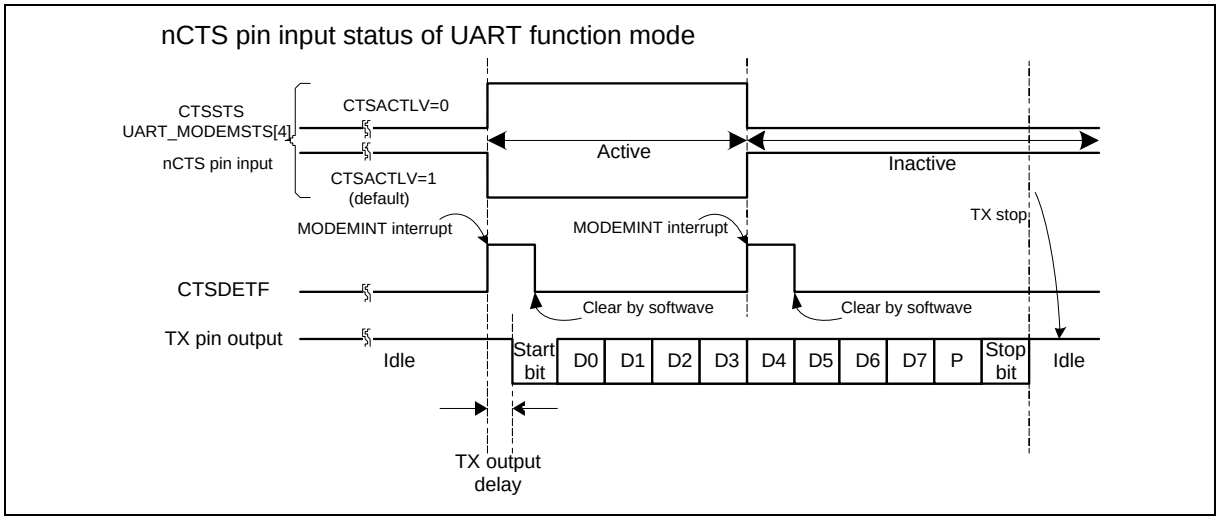


Figure 6.19-12 UART nCTS Auto-Flow Control Enabled

As shown in Figure 6.19-13, in UART nRTS auto-flow control mode (ATORTSEN(UART_INTEN[12])=1), the nRTS internal signal is controlled by UART FIFO controller with RTSTRGLV(UART_FIFO[19:16]) trigger level.

Setting RTSACTLV(UART_MODEM[9]) can control the nRTS pin output is inverse or non-inverse from nRTS signal. User can read the RTSSTS (UART_MODEM[13]) bit to get real nRTS pin output voltage logic status.

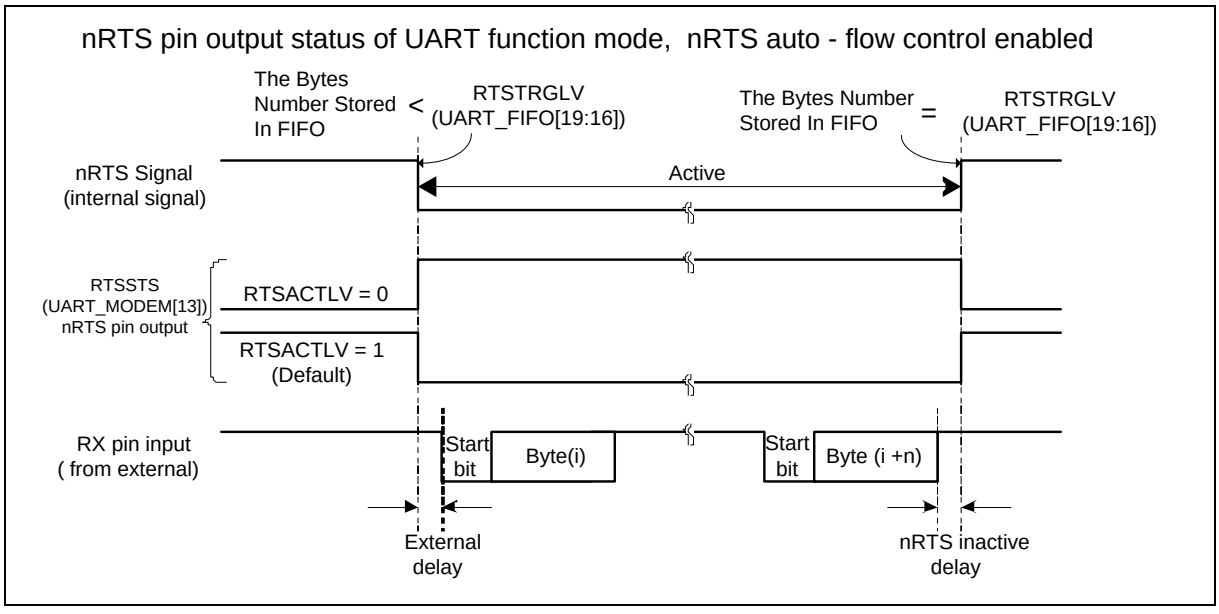


Figure 6.19-13 UART nRTS Auto-Flow Control Enabled

As shown in Figure 6.19-14, in software mode (ATORTSEN(UART_INTEN[12])=0), the nRTS flow is directly controlled by software programming of RTS(UART_MODEM[1]) control bit.

Setting RTSACTLV(UART_MODEM[9]) can control the nRTS pin output is inverse or non-inverse from RTS(UART_MODEM[1]) control bit. User can read the RTSSTS(UART_MODEM[13]) bit to get real nRTS pin output voltage logic status.

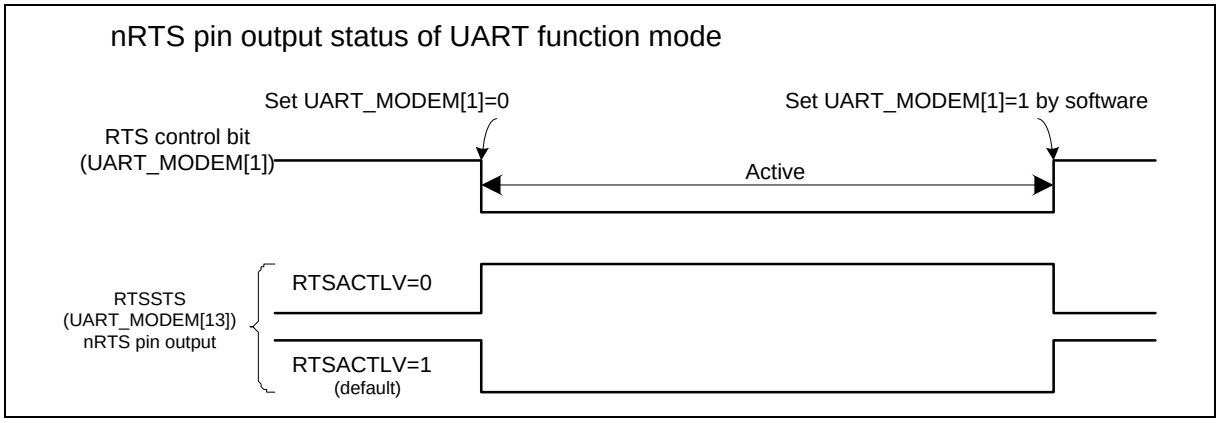


Figure 6.19-14 UART nRTS Auto-Flow with Software Control

6.19.5.9 IrDA Function Mode

The UART controller also provides Serial IrDA (SIR, Serial Infrared) function (Setting UART_FUNCSEL [2:0] to '010' to enable the IrDA function). The SIR specification defines a short-range infrared asynchronous serial transmission mode with one start bit, 8 data bits, and 1 stop bit. The maximum data rate is 115.2 kbps. The IrDA SIR block contains an IrDA SIR protocol encoder/decoder. The IrDA SIR protocol is half-duplex only. So, it cannot transmit and receive data at the same time. The IrDA SIR physical layer specifies a minimum 10 ms transfer delay between transmission and reception, and this delay feature must be implemented by software.

In IrDA mode, the BAUDM1 (UART_BAUD [29]) must be cleared.

Baud Rate = Clock / (16 * (BRD +2)), where BRD (UART_BAUD[15:0]) is Baud Rate Divider in UART_BAUD register.

Note: The tolerance of baud-rate is ±5% between IrDA master and IrDA slave.

The IrDA control block diagram is shown in Figure 6.19-15.

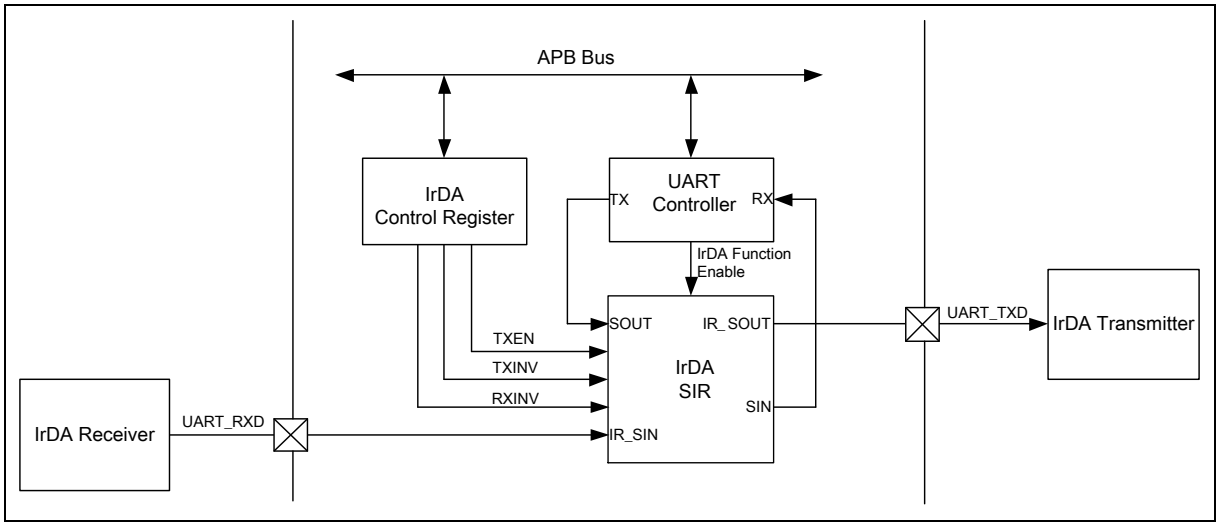


Figure 6.19-15 IrDA Control Block Diagram

IrDA SIR Transmit Encoder

The IrDA SIR Transmit Encoder modulates Non-Return-to-Zero (NRZ) transmit bit stream output from UART. The IrDA SIR physical layer specifies the use of Return-to-Zero, Inverted (RZI) modulation

scheme which represents logic 0 as an infra light pulse. The modulated output pulse stream is transmitted to an external output driver and infrared light emitting diode.

The transmitted pulse width is specified as 3/16 period of baud rate.

IrDA SIR Receive Decoder

The IrDA SIR Receive Decoder demodulates the Return-to-Zero bit stream from the input detector and outputs the NRZ serial bits stream to the UART received data input.

In idle state, the decoder input is high. A start bit is detected when the decoder input is LOW. In normal operation, the RXINV (UART_IRDA[6]) is set to '1' and TXINV (UART_IRDA[5]) is set to '0'.

IrDA SIR Operation

The IrDA SIR encoder/decoder provides functionality which converts between UART data stream and half-duplex serial SIR interface. Figure 6.19-16 is IrDA encoder/decoder waveform.

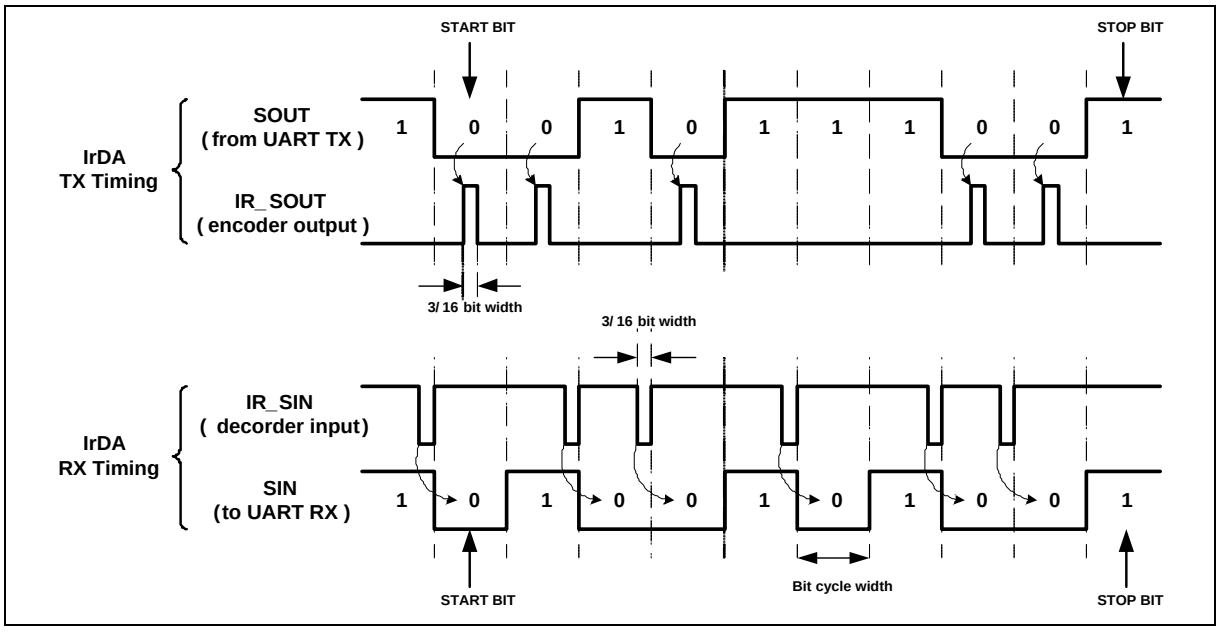


Figure 6.19-16 IrDA TX/RX Timing Diagram

6.19.5.10 LIN Function Mode (Local Interconnection Network)

The UART Controller supports LIN function. Setting FUNCSEL (UART_FUNCSEL[2:0]) to '001' to select LIN mode operation. The UART Controller supports LIN break/delimiter generation and break/delimiter detection in LIN master mode, and supports header detection and automatic resynchronization in LIN Slave mode.

Structure of LIN Frame

According to the LIN protocol, all information transmitted is packed as frames; a frame consists of a header (provided by the master task) and a response (provided by a slave task). The header (provided by the master task) consists of a break field and a sync field followed by a frame identifier (frame ID). The frame identifier uniquely defines the purpose of the frame. The slave task is appointed for providing the response associated with the frame ID. The response consists of a data field and a checksum field. Figure 6.19-17 is the structure of LIN Frame.

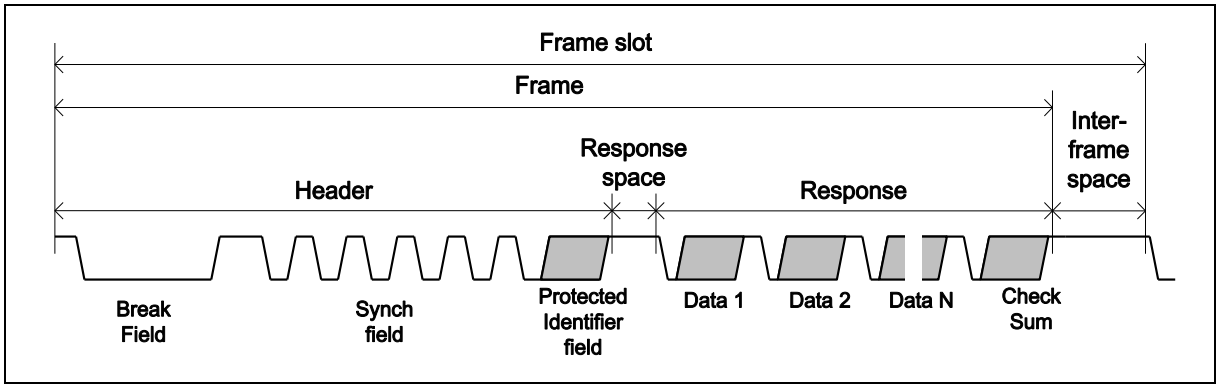


Figure 6.19-17 Structure of LIN Frame

Structure of LIN Byte

In LIN mode, each byte field is initiated by a START bit with value 0 (dominant), followed by 8 data bits and no parity bit, LSB is first and ended by 1 stop bit with value 1 (recessive) in accordance with the LIN standard. The structure of Byte is shown in Figure 6.19-18.

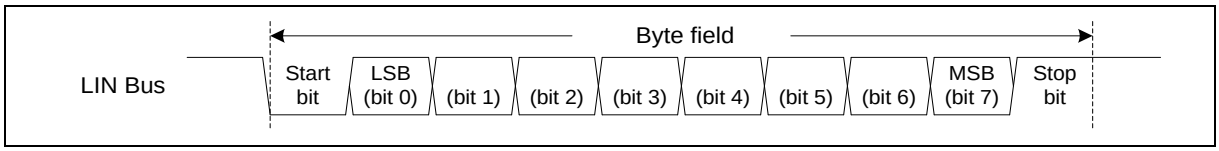


Figure 6.19-18 Structure of LIN Byte

LIN Master Mode

The UART Controller supports LIN Master mode. To enable and initialize the LIN Master mode, the following steps are necessary:

1. Set the UART_BAUD register to select the desired baud rate.
2. Set WLS (UART_LINE[1:0]) to '11' to configure the word length with 8 bits, clearing PBE (UART_LINE[3]) bit to disable parity check and clearing NSB (UART_LINE[2]) bit to configure with one stop bit.
3. Set FUNCSEL (UART_FUNCSEL[2:0]) to '001' to select LIN function mode operation.

A complete header consists of a break field and sync field followed by a frame identifier (frame ID). The UART controller can be selected header sending by three header selected modes. The header selected mode can be "break field" or "break field and sync field" or "break field, sync field and frame ID field" by setting HSEL (UART_LINCTL[23:22]). If the selected header is "break field", software must handle the following sequence to send a complete header to bus by filling sync data (0x55) and frame ID data to the UART_DAT register. If the selected header is "break field and sync field", software must handle the sequence to send a complete header to bus by filling the frame ID data to UART_DAT register, and if the selected header is "break field, sync field and frame ID field", hardware will control the header sending sequence automatically but software must filled frame ID data to PID (UART_LINCTL [31:24]). When operating in header selected mode in which the selected header is "break field, sync field and frame ID field", the frame ID parity bit can be calculated by software or hardware depending whether the IDPEN (UART_LINCTL[9]) bit is set or not.

HSEL	Break Field	Sync Field	ID Field
0	Generated by Hardware	Handled by Software	Handled by Software
1	Generated by Hardware	Generated by Hardware	Handled by Software

2	Generated by Hardware	Generated by Hardware	Generated by Hardware (But Software needs to fill ID to PID (UART_LINCTL[31:24]) first)

Table 6.19-12 LIN Header Selection in Master Mode

When UART is operated in LIN data transmission, LIN bus transfer state can be monitored by hardware or software. User can enable hardware monitoring by setting BITERREN (UART_LINCTL [12]) to "1", if the input pin (UART_RX) state is not equal to the output pin (UART_TX) state in LIN transmitter state that hardware will generate an interrupt to CPU. Software can also monitor the LIN bus transfer state by checking the read back data in UART_DAT register. The following sequence is a program sequence example.

The procedure without software error monitoring in Master mode:

1. Fill Protected Identifier to PID (UART_LINCTL[31:24]).
2. Select the hardware transmission header field including "break field + sync field + protected identifier field" by setting HSEL (UART_LINCTL [23:22]) to "10".
3. Set SENDH (UART_LINCTL[8]) bit to 1 for requesting header transmission.
4. Wait until SENDH (UART_LINCTL[8]) bit cleared by hardware.
5. Wait until TXEMPTYF (UART_FIFOSTS[28]) set to 1 by hardware.

Note1: The default setting of break field is 12 dominant bits (break field) and 1 recessive bit break/sync delimiter. Setting BRKFL (UART_LINCTL [19:16]) and BSL (UART_LINCTL[21:20]) to change the LIN break field length and break/sync delimiter length.

Note2: The default setting of break/sync delimiter length is 1-bit time and the inter-byte spaces default setting is also 1-bit time. Setting BSL (UART_LINCTL[21:20]) and DLY(UART_TOUT[15:8]) can change break/sync delimiter length and inter-byte spaces.

Note3: If the header includes the "break field, sync field and frame ID field", software must fill frame ID to PID (UART_LINCTL[31:24]) before trigger header transmission (setting the SENDH (UART_LINCTL[8])). The frame ID parity can be generated by software or hardware depending on IDPEN (UART_LINCTL[9]) setting. If the parity generated by software with IDPEN (UART_LINCTL[9]) is set to '0', software must fill 8 bit data (include 2 bit parity) in this field. If the parity generated by hardware with IDPEN (UART_LINCTL[9]) is set to '1', software fills ID0~ID5 and hardware calculates P0 and P1.

Procedure with software error monitoring in Master mode:

1. Choose the hardware transmission header field to only include "break field" by setting HSEL (UART_LINCTL [23:22])] to '00'.
2. Enable break detection function by setting BRKDETEN (UART_LINCTL[10]).
3. Request break + break/sync delimiter transmission by setting the SENDH (UART_LINCTL[8]).
4. Wait until the BRKDETF (UART_LINSTS[8]) flag is set to "1" by hardware.
5. Request sync field transmission by writing 0x55 into UART_DAT register.
6. Wait until the RDAIF (UART_INTSTS[0]) is set to "1" by hardware and then read back the UART_DAT register.
7. Request header frame ID transmission by writing the protected identifier value to UART_DAT register.
8. Wait until the RDAIF (UART_INTSTS[0]) is set to "1" by hardware and then read back the UART_DAT register.

LIN Break and Delimiter Detection

When software enables the break detection function by setting BRKDETEN (UART_LINCTL[10]), the break detection circuit is activated. The break detection circuit is totally independent from the UART receiver.

When the break detection function is enabled, the circuit looks at the input UART_RX pin for a start signal. If UART LIN controller detects consecutive dominant is greater than 11 bits dominant followed by a recessive bit (delimiter), the BRKDETF (UART_LINSTS[8]) flag is set at the end of break field. If the LINIEN (UART_INTEN[8]) bit is set to 1, an interrupt LININT (UART_INTSTS[15]) will be generated. The behavior of the break detection and break flag are shown in Figure 6.19-19.

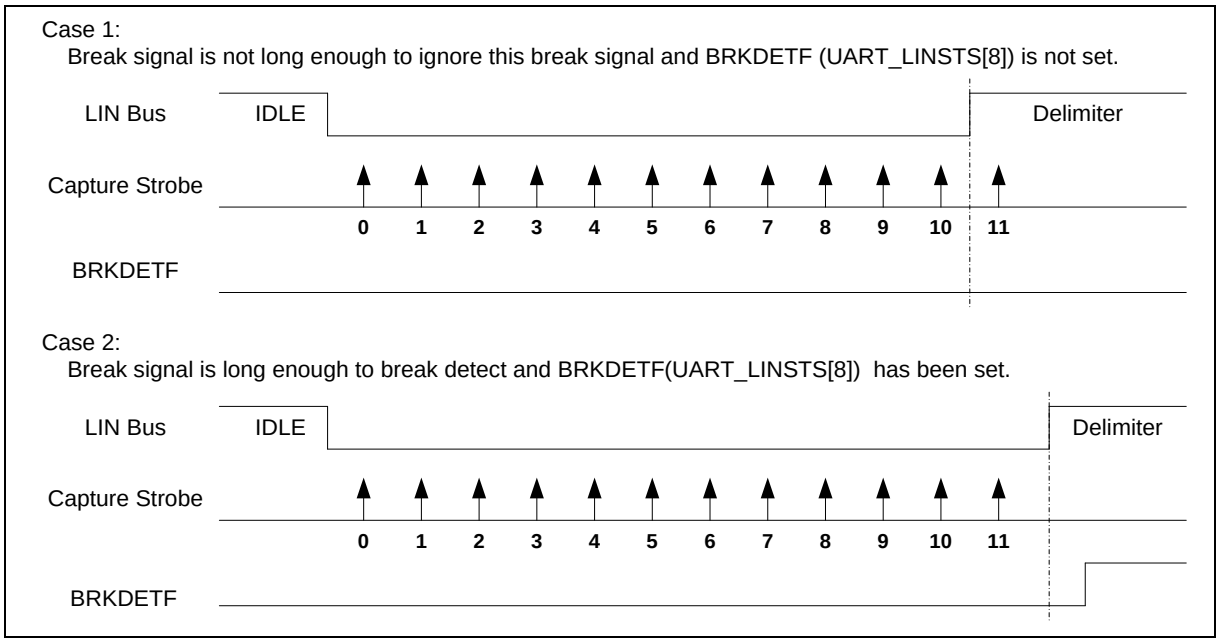


Figure 6.19-19 Break Detection in LIN Mode

LIN Frame ID and Parity Format

The LIN frame ID value in LIN function mode is shown, the frame ID parity can be generated by software or hardware depends on IDPEN (UART_LINCTL[9]).

If the parity generated by hardware (IDPEN (UART_LINCTL[9])=1), user fill ID0~ID5 (UART_LINCTL [29:24]) hardware will calculate P0 (UART_LINCTL[30]) and P1 (UART_LINCTL[31]) otherwise user must filled frame ID and parity in this field.

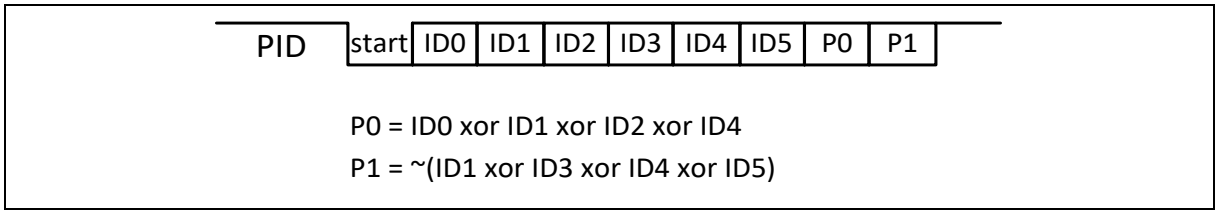


Figure 6.19-20 LIN Frame ID and Parity Format

LIN Slave Mode

The UART Controller supports LIN Slave mode. To enable and initialize the LIN Slave mode, the following steps are necessary:

1. Set the UART_BAUD register to select the desired baud rate.

2. Configure the data length to 8 bits by setting WLS (UART_LINE[1:0]) to '11' and disable parity check by clearing PBE (UART_LINE[3]) bit and configure with one stop bit by clearing NSB (UART_LINE[2]) bit.
3. Select LIN function mode by setting FUNCSEL (UART_FUNCSEL[2:0]) to '001'.
4. Enable LIN slave mode by setting the SLVEN (UART_LINCTL[0]) to 1.

LIN Header Reception

According to the LIN protocol, a slave node must wait for a valid header which comes from the master node. Next the slave task will take one of following actions (depend on the master header frame ID value).

- Receive the response.
- Transmit the response.
- Ignore the response and wait for next header.

In LIN Slave mode, user can enable the slave header detection function by setting the SLVHDEN (UART_LINCTL[1]) to detect complete frame header (receive "break field", "sync field" and "frame ID field"). When a LIN header is received, the SLVHDETF (UART_LINSTS[0]) flag will be set. If the LINIEN (UART_INTEN[8]) bit is set to 1, an interrupt will be generated. User can enable the frame ID parity check function by setting IDPEN (UART_LINCTL[9]). If only received frame ID parity is not correct (break and sync filed are correct), the SLVIDPEF (UART_LINSTS[2]) flag is set to '1'. If the LINIEN(UART_INTEN[8]) is set to 1, an interrupt will be generated and SLVHDETF (UART_LINSTS[0]) is set to '1'. User can also put LIN in mute mode by setting MUTE (UART_LINCTL[4]) to '1'. This mode allows detection of headers only (break + sync + frame ID) and prevents the reception of any other characters. In order to avoid bit rate tolerance, the controller supports automatic resynchronization function to avoid clock deviation error, user can enable this feature by setting SLVAREN (UART_LINCTL[2]).

LIN Response Transmission

The LIN slave node can transmit response and receive response. When slave node is the publisher of the response, the slave node sends response by filling data to the UART_DAT register. If the slave node is the subscriber of the response, the slave node receives data from LIN bus.

LIN Header Time-out Error

The LIN slave controller contains a header time-out counter. If the entire header is not received within the maximum time limit of 57 bit times, the header error flag SLVHEF (UART_LINSTS [1]) will be set. The time-out counter is enabled at each break detect edge and stopped in the following conditions.

- A LIN frame ID field has been received.
- The header error flag asserts.
- Writing 1 to the SLVSYNCF (UART_LINSTS[3]) to re-search a new frame header.

Mute Mode and LIN Exit from Mute Mode Condition

In Mute mode, a LIN slave node will not receive any data until specified condition occurred. It allows header detection only and prevents the reception of any other characters. User can enable Mute mode by setting the MUTE (UART_LINCTL[4]) and exiting from Mute mode condition can be selected by HSEL (UART_LINCTL[23:22]).

Note: It is recommended to set LIN slave node to Mute mode after checksum transmission.

The LIN slave controller exiting from Mute mode is described as follows: If HSEL (UART_LINCTL[23:22]) is set to "break field", when LIN slave controller detects a valid LIN break + delimiter, the controller will enable the receiver (exit from Mute mode) and subsequent data (sync data,

frame ID data, response data) are received in RX FIFO.

If HSEL (UART_LINCTL[23:22]) is set to “break field and sync field”, when the LIN slave controller detects a valid LIN break + delimiter followed by a valid sync field without frame error, the controller will enable the receiver (exit from mute mode) and subsequent data(ID data, response data) are received in RX FIFO. If HSEL (UART_LINCTL[23:22]) is set to “break field, sync field and ID field”, when the LIN slave controller detects a valid LIN break + delimiter and valid sync field without frame error followed by ID data without frame error and received ID data matched PID (UART_LINCTL[31:24]) value. The controller will enable the receiver (exit from mute mode) and subsequent data (response data) are received in RX FIFO.

Slave Mode Non-automatic Resynchronization (NAR)

User can disable the automatic resynchronization function to fix the communication baud rate. When operating in Non-Automatic Resynchronization mode, software needs some initial process, and the initialization process flow of Non-Automatic Resynchronization mode is shown as follows:

1. Select the desired baud rate by setting the UART_BAUD register.
2. Select LIN function mode by setting FUNCSEL (UART_FUNCSEL[2:0]) to ‘001’.
3. Disable automatic resynchronization function by setting SLVAREN (UART_LINCTL[2]) is set to 0.
4. Enable LIN slave mode by setting the SLVEN (UART_LINCTL[0]) is set to 1.

Slave Mode with Automatic Resynchronization (AR)

In Automatic Resynchronization (AR) mode, the controller will adjust the baud rate generator after each sync field reception. The initialization process flow of Automatic Resynchronization mode is shown as follows:

1. Select the desired baud rate by setting the UART_BAUD register.
2. Select LIN function mode by setting FUNCSEL (UART_FUNCSEL[2:0]) to ‘001’.
3. Enable automatic resynchronization function by setting SLVAREN (UART_LINCTL[2]) to ‘1’.
4. Enable LIN slave mode by setting the SLVEN (UART_LINCTL[0]) is set to ‘1’.

When the automatic resynchronization function is enabled, after each LIN break field, the time duration between five falling edges is sampled on peripheral clock and the result of this measurement is stored in an internal 13-bit register and the UART_BAUD register value will be automatically updated at the end of the fifth falling edge. If the measure timer (13-bit) overflows before five falling edges, then the header error flag SLVHEF (UART_LINSTS [1]) will be set.

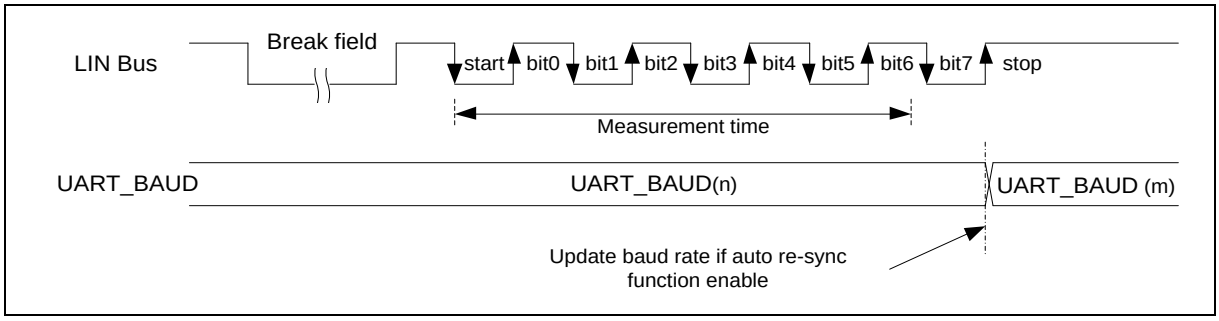


Figure 6.19-21 LIN Sync Field Measurement

When operating in Automatic Resynchronization (AR) mode, software must select the desired baud rate by setting the UART_BAUD register and hardware will store it at internal TEMP_REG register, after each LIN break field, the time duration between five falling edges is sampled on peripheral clock

and the result of this measurement is stored in an internal 13-bit register BAUD_LIN and the result will be updated to UART_BAUD register automatically.

To guarantee the transmission baud rate, the baud rate generator must reload the initial value before each new break reception. The initial value is programmed by the application during initialization (TEMP_REG). User can set SLVDUEN (UART_LINCTL [3]) to enable auto reload initial baud rate value function. If the SLVDUEN (UART_LINCTL [3]) is set, when received the next character, hardware will auto reload the initial value to UART_BAUD, and when the UART_BAUD be updated, the SLVDUEN (UART_LINCTL [3]) will be cleared automatically. The behavior of LIN updated method as shown Figure 6.19-22.

Note1: It is recommended to set the SLVDUEN bit before every checksum reception.

Note2: When a header error is detected, user must write 1 to SLVSYNCF (UART_LINSTS[3]) to re-search new frame header. When writing 1 to it, hardware will reload the initial baud rate TEMP_REG and re-search new frame header.

Note3: When operating in Automatic Resynchronization mode, the baud rate setting must be operated at mode2 (BAUDM1 (UART_BAUD [29]) and BAUDM0 (UART_BAUD[28]) must be 1).

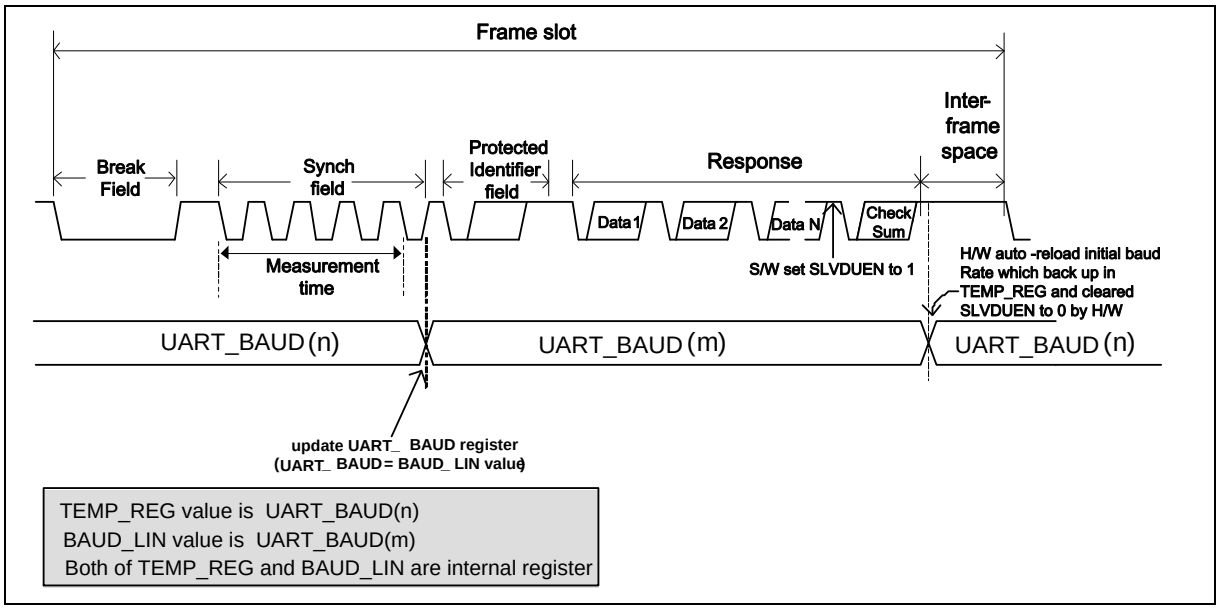


Figure 6.19-22 UART_BAUD Update Sequence in AR mode if SLVDUEN is 1

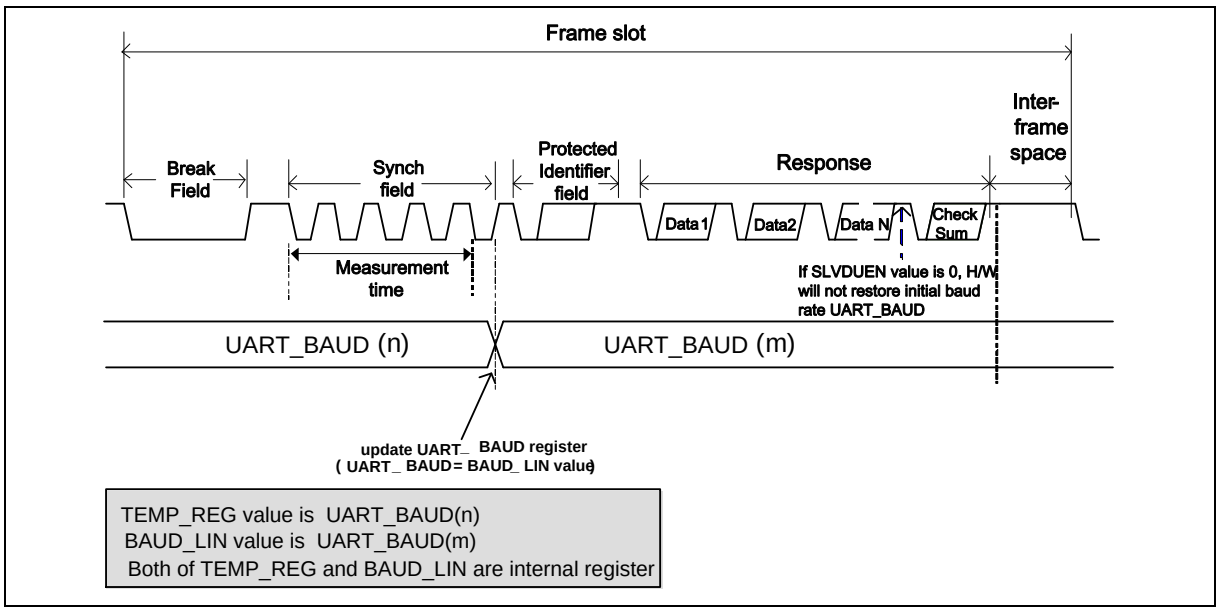


Figure 6.19-23 UART_BAUD Update Sequence in AR mode if SLVDUEN is 0

Deviation Error on the Sync Field

When operating in Automatic Resynchronization mode, the controller will check the deviation error on the sync field. The deviation error is checked by comparing the current baud rate with the received sync field. Two checks are performed in parallel.

Check1: Based on measurement between the first falling edge and the last falling edge of the sync field.

- If the difference is more than 14.84%, the header error flag SLVHEF (UART_LINSTS[1]) will be set.
- If the difference is less than 14.06%, the header error flag SLVHEF (UART_LINSTS[1]) will not be set.
- If the difference is between 14.84% and 14.06%, the header error flag SLVHEF (UART_LINSTS[1]) may either set or not.

Check2: Based on measurement of time between each falling edge of the sync field.

- If the difference is more than 18.75%, the header error flag SLVHEF (UART_LINSTS[1]) will be set.
- If the difference is less than 15.62%, the header error flag SLVHEF (UART_LINSTS[1]) will not be set.
- If the difference is between 18.75% and 15.62%, the header error flag SLVHEF (UART_LINSTS[1]) may either set or not.

Note: The deviation check is based on the current baud rate clock. Therefore, in order to guarantee correct deviation checking, the baud rate must reload the nominal value before each new break reception by setting SLVDUEN (UART_LINCTL[3]) register (It is recommend setting the SLVDUEN (UART_LINCTL[3]) bit before every checksum reception).

LIN Header Error Detection

In LIN Slave function mode, when user enables the header detection function by setting the SLVHDEN (UART_LINCTL[1]), hardware will handle the header detect flow. If the header has an error, the LIN header error flag SLVHEF (UART_LINSTS[1]) will be set and an interrupt is generated if the LINIEN

(UART_INTEN[8]) bit is set. When header error is detected, user must reset the detect circuit to re-search a new frame header by writing 1 to SLVSYNCF (UART_LINSTS[3]) to re-search a new frame header.

The LIN header error flag SLVHEF (UART_LINSTS[1]) is set if one of the following conditions occurs:

- Break Delimiter is too short (less than 0.5-bit time).
- Frame error in sync field or Identifier field.
- The sync field data is not 0x55 (Non-Automatic Resynchronization mode).
- The sync field deviation error (With Automatic Resynchronization mode).
- The sync field measure time-out (With Automatic Resynchronization mode).
- LIN header reception time-out.

6.19.5.11 RS-485 Function Mode

Another alternate function of UART controller is RS-485 function (user must set UART_FUNCSEL [2:0] to '011' to enable RS-485 function), and direction control provided by nRTS pin from an asynchronous serial port. The RS-485 transceiver control is implemented by using the nRTS control signal to enable the RS-485 driver. Many characteristics of the RX and TX are same as UART in RS-485 mode.

The UART controller can be configured as an RS-485 addressable slave and the RS-485 master transmitter will identify an address character by setting the parity (9-th bit) to 1. For data characters, the parity is set to 0. Software can use UART_LINE register to control the 9-th bit (When the PBE, EPE and SPE are set, the 9-th bit is transmitted 0 and when PBE and SPE are set and EPE is cleared, the 9-th bit is transmitted 1).

The controller supports three operation modes: RS-485 Normal Multidrop Operation Mode (NMM), RS-485 Auto Address Detection Operation Mode (AAD) and RS-485 Auto Direction Control Operation Mode (AUD). Software can choose any operation mode by programming the UART_ALTCTL register, and drive the transfer delay time between the last stop bit leaving the TX FIFO and the de-assertion of by setting DLY (UART_TOUT [15:8]) register.

RS-485 Normal Multidrop Operation Mode (NMM)

In RS-485 Normal Multidrop Operation Mode (RS485NMM (UART_ALTCTL[8]) = 1), in first, software must decide the data which before the address byte be detected will be stored in RX FIFO or not. If software wants to ignore any data before address byte detected, the flow is set RXOFF (UART_FIFO [8]) then enable RS485NMM (UART_ALTCTL [8]) and the receiver will ignore any data until an address byte is detected (bit 9 = 1) and the address byte data will be stored in the RX FIFO. If software wants to receive any data before address byte detected, the flow is disables RXOFF (UART_FIFO [8]) then enable RS485NMM (UART_ALTCTL [8]) and the receiver will received any data.

If an address byte is detected (bit 9 = 1), it will generate an interrupt to CPU and RXOFF (UART_FIFO [8]) can decide whether accepting the following data bytes are stored in the RX FIFO. If software disables receiver by setting RXOFF (UART_FIFO [8]) register, when a next address byte is detected, the controller will clear the RXOFF (UART_FIFO [8]) bit and the address byte data will be stored in the RX FIFO.

RS-485 Auto Address Detection Operation Mode (AAD)

In RS-485 Auto Address Detection Operation Mode (RS485AAD (UART_ALTCTL[9]) = 1), the receiver will ignore any data until an address byte is detected (bit 9 = 1) and the address byte data matches the ADDRMV (UART_ALTCTL[31:24]) value. The address byte data will be stored in the RX FIFO. The all received byte data will be accepted and stored in the RX FIFO until an address byte data not match the ADDRMV (UART_ALTCTL[31:24]) value.

RS-485 Auto Direction Function (AUD)

Another option function of RS-485 controllers is RS-485 auto direction control function (RS485AUD (UART_ALTCTL[10) = 1). The RS-485 transceiver control is implemented by using the nRTS control signal from an asynchronous serial port. The nRTS line is connected to the RS-485 transceiver enable pin such that setting the nRTS line to high (logic 1) enables the RS-485 transceiver. Setting the nRTS line to low (logic 0) puts the transceiver into the tri-state condition to disabled. User can set RTSACTLV in UART_MODEM register to change the nRTS driving level.

Figure 6.19-24 demonstrates the RS-485 nRTS driving level in AUD mode. The nRTS pin will be automatically driven during TX data transmission.

Setting RTSACTLV(UART_MODEM[9]) can control nRTS pin output driving level. User can read the RTSSTS(UART_MODEM[13]) bit to get real nRTS pin output voltage logic status.

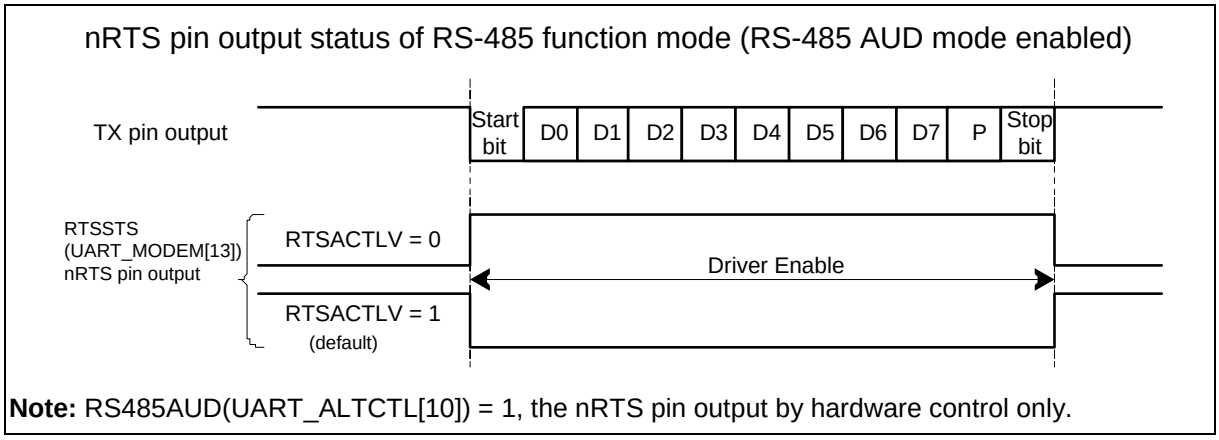


Figure 6.19-24 RS-485 nRTS Driving Level in Auto Direction Mode

Figure 6.19-25 demonstrates the RS-485 nRTS driving level in software control (RS485AUD (UART_ALTCTL[10])=0). The nRTS driving level is controlled by programing the RTS(UART_MODEM[1]) control bit.

Setting RTSACTLV (UART_MODEM[9]) can control the nRTS pin output is inverse or non-inverse from RTS(UART_MODEM[1]) control bit. User can read the RTSSTS (UART_MODEM[13]) bit to get real nRTS pin output voltage logic status. The structure of RS-485 frame is shown in Figure 6.19-26.

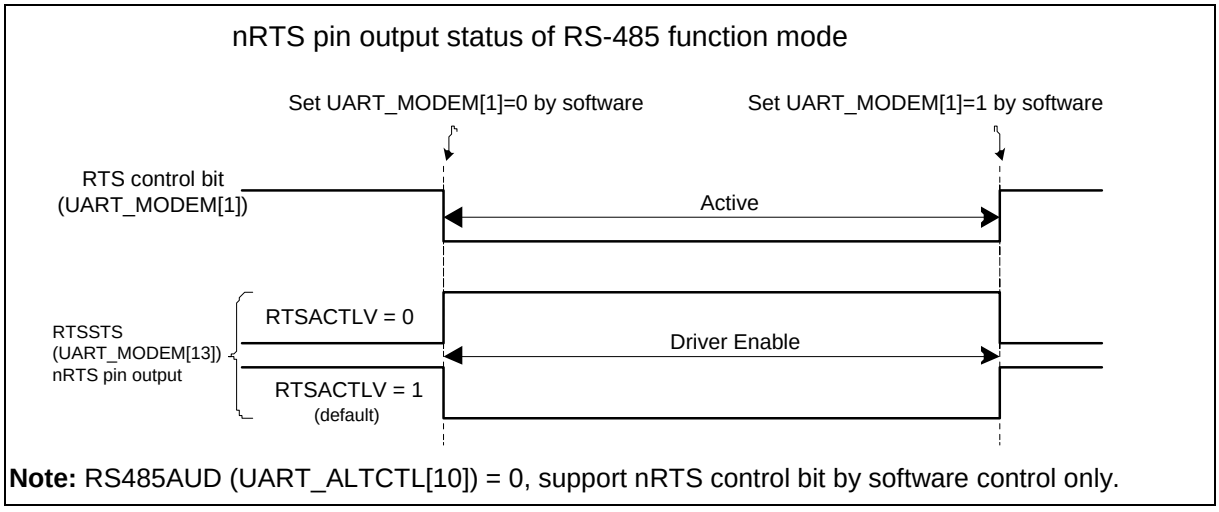
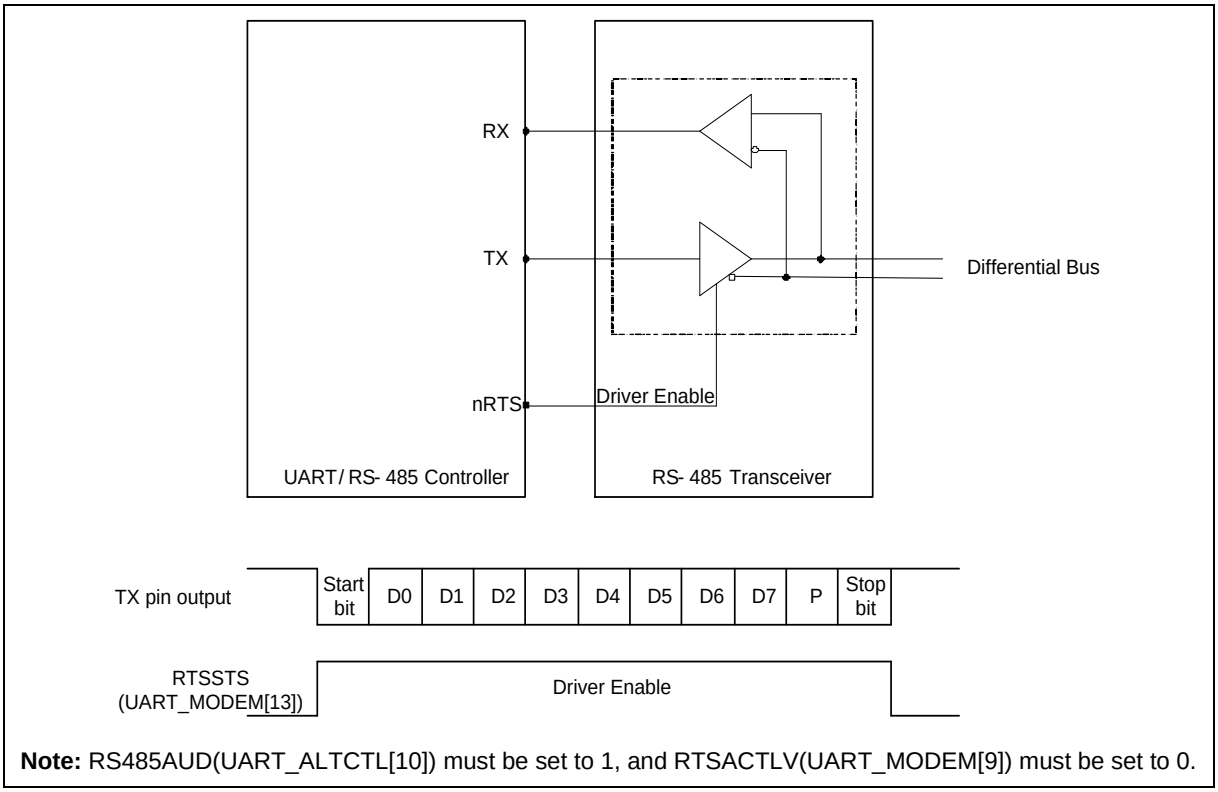


Figure 6.19-25 RS-485 nRTS Driving Level with Software Control

Programming Sequence Example:

1. Program FUNCSEL in UART_FUNCSEL to select RS-485 function.
2. Program the RXOFF (UART_FIFO[8]) to determine enable or disable the receiver RS-485 receiver.
3. Program the RS485NMM (UART_ALTCTL[8]) or RS485AAD (UART_ALTCTL[9]) mode.
4. If the RS485AAD (UART_ALTCTL[9]) mode is selected, the ADDRMV (UART_ALTCTL[31:24]) is programmed for auto address match value.
5. Determine auto direction control by programming RS485AUD (UART_ALTCTL[10]).



Note: RS485AUD(UART_ALTCTL[10]) must be set to 1, and RTSACTLV(UART_MODEM[9]) must be set to 0.

Figure 6.19-26 Structure of RS-485 Frame

6.19.5.12 UART Single-wire Half Duplex

The UART controller provides single-wire half duplex function in UART function mode. Setting UART_FUNCSEL [2:0] to '3'b100' to enable the UART Single-wire function. The single-wire bus is idle (RXIDLE(UART_FIFOSTS[29]) = 1) in RX state. Before writing data to transmit buffer (UART_DAT[7:0]), the bus state should be checked in idle (RXIDLE(UART_FIFOSTS[29]). By writing data to transmit buffer, the bus state transfers to TX state immediately. After the transmission, the bus state transfers from TX state to RX state.

The UART will not allowed to receive data in single-wire half duplex function TX mode. If nRTS is asserted in TX mode, nRTS will make the docking UART device send out data and cause bus confliction. To reduce the bus confliction, the UART controller supports flow control function and bit error detection but do not support auto-flow control function. The nRTS is automatically inactivated in single-wire TX state. In TX state, the UART controller will monitor bus state. If the bus state is not equal to TX state, the SWBEIF(UART_INTSTS[16]) is set.

6.19.5.13 PDMA Transfer Function

The UART controller supports PDMA transfer function.

By configuring PDMA parameter and set UART_DAT as the PDMA destination address. When TXPDMAEN (UART_INTEN[14]) is set to 1, the controller will issue request to PDMA controller to start the PDMA transmission process automatically.

By configuring PDMA parameter and set UART_DAT as the PDMA source address. When RXPDMAEN (UART_INTEN[15]) is set to 1, the controller will start the PDMA reception process. UART controller will issue request to PDMA controller automatically when there is data in the RX FIFO buffer.

Note: If STOPn (PDMA_STOP[n]) is set to stop UART RXPDMA task and the UART receive is not finish. UART controller will complete the transfer and stored current receive data in receive buffer. By reading RXEMPTY (UART_FIFOSTS[14]) to check there is valid data in receive buffer or not.

6.19.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
UART Base Address: **UARTx_BA = 0x4007_0000 + (0x1000 * x)** **x=0,1,2,3,4,5** **UART non-secure base address is UARTx_BA + 0x1000_0000.**				
UART_DAT **x=0,1,2,3,4,5**	UARTx_BA+0x00	R/W	UART Receive/Transmit Buffer Register	Undefined
UART_INTEN **x=0,1,2,3,4,5**	UARTx_BA+0x04	R/W	UART Interrupt Enable Register	0x0000_0000
UART_FIFO **x=0,1,2,3,4,5**	UARTx_BA+0x08	R/W	UART FIFO Control Register	0x0000_0101
UART_LINE **x=0,1,2,3,4,5**	UARTx_BA+0x0C	R/W	UART Line Control Register	0x0000_0000
UART_MODEM **x=0,1,2,3,4,5**	UARTx_BA+0x10	R/W	UART Modem Control Register	0x0000_0200
UART_MODEMSTS **x=0,1,2,3,4,5**	UARTx_BA+0x14	R/W	UART Modem Status Register	0x0000_0110
UART_FIFOSTS **x=0,1,2,3,4,5**	UARTx_BA+0x18	R/W	UART FIFO Status Register	0xB040_4000
UART_INTSTS **x=0,1,2,3,4,5**	UARTx_BA+0x1C	R/W	UART Interrupt Status Register	0x0040_0002
UART_TOUT **x=0,1,2,3,4,5**	UARTx_BA+0x20	R/W	UART Time-out Register	0x0000_0000
UART_BAUD **x=0,1,2,3,4,5**	UARTx_BA+0x24	R/W	UART Baud Rate Divider Register	0x0F00_0000
UART_IRDA **x=0,1,2,3,4,5**	UARTx_BA+0x28	R/W	UART IrDA Control Register	0x0000_0040
UART_ALTCTL **x=0,1,2,3,4,5**	UARTx_BA+0x2C	R/W	UART Alternate Control/Status Register	0x0000_000C
UART_FUNCSEL **x=0,1,2,3,4,5**	UARTx_BA+0x30	R/W	UART Function Select Register	0x0000_0000
UART_LINCTL **x=0,1**	UARTx_BA+0x34	R/W	UART LIN Control Register	0x000C_0000
UART_LINSTS **x=0,1**	UARTx_BA+0x38	R/W	UART LIN Status Register	0x0000_0000

UART_BRCOMP **x=0,1,2,3,4,5**	UARTx_BA+0x3C	R/W	UART Baud Rate Compensation Register	0x0000_0000
UART_WKCTL **x=0,1,2,3,4,5**	UARTx_BA+0x40	R/W	UART Wake-up Control Register	0x0000_0000
UART_WKSTS **x=0,1,2,3,4,5**	UARTx_BA+0x44	R/W	UART Wake-up Status Register	0x0000_0000
UART_DWKCOMP **x=0,1,2,3,4,5**	UARTx_BA+0x48	R/W	UART Incoming Data Wake-up Compensation Register	0x0000_0000

6.19.7 Register Description

UART Receive/Transmit Buffer Register (UART_DAT)

Register	Offset	R/W	Description	Reset Value
UART_DAT **x=0,1,2,3,4,5**	UARTx_BA+0x00	R/W	UART Receive/Transmit Buffer Register	Undefined

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

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**PARITY**	**Parity Bit Receive/Transmit Buffer** Write Operation: By writing to this bit, the parity bit will be stored in transmitter FIFO. If PBE (UART_LINE[3]) and PSS (UART_LINE[7]) are set, the UART controller will send out this bit follow the DAT (UART_DAT[7:0]) through the UART_TXD. Read Operation: If PBE (UART_LINE[3]) and PSS (UART_LINE[7]) are enabled, the parity bit can be read by this bit. **Note:** This bit has effect only when PBE (UART_LINE[3]) and PSS (UART_LINE[7]) are set.
[7:0]	**DAT**	**Data Receive/Transmit Buffer** Write Operation: By writing one byte to this register, the data byte will be stored in transmitter FIFO. The UART controller will send out the data stored in transmitter FIFO top location through the UART_TXD. Read Operation: By reading this register, the UART controller will return an 8-bit data received from receiver FIFO.

UART Interrupt Enable Register (UART_INTEN)

Register	Offset	R/W	Description	Reset Value
UART_INTEN x=0,1,2,3,4,5	UARTx_BA+0x04	R/W	UART Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved	TXENDIEN	Reserved			ABRIEN	Reserved	SWBEIEN
15	14	13	12	11	10	9	8
RXPDMAEN	TXPDMAEN	ATOCTSEN	ATORTSEN	TOCNTEN	Reserved		LINIEN
7	6	5	4	3	2	1	0
Reserved	WKIEN	BUFERRIEN	RXTOIEN	MODEMIEN	RLSIEN	THREIEN	RDAIEN

Bits	Description	
[31:23]	**Reserved**	Reserved.
[22]	**TXENDIEN**	**Transmitter Empty Interrupt Enable Bit** If TXENDIEN (UART_INTEN[22]) is enabled, the Transmitter Empty interrupt TXENDINT (UART_INTSTS[30]) will be generated when TXENDIF (UART_INTSTS[22]) is set (TX FIFO (UART_DAT) is empty and the STOP bit of the last byte has been transmitted). 0 = Transmitter empty interrupt Disabled. 1 = Transmitter empty interrupt Enabled.
[21:19]	**Reserved**	Reserved.
[18]	**ABRIEN**	**Auto-baud Rate Interrupt Enable Bit** 0 = Auto-baud rate interrupt Disabled. 1 = Auto-baud rate interrupt Enabled.
[17]	**Reserved**	Reserved.
[16]	**SWBEIEN**	**Single-wire Bit Error Detection Interrupt Enable Bit** Set this bit, the Single-wire Half Duplex Bit Error Detection Interrupt SWBEINT(UART_INTSTS[24]) is generated when Single-wire Bit Error Detection SWBEIF(UART_INTSTS[16]) is set. 0 = Single-wire Bit Error Detect Interrupt Disabled. 1 = Single-wire Bit Error Detect Interrupt Enabled. **Note:** This bit is valid when FUNCSEL (UART_FUNCSEL[2:0]) is select UART Single-wire mode.
[15]	**RXPDMAEN**	**RX PDMA Enable Bit** This bit can enable or disable RX PDMA service. 0 = RX PDMA Disabled. 1 = RX PDMA Enabled. **Note:** If RLSIEN (UART_INTEN[2]) is enabled and HWRLSINT (UART_INTSTS[26]) is set to 1, the RLS (Receive Line Status) Interrupt is caused. If RLS interrupt is caused by Break Error Flag BIF(UART_FIFOSTS[6]), Frame Error Flag FEF(UART_FIFO[5]) or Parity Error Flag PEF(UART_FIFOSTS[4]), UART PDMA receive request operation is stopped. Clear Break Error Flag BIF or Frame Error Flag FEF or Parity Error Flag PEF by writing “1” to corresponding BIF, FEF and PEF to

		make UART PDMA receive request operation continue.
[14]	**TXPDMAEN**	**TX PDMA Enable Bit** 0 = TX PDMA Disabled. 1 = TX PDMA Enabled. **Note:** If RLSIEN (UART_INTEN[2]) is enabled and HWRLSINT (UART_INTSTS[26]) is set to 1, the RLS (Receive Line Status) Interrupt is caused. If RLS interrupt is caused by Break Error Flag BIF(UART_FIFOSTS[6]), Frame Error Flag FEF(UART_FIFO[5]) or Parity Error Flag PEF(UART_FIFOSTS[4]), UART PDMA transmit request operation is stopped. Clear Break Error Flag BIF or Frame Error Flag FEF or Parity Error Flag PEF by writing “1” to corresponding BIF, FEF and PEF to make UART PDMA transmit request operation continue.
[13]	**ATOCTSEN**	**nCTS Auto-flow Control Enable Bit** 0 = nCTS auto-flow control Disabled. 1 = nCTS auto-flow control Enabled. **Note:** When nCTS auto-flow is enabled, the UART will send data to external device if nCTS input assert (UART will not send data to device until nCTS is asserted).
[12]	**ATORTSEN**	**nRTS Auto-flow Control Enable Bit** 0 = nRTS auto-flow control Disabled. 1 = nRTS auto-flow control Enabled. **Note:** When nRTS auto-flow is enabled, if the number of bytes in the RX FIFO equals the RTSTRGLV (UART_FIFO[19:16]), the UART will de-assert nRTS signal.
[11]	**TOCNTEN**	**Receive Buffer Time-out Counter Enable Bit** 0 = Receive Buffer Time-out counter Disabled. 1 = Receive Buffer Time-out counter Enabled.
[10:9]	**Reserved**	Reserved.
[8]	**LINIEN**	**LIN Bus Interrupt Enable Bit** 0 = LIN bus interrupt Disabled. 1 = LIN bus interrupt Enabled. **Note:** This bit is used for LIN function mode.
[7]	**Reserved**	Reserved.
[6]	**WKIEN**	**Wake-up Interrupt Enable Bit** 0 = Wake-up Interrupt Disabled. 1 = Wake-up Interrupt Enabled.
[5]	**BUFERRIEN**	**Buffer Error Interrupt Enable Bit** 0 = Buffer error interrupt Disabled. 1 = Buffer error interrupt Enabled.
[4]	**RXTOIEN**	**RX Time-out Interrupt Enable Bit** 0 = RX time-out interrupt Disabled. 1 = RX time-out interrupt Enabled.
[3]	**MODEMIEN**	**Modem Status Interrupt Enable Bit** 0 = Modem status interrupt Disabled. 1 = Modem status interrupt Enabled.
[2]	**RLSIEN**	**Receive Line Status Interrupt Enable Bit** 0 = Receive Line Status interrupt Disabled. 1 = Receive Line Status interrupt Enabled.
[1]	**THREIEN**	**Transmit Holding Register Empty Interrupt Enable Bit**

		0 = Transmit holding register empty interrupt Disabled. 1 = Transmit holding register empty interrupt Enabled.
[0]	**RDAIEN**	**Receive Data Available Interrupt Enable Bit** 0 = Receive data available interrupt Disabled. 1 = Receive data available interrupt Enabled.

UART FIFO Control Register (UART_FIFO)

Register	Offset	R/W	Description	Reset Value
UART_FIFO **x=0,1,2,3,4,5**	UARTx_BA+0x08	R/W	UART FIFO Control Register	0x0000_0101

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved				**RTSTRGLV**			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							**RXOFF**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RFITL				**Reserved**	**TXRST**	**RXRST**	**Reserved**

Bits	Description	
[31:20]	**Reserved**	Reserved.
[19:16]	**RTSTRGLV**	**nRTS Trigger Level for Auto-flow Control** 0000 = nRTS Trigger Level is 1 byte. 0001 = nRTS Trigger Level is 4 bytes. 0010 = nRTS Trigger Level is 8 bytes. 0011 = nRTS Trigger Level is 14 bytes. Others = Reserved. **Note:** This field is used for auto nRTS flow control.
[15:9]	**Reserved**	Reserved.
[8]	**RXOFF**	**Receiver Disable Bit** The receiver is disabled or not (set 1 to disable receiver). 0 = Receiver Enabled. 1 = Receiver Disabled. **Note:** This bit is used for RS-485 Normal Multi-drop mode. It should be programmed before RS485NMM (UART_ALTCTL [8]) is programmed.
[7:4]	**RFITL**	**RX FIFO Interrupt Trigger Level** When the number of bytes in the receive FIFO equals the RFITL, the RDAIF (UART_INTSTS[0]) will be set (if RDAIEN (UART_INTEN [0]) enabled, and an interrupt will be generated). 0000 = RX FIFO Interrupt Trigger Level is 1 byte. 0001 = RX FIFO Interrupt Trigger Level is 4 bytes. 0010 = RX FIFO Interrupt Trigger Level is 8 bytes. 0011 = RX FIFO Interrupt Trigger Level is 14 bytes. Others = Reserved.
[3]	**Reserved**	Reserved.
[2]	**TXRST**	**TX Field Software Reset**

		When TXRST (UART_FIFO[2]) is set, all the byte in the transmit FIFO and TX internal state machine are cleared. 0 = No effect. 1 = Reset the TX internal state machine and pointers. **Note 1:** This bit will automatically clear at least 3 UART peripheral clock cycles. **Note 2:** Before setting this bit, it should wait for the TXEMPTYF (UART_FIFOSTS[28]) be set.
[1]	**RXRST**	**RX Field Software Reset** When RXRST (UART_FIFO[1]) is set, all the byte in the receiver FIFO and RX internal state machine are cleared. 0 = No effect. 1 = Reset the RX internal state machine and pointers. **Note 1:** This bit will automatically clear at least 3 UART peripheral clock cycles. **Note 2:** Before setting this bit, it should wait for the RXIDLE (UART_FIFOSTS[29]) be set.
[0]	**Reserved**	Reserved.

UART Line Control Register (UART_LINE)

Register	Offset	R/W	Description	Reset Value
UART_LINE **x=0,1,2,3,4,5**	UARTx_BA+0x0C	R/W	UART Line Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved						**RXDINV**	**TXDINV**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PSS	**BCB**	**SPE**	**EPE**	**PBE**	**NSB**	**WLS**	

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**RXDINV**	**RX Data Inverted** 0 = Received data signal inverted Disabled. 1 = Received data signal inverted Enabled. **Note 1:** Before setting this bit, TXRXDIS (UART_FUNCSEL[3]) should be set then waited for TXRXACT (UART_FIFOSTS[31]) is cleared. When the configuration is done, cleared TXRXDIS (UART_FUNCSEL[3]) to activate UART controller. **Note 2:** This bit is valid when FUNCSEL (UART_FUNCSEL[2:0]) is select UART, LIN or RS485 function.
[8]	**TXDINV**	**TX Data Inverted** 0 = Transmitted data signal inverted Disabled. 1 = Transmitted data signal inverted Enabled. **Note 1:** Before setting this bit, TXRXDIS (UART_FUNCSEL[3]) should be set then waited for TXRXACT (UART_FIFOSTS[31]) is cleared. When the configuration is done, cleared TXRXDIS (UART_FUNCSEL[3]) to activate UART controller. **Note 2:** This bit is valid when FUNCSEL (UART_FUNCSEL[2:0]) is select UART, LIN or RS485 function.
[7]	**PSS**	**Parity Bit Source Selection** The parity bit can be selected to be generated and checked automatically or by software. 0 = Parity bit is generated by EPE (UART_LINE[4]) and SPE (UART_LINE[5]) setting and checked automatically. 1 = Parity bit generated and checked by software. **Note 1:** This bit has effect only when PBE (UART_LINE[3]) is set. **Note 2:** If PSS is 0, the parity bit is transmitted and checked automatically. If PSS is 1, the transmitted parity bit value can be determined by writing PARITY (UART_DAT[8]) and the parity bit can be read by reading PARITY (UART_DAT[8]).
[6]	**BCB**	**Break Control Bit** 0 = Break Control Disabled. 1 = Break Control Enabled.

		Note: When this bit is set to logic 1, the transmitted serial data output (TX) is forced to the Spacing State (logic 0). This bit acts only on TX line and has no effect on the transmitter logic.
[5]	**SPE**	**Stick Parity Enable Bit** 0 = Stick parity Disabled. 1 = Stick parity Enabled. **Note:** If PBE (UART_LINE[3]) and EPE (UART_LINE[4]) are logic 1, the parity bit is transmitted and checked as logic 0. If PBE (UART_LINE[3]) is 1 and EPE (UART_LINE[4]) is 0 then the parity bit is transmitted and checked as 1.
[4]	**EPE**	**Even Parity Enable Bit** 0 = Odd number of logic 1's is transmitted and checked in each word. 1 = Even number of logic 1's is transmitted and checked in each word. **Note:** This bit has effect only when PBE (UART_LINE[3]) is set.
[3]	**PBE**	**Parity Bit Enable Bit** 0 = Parity bit generated Disabled. 1 = Parity bit generated Enabled. **Note:** Parity bit is generated on each outgoing character and is checked on each incoming data.
[2]	**NSB**	**Number of "STOP Bit"** 0 = One "STOP bit" is generated in the transmitted data. 1 = When select 5-bit word length, 1.5 "STOP bit" is generated in the transmitted data. When select 6-, 7- and 8-bit word length, 2 "STOP bit" is generated in the transmitted data.
[1:0]	**WLS**	**Word Length Selection** This field sets UART word length. 00 = 5 bits. 01 = 6 bits. 10 = 7 bits. 11 = 8 bits.

UART Modem Control Register (UART_MODEM)

Register	Offset	R/W	Description	Reset Value
UART_MODEM **x=0,1,2,3,4,5**	UARTx_BA+0x10	R/W	UART Modem Control Register	0x0000_0200

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

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**RTSSTS**	**nRTS Pin Status (Read Only)** This bit mirror from nRTS pin output of voltage logic status. 0 = nRTS pin output is low level voltage logic state. 1 = nRTS pin output is high level voltage logic state.
[12:10]	**Reserved**	Reserved.
[9]	**RTSACTLV**	**nRTS Pin Active Level** This bit defines the active level state of nRTS pin output. 0 = nRTS pin output is high level active. 1 = nRTS pin output is low level active. (Default) **Note 1:** Refer to Figure 6.19-13 and Figure 6.19-14 for UART function mode. **Note 2:** Refer to Figure 6.19-24 and Figure 6.19-25 for RS-485 function mode. **Note 3:** Before setting this bit, TXRXDIS (UART_FUNCSEL[3]) should be set then waited for TXRXACT (UART_FIFOSTS[31]) is cleared. When the configuration is done, cleared TXRXDIS (UART_FUNCSEL[3]) to activate UART controller.
[8:2]	**Reserved**	Reserved.
[1]	**RTS**	**nRTS (Request-to-send) Signal Control** This bit is direct control internal nRTS signal active or not, and then drive the nRTS pin output with RTSACTLV bit configuration. 0 = nRTS signal is active. 1 = nRTS signal is inactive. **Note 1:** The nRTS signal control bit is not effective when nRTS auto-flow control is enabled in UART function mode. **Note 2:** The nRTS signal control bit is not effective when RS-485 auto direction mode (AUD) is enabled in RS-485 function mode. **Note 3:** Single-wire mode is support this feature.

[0]	**Reserved**	Reserved.

UART Modem Status Register (UART_MODEMSTS)

Register	Offset	R/W	Description	Reset Value
UART_MODEMSTS **x=0,1,2,3,4,5**	UARTx_BA+0x14	R/W	UART Modem Status Register	0x0000_0110

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							**CTSACTLV**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			**CTSSTS**	**Reserved**			**CTSDETF**

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**CTSACTLV**	**nCTS Pin Active Level** This bit defines the active level state of nCTS pin input. 0 = nCTS pin input is high level active. 1 = nCTS pin input is low level active. (Default) **Note:** Before setting this bit, TXRXDIS (UART_FUNCSEL[3]) should be set then waited for TXRXACT (UART_FIFOSTS[31]) is cleared. When the configuration is done, cleared TXRXDIS (UART_FUNCSEL[3]) to activate UART controller.
[7:5]	**Reserved**	Reserved.
[4]	**CTSSTS**	**nCTS Pin Status (Read Only)** This bit mirror from nCTS pin input of voltage logic status. 0 = nCTS pin input is low level voltage logic state. 1 = nCTS pin input is high level voltage logic state. **Note:** This bit echoes when UART controller peripheral clock is enabled, and nCTS multi-function port is selected.
[3:1]	**Reserved**	Reserved.
[0]	**CTSDETF**	**Detect nCTS State Change Flag** This bit is set whenever nCTS input has change state, and it will generate Modem interrupt to CPU when MODEMIEN (UART_INTEN [3]) is set to 1. 0 = nCTS input has not change state. 1 = nCTS input has change state. **Note:** This bit can be cleared by writing “1” to it.

UART FIFO Status Register (UART_FIFOSTS)

Register	Offset	R/W	Description	Reset Value
UART_FIFOSTS x=0,1,2,3,4,5	UARTx_BA+0x18	R/W	UART FIFO Status Register	0xB040_4000

31	30	29	28	27	26	25	24
TXRXACT	Reserved	RXIDLE	TXEMPTYF	Reserved			TXOVIF
23	22	21	20	19	18	17	16
TXFULL	TXEMPTY	TXPTR					
15	14	13	12	11	10	9	8
RXFULL	RXEMPTY	RXPTR					
7	6	5	4	3	2	1	0
Reserved	BIF	FEF	PEF	ADDRDETF	ABRDTOIF	ABRDIF	RXOVIF

Bits	Description	
[31]	**TXRXACT**	**TX and RX Active Status (Read Only)** This bit indicates TX and RX are active or inactive. 0 = TX and RX are inactive. 1 = TX and RX are active. (Default) **Note:** When TXRXDIS (UART_FUNCSEL[3]) is set and both TX and RX are in idle state, this bit is cleared. The UART controller cannot transmit or receive data at this moment. Otherwise this bit is set.
[30]	**Reserved**	Reserved.
[29]	**RXIDLE**	**RX Idle Status (Read Only)** This bit is set by hardware when RX is idle. 0 = RX is busy. 1 = RX is idle. (Default)
[28]	**TXEMPTYF**	**Transmitter Empty Flag (Read Only)** This bit is set by hardware when TX FIFO (UART_DAT) is empty and the STOP bit of the last byte has been transmitted. 0 = TX FIFO is not empty or the STOP bit of the last byte has been not transmitted. 1 = TX FIFO is empty and the STOP bit of the last byte has been transmitted. **Note:** This bit is cleared automatically when TX FIFO is not empty or the last byte transmission has not completed.
[27:25]	**Reserved**	Reserved.
[24]	**TXOVIF**	**TX Overflow Error Interrupt Flag** If TX FIFO (UART_DAT) is full, an additional write to UART_DAT will cause this bit to logic 1. 0 = TX FIFO is not overflow. 1 = TX FIFO is overflow. **Note:** This bit can be cleared by writing “1” to it.
[23]	**TXFULL**	**Transmitter FIFO Full (Read Only)**

		This bit indicates TX FIFO full or not. 0 = TX FIFO is not full. 1 = TX FIFO is full. **Note:** This bit is set when the number of usage in TX FIFO Buffer is equal to 16, otherwise it is cleared by hardware.
[22]	**TXEMPTY**	**Transmitter FIFO Empty (Read Only)** This bit indicates TX FIFO empty or not. 0 = TX FIFO is not empty. 1 = TX FIFO is empty. **Note:** When the last byte of TX FIFO has been transferred to Transmitter Shift Register, hardware sets this bit high. It will be cleared when writing data into UART_DAT (TX FIFO not empty).
[21:16]	**TXPTR**	**TX FIFO Pointer (Read Only)** This field indicates the TX FIFO Buffer Pointer. When CPU writes one byte into UART_DAT, TXPTR increases one. When one byte of TX FIFO is transferred to Transmitter Shift Register, TXPTR decreases one. The Maximum value shown in TXPTR is 15. When the using level of TX FIFO Buffer equal to 16, the TXFULL bit is set to 1 and TXPTR will show 0. As one byte of TX FIFO is transferred to Transmitter Shift Register, the TXFULL bit is cleared to 0 and TXPTR will show 15.
[15]	**RXFULL**	**Receiver FIFO Full (Read Only)** This bit initiates RX FIFO full or not. 0 = RX FIFO is not full. 1 = RX FIFO is full. **Note:** This bit is set when the number of usage in RX FIFO Buffer is equal to 16, otherwise it is cleared by hardware.
[14]	**RXEMPTY**	**Receiver FIFO Empty (Read Only)** This bit initiate RX FIFO empty or not. 0 = RX FIFO is not empty. 1 = RX FIFO is empty. **Note:** When the last byte of RX FIFO has been read by CPU, hardware sets this bit high. It will be cleared when UART receives any new data.
[13:8]	**RXPTR**	**RX FIFO Pointer (Read Only)** This field indicates the RX FIFO Buffer Pointer. When UART receives one byte from external device, RXPTR increases one. When one byte of RX FIFO is read by CPU, RXPTR decreases one. The Maximum value shown in RXPTR is 15. When the using level of RX FIFO Buffer equal to 16, the RXFULL bit is set to 1 and RXPTR will show 0. As one byte of RX FIFO is read by CPU, the RXFULL bit is cleared to 0 and RXPTR will show 15.
[7]	**Reserved**	Reserved.
[6]	**BIF**	**Break Interrupt Flag** This bit is set to logic 1 whenever the received data input (RX) is held in the “spacing state” (logic 0) for longer than a full word transmission time (that is, the total time of “start bit” + data bits + parity + stop bits). 0 = No Break interrupt is generated. 1 = Break interrupt is generated. **Note:** This bit can be cleared by writing “1” to it.
[5]	**FEF**	**Framing Error Flag** This bit is set to logic 1 whenever the received character does not have a valid “stop bit” (that is, the stop bit following the last data bit or parity bit is detected as logic 0). 0 = No framing error is generated. 1 = Framing error is generated. **Note:** This bit can be cleared by writing “1” to it.

[4]	**PEF**	**Parity Error Flag** This bit is set to logic 1 whenever the received character does not have a valid “parity bit”. 0 = No parity error is generated. 1 = Parity error is generated. **Note:** This bit can be cleared by writing “1” to it.
[3]	**ADDRDETF**	**RS-485 Address Byte Detect Flag** 0 = Receiver detects a data that is not an address bit (bit 9 =’0’). 1 = Receiver detects a data that is an address bit (bit 9 =’1’). **Note 1:** This field is used for RS-485 function mode and ADDRDEN (UART_ALTCTL[15]) is set to 1 to enable Address detection mode. **Note 2:** This bit can be cleared by writing “1” to it.
[2]	**ABRDTOIF**	**Auto-baud Rate Detect Time-out Interrupt Flag** This bit is set to logic “1” in Auto-baud Rate Detect mode when the baud rate counter is overflow. 0 = Auto-baud rate counter is underflow. 1 = Auto-baud rate counter is overflow. **Note:** This bit can be cleared by writing “1” to it.
[1]	**ABRDIF**	**Auto-baud Rate Detect Interrupt Flag** This bit is set to logic “1” when auto-baud rate detect function is finished. 0 = Auto-baud rate detect function is not finished. 1 = Auto-baud rate detect function is finished. **Note:** This bit can be cleared by writing “1” to it.
[0]	**RXOVIF**	**RX Overflow Error Interrupt Flag** This bit is set when RX FIFO overflow. If the number of bytes of received data is greater than RX_FIFO (UART_DAT) size 16 bytes, this bit will be set. 0 = RX FIFO is not overflow. 1 = RX FIFO is overflow. **Note:** This bit can be cleared by writing “1” to it.

UART Interrupt Status Register (UART_INTSTS)

Register	Offset	R/W	Description	Reset Value
UART_INTSTS x=0,1,2,3,4,5	UARTx_BA+0x1C	R/W	UART Interrupt Status Register	0x0040_0002

31	30	29	28	27	26	25	24
ABRINT	TXENDINT	HWBUFEINT	HWTOINT	HWMODINT	HWRLSINT	Reserved	SWBEINT
23	22	21	20	19	18	17	16
Reserved	TXENDIF	HWBUFEIF	HWTOIF	HWMODIF	HWRLSIF	Reserved	SWBEIF
15	14	13	12	11	10	9	8
LININT	WKINT	BUFERRINT	RXTOINT	MODEMINT	RLSINT	THREINT	RDAINT
7	6	5	4	3	2	1	0
LINIF	WKIF	BUFERRIF	RXTOIF	MODEMIF	RLSIF	THREIF	RDAIF

Bits	Description	
[31]	**ABRINT**	**Auto-baud Rate Interrupt Indicator (Read Only)** This bit is set if ABRIEN (UART_INTEN[18]) and ABRIF (UART_ALTCTL[17]) are both set to 1. 0 = No Auto-baud Rate interrupt is generated. 1 = The Auto-baud Rate interrupt is generated.
[30]	**TXENDINT**	**Transmitter Empty Interrupt Indicator (Read Only)** This bit is set if TXENDIEN (UART_INTEN[22]) and TXENDIF(UART_INTSTS[22]) are both set to 1. 0 = No Transmitter Empty interrupt is generated. 1 = Transmitter Empty interrupt is generated.
[29]	**HWBUFEINT**	**PDMA Mode Buffer Error Interrupt Indicator (Read Only)** This bit is set if BUFERRIEN (UART_INTEN[5]) and HWBUFEIF (UART_INTSTS[21]) are both set to 1. 0 = No buffer error interrupt is generated in PDMA mode. 1 = Buffer error interrupt is generated in PDMA mode.
[28]	**HWTOINT**	**PDMA Mode RX Time-out Interrupt Indicator (Read Only)** This bit is set if RXTOIEN (UART_INTEN[4]) and HWTOIF(UART_INTSTS[20]) are both set to 1. 0 = No RX time-out interrupt is generated in PDMA mode. 1 = RX time-out interrupt is generated in PDMA mode.
[27]	**HWMODINT**	**PDMA Mode MODEM Status Interrupt Indicator (Read Only)** This bit is set if MODEMIEN (UART_INTEN[3]) and HWMODIF(UART_INTSTS[19]) are both set to 1. 0 = No Modem interrupt is generated in PDMA mode. 1 = Modem interrupt is generated in PDMA mode.
[26]	**HWRLSINT**	**PDMA Mode Receive Line Status Interrupt Indicator (Read Only)** This bit is set if RLSIEN (UART_INTEN[2]) and HWRLSIF(UART_INTSTS[18]) are both set to 1. 0 = No RLS interrupt is generated in PDMA mode. 1 = RLS interrupt is generated in PDMA mode.

[25]	**Reserved**	Reserved.
[24]	**SWBEINT**	**Single-wire Bit Error Detect Interrupt Indicator (Read Only)** This bit is set if SWBEIEN (UART_INTEN[16]) and SWBEIF (UART_INTSTS[16]) are both set to 1. 0 = No Single-wire Bit Error Detection Interrupt generated. 1 = Single-wire Bit Error Detection Interrupt generated.
[23]	**Reserved**	Reserved.
[22]	**TXENDIF**	**Transmitter Empty Interrupt Flag** This bit is set when TX FIFO (UART_DAT) is empty and the STOP bit of the last byte has been transmitted (TXEMPTYF (UART_FIFOSTS[28]) is set). If TXENDIEN (UART_INTEN[22]) is enabled, the Transmitter Empty interrupt will be generated. 0 = No transmitter empty interrupt flag is generated. 1 = Transmitter empty interrupt flag is generated. **Note:** This bit is cleared automatically when TX FIFO is not empty or the last byte transmission has not completed.
[21]	**HWBUFEIF**	**PDMA Mode Buffer Error Interrupt Flag (Read Only)** This bit is set when the TX or RX FIFO overflows (TXOVIF (UART_FIFOSTS [24]) or RXOVIF (UART_FIFOSTS[0]) is set). When BUFERRIF (UART_INTSTS[5]) is set, the transfer maybe is not correct. If BUFERRIEN (UART_INTEN [5]) is enabled, the buffer error interrupt will be generated. 0 = No buffer error interrupt flag is generated in PDMA mode. 1 = Buffer error interrupt flag is generated in PDMA mode. **Note:** This bit is cleared when both TXOVIF (UART_FIFOSTS[24]) and RXOVIF (UART_FIFOSTS[0]) are cleared.
[20]	**HWTOIF**	**PDMA Mode RX Time-out Interrupt Flag (Read Only)** This bit is set when the RX FIFO is not empty and no activities occurred in the RX FIFO and the time-out counter equal to TOIC (UART_TOUT[7:0]). If RXTOIEN (UART_INTEN [4]) is enabled, the RX time-out interrupt will be generated . 0 = No RX time-out interrupt flag is generated in PDMA mode. 1 = RX time-out interrupt flag is generated in PDMA mode. **Note:** This bit is read only and user can read UART_DAT (RX is in active) to clear it.
[19]	**HWMODIF**	**PDMA Mode MODEM Interrupt Flag (Read Only)** This bit is set when the nCTS pin has state change (CTSDETF (UART_MODEMSTS [0] =1)). If MODEMIEN (UART_INTEN [3]) is enabled, the Modem interrupt will be generated. 0 = No Modem interrupt flag is generated in PDMA mode. 1 = Modem interrupt flag is generated in PDMA mode. **Note:** This bit is read only and reset to 0 when the bit CTSDETF (UART_MODEMSTS[0]) is cleared by writing 1 on CTSDETF (UART_MODEMSTS [0]).
[18]	**HWRLSIF**	**PDMA Mode Receive Line Status Flag (Read Only)** This bit is set when the RX receive data have parity error, frame error or break error (at least one of 3 bits, BIF (UART_FIFOSTS[6]), FEF (UART_FIFOSTS[5]) and PEF (UART_FIFOSTS[4]) is set). If RLSIEN (UART_INTEN [2]) is enabled, the RLS interrupt will be generated. 0 = No RLS interrupt flag is generated in PDMA mode. 1 = RLS interrupt flag is generated in PDMA mode. **Note 1:** In RS-485 function mode, this field include “receiver detect any address byte received address byte character (bit9 = ‘1’) bit". **Note 2:** In UART function mode, this bit is read only and reset to 0 when all bits of BIF(UART_FIFOSTS[6]), FEF(UART_FIFOSTS[5]) and PEF(UART_FIFOSTS[4]) are cleared. **Note 3:** In RS-485 function mode, this bit is read only and reset to 0 when all bits of BIF(UART_FIFOSTS[6]), FEF(UART_FIFOSTS[5]), PEF(UART_FIFOSTS[4]) and ADDRDETF (UART_FIFOSTS[3]) are cleared.

[17]	**Reserved**	Reserved.
[16]	**SWBEIF**	**Single-wire Bit Error Detection Interrupt Flag** This bit is set when the single wire bus state not equals to UART controller TX state in Single-wire mode. 0 = No single-wire bit error detection interrupt flag is generated. 1 = Single-wire bit error detection interrupt flag is generated. **Note 1:** This bit is active when FUNCSEL (UART_FUNCSEL[2:0]) is select UART Single-wire mode. **Note 2:** This bit can be cleared by writing “1” to it.
[15]	**LININT**	**LIN Bus Interrupt Indicator (Read Only)** This bit is set if LINIEN (UART_INTEN[8]) and LINIF(UART_INTSTS[7]) are both set to 1. 0 = No LIN Bus interrupt is generated. 1 = The LIN Bus interrupt is generated.
[14]	**WKINT**	**UART Wake-up Interrupt Indicator (Read Only)** This bit is set if WKIEN (UART_INTEN[6]) and WKIF (UART_INTSTS[6]) are both set to 1. 0 = No UART wake-up interrupt is generated. 1 = UART wake-up interrupt is generated.
[13]	**BUFERRINT**	**Buffer Error Interrupt Indicator (Read Only)** This bit is set if BUFERRIEN(UART_INTEN[5]) and BUFERRIF(UART_ INTSTS[5]) are both set to 1. 0 = No buffer error interrupt is generated. 1 = Buffer error interrupt is generated.
[12]	**RXTOINT**	**RX Time-out Interrupt Indicator (Read Only)** This bit is set if RXTOIEN (UART_INTEN[4]) and RXTOIF(UART_INTSTS[4]) are both set to 1. 0 = No RX time-out interrupt is generated. 1 = RX time-out interrupt is generated.
[11]	**MODEMINT**	**MODEM Status Interrupt Indicator (Read Only)** This bit is set if MODEMIEN(UART_INTEN[3]) and MODEMIF(UART_INTSTS[3]) are both set to 1 0 = No Modem interrupt is generated. 1 = Modem interrupt is generated..
[10]	**RLSINT**	**Receive Line Status Interrupt Indicator (Read Only)** This bit is set if RLSIEN (UART_INTEN[2]) and RLSIF(UART_INTSTS[2]) are both set to 1. 0 = No RLS interrupt is generated. 1 = RLS interrupt is generated.
[9]	**THREINT**	**Transmit Holding Register Empty Interrupt Indicator (Read Only)** This bit is set if THREIEN (UART_INTEN[1]) and THREIF(UART_INTSTS[1]) are both set to 1. 0 = No THRE interrupt is generated. 1 = THRE interrupt is generated.
[8]	**RDAINT**	**Receive Data Available Interrupt Indicator (Read Only)** This bit is set if RDAIEN (UART_INTEN[0]) and RDAIF (UART_INTSTS[0]) are both set to 1. 0 = No RDA interrupt is generated. 1 = RDA interrupt is generated.
[7]	**LINIF**	**LIN Bus Interrupt Flag** This bit is set when LIN slave header detect (SLVHDETF (UART_LINSTS[0] =1)), LIN break detect (BRKDETF(UART_LINSTS[8]=1)), bit error detect (BITEF(UART_LINSTS[9]=1)), LIN slave ID parity error (SLVIDPEF(UART_LINSTS[2] = 1)) or LIN slave header error detect (SLVHEF (UART_LINSTS[1])). If LINIEN (UART_INTEN [8]) is enabled the LIN interrupt will be generated.

		0 = None of SLVHDETF, BRKDETF, BITEF, SLVIDPEF and SLVHEF is generated. 1 = At least one of SLVHDETF, BRKDETF, BITEF, SLVIDPEF and SLVHEF is generated. **Note:** This bit is cleared when SLVHDETF(UART_LINSTS[0]), BRKDETF(UART_LINSTS[8]), BITEF(UART_LINSTS[9]), SLVIDPEF (UART_LINSTS[2]) and SLVHEF(UART_LINSTS[1]) all are cleared and software writing '1' to LINIF(UART_INTSTS[7]).
[6]	**WKIF**	**UART Wake-up Interrupt Flag (Read Only)** This bit is set when TOUTWKF (UART_WKSTS[4]), RS485WKF (UART_WKSTS[3]), RFRTWKF (UART_WKSTS[2]), DATWKF (UART_WKSTS[1]) or CTSWKF(UART_WKSTS[0]) is set to 1. 0 = No UART wake-up interrupt flag is generated. 1 = UART wake-up interrupt flag is generated. **Note:** This bit is cleared if all of TOUTWKF, RS485WKF, RFRTWKF, DATWKF and CTSWKF are cleared to 0 by writing 1 to the corresponding interrupt flag.
[5]	**BUFERRIF**	**Buffer Error Interrupt Flag (Read Only)** This bit is set when the TX FIFO or RX FIFO overflows (TXOVIF (UART_FIFOSTS[24]) or RXOVIF (UART_FIFOSTS[0]) is set). When BUFERRIF (UART_INTSTS[5]) is set, the transfer is not correct. If BUFERRIEN (UART_INTEN [5]) is enabled, the buffer error interrupt will be generated. 0 = No buffer error interrupt flag is generated. 1 = Buffer error interrupt flag is generated. **Note:** This bit is cleared if both of RXOVIF(UART_FIFOSTS[0]) and TXOVIF(UART_FIFOSTS[24]) are cleared to 0 by writing 1 to RXOVIF(UART_FIFOSTS[0]) and TXOVIF(UART_FIFOSTS[24]).
[4]	**RXTOIF**	**RX Time-out Interrupt Flag (Read Only)** This bit is set when the RX FIFO is not empty and no activities occurred in the RX FIFO and the time-out counter equal to TOIC (UART_TOUT[7:0]). If RXTOIEN (UART_INTEN [4]) is enabled, the RX time-out interrupt will be generated. 0 = No RX time-out interrupt flag is generated. 1 = RX time-out interrupt flag is generated. **Note:** This bit is read only and user can read UART_DAT (RX is in active) to clear it.
[3]	**MODEMIF**	**MODEM Interrupt Flag (Read Only)** This bit is set when the nCTS pin has state change (CTSDETF (UART_MODEMSTS[0]) = 1). If MODEMIEN (UART_INTEN [3]) is enabled, the Modem interrupt will be generated. 0 = No Modem interrupt flag is generated. 1 = Modem interrupt flag is generated. **Note:** This bit is read only and reset to 0 when bit CTSDETF is cleared by a write 1 on CTSDETF(UART_MODEMSTS[0]).
[2]	**RLSIF**	**Receive Line Interrupt Flag (Read Only)** This bit is set when the RX receive data have parity error, frame error or break error (at least one of 3 bits, BIF(UART_FIFOSTS[6]), FEF(UART_FIFOSTS[5]) and PEF(UART_FIFOSTS[4]), is set). If RLSIEN (UART_INTEN [2]) is enabled, the RLS interrupt will be generated. 0 = No RLS interrupt flag is generated. 1 = RLS interrupt flag is generated. **Note 1:** In RS-485 function mode, this field is set include "receiver detect and received address byte character (bit9 = '1') bit". At the same time, the bit of ADDRDETF (UART_FIFOSTS[3]) is also set. **Note 2:** This bit is read only and reset to 0 when all bits of BIF (UART_FIFOSTS[6]), FEF(UART_FIFOSTS[5]) and PEF(UART_FIFOSTS[4]) are cleared. **Note 3:** In RS-485 function mode, this bit is read only and reset to 0 when all bits of BIF (UART_FIFOSTS[6]), FEF(UART_FIFOSTS[5]), PEF(UART_FIFOSTS[4]) and ADDRDETF (UART_FIFOSTS[3]) are cleared.
[1]	**THREIF**	**Transmit Holding Register Empty Interrupt Flag** This bit is set when the last data of TX FIFO is transferred to Transmitter Shift Register. If THREIEN (UART_INTEN[1]) is enabled, the THRE interrupt will be generated. 0 = No THRE interrupt flag is generated.

		1 = THRE interrupt flag is generated. **Note:** This bit is read only and it will be cleared when writing data into UART_DAT (TX FIFO not empty).
[0]	**RDAIF**	**Receive Data Available Interrupt Flag** When the number of bytes in the RX FIFO equals the RFITL then the RDAIF(UART_INTSTS[0]) will be set. If RDAIEN (UART_INTEN [0]) is enabled, the RDA interrupt will be generated. 0 = No RDA interrupt flag is generated. 1 = RDA interrupt flag is generated. **Note:** This bit is read only and it will be cleared when the number of unread bytes of RX FIFO drops below the threshold level (RFITL(UART_FIFO[7:4]).

UART Time-out Register (UART_TOUT)

Register	Offset	R/W	Description	Reset Value
UART_TOUT **x=0,1,2,3,4,5**	UARTx_BA+0x20	R/W	UART Time-out Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**DLY**	**TX Delay Time Value** This field is used to programming the transfer delay time between the last stop bit and next start bit. The unit is bit time.
[7:0]	**TOIC**	**Time-out Interrupt Comparator** The time-out counter resets and starts counting (the counting clock = baud rate) whenever the RX FIFO receives a new data word if time out counter is enabled by setting TOCNTEN (UART_INTEN[11]). Once the content of time-out counter is equal to that of time-out interrupt comparator (TOIC (UART_TOUT[7:0])), a receiver time-out interrupt (RXTOINT(UART_INTSTS[12])) is generated if RXTOIEN (UART_INTEN [4]) enabled. A new incoming data word or RX FIFO empty will clear RXTOIF (UART_INTSTS[4]). In order to avoid receiver time-out interrupt generation immediately during one character is being received, TOIC value should be set between 40 and 255. So, for example, if TOIC is set with 40, the time-out interrupt is generated after four characters are not received when 1 stop bit and no parity check is set for UART transfer.

UART Baud Rate Divider Register (UART_BAUD)

Register	Offset	R/W	Description	Reset Value
UART_BAUD **x=0,1,2,3,4,5**	UARTx_BA+0x24	R/W	UART Baud Rate Divider Register	0x0F00_0000

31	30	29	28	27	26	25	24
Reserved		BAUDM1	BAUDM0	EDIVM1			
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
BRD							
7	6	5	4	3	2	1	0
BRD							

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29]	**BAUDM1**	**BAUD Rate Mode Selection Bit 1** This bit is baud rate mode selection bit 1. UART provides three baud rate calculation modes. This bit combines with BAUDM0 (UART_BAUD[28]) to select baud rate calculation mode. The detail description is shown in Table 6.19-4. **Note:** In IrDA mode must be operated in mode 0.
[28]	**BAUDM0**	**BAUD Rate Mode Selection Bit 0** This bit is baud rate mode selection bit 0. UART provides three baud rate calculation modes. This bit combines with BAUDM1 (UART_BAUD[29]) to select baud rate calculation mode. The detail description is shown in Table 6.19-4.
[27:24]	**EDIVM1**	**Extra Divider for BAUD Rate Mode 1** This field is used for baud rate calculation in mode 1 and has no effect for baud rate calculation in mode 0 and mode 2. The detail description is shown in Table 6.19-4.
[23:16]	**Reserved**	Reserved.
[15:0]	**BRD**	**Baud Rate Divider** The field indicates the baud rate divider. This filed is used in baud rate calculation. The detail description is shown in Table 6.19-4.

UART IrDA Control Register (UART_IRDA)

Register	Offset	R/W	Description	Reset Value
UART_IRDA **x=0,1,2,3,4,5**	UARTx_BA+0x28	R/W	UART IrDA Control Register	0x0000_0040

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**RXINV**	**TXINV**	**Reserved**			**TXEN**	**Reserved**

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**RXINV**	**IrDA Inverse Receive Input Signal** 0 = None inverse receiving input signal. 1 = Inverse receiving input signal. (Default) **Note 1:** Before setting this bit, TXRXDIS (UART_FUNCSEL[3]) should be set then waited for TXRXACT (UART_FIFOSTS[31]) is cleared. When the configuration is done, cleared TXRXDIS (UART_FUNCSEL[3]) to activate UART controller. **Note 2:** This bit is valid when FUNCSEL (UART_FUNCSEL[2:0]) is select IrDA function.
[5]	**TXINV**	**IrDA Inverse Transmitting Output Signal** 0 = None inverse transmitting signal. (Default). 1 = Inverse transmitting output signal. **Note 1:** Before setting this bit, TXRXDIS (UART_FUNCSEL[3]) should be set then waited for TXRXACT (UART_FIFOSTS[31]) is cleared. When the configuration is done, cleared TXRXDIS (UART_FUNCSEL[3]) to activate UART controller. **Note 2:** This bit is valid when FUNCSEL (UART_FUNCSEL[2:0]) is select IrDA function.
[4:2]	**Reserved**	Reserved.
[1]	**TXEN**	**IrDA Receiver/Transmitter Selection Enable Bit** 0 = IrDA Transmitter Disabled and Receiver Enabled. (Default) 1 = IrDA Transmitter Enabled and Receiver Disabled.
[0]	**Reserved**	Reserved.

UART Alternate Control/Status Register (UART_ALTCTL)

Register	Offset	R/W	Description	Reset Value
UART_ALTCTL x=0,1,2,3,4,5	UARTx_BA+0x2C	R/W	UART Alternate Control/Status Register	0x0000_000C

31	30	29	28	27	26	25	24
ADDRMV							
23	22	21	20	19	18	17	16
Reserved			ABRDBITS		ABRDEN	ABRIF	Reserved
15	14	13	12	11	10	9	8
ADDRDEN	Reserved				RS485AUD	RS485AAD	RS485NMM
7	6	5	4	3	2	1	0
LINTXEN	LINRXEN	Reserved		BRKFL			

Bits	Description	
[31:24]	**ADDRMV**	**Address Match Value** This field contains the RS-485 address match values. **Note:** This field is used for RS-485 auto address detection mode.
[23:21]	**Reserved**	Reserved.
[20:19]	**ABRDBITS**	**Auto-baud Rate Detect Bit Length** 00 = 1-bit time from Start bit to the 1st rising edge. The input pattern shall be 0x01. 01 = 2-bit time from Start bit to the 1st rising edge. The input pattern shall be 0x02. 10 = 4-bit time from Start bit to the 1st rising edge. The input pattern shall be 0x08. 11 = 8-bit time from Start bit to the 1st rising edge. The input pattern shall be 0x80. **Note :** The calculation of bit number includes the START bit.
[18]	**ABRDEN**	**Auto-baud Rate Detect Enable Bit** 0 = Auto-baud rate detect function Disabled. 1 = Auto-baud rate detect function Enabled. **Note :** This bit is cleared automatically after auto-baud detection is finished.
[17]	**ABRIF**	**Auto-baud Rate Interrupt Flag (Read Only)** This bit is set when auto-baud rate detection function finished or the auto-baud rate counter was overflow and if ABRIEN(UART_INTEN [18]) is set then the auto-baud rate interrupt will be generated. 0 = No auto-baud rate interrupt flag is generated. 1 = Auto-baud rate interrupt flag is generated. **Note:** This bit is read only, but it can be cleared by writing “1” to ABRDTOIF (UART_FIFOSTS[2]) and ABRDIF(UART_FIFOSTS[1]).
[16]	**Reserved**	Reserved.
[15]	**ADDRDEN**	**RS-485 Address Detection Enable Bit** This bit is used to enable RS-485 Address Detection mode. 0 = Address detection mode Disabled.

		1 = Address detection mode Enabled. **Note:** This bit is used for RS-485 any operation mode.
[14:11]	**Reserved**	Reserved.
[10]	**RS485AUD**	**RS-485 Auto Direction Function (AUD)** 0 = RS-485 Auto Direction Operation function (AUD) Disabled. 1 = RS-485 Auto Direction Operation function (AUD) Enabled. **Note:** It can be active with RS-485_AAD or RS-485_NMM operation mode.
[9]	**RS485AAD**	**RS-485 Auto Address Detection Operation Mode (AAD)** 0 = RS-485 Auto Address Detection Operation mode (AAD) Disabled. 1 = RS-485 Auto Address Detection Operation mode (AAD) Enabled. **Note:** It cannot be active with RS-485_NMM operation mode.
[8]	**RS485NMM**	**RS-485 Normal Multi-drop Operation Mode (NMM)** 0 = RS-485 Normal Multi-drop Operation mode (NMM) Disabled. 1 = RS-485 Normal Multi-drop Operation mode (NMM) Enabled. **Note:** It cannot be active with RS-485_AAD operation mode.
[7]	**LINTXEN**	**LIN TX Break Mode Enable Bit** 0 = LIN TX Break mode Disabled. 1 = LIN TX Break mode Enabled. **Note:** When TX break field transfer operation finished, this bit will be cleared automatically.
[6]	**LINRXEN**	**LIN RX Enable Bit** 0 = LIN RX mode Disabled. 1 = LIN RX mode Enabled.
[5:4]	**Reserved**	Reserved.
[3:0]	**BRKFL**	**UART LIN Break Field Length** This field indicates a 4-bit LIN TX break field count. **Note 1:** This break field length is BRKFL + 1. **Note 2:** According to LIN spec, the reset value is 0xC (break field length = 13).

UART Function Select Register (UART_FUNCSEL)

Register	Offset	R/W	Description	Reset Value
UART_FUNCSEL **x=0,1,2,3,4,5**	UARTx_BA+0x30	R/W	UART Function Select Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**DGE**	**Reserved**		**TXRXDIS**	**FUNCSEL**		

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**DGE**	**Deglitch Enable Bit** 0 = Deglitch Disabled. 1 = Deglitch Enabled. **Note:** When this bit is set to logic 1, any pulse width less than about 300 ns will be considered a glitch and will be removed in the serial data input (RX). This bit acts only on RX line and has no effect on the transmitter logic.
[5:4]	**Reserved**	Reserved.
[3]	**TXRXDIS**	**TX and RX Disable Bit** Setting this bit can disable TX and RX. 0 = TX and RX Enabled. 1 = TX and RX Disabled. **Note:** The TX and RX will not disable immediately when this bit is set. The TX and RX complete current task before disable TX and RX. When TX and RX disable, the TXRXACT (UART_FIFOSTS[31]) is cleared.
[2:0]	**FUNCSEL**	**Function Select** 000 = UART function. 001 = LIN function. 010 = IrDA function. 011 = RS-485 function. 100 = UART Single-wire function. Others = Reserved.

UART LIN Control Register (UART_LINCTL)

Register	Offset	R/W	Description	Reset Value
UART_LINCTL x=0,1	UARTx_BA+0x34	R/W	UART LIN Control Register	0x000C_0000

31	30	29	28	27	26	25	24
PID							
23	22	21	20	19	18	17	16
HSEL		BSL		BRKFL			
15	14	13	12	11	10	9	8
Reserved			BITERREN	LINRXOFF	BRKDETEN	IDPEN	SENDH
7	6	5	4	3	2	1	0
Reserved			MUTE	SLVDUEN	SLVAREN	SLVHDEN	SLVEN

Bits	Description	
[31:24]	**PID**	**LIN PID Bits** This field contains the LIN frame ID value in LIN function mode, and the frame ID parity can be generated by software or hardware depends on IDPEN (UART_LINCTL[9]) = 1. If the parity generated by hardware, user fill ID0~ID5 (PID [29:24]), hardware will calculate P0 (PID[30]) and P1 (PID[31]), otherwise user must filled frame ID and parity in this field. **Note 1:** User can fill any 8-bit value to this field and the bit 24 indicates ID0 (LSB first). **Note 2:** This field can be used for LIN master mode or slave mode.
[23:22]	**HSEL**	**LIN Header Select** 00 = The LIN header includes “break field”. 01 = The LIN header includes “break field” and “sync field”. 10 = The LIN header includes “break field”, “sync field” and “frame ID field”. 11 = Reserved. **Note:** This bit is used to master mode for LIN to send header field (SENDH (UART_LINCTL [8]) = 1) or used to slave to indicates exit from mute mode condition (MUTE (UART_LINCTL[4] = 1).
[21:20]	**BSL**	**LIN Break/Sync Delimiter Length** 00 = The LIN break/sync delimiter length is 1-bit time. 01 = The LIN break/sync delimiter length is 2-bit time. 10 = The LIN break/sync delimiter length is 3-bit time. 11 = The LIN break/sync delimiter length is 4-bit time. **Note:** This bit used for LIN master to sending header field.
[19:16]	**BRKFL**	**LIN Break Field Length** This field indicates a 4-bit LIN TX break field count. **Note 1:** These registers are shadow registers of BRKFL (UART_ALTCTL[3:0]), User can read/write it by setting BRKFL (UART_ALTCTL[3:0]) or BRKFL (UART_LINCTL[19:16]). **Note 2:** This break field length is BRKFL + 1. **Note 3:** According to LIN spec, the reset value is 12 (break field length = 13).

[15:13]	**Reserved**	Reserved.
[12]	**BITERREN**	**Bit Error Detect Enable Bit** 0 = Bit error detection function Disabled. 1 = Bit error detection function Enabled. **Note:** In LIN function mode, when occur bit error, the BITEF (UART_LINSTS[9]) flag will be asserted. If the LINIEN (UART_INTEN[8]) = 1, an interrupt will be generated.
[11]	**LINRXOFF**	**LIN Receiver Disable Bit** If the receiver is enabled (LINRXOFF (UART_LINCTL[11]) = 0), all received byte data will be accepted and stored in the RX FIFO, and if the receiver is disabled (LINRXOFF (UART_LINCTL[11] = 1), all received byte data will be ignore. 0 = LIN receiver Enabled. 1 = LIN receiver Disabled. **Note:** This bit is only valid when operating in LIN function mode (FUNCSEL (UART_FUNCSEL[2:0]) = 001).
[10]	**BRKDETEN**	**LIN Break Detection Enable Bit** When detect consecutive dominant greater than 11 bits, and are followed by a delimiter character, the BRKDETF (UART_LINSTS[8]) flag is set at the end of break field. If the LINIEN (UART_INTEN [8])=1, an interrupt will be generated. 0 = LIN break detection Disabled. 1 = LIN break detection Enabled.
[9]	**IDPEN**	**LIN ID Parity Enable Bit** 0 = LIN frame ID parity Disabled. 1 = LIN frame ID parity Enabled. **Note 1:** This bit can be used for LIN master to sending header field (SENDH (UART_LINCTL[8])) = 1 and HSEL (UART_LINCTL[23:22]) = 10 or be used for enable LIN slave received frame ID parity checked. **Note 2:** This bit is only used when the operation header transmitter is in HSEL (UART_LINCTL[23:22]) = 10.
[8]	**SENDH**	**LIN TX Send Header Enable Bit** The LIN TX header can be “break field” or “break and sync field” or “break, sync and frame ID field”, it is depend on setting HSEL (UART_LINCTL[23:22]). 0 = Send LIN TX header Disabled. 1 = Send LIN TX header Enabled. **Note 1:** This bit is shadow bit of LINTXEN (UART_ALTCTL [7]); user can read/write it by setting LINTXEN (UART_ALTCTL [7]) or SENDH (UART_LINCTL [8]). **Note 2:** When transmitter header field (it may be “break” or “break + sync” or “break + sync + frame ID” selected by HSEL (UART_LINCTL[23:22]) field) transfer operation finished, this bit will be cleared automatically.
[7:5]	**Reserved**	Reserved.
[4]	**MUTE**	**LIN Mute Mode Enable Bit** 0 = LIN mute mode Disabled. 1 = LIN mute mode Enabled. **Note:** The exit from mute mode condition and each control and interactions of this field are explained in 6.19.5.10 (LIN slave mode).
[3]	**SLVDUEN**	**LIN Slave Divider Update Method Enable Bit** 0 = UART_BAUD updated is written by software (if no automatic resynchronization update occurs at the same time). 1 = UART_BAUD is updated at the next received character. User must set the bit before checksum reception. **Note 1:** This bit only is valid in LIN slave mode (SLVEN (UART_LINCTL[0]) = 1).

		Note 2: This bit used for LIN Slave Automatic Resynchronization mode. (for Non-Automatic Resynchronization mode, this bit should be kept cleared) **Note 3:** The control and interactions of this field are explained in 6.19.5.10 (Slave mode with automatic resynchronization).
[2]	**SLVAREN**	**LIN Slave Automatic Resynchronization Mode Enable Bit** 0 = LIN automatic resynchronization Disabled. 1 = LIN automatic resynchronization Enabled. **Note 1:** This bit only is valid in LIN slave mode (SLVEN (UART_LINCTL[0]) = 1). **Note 2:** When operation in Automatic Resynchronization mode, the baud rate setting must be mode2 (BAUDM1 (UART_BAUD [29]) and BAUDM0 (UART_BAUD [28]) must be 1). **Note 3:** The control and interactions of this field are explained in 6.19.5.10 (Slave mode with automatic resynchronization).
[1]	**SLVHDEN**	**LIN Slave Header Detection Enable Bit** 0 = LIN slave header detection Disabled. 1 = LIN slave header detection Enabled. **Note 1:** This bit is only valid in LIN slave mode (SLVEN (UART_LINCTL[0]) = 1). **Note 2:** In LIN function mode, when detect header field (break + sync + frame ID), SLVHDETF (UART_LINSTS [0]) flag will be asserted. If the LINIEN (UART_INTEN[8]) = 1, an interrupt will be generated.
[0]	**SLVEN**	**LIN Slave Mode Enable Bit** 0 = LIN slave mode Disabled. 1 = LIN slave mode Enabled.

UART LIN Status Register (UART_LINSTS)

Register	Offset	R/W	Description	Reset Value
UART_LINSTS **x=0,1**	UARTx_BA+0x38	R/W	UART LIN Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						BITEF	BRKDETF
7	6	5	4	3	2	1	0
Reserved				SLVSYNCF	SLVIDPEF	SLVHEF	SLVHDETF

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**BITEF**	**Bit Error Detect Status Flag** At TX transfer state, hardware will monitor the bus state, if the input pin (UART_RXD) state not equals to the output pin (UART_TXD) state, BITEF (UART_LINSTS[9]) will be set. When occur bit error, if the LINIEN (UART_INTEN[8]) = 1, an interrupt will be generated. 0 = Bit error not detected. 1 = Bit error detected. **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** This bit is only valid when enable bit error detection function (BITERREN (UART_LINCTL [12]) = 1).
[8]	**BRKDETF**	**LIN Break Detection Flag** This bit is set by hardware when a break is detected and be cleared by writing 1 to it through software. 0 = LIN break not detected. 1 = LIN break detected. **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** This bit is only valid when LIN break detection function is enabled (BRKDETEN (UART_LINCTL[10]) =1).
[7:4]	**Reserved**	Reserved.
[3]	**SLVSYNCF**	**LIN Slave Sync Field** This bit indicates that the LIN sync field is being analyzed in Automatic Resynchronization mode. When the receiver header have some error been detect, user must reset the internal circuit to re-search new frame header by writing 1 to this bit. 0 = The current character is not at LIN sync state. 1 = The current character is at LIN sync state. **Note 1:** This bit is only valid in LIN Slave mode (SLVEN(UART_LINCTL[0]) = 1). **Note 2:** This bit can be cleared by writing 1 to it.

		Note 3: When writing 1 to it, hardware will reload the initial baud rate and re-search a new frame header.
[2]	**SLVIDPEF**	**LIN Slave ID Parity Error Flag** This bit is set by hardware when receipted frame ID parity is not correct. 0 = No active. 1 = Receipted frame ID parity is not correct. **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** This bit is only valid in LIN slave mode (SLVEN (UART_LINCTL [0])= 1) and enable LIN frame ID parity check function IDPEN (UART_LINCTL [9]).
[1]	**SLVHEF**	**LIN Slave Header Error Flag** This bit is set by hardware when a LIN header error is detected in LIN slave mode and be cleared by writing 1 to it. The header errors include “break delimiter is too short (less than 0.5 bit time)”, “frame error in sync field or Identifier field”, “sync field data is not 0x55 in Non-Automatic Resynchronization mode”, “sync field deviation error with Automatic Resynchronization mode”, “sync field measure time-out with Automatic Resynchronization mode” and “LIN header reception time-out”. 0 = LIN header error not detected. 1 = LIN header error detected. **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** This bit is only valid when UART is operated in LIN slave mode (SLVEN (UART_LINCTL [0]) = 1) and enables LIN slave header detection function (SLVHDEN (UART_LINCTL [1])).
[0]	**SLVHDETF**	**LIN Slave Header Detection Flag** This bit is set by hardware when a LIN header is detected in LIN slave mode and be cleared by writing 1 to it. 0 = LIN header not detected. 1 = LIN header detected (break + sync + frame ID). **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** This bit is only valid in LIN slave mode (SLVEN (UART_LINCTL [0]) = 1) and enable LIN slave header detection function (SLVHDEN (UART_LINCTL [1])). **Note 3:** When enable ID parity check IDPEN (UART_LINCTL [9]), if hardware detect complete header (“break + sync + frame ID”), the SLVHDETF will be set whether the frame ID correct or not.

UART Baud Rate Compensation Register (UART_BRCOMP)

Register	Offset	R/W	Description	Reset Value
UART_BRCOMP x=0,1,2,3,4,5	UARTx_BA+0x3C	R/W	UART Baud Rate Compensation Register	0x0000_0000

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

Bits	Description	
[31]	**BRCOMPDEC**	**Baud Rate Compensation Decrease** 0 = Positive (increase one module clock) compensation for each compensated bit. 1 = Negative (decrease one module clock) compensation for each compensated bit.
[30:9]	**Reserved**	Reserved.
[8:0]	**BRCOMP**	**Baud Rate Compensation Patten** These 9-bits are used to define the relative bit is compensated or not. BRCOMP[7:0] is used to define the compensation of UART_DAT[7:0] and BRCOM[8] is used to define the parity bit.

UART Wake-up Control Register (UART_WKCTL)

Register	Offset	R/W	Description	Reset Value
UART_WKCTL **x=0,1,2,3,4,5**	UARTx_BA+0x40	R/W	UART Wake-up Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			**WKTOUTEN**	**WKRS485EN**	**WKRFRTEN**	**WKDATEN**	**WKCTSEN**

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**WKTOUTEN**	**Received Data FIFO Reached Threshold Time-out Wake-up Enable Bit** 0 = Received Data FIFO reached threshold time-out wake-up system function Disabled. 1 = Received Data FIFO reached threshold time-out wake-up system function Enabled. **Note 1:** When the system is in Power-down mode, Received Data FIFO reached threshold time-out will wake up system from Power-down mode. **Note 2:** It is suggested the function is enabled when the WKRFRTEN (UART_WKCTL[2]) is set to 1.
[3]	**WKRS485EN**	**RS-485 Address Match (AAD Mode) Wake-up Enable Bit** 0 = RS-485 Address Match (AAD mode) wake-up system function Disabled. 1 = RS-485 Address Match (AAD mode) wake-up system function Enabled. **Note 1:** When the system is in .Power-down mode, RS-485 Address Match will wake -up system from Power-down mode. **Note 2:** This bit is used for RS-485 Auto Address Detection (AAD) mode in RS-485 function mode and ADDRDEN (UART_ALTCTL[15]) is set to 1.
[2]	**WKRFRTEN**	**Received Data FIFO Reached Threshold Wake-up Enable Bit** 0 = Received Data FIFO reached threshold wake-up system function Disabled. 1 = Received Data FIFO reached threshold wake-up system function Enabled. **Note:** When the system is in Power-down mode, Received Data FIFO reached threshold will wake-up system from Power-down mode.
[1]	**WKDATEN**	**Incoming Data Wake-up Enable Bit** 0 = Incoming data wake-up system function Disabled. 1 = Incoming data wake-up system function Enabled. **Note:** When the system is in Power-down mode, incoming data will wake-up system from Power-down mode.
[0]	**WKCTSEN**	**nCTS Wake-up Enable Bit** 0 = nCTS Wake-up system function Disabled.

		1 = nCTS Wake-up system function Enabled. **Note:** When the system is in Power-down mode, an external nCTS change will wake up system from Power-down mode.

UART Wake-up Status Register (UART_WKSTS)

Register	Offset	R/W	Description	Reset Value
UART_WKSTS **x=0,1,2,3,4,5**	UARTx_BA+0x44	R/W	UART Wake-up Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			**TOUTWKF**	**RS485WKF**	**RFRTWKF**	**DATWKF**	**CTSWKF**

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**TOUTWKF**	**Received Data FIFO Threshold Time-out Wake-up Flag** This bit is set if chip wake-up from power-down state by Received Data FIFO Threshold Time-out wake-up. 0 = Chip stays in power-down state. 1 = Chip wake-up from power-down state by Received Data FIFO reached threshold time-out. **Note 1:** If WKTOUTEN (UART_WKCTL[4]) is enabled, the Received Data FIFO reached threshold time-out wake-up cause this bit is set to '1'. **Note 2:** This bit can be cleared by writing '1' to it.
[3]	**RS485WKF**	**RS-485 Address Match (AAD Mode) Wake-up Flag** This bit is set if chip wake-up from power-down state by RS-485 Address Match (AAD mode). 0 = Chip stays in power-down state. 1 = Chip wake-up from power-down state by RS-485 Address Match (AAD mode) wake-up. **Note 1:** If WKRS485EN (UART_WKCTL[3]) is enabled, the RS-485 Address Match (AAD mode) wake-up cause this bit is set to '1'. **Note 2:** This bit can be cleared by writing '1' to it.
[2]	**RFRTWKF**	**Received Data FIFO Reached Threshold Wake-up Flag** This bit is set if chip wake-up from power-down state by Received Data FIFO reached threshold wake-up. 0 = Chip stays in power-down state. 1 = Chip wake-up from power-down state by Received Data FIFO Reached Threshold wake-up. **Note 1:** If WKRFRTEN (UART_WKCTL[2]) is enabled, the Received Data FIFO Reached Threshold wake-up cause this bit is set to '1'. **Note 2:** This bit can be cleared by writing '1' to it.
[1]	**DATWKF**	**Incoming Data Wake-up Flag** This bit is set if chip wake-up from power-down state by data wake-up. 0 = Chip stays in power-down state.

		1 = Chip wake-up from power-down state by Incoming Data wake-up. **Note 1:** If WKDATEN (UART_WKCTL[1]) is enabled, the Incoming Data wake-up cause this bit is set to '1'. **Note 2:** This bit can be cleared by writing '1' to it.
[0]	**CTSWKF**	**nCTS Wake-up Flag** This bit is set if chip wake-up from power-down state by nCTS wake-up. 0 = Chip stays in power-down state. 1 = Chip wake-up from power-down state by nCTS wake-up. **Note 1:** If WKCTSEN (UART_WKCTL[0]) is enabled, the nCTS wake-up cause this bit is set to '1'. **Note 2:** This bit can be cleared by writing '1' to it.

UART Incoming Data Wake-up Compensation Register (UART_DWKCOMP)

Register	Offset	R/W	Description	Reset Value
UART_DWKCOMP x=0,1,2,3,4,5	UARTx_BA+0x48	R/W	UART Incoming Data Wake-up Compensation Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**STCOMP**	**Start Bit Compensation Value** These bits field indicate how many clock cycle selected by UART_CLK do the UART controller can get the 1st bit (start bit) when the device is woken up from Power-down mode. **Note:** It is valid only when WKDATEN (UART_WKCTL[1]) is set.

6.20 Smart Card Host Interface (SC)

6.20.1 Overview

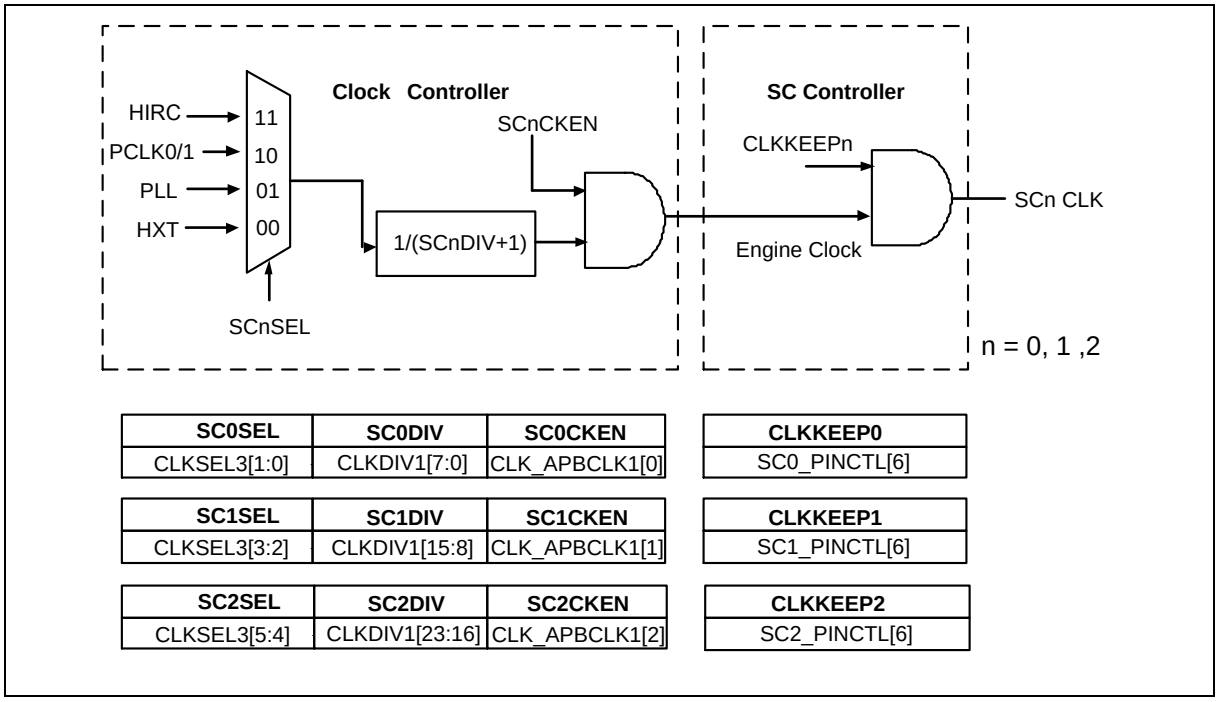
The Smart Card Interface controller (SC controller) is based on ISO/IEC 7816-3 standard and fully compliant with PC/SC Specifications. It also provides status of card insertion/removal.

6.20.2 Features

- ISO 7816-3 T = 0, T = 1 compliant
- EMV2000 compliant
- Three ISO 7816-3 ports
- Separates receive/transmit 4 byte entry FIFO for data payloads
- Programmable transmission clock frequency
- Programmable receiver buffer trigger level
- Programmable guard time selection (11 ETU ~ 267 ETU)
- One 24-bit timer and two 8-bit timers for Answer to Request (ATR) and waiting times processing
- Supports auto direct / inverse convention function
- Supports transmitter and receiver error retry and error number limiting function
- Supports hardware activation sequence process, and the time between PWR on and CLK start is configurable
- Supports hardware warm reset sequence process
- Supports hardware deactivation sequence process
- Supports hardware auto deactivation sequence when detected the card removal
- Supports UART mode
  - Full duplex, asynchronous communications
  - Separates receiving / transmitting 4 bytes entry FIFO for data payloads
  - Supports programmable baud rate generator
  - Supports programmable receiver buffer trigger level
  - Programmable transmitting data delay time between the last stop bit leaving the TX-FIFO and the de-assertion by setting EGT (SCn_EGT[7:0])
  - Programmable even, odd or no parity bit generation and detection
  - Programmable stop bit, 1- or 2- stop bit generation

6.20.3 Block Diagram

The SC clock control and block diagram are shown in SC Clock Control Diagram (8-bit Pre-scale Counter in Clock Controller) and SC Controller Block Diagram. The SC controller is completely asynchronous design with two clock domains, PCLK and engine clock. Note that the PCLK should be higher than or equal to the frequency of engine clock.



SC0SEL	SC0DIV	SC0CKEN	CLKKEEP0
CLKSEL3[1:0]	CLKDIV1[7:0]	CLK_APBCLK1[0]	SC0_PINCTL[6]
SC1SEL	**SC1DIV**	**SC1CKEN**	**CLKKEEP1**
CLKSEL3[3:2]	CLKDIV1[15:8]	CLK_APBCLK1[1]	SC1_PINCTL[6]
SC2SEL	**SC2DIV**	**SC2CKEN**	**CLKKEEP2**
CLKSEL3[5:4]	CLKDIV1[23:16]	CLK_APBCLK1[2]	SC2_PINCTL[6]

Figure 6.20-1 SC Clock Control Diagram (8-bit Pre-scale Counter in Clock Controller)

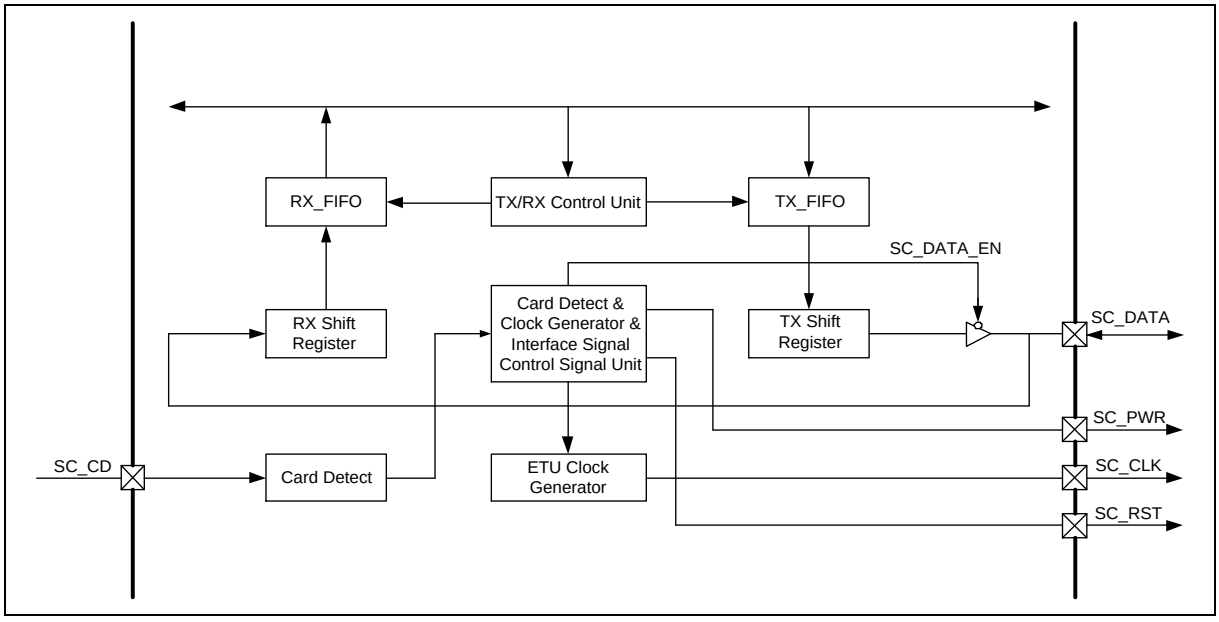


Figure 6.20-2 SC Controller Block Diagram

6.20.4 Basic Configuration

SC Host Controller Pin description is shown in:

Pin	Type	Description
SCn_DATA	Bi-direction	SC Host Controller DATA
SCn_CD	Input	SC Host Controller Card Detect
SCn_PWR	Output	SC Host Controller Power ON/OFF
SCn_CLK	Output	SC Host Controller Clock
SCn_RST	Output	SC Host Controller Reset

Table 6.20-1 SC Host Controller Pin Description

UART Mode Pin description is shown in UART Mode Pin Description:

Pin	Type	Description
SCn_DATA	Input	UART Receive Data
SCn_CLK	Output	UART Transmit Data

Table 6.20-2 UART Mode Pin Description

6.20.4.1 SC0 Basic Configuration

- Clock Source Configuration
  - Select the source of SC0 peripheral clock on SC0SEL (CLK_CLKSEL3[1:0]).
  - Select the clock divider number of SC0 peripheral clock on SC0DIV (CLK_CLKDIV1[7:0]).
  - Enable SC0 peripheral clock in SC0CKEN (CLK_APBCLK1[0]).
- Reset Configuration
  - Reset SC0 controller in SC0RST (SYS_IPRST2[0]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
SC0	SC0_CLK	PB.5	MFP9
		PF.6	MFP3
		PA.0	MFP6
		PE.2	MFP6
	SC0_DAT	PB.4	MFP9
		PF.7	MFP3
		PA.1	MFP6
		PE.3	MFP6
	SC0_PWR	PB.2	MFP9
		PF.9	MFP3

		PA.3	MFP6
		PE.5	MFP6
	SC0_RST	PB.3	MFP9
		PF.8	MFP3
		PA.2	MFP6
		PE.4	MFP6
	SC0_nCD	PC.12	MFP9
		PF.10	MFP3
		PA.4	MFP6
		PE.6	MFP6

6.20.4.2 SC1 Basic Configuration

- Clock Source Configuration
  - Select the source of SC1 peripheral clock on SC1SEL (CLK_CLKSEL3[3:2]).
  - Select the clock divider number of SC1 peripheral clock on SC1DIV (CLK_CLKDIV1[15:8]).
  - Enable SC1 peripheral clock in SC1CKEN (CLK_APBCLK1[1]).
- Reset Configuration
  - Reset SC1 controller in SC1RST (SYS_IPRST2[1]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
SC1	SC1_CLK	PC.0	MFP5
		PD.4	MFP8
		PB.12	MFP3
	SC1_DAT	PC.1	MFP5
		PD.5	MFP8
		PB.13	MFP3
	SC1_PWR	PC.3	MFP5
		PD.7	MFP8
		PB.15	MFP3
	SC1_RST	PC.2	MFP5
		PD.6	MFP8
		PB.14	MFP3
	SC1_nCD	PC.4	MFP5
		PD.3	MFP8
		PD.14	MFP4

6.20.4.3 SC2 Basic Configuration

- Clock Source Configuration
  - Select the source of SC2 peripheral clock on SC2SEL (CLK_CLKSEL3[5:4]).
  - Select the clock divider number of SC2 peripheral clock on SC2DIV (CLK_CLKDIV1[23:16]).
  - Enable SC2 peripheral clock in SC2CKEN (CLK_APBCLK1[2]).
- Reset Configuration
  - Reset SC2 controller in SC2RST (SYS_IPRST2[2]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
SC2	SC2_CLK	PA.8	MFP3
		PA.6	MFP6
		PD.0	MFP7
		PA.15	MFP7
		PE.0	MFP4
	SC2_DAT	PA.9	MFP3
		PA.7	MFP6
		PD.1	MFP7
		PA.14	MFP7
		PE.1	MFP4
	SC2_PWR	PA.11	MFP3
		PC.7	MFP6
		PD.3	MFP7
		PA.12	MFP7
		PH.8	MFP4
	SC2_RST	PA.10	MFP3
		PC.6	MFP6
		PD.2	MFP7
		PA.13	MFP7
		PH.9	MFP4
	SC2_nCD	PC.13	MFP3
		PA.5	MFP6
		PH.10	MFP4

6.20.5 Functional Description

Basically, the Smart Card Interface acts as a half-duplex asynchronous communication port and its data format is composed of ten consecutive bits which is shown in SC Data Character.

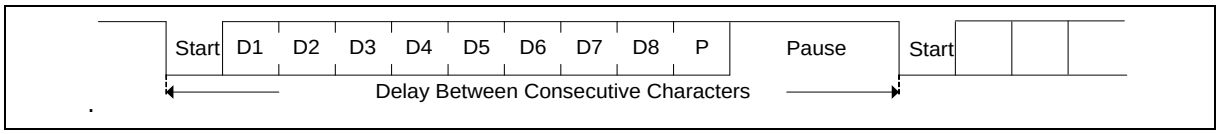


Figure 6.20-3 SC Data Character

The Smart Card Interface controller supports hardware activation, warm reset and deactivation sequence. The activation, warm reset and deactivation sequence are shown as follows.

6.20.5.1 Smart Card Pin Configuration

The Smart Card Interface pin status can be observed by polling following registers.

1. SCn_RST is a output pin. Its status can be observed by RSTSTS (SCn_PINCTL[18]). Programming RSTEN (SCn_PINCTL[1]) '0' or '1' can drive this output pin low or high.
2. SCn_PWR is a output pin. Its status can be observed by PWRSTS (SCn_PINCTL[17]). Programming PWREN (SCn_PINCTL[0]) to '0' or '1' can drive this output pin low or high. PWRINV (SCn_PINCTL[11]) can inverse the SCn_PWR output. User must select PWRINV (SCn_PINCTL[11]) before smart card is enabled by SCEN (SCn_CTL[0]).
3. SCn_DATA is a bidirectional pin, DATASTS (SCn_PINCTL[16]) shows the pin status when SC is receiving data. Programming SCDATA (SCn_PINCTL[9]) to '0' or '1' can drive this pin output low or high.
4. SCn_CLK is a output pin. It outputs smart card clock SCn CLK. Programming CLKKEEP (SCn_PINCTL[6]) '0' or '1' to disable or enable this pin. Programming CSTOPLV (SCn_PINCTL[5]) can determine the SCn_CLK is stopped at high or low when this pin is disable.
5. SCn_CD (Card Detect Pin) state represent the status of the card is inserted or not. SCn_CD pin status can be observed by CDPINSTS (SCn_STATUS[13]). SCn_CD related function can be set by CDLV (SCn_CTL[26]), CDDBSEL (SCn_CTL[25:24]), CDIF (SCn_INTSTS[7]), CDIEN (SCn_INTEN[7]).
6. CDLV (SCn_CTL[26]) determines what kind of pin level change represents the card insertion. CDDBSEL (SCn_CTL[25:24]) determines the de-bounce cycles. When the card status CINSERT (SCn_STATUS[12]) or CREMOVE (SCn_STATUS[11]) is detected, CDIF will set to 1. If CDIEN is enable, SC will deliver a interrupt to CPU when CDIF is set to 1. Card Detect Pin is recommend setting before enable SC.

6.20.5.2 Activation, Warm Reset and Deactivation Sequence

Activation

The activation sequence is shown in Figure 6.20-4:

1. Set SCn_RST to low by programming RSTEN (SCn_PINCTL[1]) to '0', and wait SYNC (SCn_PINCTL[30]) is cleared to 0.
2. Set SCn_PWR at high level by programming PWREN (SCn_PINCTL[0]) to '1' and SCn_DATA at high level (reception mode) by programming SCDATA (SCn_PINCTL[9]) to '1', and wait SYNC (SCn_PINCTL[30]) is cleared to 0.
3. Enable SCn_CLK clock by programming CLKKEEP (SCn_PINCTL[6]) to '1', and wait SYNC (SCn_PINCTL[30]) is cleared to 0.
4. De-assert SCn_RST to high by programming RSTEN (SCn_PINCTL[1]) to '1', and wait SYNC (SCn_PINCTL[30]) is cleared to 0.

The activation sequence can be controlled in two ways. The procedure is shown as follows:

- Software Timing Control:

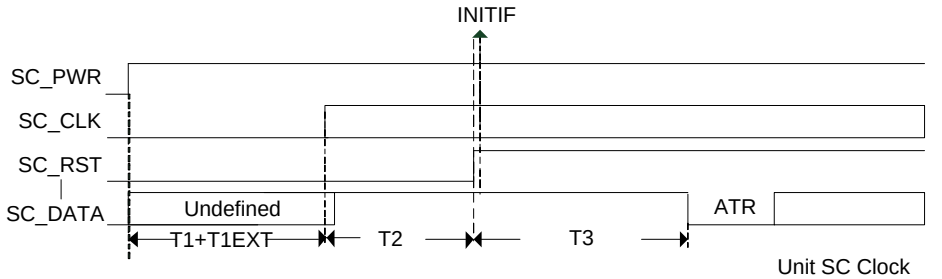
  Set SCn_PINCTL and SCn_TMRCTLx (x = 0, 1, 2) to process the activation sequence. SCn_PWR, SCn_CLK, SCn_RST and SCn_DATA pin state can be programmed by SCn_PINCTL. The programming method is shown in activation sequence. The activation sequence timing can be controlled by setting SCn_TMRCTLx (x = 0, 1, 2). This programming procedure provides user with a flexible timing setting for activation sequence.

- Hardware Timing Control:

  Set ACTEN (SCn_ALTCTL[3]) to '1' and the interface will perform the activation sequence by hardware. The SCn_PWR to SCn_CLK start (T1) and SCn_CLK_start to SCn_RST assert (T2) can be selected by programming INITSEL (SCn_ALTCTL[9:8]). The SCn_PWR to SCn_CLK length can be configure by setting T1EXT (SCn_ACTCTL[4:0]). This programming procedure provides user with a simple setting for activation sequence. During the hardware activation, RX receive is disabled and can not receive data.

The following is activation control sequence in hardware activation mode:

1. Set activation timing by setting INITSEL (SCn_ALTCTL[9:8]).
2. Timer0 can be selected by setting TMRSEL (SCn_CTL[14:13]) is 11.
3. Set operation mode OPMODE (SCn_TMRCTL0[27:24]) to 0011 and give an Answer to Request (ATR) value by setting CNT (SCn_TMRCTL0[23:0]) register.
4. When hardware de-asserts SCn_RST to high, hardware will generator an interrupt INITIF (SCn_INTSTS[8]) to CPU at the same time if INITIEN (SCn_INTEN[8]) is 1.
5. If the Timer0 decreases the counter to "0" (started from SCn_RST de-assert) and the card does not response ATR before that time, hardware will generate an interrupt flag TMR0IF (SCn_INTSTS[3]).



Time	Comment
T1	SC_PWR to SC_CLK Start
T1EXT	Configurable cycles for T1
T2	SC_CLK Start to SC_RST Assert
T3	SC_CLK Start to ATR Appear

INITSEL (SC_ALTCTL[9:8])	T1	T2
00	83.5	491
01	133	537
10	165	569
11	165	42060

Note : The values are measured by chip I / O pin and the real value will depend on system design

Figure 6.20-4 SC Activation Sequence

Warm Reset

The warm reset sequence is shown in Figure 6.20-5 :

1. Set SCn_RST to low by programming RSTEN (SCn_PINCTL[1]) to '0', and wait SYNC (SCn_PINCTL[30]) is cleared to 0.
2. Set SCn_DATA to high by programming SCDATA (SCn_PINCTL[9]) to '1', and wait SYNC (SCn_PINCTL[30]) is cleared to 0.
3. Set SCn_RST to high by programming RSTEN (SCn_PINCTL[1]) to '1', and wait SYNC (SCn_PINCTL[30]) is cleared to 0.

The warm reset sequence can be controlled in two ways. The procedure is shown as follows.

- Software Timing Control:

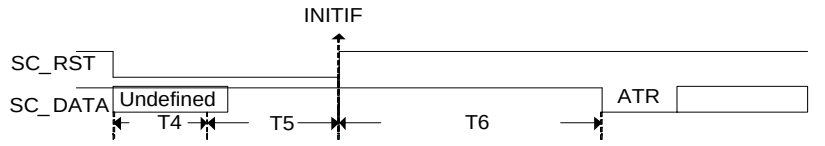
  Set SCn_PINCTL and SCn_TMRCTLx (x = 0, 1, 2) to process the warm reset sequence. The SCn_RST and SCn_DATA pin state can be programmed by SCn_PINCTL. The warm reset sequence timing can be controlled by setting SCn_TMRCTLx (x = 0, 1, 2). This programming procedure provides user with a flexible timing setting for warm reset sequence.

- Hardware Timing Control:

  Set WARSTEN (SCn_ALTCTL[4]) to '1' and the interface will perform the warm reset sequence by hardware. The SCn_RST to SCn_DATA reception mode (T4) and SCn_DATA reception mode to SCn_RST assert (T5) can be selected by programming INITSEL (SCn_ALTCTL[9:8]). This programming procedure provides user with a simple setting for warm reset sequence. During the hardware warm reset, RX receive is disabled and can not receive data.

The following is the warm reset control sequence by hardware:

1. Set warm reset timing by setting INITSEL (SCn_ALTCTL[9:8]).
2. Select Timer0 by setting TMRSEL (SCn_CTL[14:13]) to 11.
3. Set operation mode OPMODE (SCn_TMRCTL0[27:24]) to 0011 and give an Answer to Request (ATR) value by setting CNT (SCn_TMRCTL0[23:0]) register.
4. Set CNTEN0 (SCn_ALTCTL[5]) and WARSTEN (SCn_ALTCTL[4]) to start counting.
5. When hardware de-asserts SCn_RST to high, hardware will generate an interrupt INITIF (SCn_INTSTS[8]) to CPU at the same time if INITIEN (SCn_INTEN[8]) is 1.
6. If the Timer0 decreases the counter to '0' (start from SCn_RST) and the card does not response ATR before that time, hardware will generate an interrupt flag TMR0IF (SCn_INTSTS[3]).



Time	Comment
T4	SC_RST to SC_DATA Reception Mode
T5	SC_DATA Reception Mode to SC_RST Assert
T6	SC_CLK Start to ATR Appear

INITSEL (SC_ALTCTL[9:8])	T4	T5
00	81	483
01	129	531
10	161	563
11	161	42106

Unit: SC Clock

Note: This value is measured by chip IO pin and the real value will depend on system design

Figure 6.20-5 SC Warm Reset Sequence

Deactivation

The deactivation sequence is shown in Figure 6.20-6.

1. Set SCn_RST to low by programming RSTEN (SCn_PINCTL[1]) to '0', wait SYNC (SCn_PINCTL[30]) is cleared to 0.
2. Stop SCn_CLK by programming CLKKEEP (SCn_PINCTL[6]) to '0', wait SYNC (SCn_PINCTL[30]) is cleared to 0.
3. Set SCn_DATA to low by programming SCDATA (SCn_PINCTL[9]) to '0', wait SYNC (SCn_PINCTL[30]) is cleared to 0.
4. Deactivate SCn_PWR by programming PWREN (SCn_PINCTL[0]) to '0', wait SYNC (SCn_PINCTL[30]) is cleared to 0.

The deactivation sequence can be controlled in two ways. The procedure is shown as follows.

- Software Timing Control:

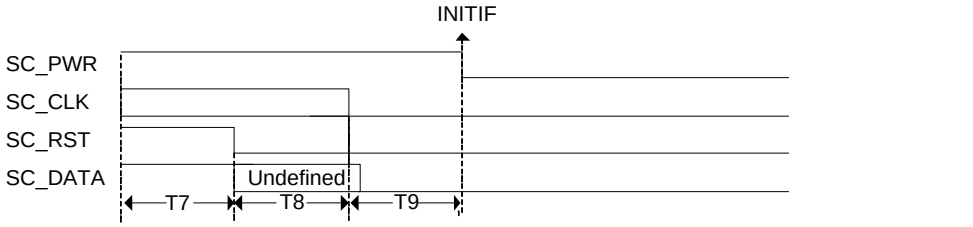
  Set SCn_PINCTL and SCn_TMRCTL0 to process the deactivation sequence. SCn_PWR, SCn_CLK, SCn_RST and SCn_DATA pin state can be programmed by SCn_PINCTL. The deactivation sequence timing can be controlled by setting SCn_TMRCTL0. This programming procedure provides user with a flexible timing setting for deactivation sequence.

- Hardware Timing Control:

  DACTEN (SCn_ALTCTL[2]) to '1' and the interface will perform the deactivation sequence by hardware. The Deactivation Trigger to SCn_RST low (T7), SMC_RST low to SCn_CLK (T8) and stop SCn_CLK to stop SCn_PWR (T9) time can be selected by programming INITSEL (SCn_ALTCTL[9:8]). This programming procedure provides user with a simple setting for deactivation sequence.

When hardware de-asserts SCn_PWR to low, the SC controller will generate an interrupt INITIF (SCn_INTSTS[8]) to CPU at the same time if INITIEN (SCn_INTEN[8]) is 1.

The SC controller also supports auto deactivation sequence when the card removal detection is enabled by setting ADACEN (SCn_ALTCTL[11]).



Time	Comment
T7	Deactivation Trigger to SC_RST Low
T8	SMC_RST Low to Stop SC_CLK
T9	Stop SC_CLK to Stop SC_PWR

INITSEL (SC_ALTCTL[9:8])	T7	T8	T9
00	97	83	87
01	145	131	135
10	177	163	167
11	177	163	167

Unit: SC Clock

Figure 6.20-6 SC Deactivation Sequence

6.20.5.3 Basic Operation Flow

Basic operation flow of smart card can be referenced from ISO 7816-3 & EMV.

The Program Sequence Flow is shown as follows:

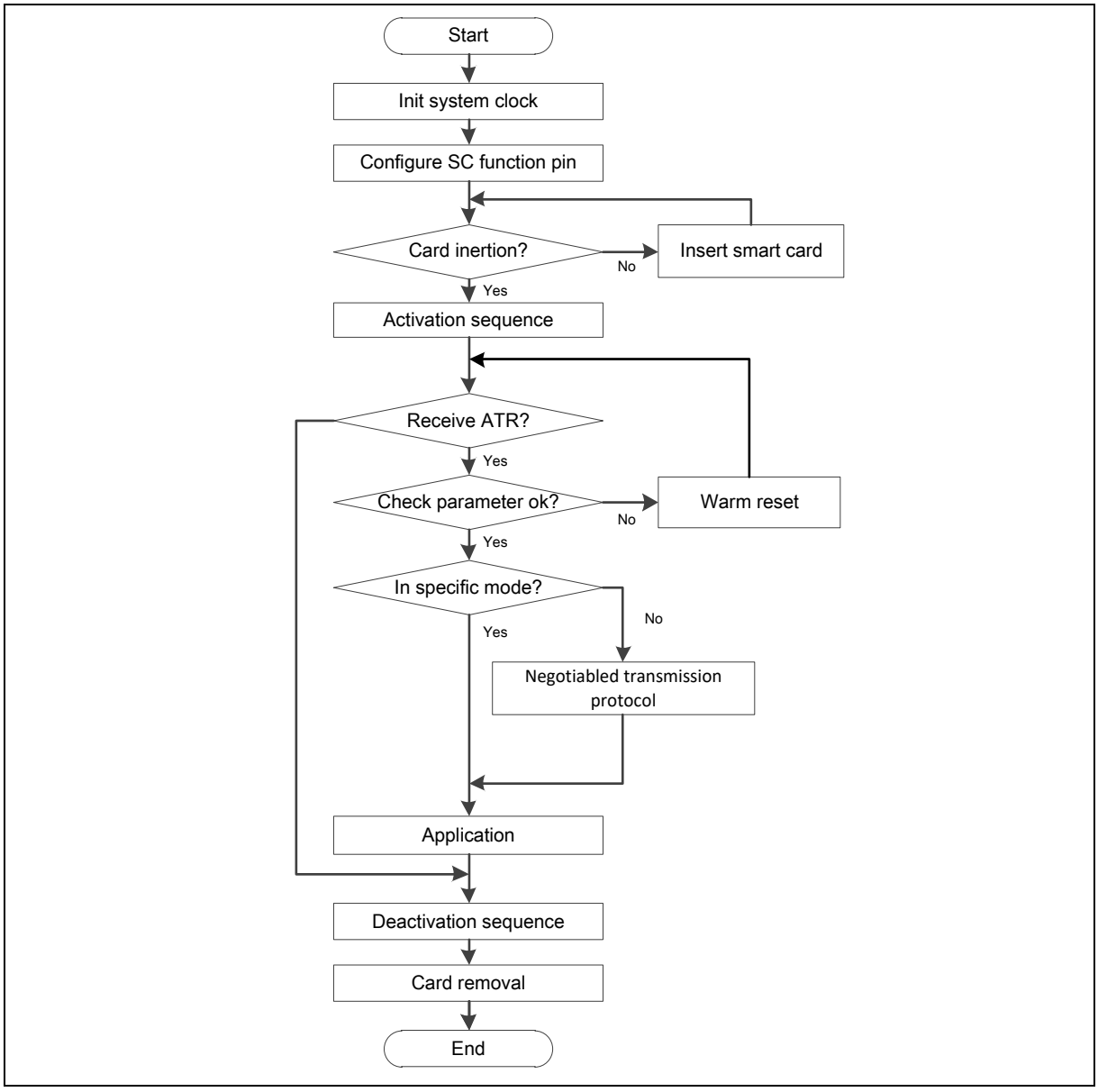


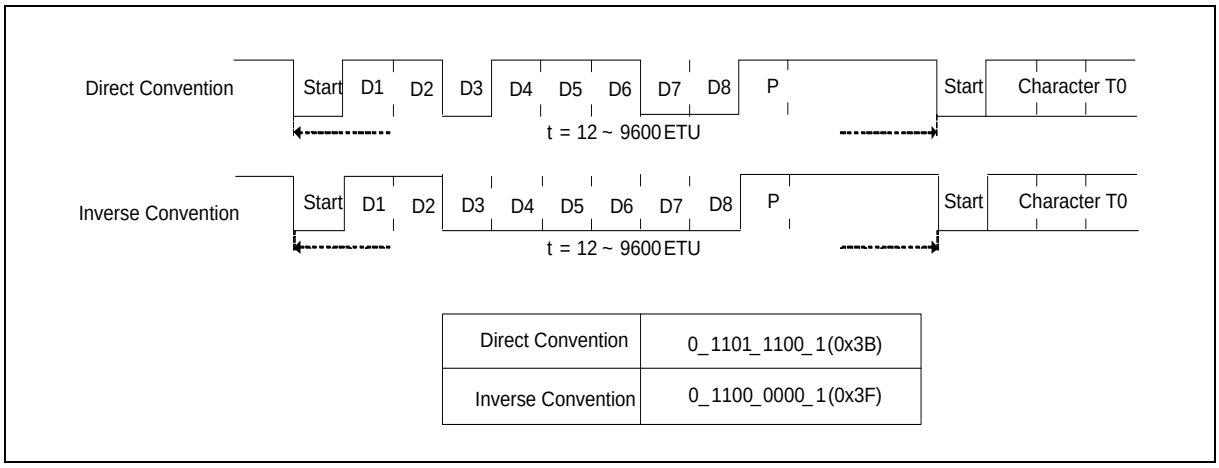
Figure 6.20-7 Basic Operation Flow

6.20.5.4 Initial Character TS

According to ISO 7816-3, the initial character TS has two possible patterns shown in Figure 6.20-8. If the TS pattern is 1100_0000, it is inverse convention. When decoded by inverse convention, the conveyed byte is equal to 0x3F. If the TS pattern is 1101_1100, it is direct convention. When decoded by direct convention, the conveyed byte is equal to 0x3B. User can set AUTOCEN (SCn_CTL[3]) and then the operating convention will be decided by hardware. User can also set the CONSEL (SCn_CTL[5:4]) register (set to '00' or '11') to change the operating convention after SC received TS of answer to request (ATR).

If auto convention function is enabled by setting AUTOCEN (SCn_CTL[3]) register, the setting step must be done before Answer to Request (ATR) state and the received first data must be 0x3B or 0x3F. After hardware received first data and stored it at SCn_DAT, the hardware will decided the convention and change the CONSEL (SCn_CTL[5:4]) register automatically. If the received first data is neither

0x3B nor 0x3F, ACERRIF (SCn_INTSTS[10] Auto Convention Error Interrupt Status Flag) will be set and the hardware will generate an interrupt to CPU if ACERRIEN (SCn_INTEN[10]) is 1.



Direct Convention	0_1101_1100_1 (0x3B)
Inverse Convention	0_1100_0000_1 (0x3F)

Figure 6.20-8 Initial Character TS

6.20.5.5 Transfer Data Flow and Data Buffer Status

After setting initial sequence, SC can start transferring data which format is corresponding to ISO 7816-3. Set ETURDIV (SCn_ETUCTL[11:0]) to 273 to make ETU (Element Timing Unit) meet ISO 7816-3. Writing data to SCn_DAT, SC will send out an 8-bit data to smart card. Reading data from SCn_DAT, SC will return an 8-bit received data from smart card.

Data buffer status show in SCn_STATUS. TXPOINT (SCn_STATUS[26:24]) and RXPOINT (SCn_STATUS[18:16]) represent how many data in transmit buffer and received buffer. TXEMPTY (SCn_STATUS[9]), TXFULL (SCn_STATUS[10]), TXOV (SCn_STATUS[8]), RXEMPTY (SCn_STATUS[1]), RXFULL (SCn_STATUS[2]), RXOV (SCn_STATUS[0]), represent the transmitted/received buffer status is full, empty, or overflow. SC even can generate interrupt for the transmit buffer empty situation by setting TBEIEN (SCn_INTEN[1]) bit. After interrupt status flag TBEIF (SCn_INTSTS[1]) is generated, user can decide to transfer data or not by polling these flag and it only can be cleared automatically when write data to SCn_DAT again.

TX and RX can be disabled separately by setting TXOFF (SCn_CTL[2]) and RXOFF (SCn_CTL[1]). TXACT (SCn_STATUS[31]) and RXACT (SCn_STATUS[23]) represent the TX transfer/RX transfer is active or not.

6.20.5.6 Receiver Buffer Time-out

The time-out down counter resets and starts counting whenever the RX buffer received a new data. Once the counter decrease to 1 and no new data is received or CPU does not read data by reading SCn_DAT, a receiver time-out flag RXTOIF (SCn_INTSTS[9]) will be set, and hardware will generate an interrupt to CPU when RXTOIEN (SCn_INTEN[9]) is enabled.

6.20.5.7 Error Signal and Character Repetition

According to ISO 7816-3 T = 0 mode description, as shown in Figure 6.20-9, if the receiver receives a wrong parity bit, it will pull the SCn_DAT to low by 1.5 bit period to inform the transmitter parity error. Then the transmitter will retransmit the character. The SC interface controller supports hardware error detection function in receiver and supports hardware re-transmit function in transmitter.

User can enable re-transmit function by setting TXRTYEN (SCn_CTL[23]). User can also define the retry (re-transmit) number limitation in TXRTY (SCn_CTL[22:20]). If the re-transmit number is between 1 and TXRTY, TXRERR (SCn_STATUS[29]) flag will be set by hardware. The re-transmit number is up to TXRTY +1 and if the re-transmit number is equal to TXRTY +1, TXOVERR (SCn_STATUS[30]) flag will be set by hardware and if TERRIEN (SCn_INTEN[2]) is enabled, SC controller will generate a

transfer error interrupt to CPU, and TERRIF (SCn_INTSTS[2]) flag will also be set.

User can also enable re-received function by setting RXRTYEN (SCn_CTL[19]).The received retry number limitation is defined in RXRTY (SCn_CTL[18:16]). If the re-received number is between 1 and RXRTY, RXRERR (SCn_STATUS[21]) flag will be set by hardware. The receiver retry number is up to RXRTY +1, if the number of received errors by receiver is equal to RXRTY +1, receiver will receive this error data to buffer and RXOVERR (SCn_STATUS[22]) flag will be set by hardware and if TERRIEN (SCn_INTEN[2]) is enabled, SC controller will generate a transfer error interrupt to CPU, and TERRIF (SCn_INTSTS[2]) flag will also be set.

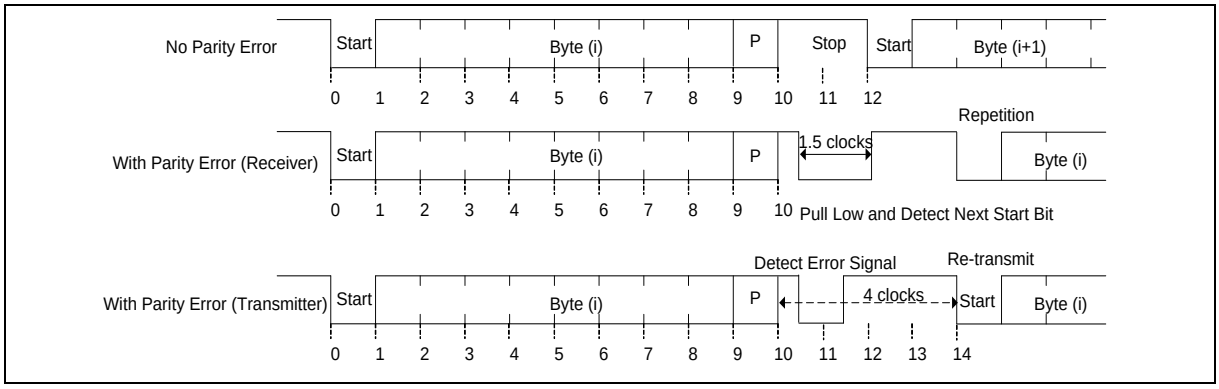


Figure 6.20-9 SC Error Signal

6.20.5.8 Internal Timer Operation Mode

The Smart Card Interface includes a 24-bit time-out counter and two 8 bit time-out counters. These counters help the SC controller in processing different real-time interval. Each counter can be set to start counting once the trigger enable bit (CNTENx in SCn_ALTCTL[7:5], x = 0, 1, 2) has been written or a START bit has been detected.

The following is the programming flow:

1. Enable counter by setting TMRSEL (SCn_CTL[14:13]) to 11.
2. Select operation mode OPMODE (SCn_TMRCTLx[27:24], x = 0, 1, 2).
3. Give a count value CNT for Timer0, Timer1 and Timer2 by setting CNT0 (SCn_TMRCTL0[23:0]), CNT1 (SCn_TMRCTL1[7:0]) and CNT2 (SCn_TMRCTL2[7:0]).
4. Set CNTEN0 (SCn_ALTCTL [5]), CNTEN1 (SCn_ALTCTL [6]) or CNTEN2 (SCn_ALTCTL [7]) to enable timer. ACTSTS0 (SCn_ALTCTL[13]), ACTSTS1 (SCn_ALTCTL[14]) and ACTSTS2 (SCn_ALTCTL[15]) represent the status that timer is enable or not.
5. Wait until the counting condition is satisfied, the timer start counting.
6. When internal timer counter satisfied interrupt conditions in different modes and TMR0IEN (SCn_INTEN[3]), TMR1IEN (SCn_INTEN[4]), TMR2IEN (SCn_INTEN[5]) are enable, SC will generate a interrupt to CPU.

The SCn_TMRCTL0, SCn_TMRCTL1 and SCn_TMRCTL2 timer operation mode are listed in Table 6.20-3.

Note1: Only Timer0 (SCn_TMRCTL0 register) supports mode 0011.

Note2: START bit can only be detected when Tx or Rx is idle or finish the last transmission.

OPMODE (SCn_TMRCTLx[27:24]), X = 0, 1, 2)	Operation Mode Description	
0000	The down counter is started when CNTENx (SCn_ALTCTL[7:5]) enabled and ended when counter time-out. The time-out counter value will be CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) +1.	
	Start	Start counting when CNTENx (SCn_ALTCTL[7:5]) enabled.
	End	When the down counter equals 0, hardware will set TMR0IF, TMR1IF, TMR2IF (SCn_INTSTS[5:3]) and clear CNTENx (SCn_ALTCTL[7:5]) automatically.
0001	The down counter is started when the first START bit (reception or transmission) detected and ended when counter time-out. It takes 2 ETU to detect first START bit after writing data to Tx or receiving data from Rx. The time-out counter value will be CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) +1.	
	Start	Start counting when the first START bit (reception or transmission) detected after CNTENx (SCn_ALTCTL[7:5]) set to 1.
	End	When the down counter equals 0, hardware will set TMR0IF, TMR1IF, TMR2IF (SCn_INTSTS[5:3]) and clear CNTENx (SCn_ALTCTL[7:5]) automatically.
0010	The down counter is started when the first START bit (reception) detected and ended when counter time-out. It takes 2 ETU to detect first START bit after receiving data from Rx. The time-out value will be CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) +1.	
	Start	Start counting when the first START bit (reception) detected bit after CNTENx (SCn_ALTCTL[7:5]) set to 1.
	End	When the down counter equals 0, hardware will set TMR0IF, TMR1IF, TMR2IF (SCn_INTSTS[5:3]) and clear CNTENx (SCn_ALTCTL[7:5]) automatically.
0011	The down counter is only used for hardware activation, warm reset sequence to measure ATR timing. The timing starts when SCn_RST de-assertion and ends when ATR response received or time-out. If the counter decreases to 0 before ATR response received, hardware will set TMR0IF (SCn_INTSTS[3]) and generate an interrupt to CPU if TMR0IEN (SCn_INTEN[3]) enabled. The time-out value will be CNT (SCn_TMRCTL0[23:0]) +1.	
	Start	Start counting when SCn_RST de-assertion after CNTEN0 (SCn_ALTCTL[5]) set to 1. It is only used for hardware activation, warm reset mode.
	End	When the down counter equals 0 before ATR response received, hardware will set TMR0IF and clear CNTEN0 (SCn_ALTCTL[5]) automatically. When ATR received and down counter does not equal to 0, hardware will clear CNTEN0 (SCn_ALTCTL[5]) automatically.
0100	Start	Start down counter counting when CNTENx (SCn_ALTCTL[7:5]) enabled.
	Recount & reload	When ACTSTSx (SCn_ALTCTL[15:13]) is 1, user can change CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL0[7:0], SCn_TMRCTL0[7:0]) value at any time. It will reload the last value which is filled into the CNT(SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) before the counter count to 0. Only when the down counter equals 0, counter reload the CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) value and start to recount.
	Interrupt	If the counter decreases to 0, hardware will set TMR0IF, TMR1IF, TMR2IF (SCn_INTSTS[5:3]) and generate an interrupt to CPU if TMRxIEN (SCn_INTEN[5:3]) enabled. The time-out value will be CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) +1.
	End	The down counter stopped when use clears CNTENx (SCn_ALTCTL[7:5]) bit.

0101	Start	The down counter is started when the first START bit (reception or transmission) detected after CNTENx (SCn_ALTCTL[7:5]) set to 1. It takes 2 ETU to detect START bit after writing data to Tx or receiving data from Rx.
	Reload	When ACTSTSx (SCn_ALTCTL[15:13]) is 1, user can change CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL0[7:0], SCn_TMRCTL0[7:0]) value at any time. It will reload the last value which is filled into the CNT(SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) before the counter count to 0. Only when the down counter equals 0, counter will reload the CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) value.
	Recount	After down counter reloads the CNT value, timer counter starts to recount only when the next START bit is detected.
	Interrupt	If the counter decreases to 0, hardware will set TMR0IF, TMR1IF, TMR2IF (SCn_INTSTS[5:3]) and generate an interrupt to CPU if TMRxIEN (SCn_INTEN[5:3]) enabled. The time-out value will be CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) +1.
	End	The down counter stopped when user clears CNTENx (SCn_ALTCTL[7:5]) bit.
0110	Start	The down counter is started when the first START bit (reception) detected after CNTENx (SCn_ALTCTL[7:5]) set to 1. It takes 2 ETU to detect START bit after writing data to Tx or receiving data from Rx.
	Reload	When ACTSTSx (SCn_ALTCTL[15:13]) is 1, user can change CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL0[7:0], SCn_TMRCTL0[7:0]) value at any time. It will reload the last value which is filled into the CNT(SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) before the counter counts to 0. Only when the down counter equals 0, counter reload the CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) value.
	Recount	After the down counter reloads the CNT value, timer counter starts to recount only when the next START bit is detected.
	Interrupt	If the counter decreases to 0, hardware will set TMR0IF, TMR1IF, TMR2IF (SCn_INTSTS[5:3]) and generate an interrupt to CPU if TMRxIEN (SCn_INTEN[5:3]) enabled. The time-out value will be CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0])+1.
	End	The down counter stopped when user clears CNTENx (SCn_ALTCTL[7:5]) bit.
0111	Start	The down counter is started when the first START bit (reception or transmission) detected after CNTENx (SCn_ALTCTL[7:5]) set to 1. It takes 2 ETU to detect START bit after writing data to Tx or receiving data from Rx.
	Reload &recount	Only when the next START bit is detected, counter will reload the new value of CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) and recount.
	Interrupt	If the counter decreases to 0 before the next START bit detected, hardware will set TMR0IF, TMR1IF, TMR2IF (SCn_INTSTS[5:3]) and generate an interrupt to CPU if TMRxIEN (SCn_INTEN[5:3]) enabled. The time-out value will be CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) +1.
	End	The down counter stopped when user clears CNTENx (SCn_ALTCTL[7:5]) bit.
1111	Start	The down counter starts counting when user sets CNTENx (SCn_ALTCTL[7:5]) bit and it will count to time-out.
	Reload	Only when the next START bit is detected, counter will reload the new value of CNT

	&recount	(SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) and recount.
	Interrupt	If the counter decreases to 0 before the next START bit detected, hardware will generate time-out interrupt flag TMR0IF, TMR1IF, TMR2IF (SCn_INTSTS[5:3]). The time-out value will be CNTx (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) +1.
	End	The down counter stopped when user clears CNTENx (SCn_ALTCTL[7:5]) bit.

Table 6.20-3 Timer0/Timer1/Timer2 Operation Mode

6.20.5.9 Block Guard Time and Extra Guard Time

Block guard time means the minimum interval between the leading edges of two consecutive characters between different transfer directions. This field indicates the counter for the bit length of block guard time. According to ISO7816-3, in T = 0 mode, user must fill 15 (real block guard time = 16.5) to this field; in T = 1 mode, user must fill 21 (real block guard time = 22.5) to it.

In transmit direction, the smart card sends data to Smart Card host controller, first. After the period is greater than BGT (SCn_CTL[12:8]), the Smart Card host controller begin to send the data.

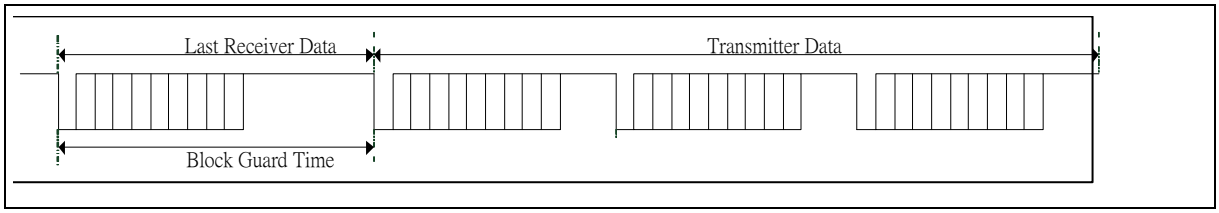


Figure 6.20-10 Transmit Direction Block Guard Time Operation

In receive direction, the Smart Card host controller sends data to smart card, first. If the smart card responses data to Smart Card host controller at the time which is less than BGT (SCn_CTL[12:8]), the block guard time interrupt BGTIF (SCn_INTSTS[6]) is generated when RXBGTEN (SCn_ALTCTL[12]) and BGTIEN (SCn_INTEN[6]) is enabled.

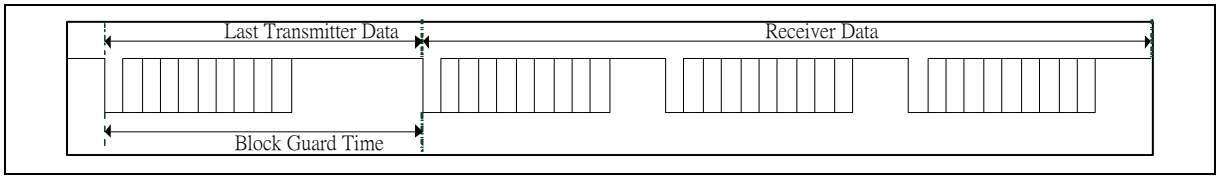


Figure 6.20-11 Receive Direction Block Guard Time Operation

Extra Guard Time is EGT (SCn_EGT[7:0]), it only affects the data transmitted by Smart Card Interface, the format is shown as Figure 6.20-12Figure 6.20-11.

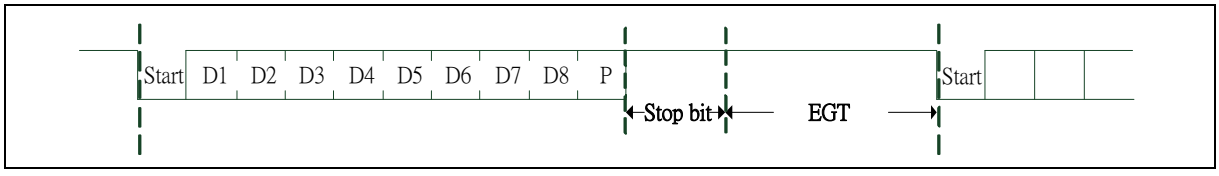


Figure 6.20-12 Extra Guard Time Operation

6.20.5.10 UART Mode

When the UARTEN (SCn_UARTCTL[0]) bit is set, the Smart Card Interface controller can also be used as basic UART function. The following is the program example for UART mode.

Programming example:

1. Set UARTEN (SCn_UARTCTL[0]) bit to enter UART mode.
2. Do user reset by setting RXRST (SCn_ALTCTL[1]) and TXRST (SCn_ALTCTL[0]) bit to ensure that all state machine return idle state.
3. Fill “0” to CONSEL (SCn_CTL[5:4]) and AUTOCEN (SCn_CTL[3]) field. In UART mode, those fields must be “0”.
4. Select the UART baud rate by setting ETURDIV (SCn_ETUCTL[11:0]) fields. For example, if smart card module clock is 12 MHz and target baud rate is 115200 bps, ETURDIV should fill with ((12000000 / 115200) - 1).
5. Select the data format include data length (by setting WLS (SCn_UARTCTL[5:4]), parity format (by setting OPE (SCn_UARTCTL[7]) and PBOFF (SCn_UARTCTL[6])) and stop bit length (by setting NSB (SCn_CTL[15] or EGT (SCn_EGT[7:0])).
6. Select the receiver buffer number trigger level by setting RXTRGLV (SCn_CTL[7:6]) field and select the receiver buffer time-out interval by setting RFTM (SCn_RXTOUT[8:0]) field.
7. Write the SCn_DAT (SCn_DAT[7:0]) (TX) register or read the SCn_DAT (SCn_DAT[7:0]) (RX) register can perform UART function.

6.20.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
SC Base Address: **SCn_BA = 0x4009_0000 + (0x1000 * n)** **n=0,1,2** **SC non-secure base address is SCn_BA + 0x1000_0000.**				
SC_DAT	SCn_BA+0x00	R/W	SC Receive/Transmit Holding Buffer Register	0xXXXX_XXXX
SC_CTL	SCn_BA+0x04	R/W	SC Control Register	0x0000_0000
SC_ALTCTL	SCn_BA+0x08	R/W	SC Alternate Control Register	0x0000_0000
SC_EGT	SCn_BA+0x0C	R/W	SC Extra Guard Time Register	0x0000_0000
SC_RXTOUT	SCn_BA+0x10	R/W	SC Receive Buffer Time-out Counter Register	0x0000_0000
SC_ETUCTL	SCn_BA+0x14	R/W	SC Element Time Unit Control Register	0x0000_0173
SC_INTEN	SCn_BA+0x18	R/W	SC Interrupt Enable Control Register	0x0000_0000
SC_INTSTS	SCn_BA+0x1C	R/W	SC Interrupt Status Register	0x0000_0002
SC_STATUS	SCn_BA+0x20	R/W	SC Transfer Status Register	0x0000_X202
SC_PINCTL	SCn_BA+0x24	R/W	SC Pin Control State Register	0x0000_0000
SC_TMRCTL0	SCn_BA+0x28	R/W	SC Internal Timer0 Control Register	0x0000_0000
SC_TMRCTL1	SCn_BA+0x2C	R/W	SC Internal Timer1 Control Register	0x0000_0000
SC_TMRCTL2	SCn_BA+0x30	R/W	SC Internal Timer2 Control Register	0x0000_0000
SC_UARTCTL	SCn_BA+0x34	R/W	SC UART Mode Control Register	0x0000_0000
SC_ACTCTL	SCn_BA+0x4C	R/W	SC Activation Control Register	0x0000_0000

6.20.7 Register Description

SC Receive/Transmit Holding Buffer Register (SC_DAT)

Register	Offset	R/W	Description	Reset Value
SC_DAT	SCn_BA+0x00	R/W	SC Receive/Transmit Holding Buffer Register	0xXXXX_XXXX

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**DAT**	**Receive/Transmit Holding Buffer** Write Operation: By writing data to DAT, the SC will send out an 8-bit data. Read Operation: By reading DAT, the SC will return an 8-bit received data. **Note:** If SCEN (SCn_CTL[0]) is not enabled, DAT cannot be programmed.

SC Control Register (SC_CTL)

Register	Offset	R/W	Description	Reset Value
SC_CTL	SCn_BA+0x04	R/W	SC Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	SYNC	Reserved			CDLV	CDDBSEL	
23	22	21	20	19	18	17	16
TXRTYEN	TXRTY			RXRTYEN	RXRTY		
15	14	13	12	11	10	9	8
NSB	TMRSEL		BGT				
7	6	5	4	3	2	1	0
RXTRGLV		CONSEL		AUTOCEN	TXOFF	RXOFF	SCEN

Bits	Description	
[31]	**Reserved**	Reserved.
[30]	**SYNC**	**SYNC Flag Indicator (Read Only)** Due to synchronization, user should check this bit before writing a new value to RXRTY and TXRTY fields. 0 = Synchronizing is completion, user can write new data to RXRTY and TXRTY. 1 = Last value is synchronizing.
[29:27]	**Reserved**	Reserved.
[26]	**CDLV**	**Card Detect Level Selection** 0 = When hardware detects the card detect pin (SCn_CD) from high to low, it indicates a card is detected. 1 = When hardware detects the card detect pin (SCn_CD) from low to high, it indicates a card is detected. **Note:** User must select card detect level before Smart Card controller enabled.
[25:24]	**CDDBSEL**	**Card Detect De-bounce Selection** This field indicates the card detect de-bounce selection. 00 = De-bounce sample card insert once per 384 (128 * 3) SC module clocks and de-bounce sample card removal once per 128 SC module clocks. Other configurations are reserved.
[23]	**TXRTYEN**	**TX Error Retry Enable Bit** This bit enables transmitter retry function when parity error has occurred. 0 = TX error retry function Disabled. 1 = TX error retry function Enabled.
[22:20]	**TXRTY**	**TX Error Retry Count Number** This field indicates the maximum number of transmitter retries that are allowed when parity error has occurred. **Note 1:** The real retry number is TXRTY + 1, so 8 is the maximum retry number. **Note 2:** This field cannot be changed when TXRTYEN enabled. The change flow is to disable TXRTYEN first and then fill in new retry value.

[19]	**RXRTYEN**	**RX Error Retry Enable Bit** This bit enables receiver retry function when parity error has occurred. 0 = RX error retry function Disabled. 1 = RX error retry function Enabled. **Note:** User must fill in the RXRTY value before enabling this bit.
[18:16]	**RXRTY**	**RX Error Retry Count Number** This field indicates the maximum number of receiver retries that are allowed when parity error has occurred **Note 1:** The real retry number is RXRTY + 1, so 8 is the maximum retry number. **Note 2:** This field cannot be changed when RXRTYEN enabled. The change flow is to disable RXRTYEN first and then fill in new retry value.
[15]	**NSB**	**Stop Bit Length** This field indicates the length of stop bit. 0 = The stop bit length is 2 ETU. 1= The stop bit length is 1 ETU. **Note 1:** The default stop bit length is 2. SC and UART adopts NSB to program the stop bit length. **Note 2:** In UART mode, RX can receive the data sequence in 1 stop bit or 2 stop bits with NSB is set to 0.
[14:13]	**TMRSEL**	**Timer Channel Selection** 00 = All internal timer function Disabled. 11 = Internal 24-bit timer and two 8-bit timers Enabled. User can configure them by setting SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0] and SCn_TMRCTL2[7:0]. Other configurations are reserved.
[12:8]	**BGT**	**Block Guard Time** Block guard time means the minimum interval between the leading edges of two consecutive characters between different transfer directions. This field indicates the counter for the bit length of block guard time. According to ISO 7816-3, in T = 0 mode, user must fill 15 (real block guard time = 16.5) to this field; in T = 1 mode, user must fill 21 (real block guard time = 22.5) to it. **Note:** The real block guard time is BGT + 1.
[7:6]	**RXTRGLV**	**Rx Buffer Trigger Level** When the number of bytes in the receiving buffer equals the RXTRGLV, the RDAIF will be set. If RDAIEN (SCn_INTEN[0]) is enabled, an interrupt will be generated to CPU. 00 = Rx Buffer Trigger Level with 01 bytes. 01 = Rx Buffer Trigger Level with 02 bytes. 10 = Rx Buffer Trigger Level with 03 bytes. 11 = Reserved.
[5:4]	**CONSEL**	**Convention Selection** 00 = Direct convention. 01 = Reserved. 10 = Reserved. 11 = Inverse convention. **Note:** If AUTOCEN (SCn_CTL[3]) is enabled, this field is ignored.
[3]	**AUTOCEN**	**Auto Convention Enable Bit** This bit is used for enable auto convention function. 0 = Auto-convention Disabled. 1 = Auto-convention Enabled. **Note 1:** If user enables auto convention function, the setting step must be done before Answer to Reset (ATR) state and the first data must be 0x3B or 0x3F. After hardware received first data and stored it at

		buffer, hardware will decided the convention and change the CONSEL (SCn_CTL[5:4]) bits automatically when received first data is 0x3B or 0x3F. If received first byte is 0x3B, TS is direct convention, CONSEL (SCn_CTL[5:4]) will be set to 00 automatically, otherwise the TS is inverse convention, and CONSEL (SCn_CTL[5:4]) will be set to 11. **Note 2:** If the first data is not 0x3B or 0x3F, hardware will set ACERRIF (SCn_INTSTS[10]) and generate an interrupt to CPU when ACERRIEN (SCn_INTEN[10]) is enabled.
[2]	**TXOFF**	**TX Transition Disable Control Bit** This bit is used for disable Tx transition function. 0 = The transceiver Enabled. 1 = The transceiver Disabled.
[1]	**RXOFF**	**RX Transition Disable Control Bit** This bit is used for disable Rx transition function. 0 = The receiver Enabled. 1 = The receiver Disabled. **Note:** If AUTOCEN (SCn_CTL[3]) is enabled, this field is ignored.
[0]	**SCEN**	**SC Controller Enable Bit** Set this bit to 1 to enable SC operation. If this bit is cleared, 0 = SC will force all transition to IDLE state. 1 = SC controller is enabled and all function can work correctly. **Note:** SCEN must be set to 1 before filling in other SC registers, or smart card will not work properly.

SC Alternate Control Register (SC_ALTCTL)

Register	Offset	R/W	Description	Reset Value
SC_ALTCTL	SCn_BA+0x08	R/W	SC Alternate Control Register	0x0000_0000

31	30	29	28	27	26	25	24
SYNC	Reserved						
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
ACTSTS2	ACTSTS1	ACTSTS0	RXBGTEN	ADACEN	Reserved	INITSEL	
7	6	5	4	3	2	1	0
CNTEN2	CNTEN1	CNTEN0	WARSTEN	ACTEN	DACTEN	RXRST	TXRST

Bits	Description	
[31]	**SYNC**	**SYNC Flag Indicator (Read Only)** Due to synchronization, user should check this bit when writing a new value to SCn_ALTCTL register. 0 = Synchronizing is completion, user can write new data to SCn_ALTCTL register. 1 = Last value is synchronizing.
[30:16]	**Reserved**	Reserved.
[15]	**ACTSTS2**	**Internal Timer2 Active Status (Read Only)** This bit indicates the timer counter status of timer2. 0 = Timer2 is not active. 1 = Timer2 is active. **Note:** Timer2 is active does not always mean timer2 is counting the CNT (SCn_TMRCTL2[7:0]).
[14]	**ACTSTS1**	**Internal Timer1 Active Status (Read Only)** This bit indicates the timer counter status of timer1. 0 = Timer1 is not active. 1 = Timer1 is active. **Note:** Timer1 is active does not always mean timer1 is counting the CNT (SCn_TMRCTL1[7:0]).
[13]	**ACTSTS0**	**Internal Timer0 Active Status (Read Only)** This bit indicates the timer counter status of timer0. 0 = Timer0 is not active. 1 = Timer0 is active. **Note:** Timer0 is active does not always mean timer0 is counting the CNT (SCn_TMRCTL0[23:0]).
[12]	**RXBGTEN**	**Receiver Block Guard Time Function Enable Bit** This bit enables the receiver block guard time function. 0 = Receiver block guard time function Disabled. 1 = Receiver block guard time function Enabled.
[11]	**ADACEN**	**Auto Deactivation When Card Removal**

		This bit is used for enable hardware auto deactivation when smart card is removed. 0 = Auto deactivation Disabled. 1 = Auto deactivation Enabled. **Note:** When the card is removed, hardware will stop any process and then do deactivation sequence if this bit is set. If auto deactivation process completes, hardware will set INITIF (SCn_INTSTS[8]) also.
[10]	**Reserved**	Reserved.
[9:8]	**INITSEL**	**Initial Timing Selection** This fields indicates the initial timing of hardware activation, warm-reset or deactivation. The unit of initial timing is SC module clock. **Activation:** refer to SC Activation Sequence in Figure 6.20-4. **Warm-reset:** refer to Warm-Reset Sequence in Figure 6.20-5. **Deactivation:** refer to Deactivation Sequence in Figure 6.20-6. **Note:** When set activation and warm reset in Timer0 operation mode 0011, it may have deviation at most 128 SC module clock cycles.
[7]	**CNTEN2**	**Internal Timer2 Start Enable Bit** This bit enables Timer 2 to start counting. User can fill 0 to stop it and set 1 to reload and count. The counter unit is ETU base. 0 = Stops counting. 1 = Start counting. **Note 1:** This field is used for internal 8-bit timer when TMRSEL (SCn_CTL[14:13]) is 11 only. Do not fill in CNTEN2 when TMRSEL (SCn_CTL[14:13]) is not equal to 11. **Note 2:** If the operation mode is not in auto-reload mode (SCn_TMRCTL2[26] = 0), this bit will be auto-cleared by hardware. **Note 3:** If SCEN (SCn_CTL[0]) is not enabled, this filed cannot be programmed.
[6]	**CNTEN1**	**Internal Timer1 Start Enable Bit** This bit enables Timer 1 to start counting. User can fill 0 to stop it and set 1 to reload and count. The counter unit is ETU base. 0 = Stops counting. 1 = Start counting. **Note 1:** This field is used for internal 8-bit timer when TMRSEL(SCn_CTL[14:13]) is 11 only. Do not fill CNTEN1 when TMRSEL (SCn_CTL[14:13]) is not equal to 11. **Note 2:** If the operation mode is not in auto-reload mode (SCn_TMRCTL1[26] = 0), this bit will be auto-cleared by hardware. **Note 3:** If SCEN (SCn_CTL[0]) is not enabled, this filed cannot be programmed.
[5]	**CNTEN0**	**Internal Timer0 Start Enable Bit** This bit enables Timer 0 to start counting. User can fill 0 to stop it and set 1 to reload and count. The counter unit is ETU base. 0 = Stops counting. 1 = Start counting. **Note 1:** This field is used for internal 24-bit timer when TMRSEL (SCn_CTL[14:13]) is 11 only. **Note 2:** If the operation mode is not in auto-reload mode (SCn_TMRCTL0[26] = 0), this bit will be auto-cleared by hardware. **Note 3:** If SCEN (SCn_CTL[0]) is not enabled, this filed cannot be programmed.
[4]	**WARSTEN**	**Warm Reset Sequence Generator Enable Bit** This bit enables SC controller to initiate the card by warm reset sequence. 0 = No effect. 1 = Warm reset sequence generator Enabled. **Note 1:** When the warm reset sequence completed, this bit will be cleared automatically and the INITIF (SCn_INTSTS[8]) will be set to 1.

		Note 2: This field will be cleared by TXRST (SCn_ALTCTL[0]) and RXRST (SCn_ALTCTL[1]). Thus, do not fill in this bit WARSTEN, TXRST and RXRST at the same time. **Note 3:** If SCEN (SCn_CTL[0]) is not enabled, this filed cannot be programmed. **Note 4:** During the warm reset sequence, RX is disabled automatically and can not receive data. After the warm reset sequence completion, RXOFF (SCn_CTL[1]) keeps the state before perform warm reset sequence.
[3]	**ACTEN**	**Activation Sequence Generator Enable Bit** This bit enables SC controller to initiate the card by activation sequence. 0 = No effect. 1 = Activation sequence generator Enabled. **Note 1:** When the activation sequence completed, this bit will be cleared automatically and the INITIF (SCn_INTSTS[8]) will be set to 1. **Note 2:** This field will be cleared by TXRST (SCn_ALTCTL[0]) and RXRST (SCn_ALTCTL[1]). Thus, do not fill in this bit ACTEN, TXRST and RXRST at the same time. **Note 3:** If SCEN (SCn_CTL[0]) is not enabled, this filed cannot be programmed. **Note 4:** During the activation sequence, RX is disabled automatically and can not receive data. After the activation sequence completion, RXOFF (SCn_CTL[1]) keeps the state before hardware activation.
[2]	**DACTEN**	**Deactivation Sequence Generator Enable Bit** This bit enables SC controller to initiate the card by deactivation sequence. 0 = No effect. 1 = Deactivation sequence generator Enabled. **Note 1:** When the deactivation sequence completed, this bit will be cleared automatically and the INITIF (SCn_INTSTS[8]) will be set to 1. **Note 2:** This field will be cleared by TXRST (SCn_ALTCTL[0]) and RXRST (SCn_ALTCTL[1]). Thus, do not fill in this bit DACTEN, TXRST and RXRST at the same time. **Note 3:** If SCEN (SCn_CTL[0]) is not enabled, this filed cannot be programmed.
[1]	**RXRST**	**Rx Software Reset** When RXRST is set, all the bytes in the receive buffer and Rx internal state machine will be cleared. 0 = No effect. 1 = Reset the Rx internal state machine and pointers. **Note:** This bit will be auto cleared after reset is complete.
[0]	**TXRST**	**TX Software Reset** When TXRST is set, all the bytes in the transmit buffer and TX internal state machine will be cleared. 0 = No effect. 1 = Reset the TX internal state machine and pointers. **Note:** This bit will be auto cleared after reset is complete.

SC Extra Guard Time Register (SC_EGT)

Register	Offset	R/W	Description	Reset Value
SC_EGT	SCn_BA+0x0C	R/W	SC Extra Guard Time Register	0x0000_0000

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**EGT**	**Extra Guard Time** This field indicates the extra guard time value. **Note:** The extra guard time unit is ETU base.

SC Receiver Buffer Time-out Register (SC_RXTOUT)

Register	Offset	R/W	Description	Reset Value
SC_RXTOUT	SCn_BA+0x10	R/W	SC Receive Buffer Time-out Counter Register	0x0000_0000

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

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8:0]	**RFTM**	**SC Receiver FIFO Time-out Counter** The time-out down counter resets and starts counting whenever the RX buffer received a new data. Once the counter decrease to 1 and no new data is received or CPU does not read data by reading SCn_DAT, a receiver time-out flag RXTOIF (SCn_INTSTS[9]) will be set, and hardware will generate an interrupt to CPU when RXTOIEN (SCn_INTEN[9]) is enabled. **Note 1:** The counter unit is ETU based and the interval of time-out is RFTM + 0.5. **Note 2:** Filling in all 0 to this field indicates to disable this function.

SC Element Time Unit Control Register (SC_ETUCTL)

Register	Offset	R/W	Description	Reset Value
SC_ETUCTL	SCn_BA+0x14	R/W	SC Element Time Unit Control Register	0x0000_0173

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

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**ETURDIV**	**ETU Rate Divider** The field is used for ETU clock rate divider. The real ETU is ETURDIV + 1. **Note:** User can configure this field, but this field must be greater than 0x04.

SC Interrupt Enable Control Register (SC_INTEN)

Register	Offset	R/W	Description	Reset Value
SC_INTEN	SCn_BA+0x18	R/W	SC Interrupt Enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					ACERRIEN	RXTOIEN	INITIEN
7	6	5	4	3	2	1	0
CDIEN	BGTIEN	TMR2IEN	TMR1IEN	TMR0IEN	TERRIEN	TBEIEN	RDAIEN

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10]	**ACERRIEN**	**Auto Convention Error Interrupt Enable Bit** This field is used to enable auto-convention error interrupt. 0 = Auto-convention error interrupt Disabled. 1 = Auto-convention error interrupt Enabled.
[9]	**RXTOIEN**	**Receiver Buffer Time-out Interrupt Enable Bit** This field is used to enable receiver buffer time-out interrupt. 0 = Receiver buffer time-out interrupt Disabled. 1 = Receiver buffer time-out interrupt Enabled.
[8]	**INITIEN**	**Initial End Interrupt Enable Bit** This field is used to enable activation (ACTEN (SCn_ALTCTL[3]) = 1), deactivation (DACTEN (SCn_ALTCTL[2]) = 1) and warm reset (WARSTEN (SCn_ALTCTL [4]) = 1) sequence complete interrupt. 0 = Initial end interrupt Disabled. 1 = Initial end interrupt Enabled.
[7]	**CDIEN**	**Card Detect Interrupt Enable Bit** This field is used to enable card detect interrupt. The card detect status is CDPINSTS (SCn_STATUS[13]). 0 = Card detect interrupt Disabled. 1 = Card detect interrupt Enabled.
[6]	**BGTIEN**	**Block Guard Time Interrupt Enable Bit** This field is used to enable block guard time interrupt in receive direction. 0 = Block guard time interrupt Disabled. 1 = Block guard time interrupt Enabled. **Note:** This bit is valid only for receive direction block guard time.
[5]	**TMR2IEN**	**Timer2 Interrupt Enable Bit** This field is used to enable Timer2 interrupt function.

		0 = Timer2 interrupt Disabled. 1 = Timer2 interrupt Enabled.
[4]	**TMR1IEN**	**Timer1 Interrupt Enable Bit** This field is used to enable the Timer1 interrupt function. 0 = Timer1 interrupt Disabled. 1 = Timer1 interrupt Enabled.
[3]	**TMR0IEN**	**Timer0 Interrupt Enable Bit** This field is used to enable Timer0 interrupt function. 0 = Timer0 interrupt Disabled. 1 = Timer0 interrupt Enabled.
[2]	**TERRIEN**	**Transfer Error Interrupt Enable Bit** This field is used to enable transfer error interrupt. The transfer error states is at SCn_STATUS register which includes receiver break error BEF (SCn_STATUS[6]), frame error FEF (SCn_STATUS[5]), parity error PEF (SCn_STATUS[4]), receive buffer overflow error RXOV (SCn_STATUS[0]), transmit buffer overflow error TXOV (SCn_STATUS[8]), receiver retry over limit error RXOVERR (SCn_STATUS[22]) and transmitter retry over limit error TXOVERR (SCn_STATUS[30]). 0 = Transfer error interrupt Disabled. 1 = Transfer error interrupt Enabled.
[1]	**TBEIEN**	**Transmit Buffer Empty Interrupt Enable Bit** This field is used to enable transmit buffer empty interrupt. 0 = Transmit buffer empty interrupt Disabled. 1 = Transmit buffer empty interrupt Enabled.
[0]	**RDAIEN**	**Receive Data Reach Interrupt Enable Bit** This field is used to enable received data reaching trigger level RXTRGLV (SCn_CTL[7:6]) interrupt. 0 = Receive data reach trigger level interrupt Disabled. 1 = Receive data reach trigger level interrupt Enabled.

SC Interrupt Status Register (SC_INTSTS)

Register	Offset	R/W	Description	Reset Value
SC_INTSTS	SCn_BA+0x1C	R/W	SC Interrupt Status Register	0x0000_0002

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					ACERRIF	RXTOIF	INITIF
7	6	5	4	3	2	1	0
CDIF	BGTIF	TMR2IF	TMR1IF	TMR0IF	TERRIF	TBEIF	RDAIF

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10]	**ACERRIF**	**Auto Convention Error Interrupt Status Flag** This field indicates auto convention sequence error. 0 = Received TS at ATR state is 0x3B or 0x3F. 1 = Received TS at ATR state is neither 0x3B nor 0x3F. **Note:** This bit can be cleared by writing 1 to it.
[9]	**RXTOIF**	**Receive Buffer Time-out Interrupt Status Flag (Read Only)** This field is used for indicate receive buffer time-out interrupt status flag. 0 = Receive buffer time-out interrupt did not occur. 1 = Receive buffer time-out interrupt occurred. **Note:** This bit is read only, user must read all receive buffer remaining data by reading SCn_DAT register to clear it.
[8]	**INITIF**	**Initial End Interrupt Status Flag** This field is used for activation (ACTEN (SCn_ALTCTL[3])), deactivation (DACTEN (SCn_ALTCTL[2])) and warm reset (WARSTEN (SCn_ALTCTL[4])) sequence interrupt status flag. 0 = Initial sequence is not complete. 1 = Initial sequence is completed. **Note:** This bit can be cleared by writing 1 to it.
[7]	**CDIF**	**Card Detect Interrupt Status Flag (Read Only)** This field is used for card detect interrupt status flag. The card detect status is CINSERT (SCn_STATUS[12]) and CREMOVE (SCn_STATUS[11]). 0 = Card detect event did not occur. 1 = Card detect event occurred. **Note:** This bit is read only, user must to clear CINSERT or CREMOVE status to clear it.
[6]	**BGTIF**	**Block Guard Time Interrupt Status Flag** This field is used for indicate block guard time interrupt status flag in receive direction.

		0 = Block guard time interrupt did not occur. 1 = Block guard time interrupt occurred. **Note 1:** This bit is valid only when RXBGTEN (SCn_ALTCTL[12]) is enabled. **Note 2:** This bit can be cleared by writing 1 to it.
[5]	**TMR2IF**	**Timer2 Interrupt Status Flag** This field is used for Timer2 interrupt status flag. 0 = Timer2 interrupt did not occur. 1 = Timer2 interrupt occurred. **Note:** This bit can be cleared by writing 1 to it.
[4]	**TMR1IF**	**Timer1 Interrupt Status Flag** This field is used for Timer1 interrupt status flag. 0 = Timer1 interrupt did not occur. 1 = Timer1 interrupt occurred. **Note:** This bit can be cleared by writing 1 to it.
[3]	**TMR0IF**	**Timer0 Interrupt Status Flag** This field is used for Timer0 interrupt status flag. 0 = Timer0 interrupt did not occur. 1 = Timer0 interrupt occurred. **Note:** This bit can be cleared by writing 1 to it.
[2]	**TERRIF**	**Transfer Error Interrupt Status Flag** This field is used for transfer error interrupt status flag. The transfer error states is at SCn_STATUS register which includes receiver break error BEF (SCn_STATUS[6]), frame error FEF (SCn_STATUS[5], parity error PEF (SCn_STATUS[4] and receive buffer overflow error RXOV (SCn_STATUS[0]), transmit buffer overflow error TXOV (SCn_STATUS[8]), receiver retry over limit error RXOVERR (SCn_STATUS[22] or transmitter retry over limit error TXOVERR (SCn_STATUS[30]). 0 = Transfer error interrupt did not occur. 1 = Transfer error interrupt occurred. **Note 1:** This field is the status flag of BEF, FEF, PEF, RXOV, TXOV, RXOVERR or TXOVERR. **Note 2:** This bit can be cleared by writing 1 to it.
[1]	**TBEIF**	**Transmit Buffer Empty Interrupt Status Flag (Read Only)** This field is used for transmit buffer empty interrupt status flag. 0 = Transmit buffer is not empty. 1 = Transmit buffer is empty. **Note:** This bit is read only. If user wants to clear this bit, user must write data to DAT (SCn_DAT[7:0]) and then this bit will be cleared automatically.
[0]	**RDAIF**	**Receive Data Reach Interrupt Status Flag (Read Only)** This field is used for received data reaching trigger level RXTRGLV (SCn_CTL[7:6]) interrupt status flag. 0 = Number of receive buffer is less than RXTRGLV setting. 1 = Number of receive buffer data equals the RXTRGLV setting. **Note:** This bit is read only. If user reads data from SCn_DAT and receiver buffer data byte number is less than RXTRGLV, this bit will be cleared automatically.

SC Transfer Status Register (SC_STATUS)

Register	Offset	R/W	Description	Reset Value
SC_STATUS	SCn_BA+0x20	R/W	SC Transfer Status Register	0x0000_X202

31	30	29	28	27	26	25	24
TXACT	TXOVERR	TXRERR	Reserved		TXPOINT		
23	22	21	20	19	18	17	16
RXACT	RXOVERR	RXRERR	Reserved		RXPOINT		
15	14	13	12	11	10	9	8
Reserved		CDPINSTS	CINSERT	CREMOVE	TXFULL	TXEMPTY	TXOV
7	6	5	4	3	2	1	0
Reserved	BEF	FEF	PEF	Reserved	RXFULL	RXEMPTY	RXOV

Bits	Description	
[31]	**TXACT**	**Transmit in Active Status Flag (Read Only)** This bit indicates Tx transmit status. 0 = This bit is cleared automatically when Tx transfer is finished or the last byte transmission has completed. 1 = Transmit is active and this bit is set by hardware when Tx transfer is in active and the STOP bit of the last byte has not been transmitted.
[30]	**TXOVERR**	**Transmitter over Retry Error** This bit is used for transmitter retry counts over than retry number limitation. 0 = Transmitter retries counts is less than TXRTY (SCn_CTL[22:20]) + 1. 1 = Transmitter retries counts is equal or over to TXRTY (SCn_CTL[22:20]) + 1. **Note:** This bit can be cleared by writing 1 to it.
[29]	**TXRERR**	**Transmitter Retry Error** This bit is used for indicate transmitter error retry and set by hardware. 0 = No Tx retry transfer. 1 = Tx has any error and retries transfer. **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** This bit is a flag and cannot generate any interrupt to CPU.
[28:27]	**Reserved**	Reserved.
[26:24]	**TXPOINT**	**Transmit Buffer Pointer Status (Read Only)** This field indicates the Tx buffer pointer status. When CPU writes data into SCn_DAT, TXPOINT increases one. When one byte of Tx buffer is transferred to transmitter shift register, TXPOINT decreases one.
[23]	**RXACT**	**Receiver in Active Status Flag (Read Only)** This bit indicates Rx transfer status. 0 = This bit is cleared automatically when Rx transfer is finished. 1 = This bit is set by hardware when Rx transfer is in active.
[22]	**RXOVERR**	**Receiver over Retry Error**

		This bit is used for receiver retry counts over than retry number limitation. 0 = Receiver retries counts is less than RXRTY (SCn_CTL[18:16]) + 1. 1 = Receiver retries counts is equal or over than RXRTY (SCn_CTL[18:16]) + 1. **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** If CPU enables receiver retries function by setting RXRTYEN (SCn_CTL[19]), hardware will not set this flag.
[21]	**RXRERR**	**Receiver Retry Error** This bit is used for receiver error retry and set by hardware. 0 = No Rx retry transfer. 1 = Rx has any error and retries transfer. **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** This bit is a flag and cannot generate any interrupt to CPU. **Note 3:** If CPU enables receiver retries function by setting RXRTYEN (SCn_CTL[19]), hardware will not set this flag.
[20:19]	**Reserved**	Reserved.
[18:16]	**RXPOINT**	**Receive Buffer Pointer Status (Read Only)** This field indicates the Rx buffer pointer status. When SC controller receives one byte from external device, RXPOINT increases one. When one byte of Rx buffer is read by CPU, RXPOINT decreases one.
[15:14]	**Reserved**	Reserved.
[13]	**CDPINSTS**	**Card Detect Pin Status (Read Only)** This bit is the pin status of SCn_CD. 0 = The SCn_CD pin state at low. 1 = The SCn_CD pin state at high.
[12]	**CINSERT**	**Card Insert Status of SCn_CD Pin** This bit is set whenever card has been inserted. 0 = No effect. 1 = Card insert. **Note 1:** This bit can be cleared by writing “1” to it. **Note 2:** The card detect function will start after SCEN (SCn_CTL[0]) set.
[11]	**CREMOVE**	**Card Removal Status of SCn_CD Pin** This bit is set whenever card has been removal. 0 = No effect. 1 = Card removed. **Note 1:** This bit can be cleared by writing “1” to it. **Note 2:** Card detect function will start after SCEN (SCn_CTL[0]) set.
[10]	**TXFULL**	**Transmit Buffer Full Status Flag (Read Only)** This bit indicates Tx buffer full or not. 0 = Tx buffer count is less than 4. 1 = Tx buffer count equals to 4.
[9]	**TXEMPTY**	**Transmit Buffer Empty Status Flag (Read Only)** This bit indicates TX buffer empty or not. 0 = Tx buffer is not empty. 1 = Tx buffer is empty, it means the last byte of Tx buffer has been transferred to Transmitter Shift Register. **Note:** This bit will be cleared when writing data into DAT (SCn_DAT[7:0]).
[8]	**TXOV**	**Transmit Overflow Error Interrupt Status Flag**

		This bit is set when Tx buffer overflow. 0 = Tx buffer is not overflow. 1 = Tx buffer is overflow when Tx buffer is full and an additional write operation to DAT (SCn_DAT[7:0]). **Note:** This bit can be cleared by writing 1 to it.
[7]	**Reserved**	Reserved.
[6]	**BEF**	**Receiver Break Error Status Flag** This bit is set to logic 1 whenever the received data input (Rx) held in the "spacing state" (logic 0) is longer than a full word transmission time (that is, the total time of “start bit” + “data bits” + “parity bit” + “stop bits”). 0 = Receiver break error flag did not occur. 1 = Receiver break error flag occurred. **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** If CPU sets receiver retries function by setting RXRTYEN (SCn_CTL[19]), hardware will not set this flag.
[5]	**FEF**	**Receiver Frame Error Status Flag** This bit is set to logic 1 whenever the received character does not have a valid "stop bit" (that is, the stop bit following the last data bit or parity bit is detected as logic 0). 0 = Receiver frame error flag did not occur. 1 = Receiver frame error flag occurred. **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** If CPU sets receiver retries function by setting RXRTYEN (SCn_CTL[19]), hardware will not set this flag.
[4]	**PEF**	**Receiver Parity Error Status Flag** This bit is set to logic 1 whenever the received character does not have a valid "parity bit". 0 = Receiver parity error flag did not occur. 1 = Receiver parity error flag occurred. **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** If CPU sets receiver retries function by setting RXRTYEN (SCn_CTL[19]), hardware will not set this flag.
[3]	**Reserved**	Reserved.
[2]	**RXFULL**	**Receive Buffer Full Status Flag (Read Only)** This bit indicates Rx buffer full or not. 0 = Rx buffer count is less than 4. 1 = Rx buffer count equals to 4.
[1]	**RXEMPTY**	**Receive Buffer Empty Status Flag (Read Only)** This bit indicates Rx buffer empty or not. 0 = Rx buffer is not empty. 1 = Rx buffer is empty, it means the last byte of Rx buffer has read from DAT (SCn_DAT[7:0]) by CPU.
[0]	**RXOV**	**Receive Overflow Error Status Flag** This bit is set when Rx buffer overflow. 0 = Rx buffer is not overflow. 1 = Rx buffer is overflow when the number of received bytes is greater than Rx buffer size (4 bytes). **Note:** This bit can be cleared by writing 1 to it.

SC Pin Control State Register (SC_PINCTL)

Register	Offset	R/W	Description	Reset Value
SC_PINCTL	SCn_BA+0x24	R/W	SC Pin Control State Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	SYNC	Reserved					
23	22	21	20	19	18	17	16
Reserved					RSTSTS	PWRSTS	DATASTS
15	14	13	12	11	10	9	8
Reserved				PWRINV	Reserved	SCDATA	Reserved
7	6	5	4	3	2	1	0
Reserved	CLKKEEP	Reserved				RSTEN	PWREN

Bits	Description	
[31]	**Reserved**	Reserved.
[30]	**SYNC**	**SYNC Flag Indicator (Read Only)** Due to synchronization, user should check this bit when writing a new value to SCn_PINCTL register. 0 = Synchronizing is completion, user can write new data to SCn_PINCTL register. 1 = Last value is synchronizing.
[29:19]	**Reserved**	Reserved.
[18]	**RSTSTS**	**SCn_RST Pin Status (Read Only)** This bit is the pin status of SCn_RST. 0 = SCn_RST pin status is low. 1 = SCn_RST pin status is high.
[17]	**PWRSTS**	**SCn_PWR Pin Status (Read Only)** This bit is the pin status of SCn_PWR. 0 = SCn_PWR pin status is low. 1 = SCn_PWR pin status is high.
[16]	**DATASTS**	**SCn_DATA Pin Status (Read Only)** This bit is the pin status of SCn_DATA. 0 = The SCn_DATA pin status is low. 1 = The SCn_DATA pin status is high.
[15:12]	**Reserved**	Reserved.
[11]	**PWRINV**	**SCn_PWR Pin Inverse** This bit is used for inverse the SCn_PWR pin. There are four kinds of combination for SCn_PWR pin setting by PWRINV (SCn_PINCTL[11]) and PWREN (SCn_PINCTL[0]). PWRINV (SCn_PINCTL[11]) is bit 1 and PWREN (SCn_PINCTL[0]) is bit 0 and all conditions as below list,

		00 = SCn_PWR pin is 0. 01 = SCn_PWR pin is 1. 10 = SCn_PWR pin is 1. 11 = SCn_PWR pin is 0. **Note:** User must select PWRINV (SCn_PINCTL[11]) before smart card is enabled by SCEN (SCn_CTL[0]).
[10]	**Reserved**	Reserved.
[9]	**SCDATA**	**SCn_DATA Pin Signal** This bit is the signal status of SCn_DATA but user can drive SCn_DATA pin to high or low by setting this bit. Write this field to drive SCn_ DATA pin. 0 = Drive SCn_DATA pin to low. 1 = Drive SCn_DATA pin to high. Read this field to get SCn_DATA signal status. 0 = SCn_DATA signal status is low. 1 = SCn_DATA signal status is high. **Note:** When SC is at activation, warm reset or deactivation mode, this bit will be changed automatically. Thus, do not fill in this field when SC is in these modes.
[8:7]	**Reserved**	Reserved.
[6]	**CLKKEEP**	**SC Clock Enable Bit** 0 = SC clock generation Disabled. 1 = SC clock always keeps free running. **Note:** When operating in activation, warm reset or deactivation mode, this bit will be changed automatically. Thus, do not fill in this field when operating in these modes.
[5:2]	**Reserved**	Reserved.
[1]	**RSTEN**	**SCn_RST Pin Signal** User can set RSTEN (SCn_PINCTL[1]) to decide SCn_RST pin is in high or low level. Write this field to drive SCn_RST pin. 0 = Drive SCn_RST pin to low. 1 = Drive SCn_RST pin to high. Read this field to get SCn_RST signal status. 0 = SCn_RST signal status is low. 1 = SCn_RST signal status is high. **Note:** When operating at activation, warm reset or deactivation mode, this bit will be changed automatically. Thus, do not fill in this field when operating in these modes.
[0]	**PWREN**	**SCn_PWR Pin Signal** User can set PWRINV (SCn_PINCTL[11]) and PWREN (SCn_PINCTL[0]) to decide SCn_PWR pin is in high or low level. Write this field to drive SCn_PWR pin Refer PWRINV (SCn_PINCTL[11]) description for programming SCn_PWR pin voltage level. Read this field to get SCn_PWR signal status. 0 = SCn_PWR signal status is low. 1 = SCn_PWR signal status is high. **Note:** When operating at activation, warm reset or deactivation mode, this bit will be changed automatically. Thus, do not fill in this field when operating in these modes.

SC Timer0 Control Register (SC_TMRCTL0)

Register	Offset	R/W	Description	Reset Value
SC_TMRCTL0	SCn_BA+0x28	R/W	SC Internal Timer0 Control Register	0x0000_0000

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

Bits	Description	
[31]	**SYNC**	**SYNC Flag Indicator (Read Only)** Due to synchronization, user should check this bit when writing a new value to the SCn_TMRCTL0 register. 0 = Synchronizing is completion, user can write new data to SCn_TMRCTL0 register. 1 = Last value is synchronizing.
[30:28]	**Reserved**	Reserved.
[27:24]	**OPMODE**	**Timer0 Operation Mode Selection** This field indicates the internal 24-bit Timer0 operation selection. Refer to Table 6.20-3 for programming Timer0.
[23:0]	**CNT**	**Timer0 Counter Value** This field indicates the internal Timer0 counter values. **Note:** Unit of Timer0 counter is ETU base.

SC Timer1 Control Register (SC_TMRCTL1)

Register	Offset	R/W	Description	Reset Value
SC_TMRCTL1	SCn_BA+0x2C	R/W	SC Internal Timer1 Control Register	0x0000_0000

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

Bits	Description	
[31]	**SYNC**	**SYNC Flag Indicator (Read Only)** Due to synchronization, software should check this bit when writing a new value to SCn_TMRCTL1 register. 0 = Synchronizing is completion, user can write new data to SCn_TMRCTL1 register. 1 = Last value is synchronizing.
[30:28]	**Reserved**	Reserved.
[27:24]	**OPMODE**	**Timer 1 Operation Mode Selection** This field indicates the internal 8-bit Timer1 operation selection. Refer to Table 6.20-3 for programming Timer1.
[23:8]	**Reserved**	Reserved.
[7:0]	**CNT**	**Timer 1 Counter Value** This field indicates the internal Timer1 counter values. **Note:** Unit of Timer1 counter is ETU base.

SC Timer2 Control Register (SC_TMRCTL2)

Register	Offset	R/W	Description	Reset Value
SC_TMRCTL2	SCn_BA+0x30	R/W	SC Internal Timer2 Control Register	0x0000_0000

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

Bits	Description	
[31]	**SYNC**	**SYNC Flag Indicator (Read Only)** Due to synchronization, user should check this bit when writing a new value to SCn_TMRCTL2 register. 0 = Synchronizing is completion, user can write new data to SCn_TMRCTL2 register. 1 = Last value is synchronizing.
[30:28]	**Reserved**	Reserved.
[27:24]	**OPMODE**	**Timer 2 Operation Mode Selection** This field indicates the internal 8-bit Timer2 operation selection Refer to Table 6.20-3 for programming Timer2.
[23:8]	**Reserved**	Reserved.
[7:0]	**CNT**	**Timer 2 Counter Value** This field indicates the internal Timer2 counter values. **Note:** Unit of Timer2 counter is ETU base.

SC UART Mode Control Register (SC_UARTCTL)

Register	Offset	R/W	Description	Reset Value
SC_UARTCTL	SCn_BA+0x34	R/W	SC UART Mode Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
OPE	PBOFF	WLS		Reserved			UARTEN

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**OPE**	**Odd Parity Enable Bit** This is used for odd/even parity selection. 0 = Even number of logic 1 are transmitted or check the data word and parity bits in receiving mode. 1 = Odd number of logic 1 are transmitted or check the data word and parity bits in receiving mode. **Note:** This bit has effect only when PBOFF bit is 0.
[6]	**PBOFF**	**Parity Bit Disable Bit** Sets this bit is used for disable parity check function. 0 = Parity bit is generated or checked between the “last data word bit” and “stop bit” of the serial data. 1 = Parity bit is not generated (transmitting data) or checked (receiving data) during transfer. **Note:** In Smart Card mode, this field must be 0 (default setting is with parity bit).
[5:4]	**WLS**	**Word Length Selection** This field is used for select UART data length. 00 = Word length is 8 bits. 01 = Word length is 7 bits. 10 = Word length is 6 bits. 11 = Word length is 5 bits. **Note:** In Smart Card mode, this WLS must be 00.
[3:1]	**Reserved**	Reserved.
[0]	**UARTEN**	**UART Mode Enable Bit** Sets this bit to enable UART mode function. 0 = Smart Card mode. 1 = UART mode. **Note 1:** When operating in UART mode, user must set CONSEL (SCn_CTL[5:4]) = 00 and AUTOCEN (SCn_CTL[3]) = 0. **Note 2:** When operating in Smart Card mode, user must set UARTEN (SCn_UARTCTL[0]) = 0.

Bits	Description	
		Note 3: When UART mode is enabled, hardware will generate a reset to reset FIFO and internal state machine.

SC Activation Control Register (SC_ACTCTL)

Register	Offset	R/W	Description	Reset Value
SC_ACTCTL	SCn_BA+0x4C	R/W	SC Activation Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			T1EXT				

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4:0]	**T1EXT**	**T1 Extend Time of Hardware Activation** This field provide the configurable cycles to extend the activation time T1 period. The cycle scaling factor is 2048. Extend cycles = (filled value * 2048) cycles. Refer to SC activation sequence in Figure 6.20-4. For example, SCLK = 4 MHz, each cycle = 0.25 us. Filled 20 to this field Extend time = 20 * 2048 * 0.25us = 10.24 ms. **Note:** Setting 0 to this field conforms to the protocol ISO/IEC 7816-3.

6.21 I²S Controller (I²S)

6.21.1 Overview

The I²S controller consists of I²S protocol to interface with external audio CODEC. Two 16-level depth FIFO for reading path and writing path respectively are capable of handling 8/16/24/32 bits audio data sizes. A PDMA controller handles the data movement between FIFO and memory.

6.21.2 Features

- Supports Master mode and Slave mode
- Capable of handling 8, 16, 24 and 32 bits data sizes in each audio channel
- Supports monaural and stereo audio data
- Supports I²S protocols: Philips standard, MSB-justified, and LSB-justified data format
- Supports PCM protocols: PCM standard, MSB-justified, and LSB-justified data format
- PCM protocol supports TDM multi-channel transmission in one audio sample, and the number of data channel can be set as 2, 4, 6, or 8
- Provides two 16-level FIFO data buffers, one for transmitting and the other for receiving
- Generates interrupt requests when buffer levels cross a programmable boundary
- Supports two PDMA requests, one for transmitting and the other for receiving

6.21.3 Block Diagram

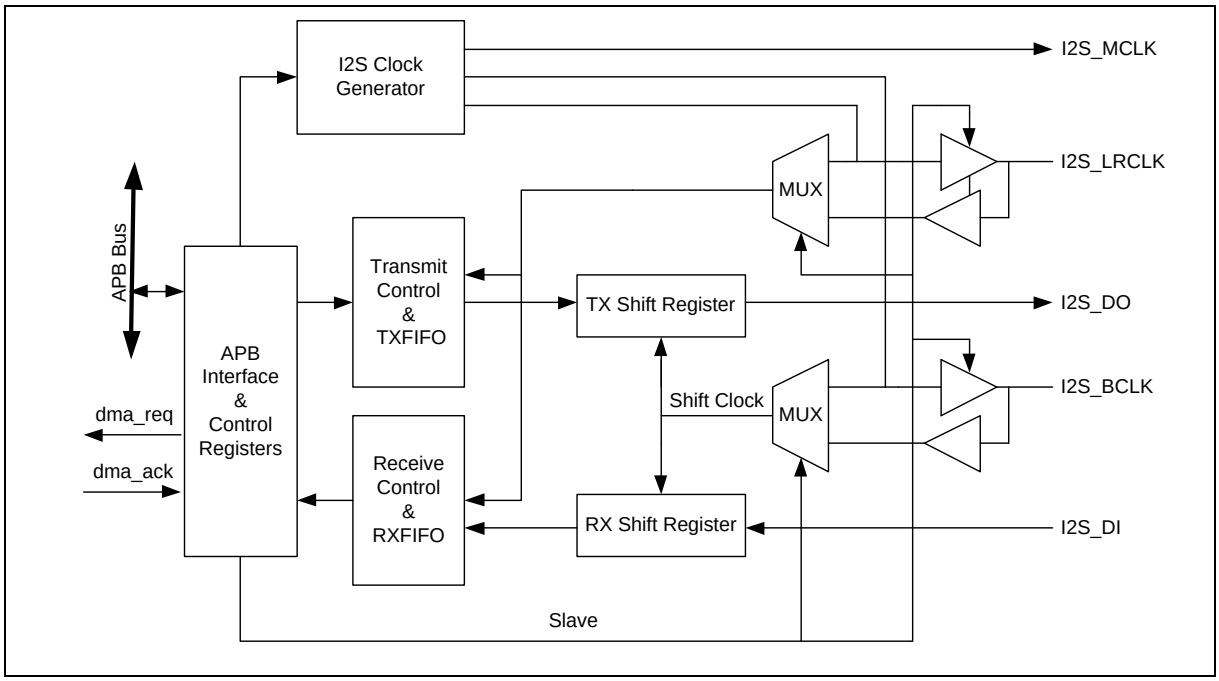


Figure 6.21-1 I²S Controller Block Diagram

6.21.4 Basic Configuration

6.21.4.1 I²S Basic Configuration

- Clock source Configuration

- Select the source of I²S peripheral clock on I2S0SEL (CLK_CLKSEL3[17:16]).
- Enable I²S peripheral clock in I2S0CKEN (CLK_APBCLK0[29]).

- Reset Configuration
  - Reset I²S controller in I2S0RST (SYS_IPRST1[29]).
- Pin configuration

Group	Pin Name	GPIO	MFP
I2S0	I2S0_BCLK	PA.12	MFP2
		PE.8, PF.10	MFP4
		PE.1	MFP5
		PC.4	MFP6
		PB.5	MFP10
	I2S0_DI	PA.14	MFP2
		PE.10, PF.8	MFP4
		PH.8	MFP5
		PC.2	MFP6
		PB.3	MFP10
	I2S0_DO	PA.15	MFP2
		PE.11, PF.7	MFP4
		PH.9	MFP5
		PC.1	MFP6
		PB.2	MFP10
	I2S0_LRCLK	PE.12, PF.6	MFP4
		PH.10	MFP5
		PC.0	MFP6
		PB.1	MFP10
	I2S0_MCLK	PA.13	MFP2
		PE.9, PF.9	MFP4
		PE.0	MFP5
		PC.3	MFP6
		PB.4	MFP10

Table 6.21-1 Pin Configuration of I²S Controller

6.21.5 Functional Description

6.21.5.1 I²S Clock

The I²S controller has four clock sources selected by I2S0SEL (CLK_CLKSEL3[17:16]). The I²S clock rate must be slower than or equal to system clock rate.

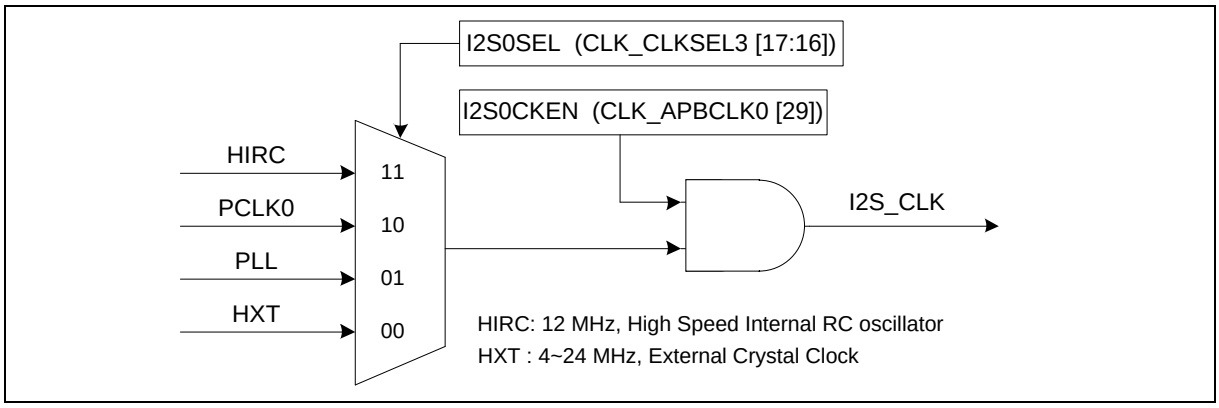


Figure 6.21-2 I^{2}S Clock Control Diagram

6.21.5.2 Master/Slave Interface

The I^{2}S function can operate as master or slave mode by setting SLAVE (I2S_CTL0[8]) to communicate with other I^{2}S slave or master. The serial bus clock I2S_BCLK is permanently generated by the master even though there is no transferring data bit at the moment. The word select signal I2S_LRCLK is also generated by the master and it indicates the beginning of a new data word and the targeted audio channel. Both the I2S_LRCLK and the transmitting data change synchronously to the falling edges of I2S_BCLK.

In some applications, especially for Audio-ADC or Audio-DAC, a master clock signal, I2S_MCLK, is required with a fixed phase relation to the I2S_BCLK. The I2S_MCLK is enabled by MCLKEN (I2S_CTL0[15]). In Master mode, the I2S_MCLK, I2S_BCLK, I2S_LRCLK is output to device slave. And if in slave mode, the I2S_MCLK is output to master, and I2S_BCLK or I2S_LRCLK is input from master.

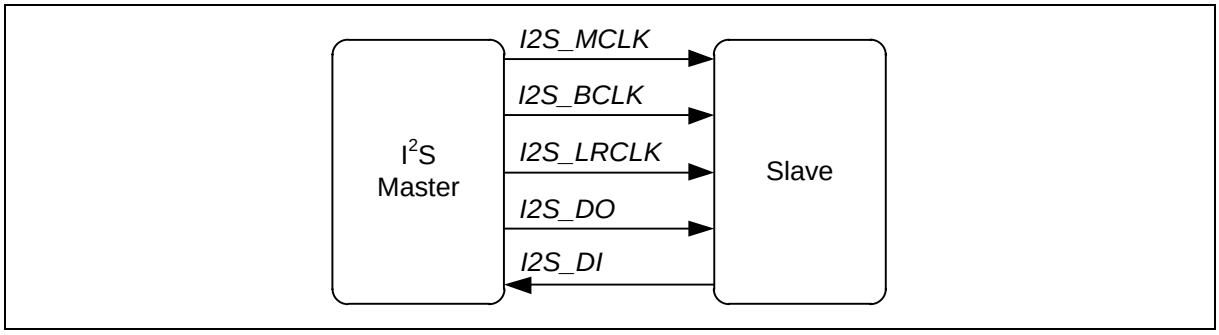


Figure 6.21-3 Master Mode Interface Block Diagram

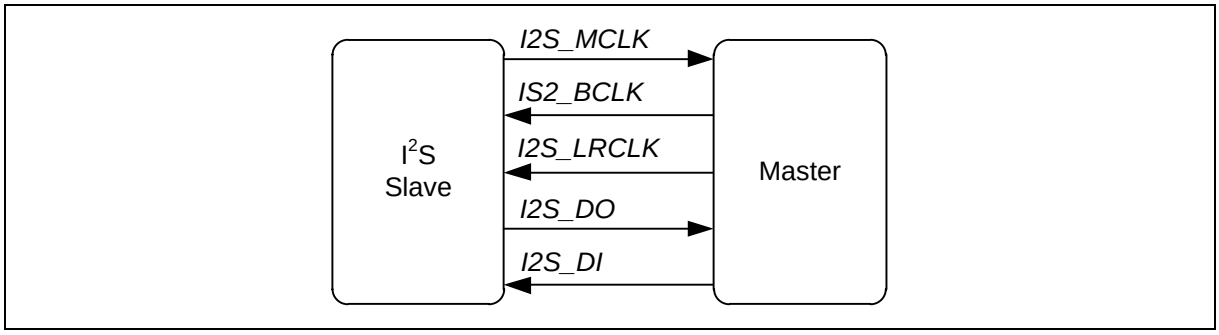


Figure 6.21-4 Slave Mode Interface Block Diagram

6.21.5.3 I2S Operation

The I2S controller supports MSB-justified, LSB-justified, and I2S Philips standard data format. The I2S_LRCLK signal indicates which audio channel is in transferring. The bit count of an audio channel is defined by CHWIDTH (I2S_CTL0[29:28]), and the bit-width of data word in an audio channel is determined by DATWIDTH (I2S_CTL0[5:4]). If CHWIDTH (I2S_CTL0[29:28]) is less than DATWIDTH (I2S_CTL0[5:4]), the hardware will set the channel bit-width to be same as data bit-width. However, there will be redundant zero bits in each audio channel if CHWIDTH (I2S_CTL0[29:28]) is greater than DATWIDTH (I2S_CTL0[5:4]).

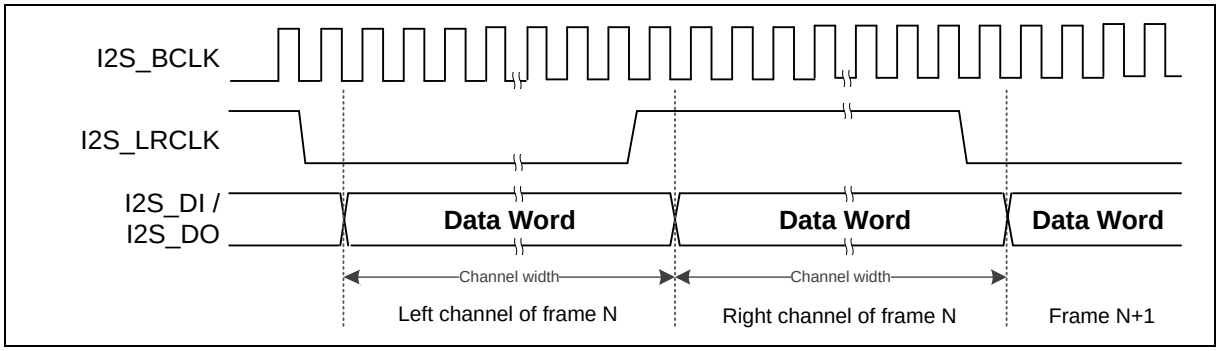


Figure 6.21-5 I2S Channel Width and Data Width (CHWIDTH≦DATWIDTH)

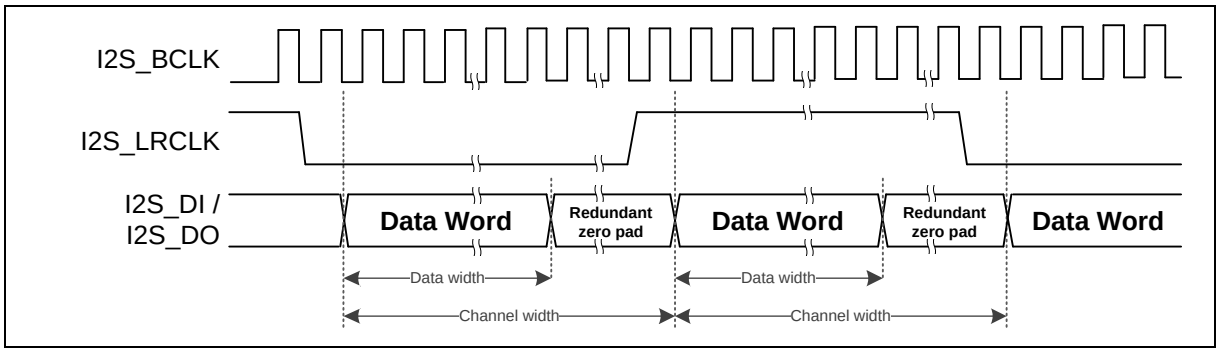


Figure 6.21-6 I2S Channel Width and Data Width (CHWIDTH > DATWIDTH)

The transferring data sequence is always started from the MSB (most significant bit) to the LSB (least significant bit). As shown in Figure 6.21-7, transmitting data are read at rising edge of I2S_BCLK and sent out at falling edge of I2S_BCLK in I2S protocol. In I2S data format, the MSB is sent and latched at the next falling edge of I2S_BCLK cycle after the transition of I2S_LRCLK. In MSB justified data format, the I2S_LRCLK changes the polarity at the transmitting of the first data bit (MSB) in each audio

channel. In LSB justified data format, the LSB is sent and latched at the last I2S_BCLK cycle of an audio channel. The MSB justified and LSB justified data format of I²S protocol can be selected by FORMAT (I2S_CTL0[26:24]).

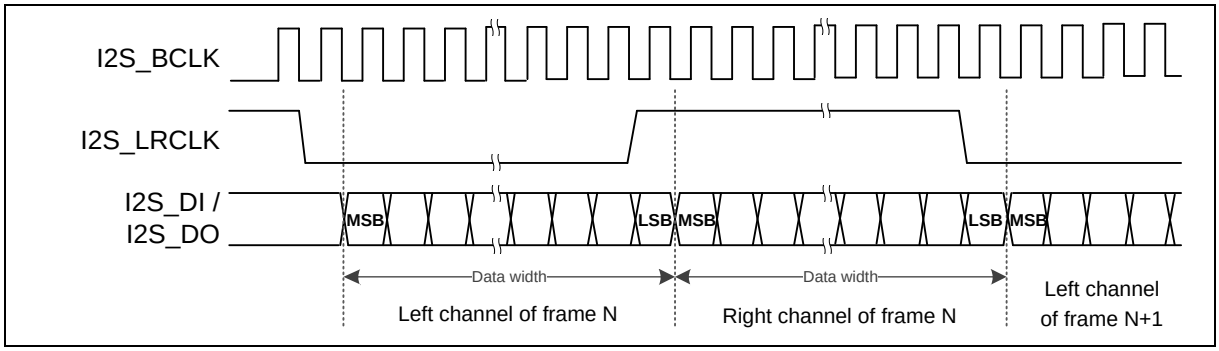


Figure 6.21-7 I²S Data Format Timing Diagram (FORMAT = 0x0 ; CHWIDTH≦DATWIDTH)

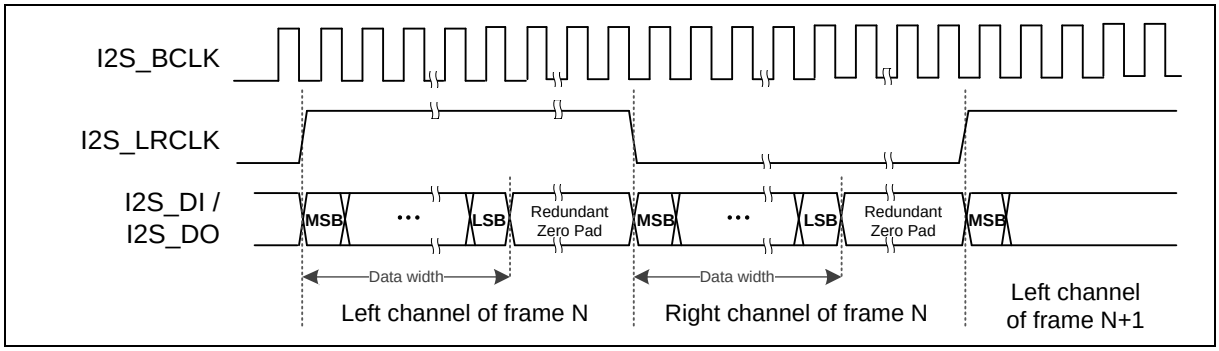


Figure 6.21-8 I²S with MSB Justified Data Format (FORMAT = 0x1 ; CHWIDTH > DATWIDTH)

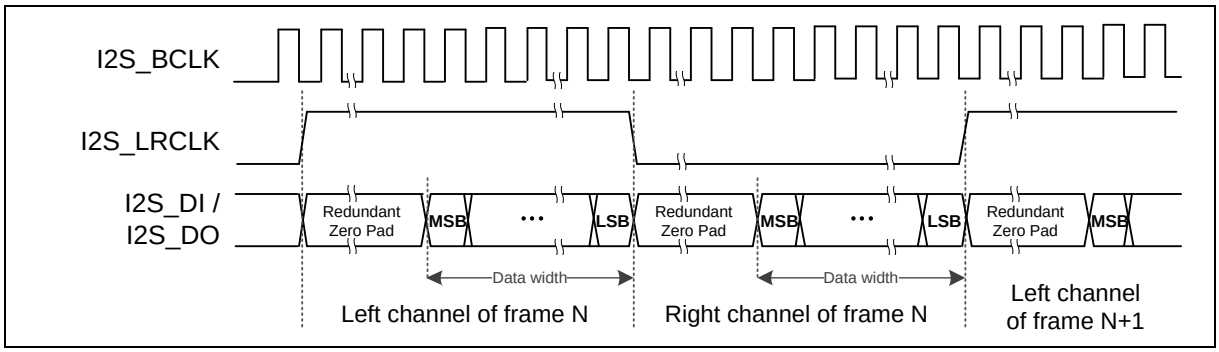


Figure 6.21-9 I²S with LSB Justified Data Format (FORMAT = 0x2 ; CHWIDTH > DATWIDTH)

The I²S controller also supports PCM audio transmission which can be selected by FORMAT (I2S_CTL0[26:24]). In PCM protocol, the function of I2S_LRCLK is simply to identify the beginning of an audio sample (or audio frame) and it is always indicated by the rising edge of the pulse. Therefore, the I2S_LRCLK in PCM protocol may be also called “frame start” or “frame sync” signal. In master device, there are two common representations for the width of the frame start pulse which can use PCMSYNC (I2S_CTL0[27]) to choose: One is equivalent to the period of a channel width and the other is equivalent to a single period of the I2S_BCLK.

Same as I²S protocol, the DATWIDTH (I2S_CTL0[5:4]) and CHWIDTH (I2S_CTL0[29:28]) can be used to configure the data bit-width and channel bit-width in PCM protocol. Besides, FORMAT (I2S_CTL0[26:24]) can also be used to select the different data formats of PCM standard mode, PCM

with MSB justified, and PCM with LSB justified data format.

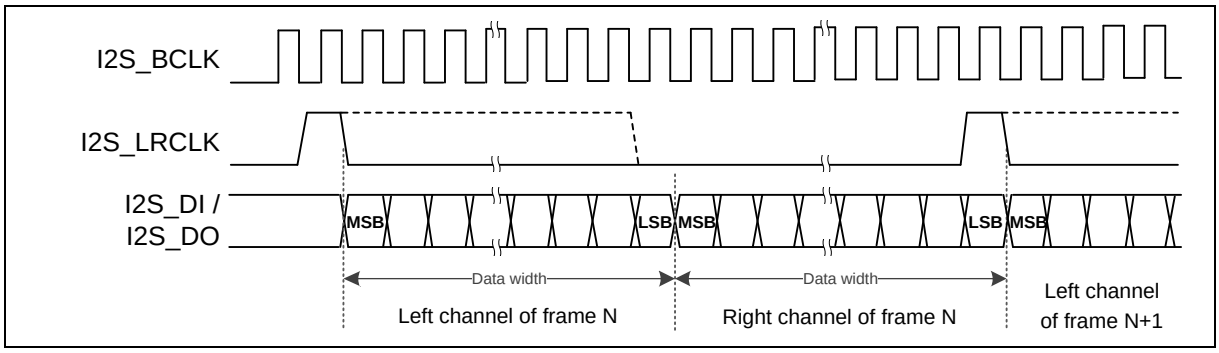


Figure 6.21-10 PCM Data Format Timing Diagram (FORMAT = 0x4 ; CHWIDTH≦DATWIDTH)

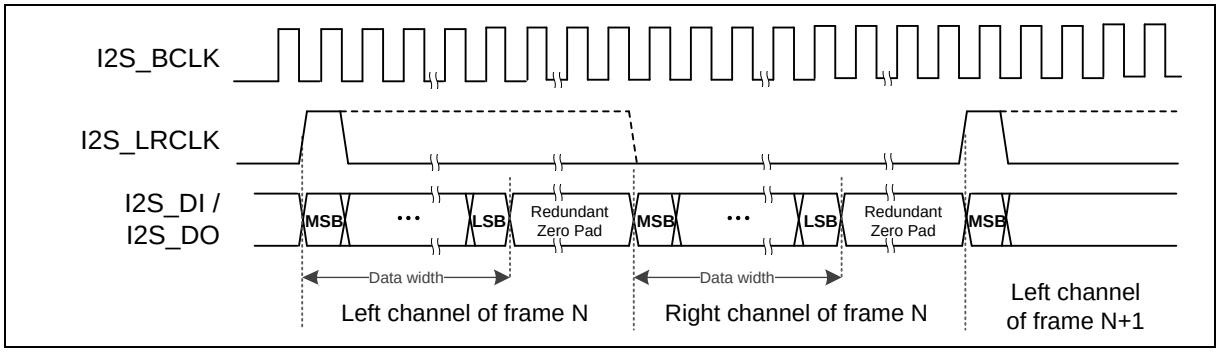


Figure 6.21-11 PCM with MSB Justified Data Format (FORMAT = 0x5 ; CHWIDTH > DATWIDTH)

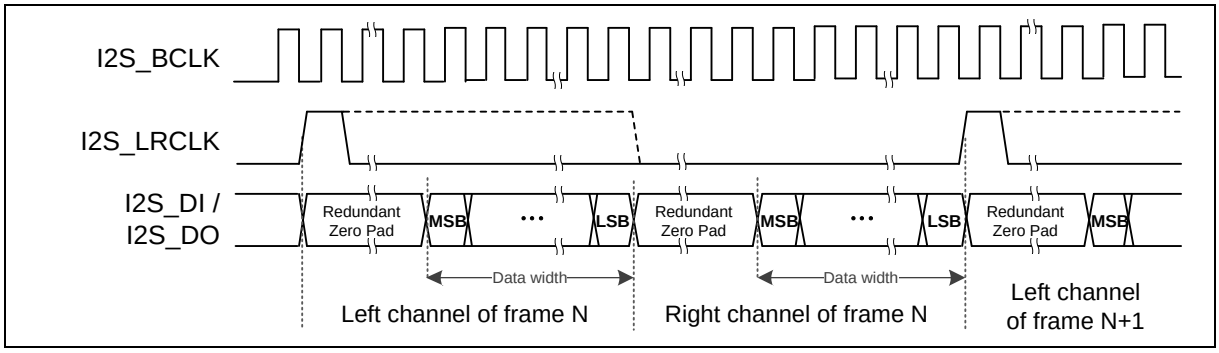


Figure 6.21-12 PCM with LSB Justified Data Format (FORMAT = 0x6 ; CHWIDTH > DATWIDTH)

6.21.5.4 TDM Multi-channel Transmission

The PCM mode in this I^{2}S controller also supports TDM transmission. The Time Division Multiplexed (TDM) method allows multiple channels of audio data to be transmitted on a single data line. The TDM interface is similar to the 2-channel PCM audio interface with the exception that more audio channels are transmitted within a sample frame which is defined by a period of the I2S_LRCLK. The channel number of TDM interface is typically 4, 6, or 8 and it is selected by TDMCHNUM (I2S_CTL0[31:30]).

Same as previous I^{2}S and PCM descriptions, each channel block is comprised of the audio data word followed by a sufficient number of zero data bits to complete one channel block. The bit-width of data word and channel block are defined by DATWIDTH (I2S_CTL0[5:4]) and CHWIDTH (I2S_CTL0[29:28]) respectively. Note that the TDM PCM mode supports 16-bit, 24-bit, 32-bit audio data word (excluding 8-bit data), and the hardware will set the bit-width of transmitting data as 16-bit if

DATWIDTH (I2S_CTL0[5:4]) is 0x0. The pulse width of frame start signal is also selected by PCMSYNC (I2S_CTL0[27]).

The examples of 6-channel TDM transmission with 24-bit audio data in 32-bit channel block are shown in Figure 6.21-13. In 2-channel audio interface, the first and second audio channels are called as left-channel and right-channel (or channel0 and channel1). In TDM multi-channel application, the first and second audio channels are called as channel0 and channel1.

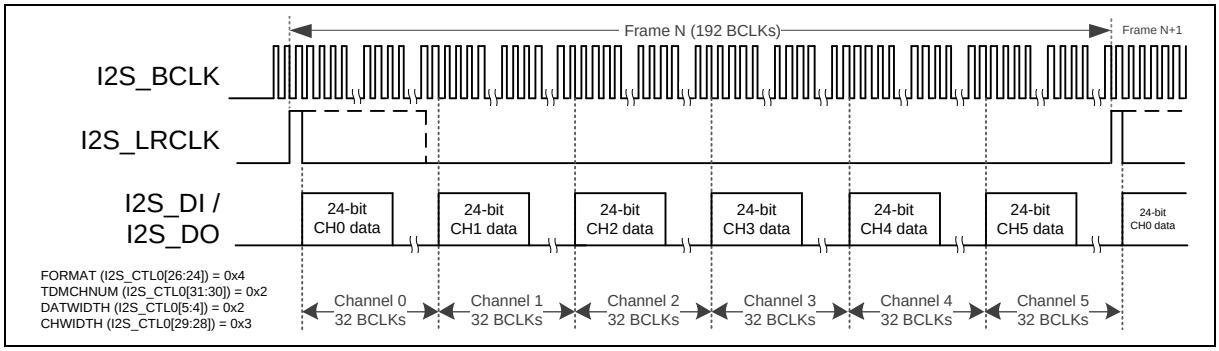


Figure 6.21-13 TDM 6-channel Audio Format with 24-bit Data in 32-bit Channel Block (PCM Standard Data Format; FORMAT=0x4)

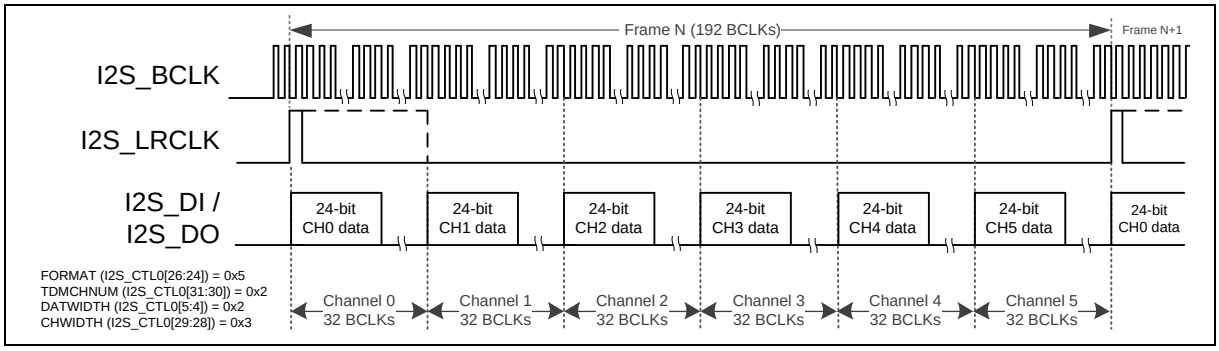


Figure 6.21-14 TDM 6-channel Audio Format with 24-bit Data in 32-bit Channel Block (PCM with MSB Justified; FORMAT=0x5)

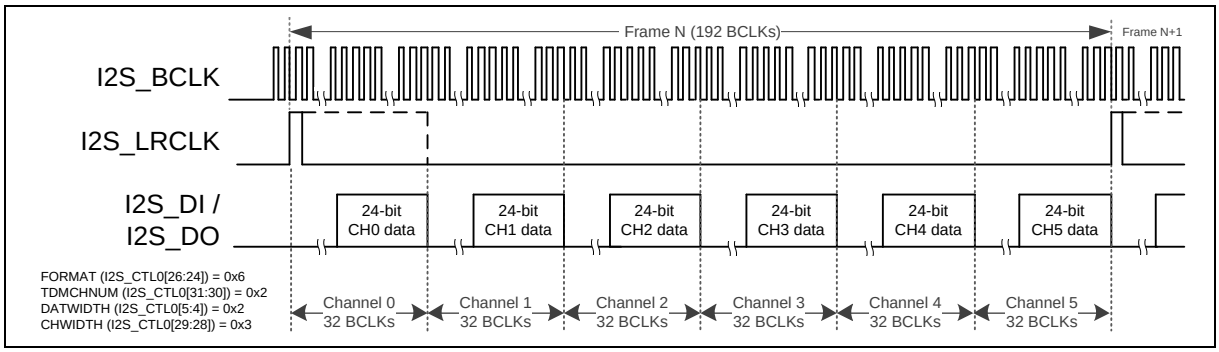


Figure 6.21-15 TDM 6-channel Audio Format with 24-bit Data in 32-bit Channel Block (PCM with LSB Justified; FORMAT=0x6)

6.21.5.5 Zero Crossing

When playing the audio by I^2S controller, the output transmitting data comes from the memory by PDMA or by CPU. However, there may be some pop noise which induces the uncomfortable hearing if

the playing sound volume is changed greatly by user. The zero-crossing event of audio data means the playing sound is relatively silent at the moment. Therefore, the zero-cross interrupt can be used for the indication of gain level adjustment in order to prevent the huge variance of sound volume.

If zero-cross detection of individual audio channel is enabled by the corresponding control bit from CH0ZCEN to CH7ZCEN (I2S_CTL1[0] to I2S_CTL1[7]), the hardware will detect the next transferring data word of the corresponding audio channel whether it is 0 or its MSB has been changed. If zero value or MSB (sign bit) changing of the transmitting audio data has been detected while zero-cross detection is enabled, the hardware will set the corresponding status bit from CH0ZCIF to CH7ZCIF (I2S_STATUS1[0] to I2S_STATUS1[7]) for the audio channel and then keep the output audio data silent (all data bit zero) automatically until the corresponding event status bit is cleared by software.

Therefore, if user wants to modify the audio playing gain, users can enable the zero crossing interrupt function, CH0ZCIEN to CH7ZCIEN (I2S_IEN[16] to I2S_IEN[23]), to indicate the zero crossing time and to change the audio gain. This will reduce the pop noise.

6.21.5.6 PDMA Mode

The I^2S function can use PDMA function for transmitting or receiving data access. If the PDMA function of transmitting data is enabled by TXPDMAEN (I2S_CTL0[20]), the I^2S controller will generate the request signal and then get transmitting audio data from memory by PDMA IP automatically while TX FIFO is not full. If the PDMA function of receiving data is enabled by RXPDMAEN (I2S_CTL0[21]), the I^2S controller will generate the request signal and then the receiving data will be moved into memory by PDMA hardware automatically while the RX FIFO is not empty. Therefore, using PDMA function will save the CPU loading to service other functions.

6.21.5.7 I^2S Interrupt Sources

The I^2S controller supports zero-cross interrupt of individual audio channel, transmit FIFO threshold level interrupt, transmit FIFO overflow interrupt and transmit FIFO underflow interrupt in transmit operation. In receive operation, it supports receive FIFO threshold level interrupt, receive FIFO overflow interrupt and receive FIFO underflow interrupt. When I^2S interrupt occurs, user can check I2STXINT (I2S_STATUS0[2]) and I2SRXINT (I2S_STATUS0[1]) flags to recognize the interrupt sources.

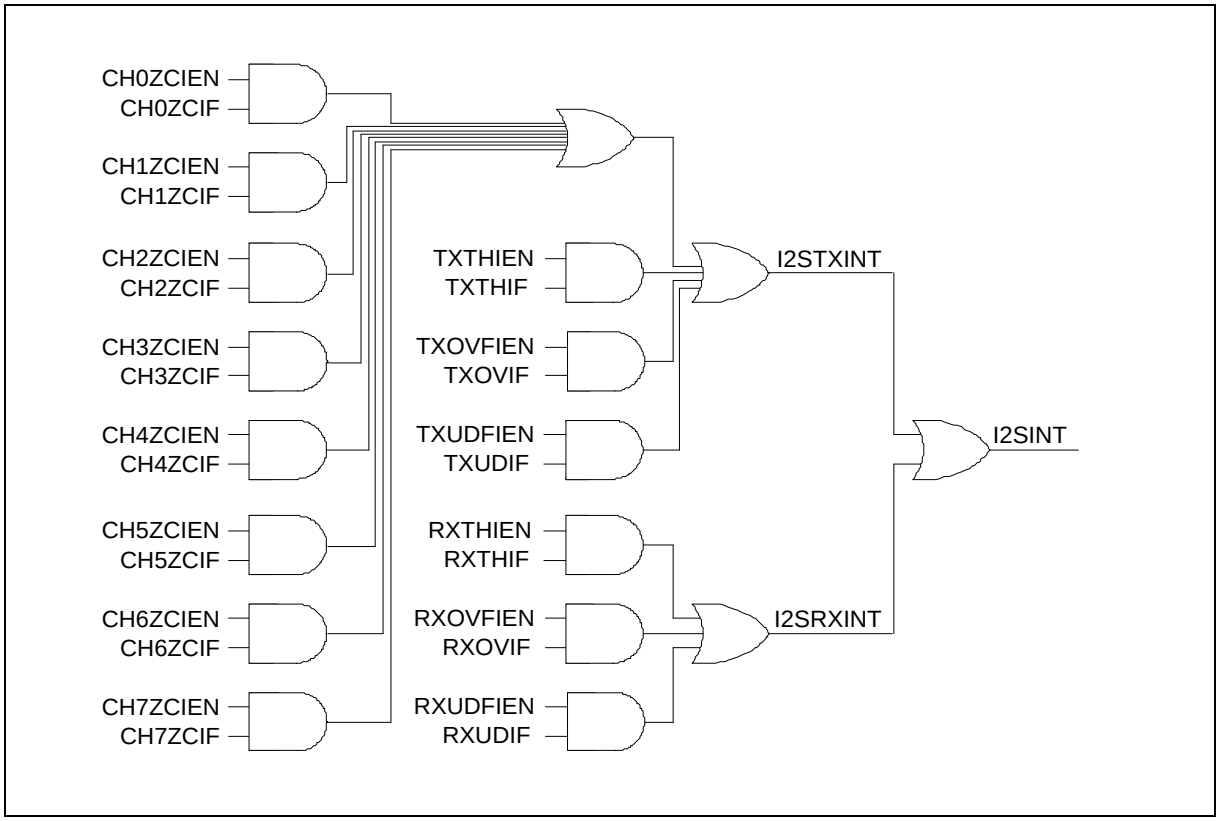


Figure 6.21-16 I^{2}S Interrupts

6.21.5.8 FIFO Operation

In 2-channel I^{2}S or PCM protocol, the bit-width of audio data in a channel block can be 8, 16, 24, or 32 bits. The memory arrangements of audio data for various settings are shown in Figure 6.21-17.

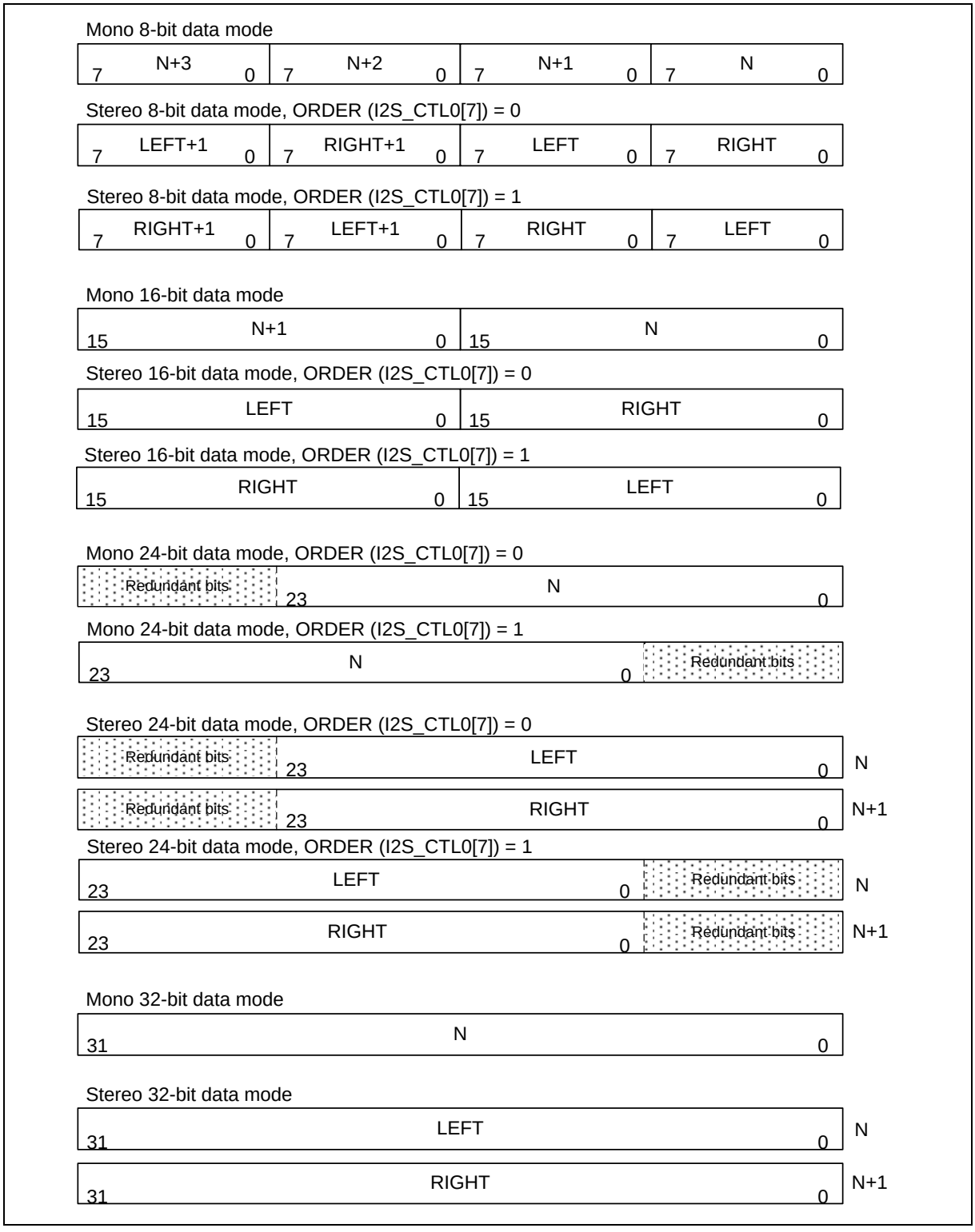


Figure 6.21-17 FIFO Contents for Various 2-channel Audio Modes

In 4-channel TDM PCM data format, the bit-width of audio data in a channel block can be 16, 24, or 32 bits. The memory arrangements of audio data for various settings are shown in Figure 6.21-18.

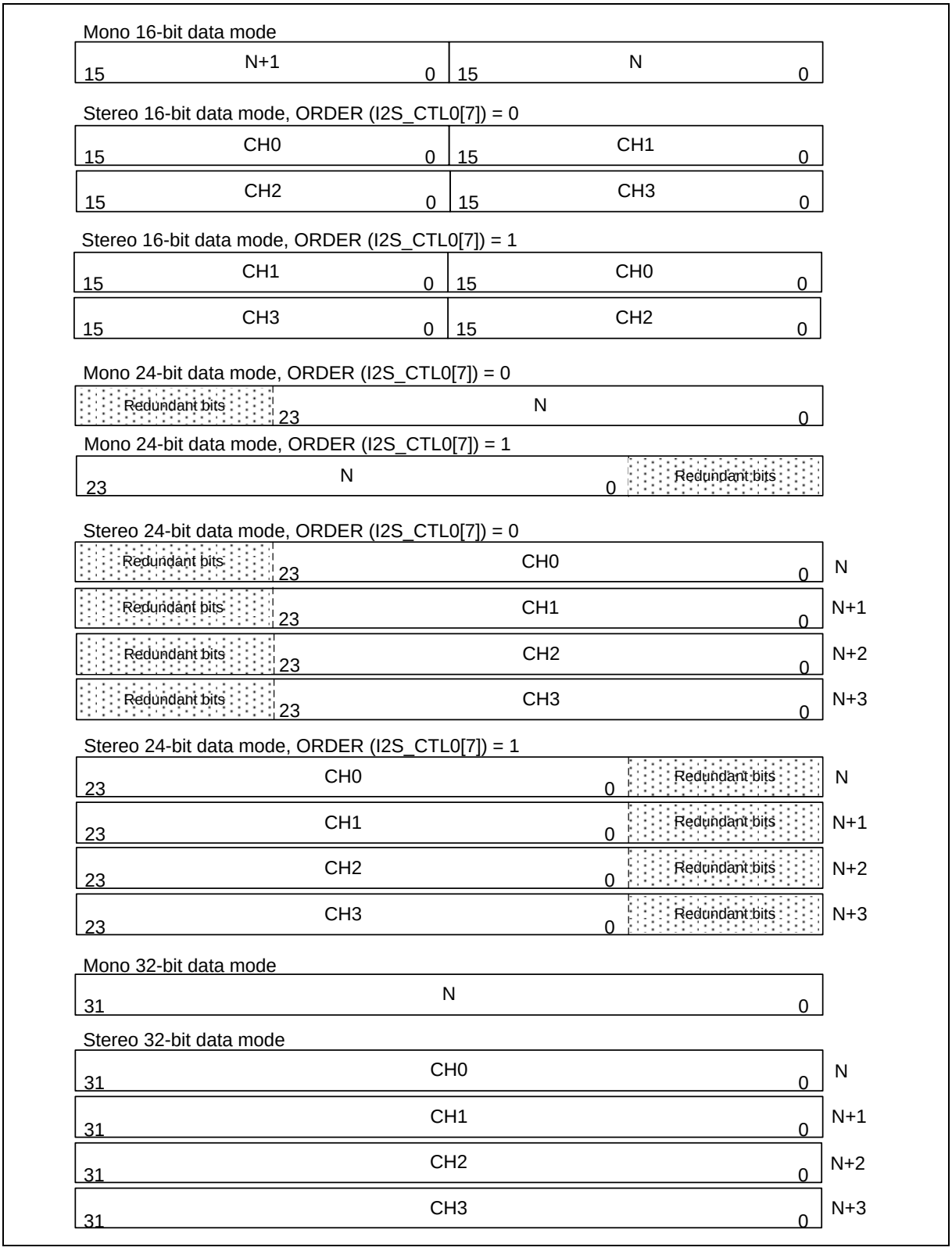


Figure 6.21-18 FIFO Contents for Various 4-channel Audio Modes

In 6-channel TDM PCM data format, the bit-width of audio data in a channel block can be 16, 24, or 32

bits. The memory arrangements of audio data for various settings are shown Figure 6.21-19. In 16-bit audio data transmission, ORDER (I2S_CTL0[7]) can be used to swap the audio data of even and odd channels which are stored in transmitting and receiving FIFO. In 24-bit audio data transmission, ORDER (I2S_CTL0[7]) can be also used to select the left-alignment or right-alignment formula of audio data which is stored in 32-bit FIFO entries.

The FIFO content of 8-channel TDM PCM data format is similar to 6-channel and it can be analogized easily.

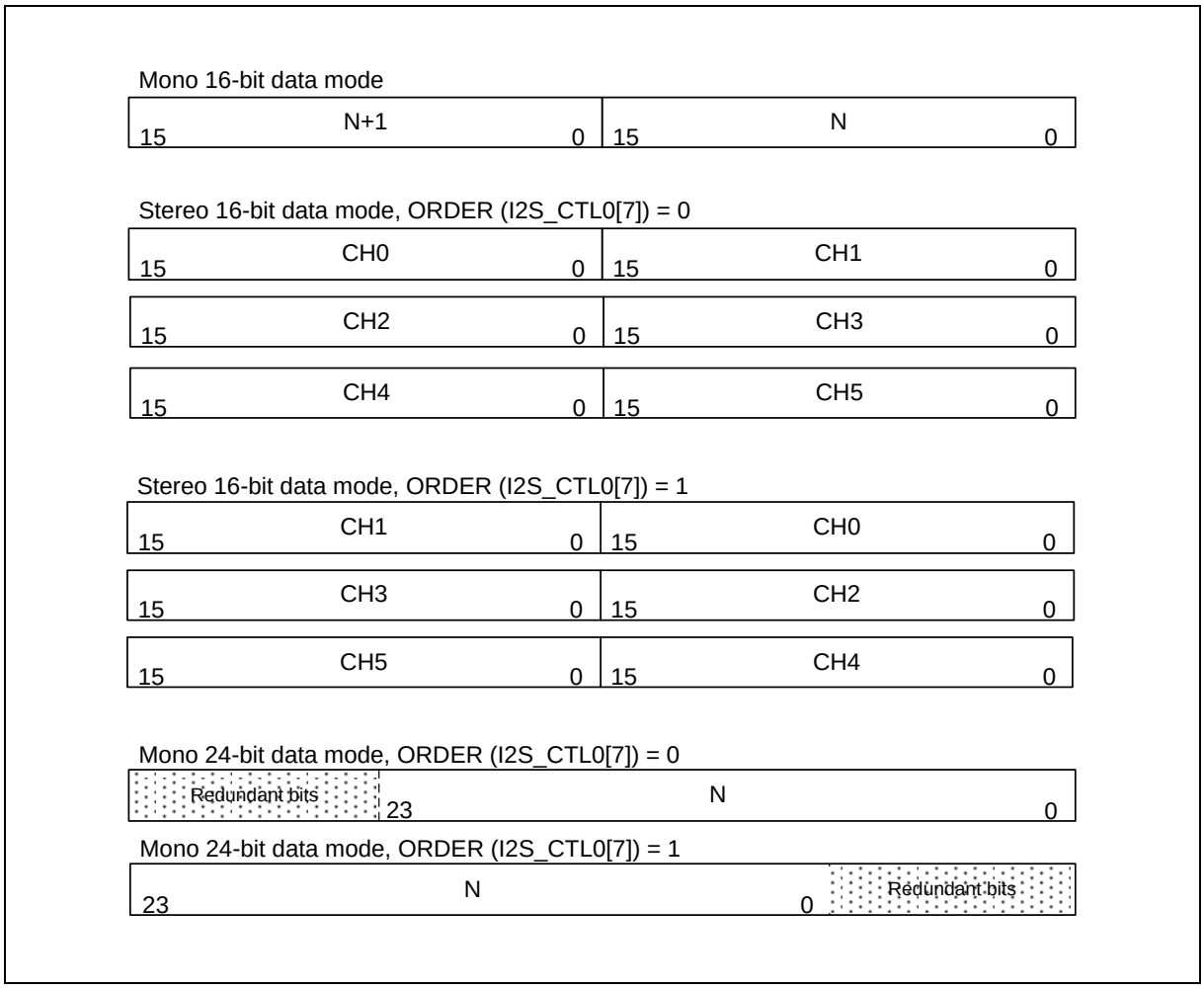


Figure 6.21-19 FIFO Contents for Various 6-channel Audio Modes (Part-1)

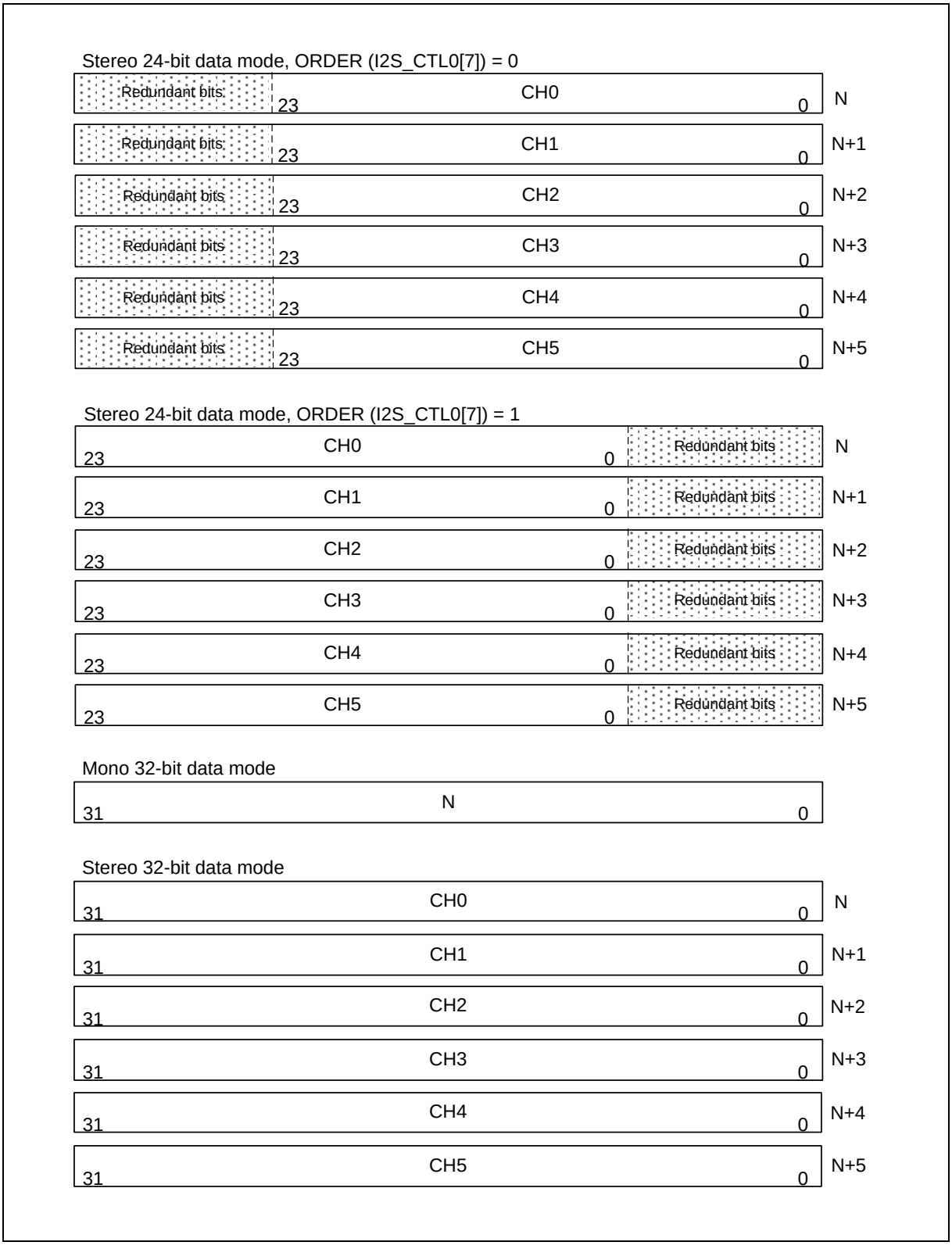


Figure 6.21-20 FIFO Contents for Various 6-channel Audio Modes (Part-2)

6.21.6 Register Map

R: *Read only,* ***W:*** *Write only,* ***R/W:*** *Both read and write*

Register	Offset	R/W	Description	Reset Value
I²S Base Address **I2S_BA = 0x4004_8000** **I²S non-secure base address is I2S_BA + 0x1000_0000.**				
I2S_CTL0	I2S_BA+0x00	R/W	I²S Control Register 0	0x0000_0000
I2S_CTL1	I2S_BA+0x20	R/W	I²S Control Register 1	0x0000_0000
I2S_CLKDIV	I2S_BA+0x04	R/W	I²S Clock Divider Register	0x0000_0000
I2S_IEN	I2S_BA+0x08	R/W	I²S Interrupt Enable Register	0x0000_0000
I2S_STATUS0	I2S_BA+0x0C	R/W	I²S Status Register 0	0x0014_1038
I2S_STATUS1	I2S_BA+0x24	R/W	I²S Status Register 1	0x0000_0000
I2S_TXFIFO	I2S_BA+0x10	W	I²S Transmit FIFO Register	0x0000_0000
I2S_RXFIFO	I2S_BA+0x14	R	I²S Receive FIFO Register	0x0000_0000

6.21.7 Register Description

I²S Control Register 0 (I2S_CTL0)

Register	Offset	R/W	Description	Reset Value
I2S_CTL0	I2S_BA+0x00	R/W	I²S Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
TDMCHNUM		CHWIDTH		PCMSYNC	FORMAT		
23	22	21	20	19	18	17	16
RXLCH	Reserved	RXPDMAEN	TXPDMAEN	RXFBCLR	TXFBCLR	Reserved	
15	14	13	12	11	10	9	8
MCLKEN	Reserved						SLAVE
7	6	5	4	3	2	1	0
ORDER	MONO	DATWIDTH		MUTE	RXEN	TXEN	I2SEN

Bits	Description	
[31:30]	**TDMCHNUM**	**TDM Channel Number** This bit fields are used to define the TDM channel number in one audio frame while PCM mode (FORMAT[2] = 1). 00 = 2 channels in audio frame. 01 = 4 channels in audio frame. 10 = 6 channels in audio frame. 11 = 8 channels in audio frame.
[29:28]	**CHWIDTH**	**Channel Width** This bit fields are used to define the length of audio channel. If CHWIDTH < DATWIDTH, the hardware will set the real channel length as the bit-width of audio data which is defined by DATWIDTH. 00 = The bit-width of each audio channel is 8-bit. 01 = The bit-width of each audio channel is 16-bit. 10 = The bit-width of each audio channel is 24-bit. 11 = The bit-width of each audio channel is 32-bit.
[27]	**PCMSYNC**	**PCM Synchronization Pulse Length Selection** This bit field is used to select the high pulse length of frame synchronization signal in PCM protocol 0 = One BCLK period. 1 = One channel period. **Note:** This bit is only available in master mode.

[26:24]	**FORMAT**	**Data Format Selection** 000 = I²S standard data format. 001 = I²S with MSB justified. 010 = I²S with LSB justified. 011 = Reserved. 100 = PCM standard data format. 101 = PCM with MSB justified. 110 = PCM with LSB justified. 111 = Reserved.
[23]	**RXLCH**	**Receive Left Channel Enable Bit** When monaural format is selected (MONO = 1), I²S will receive channel1 data if RXLCH is set to 0, and receive channel0 data if RXLCH is set to 1. 0 = Receive channel1 data in MONO mode. 1 = Receive channel0 data in MONO mode.
[22]	**Reserved**	Reserved.
[21]	**RXPDMAEN**	**Receive PDMA Enable Bit** 0 = Receive PDMA function Disabled. 1 = Receive PDMA function Enabled.
[20]	**TXPDMAEN**	**Transmit PDMA Enable Bit** 0 = Transmit PDMA function Disabled. 1 = Transmit PDMA function Enabled.
[19]	**RXFBCLR**	**Receive FIFO Buffer Clear** 0 = No Effect. 1 = Clear RX FIFO. **Note 1:** Write 1 to clear receive FIFO, internal pointer is reset to FIFO start point, and RXCNT (I2S_STATUS1[20:16]) returns 0 and receive FIFO becomes empty. **Note 2:** This bit is cleared by hardware automatically, read it return 0.
[18]	**TXFBCLR**	**Transmit FIFO Buffer Clear** 0 = No Effect. 1 = Clear TX FIFO. **Note 1:** Write 1 to clear transmit FIFO, internal pointer is reset to FIFO start point, and TXCNT (I2S_STATUS1[12:8]) returns 0 and transmit FIFO becomes empty but data in transmit FIFO is not changed. **Note 2:** This bit is clear by hardware automatically, read it return 0.
[17:16]	**Reserved**	Reserved.
[15]	**MCLKEN**	**Master Clock Enable Bit** If MCLKEN is set to 1, I²S controller will generate master clock on I2S_MCLK pin for external audio devices. 0 = Master clock Disabled. 1 = Master clock Enabled.
[14:9]	**Reserved**	Reserved.

[8]	**SLAVE**	**Slave Mode Enable Bit** 0 = Master mode. 1 = Slave mode. **Note:** I²S can operate as master or slave. For Master mode, I2S_BCLK and I2S_LRCLK pins are output mode and send out bit clock to Audio CODEC chip. In Slave mode, I2S_BCLK and I2S_LRCLK pins are input mode and I2S_BCLK and I2S_LRCLK signals are received from outer Audio CODEC chip.
[7]	**ORDER**	**Stereo Data Order in FIFO** In 8-bit/16-bit data width, this bit is used to select whether the even or odd channel data is stored in higher byte. In 24-bit data width, this is used to select the left/right alignment method of audio data which is stored in data memory consisted of 32-bit FIFO entries. 0 = Even channel data at high byte in 8-bit/16-bit data width. LSB of 24-bit audio data in each channel is aligned to right side in 32-bit FIFO entries. 1 = Even channel data at low byte in 8-bit/16-bit data width. MSB of 24-bit audio data in each channel is aligned to left side in 32-bit FIFO entries.
[6]	**MONO**	**Monaural Data Control** 0 = Data is stereo format. 1 = Data is monaural format. **Note:** When chip records data, RXLCH (I2S_CTL0[23]) indicates which channel data will be saved if monaural format is selected.
[5:4]	**DATWIDTH**	**Data Width** This bit field is used to define the bit-width of data word in each audio channel 00 = The bit-width of data word is 8-bit. 01 = The bit-width of data word is 16-bit. 10 = The bit-width of data word is 24-bit. 11 = The bit-width of data word is 32-bit.
[3]	**MUTE**	**Transmit Mute Enable Bit** 0 = Transmit data is shifted from buffer. 1 = Send zero on transmit channel.
[2]	**RXEN**	**Receive Enable Bit** 0 = Data receiving Disabled. 1 = Data receiving Enabled.
[1]	**TXEN**	**Transmit Enable Bit** 0 = Data transmission Disabled. 1 = Data transmission Enabled.
[0]	**I2SEN**	**I²S Controller Enable Bit** 0 = I²S controller Disabled. 1 = I²S controller Enabled.

I²S Control Register 1 (I2S_CTL1)

Register	Offset	R/W	Description	Reset Value
I2S_CTL1	I2S_BA+0x20	R/W	I²S Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved						PB16ORD	PBWIDTH
23	22	21	20	19	18	17	16
Reserved				RXTH			
15	14	13	12	11	10	9	8
Reserved				TXTH			
7	6	5	4	3	2	1	0
CH7ZCEN	CH6ZCEN	CH5ZCEN	CH4ZCEN	CH3ZCEN	CH2ZCEN	CH1ZCEN	CH0ZCEN

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25]	**PB16ORD**	**FIFO Read/Write Order in 16-bit Width of Peripheral Bus** When PBWIDTH = 1, the data FIFO will be increased or decreased by two peripheral bus access. This bit is used to select the order of FIFO access operations to meet the 32-bit transmitting/receiving FIFO entries. 0 = Low 16-bit read/write access first. 1 = High 16-bit read/write access first. **Note:** This bit is available while PBWIDTH = 1.
[24]	**PBWIDTH**	**Peripheral Bus Data Width Selection** This bit is used to choice the available data width of APB bus. It must be set to 1 while PDMA function is enabled and it is set to 16-bit transmission mode 0 = 32 bits data width. 1 = 16 bits data width. **Note 1:** If PBWIDTH=1, the low 16 bits of 32-bit data bus are available. **Note 2:** If PBWIDTH=1, the transmitting FIFO level will be increased after two FIFO write operations. **Note 3:** If PBWIDTH=1, the receiving FIFO level will be decreased after two FIFO read operations.
[23:20]	**Reserved**	Reserved.
[19:16]	**RXTH**	**Receive FIFO Threshold Level** 0000 = 1 data word in receive FIFO. 0001 = 2 data words in receive FIFO. 0010 = 3 data words in receive FIFO. …. 1110 = 15 data words in receive FIFO. 1111 = 16 data words in receive FIFO. **Note:** When received data word number in receive buffer is larger than threshold level then RXTHIF (I2S_STATUS0[10]) flag is set.

[15:12]	**Reserved**	Reserved.
[11:8]	**TXTH**	**Transmit FIFO Threshold Level** 0000 = 0 data word in transmit FIFO. 0001 = 1 data word in transmit FIFO. 0010 = 2 data words in transmit FIFO. .... 1110 = 14 data words in transmit FIFO. 1111 = 15 data words in transmit FIFO. **Note:** If remain data word number in transmit FIFO is less than or equal to threshold level then TXTHIF (I2S_STATUS0[18]) flag is set.
[7]	**CH7ZCEN**	**Channel7 Zero-cross Detect Enable Bit** 0 = channel7 zero-cross detect Disabled. 1 = channel7 zero-cross detect Enabled. **Note 1:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x3. **Note 2:** If this bit is set to 1, when channel7 data sign bit change or next shift data bits are all 0 then CH7ZCIF (I2S_STATUS1[7]) flag is set to 1. **Note 3:** If CH7ZCIF flag is set to 1, the channel7 will be mute.
[6]	**CH6ZCEN**	**Channel6 Zero-cross Detect Enable Bit** 0 = channel6 zero-cross detect Disabled. 1 = channel6 zero-cross detect Enabled. **Note 1:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x3. **Note 2:** If this bit is set to 1, when channel6 data sign bit change or next shift data bits are all 0 then CH6ZCIF(I2S_STATUS1[6]) flag is set to 1. **Note 3:** If CH6ZCIF flag is set to 1, the channel6 will be mute.
[5]	**CH5ZCEN**	**Channel5 Zero-cross Detect Enable Bit** 0 = channel5 zero-cross detect Disabled. 1 = channel5 zero-cross detect Enabled. **Note 1:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x2, 0x3. **Note 2:** If this bit is set to 1, when channel5 data sign bit change or next shift data bits are all 0 then CH5ZCIF(I2S_STATUS1[5]) flag is set to 1. **Note 3:** If CH5ZCIF flag is set to 1, the channel5 will be mute.
[4]	**CH4ZCEN**	**Channel4 Zero-cross Detect Enable Bit** 0 = channel4 zero-cross detect Disabled. 1 = channel4 zero-cross detect Enabled. **Note 1:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x2, 0x3. **Note 2:** If this bit is set to 1, when channel4 data sign bit change or next shift data bits are all 0 then CH4ZCIF(I2S_STATUS1[4]) flag is set to 1. **Note 3:** If CH4ZCIF flag is set to 1, the channel4 will be mute.
[3]	**CH3ZCEN**	**Channel3 Zero-cross Detect Enable Bit** 0 = channel3 zero-cross detect Disabled. 1 = channel3 zero-cross detect Enabled. **Note 1:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x1, 0x2, 0x3. **Note 2:** If this bit is set to 1, when channel3 data sign bit change or next shift data bits are all 0 then CH3ZCIF(I2S_STATUS1[3]) flag is set to 1. **Note 3:** If CH3ZCIF flag is set to 1, the channel3 will be mute.

[2]	**CH2ZCEN**	**Channel2 Zero-cross Detect Enable Bit** 0 = channel2 zero-cross detect Disabled. 1 = channel2 zero-cross detect Enabled. **Note 1:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x1, 0x2, 0x3. **Note 2:** If this bit is set to 1, when channel2 data sign bit change or next shift data bits are all 0 then CH2ZCIF(I2S_STATUS1[2]) flag is set to 1. **Note 3:** If CH2ZCIF flag is set to 1, the channel2 will be mute.
[1]	**CH1ZCEN**	**Channel1 Zero-cross Detect Enable Bit** 0 = channel1 zero-cross detect Disabled. 1 = channel1 zero-cross detect Enabled. **Note 1:** Channel1 also means right audio channel while I^2S (FORMAT[2]=0) or 2-channel PCM mode. **Note 2:** If this bit is set to 1, when channel1 data sign bit change or next shift data bits are all 0 then CH1ZCIF(I2S_STATUS1[1]) flag is set to 1. **Note 3:** If CH1ZCIF flag is set to 1, the channel1 will be mute.
[0]	**CH0ZCEN**	**Channel0 Zero-cross Detection Enable Bit** 0 = channel0 zero-cross detect Disabled. 1 = channel0 zero-cross detect Enabled. **Note 1:** Channel0 also means left audio channel while I^2S (FORMAT[2]=0) or 2-channel PCM mode. **Note 2:** If this bit is set to 1, when channel0 data sign bit change or next shift data bits are all 0 then CH0ZCIF(I2S_STATUS1[0]) flag is set to 1. **Note 3:** If CH0ZCIF flag is set to 1, the channel0 will be mute.

I²S Clock Divider (I2S_CLKDIV)

Register	Offset	R/W	Description	Reset Value
I2S_CLKDIV	I2S_BA+0x04	R/W	I²S Clock Divider Register	0x0000_0000

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

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17:8]	**BCLKDIV**	**Bit Clock Divider** The I²S controller will generate bit clock in Master mode. Software can program these bit fields to generate sampling rate clock frequency. F_BCLK= F_I2SCLK / (2 x (BCLKDIV + 1)). **Note:** F_BCLK is the frequency of BCLK and F_I2SCLK is the frequency of I2S_CLK.
[7]	**Reserved**	Reserved.
[6:0]	**MCLKDIV**	**Master Clock Divider** If chip external crystal frequency is (2 x MCLKDIV) x 256fs then software can program these bits to generate 256fs clock frequency to audio codec chip. If MCLKDIV is set to 0, MCLK is the same as external clock input. For example, sampling rate is 24 kHz and chip external crystal clock is 12.288 MHz, set MCLKDIV = 1. F_MCLK = F_I2SCLK / (2 x MCLKDIV) (When MCLKDIV is >= 1). F_MCLK = F_I2SCLK (When MCLKDIV is set to 0). **Note:** F_MCLK is the frequency of MCLK, and F_I2SCLK is the frequency of the I2S_CLK.

I²S Interrupt Enable Register (I2S_IEN)

Register	Offset	R/W	Description	Reset Value
I2S_IEN	I2S_BA+0x08	R/W	I²S Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
CH7ZCIEN	CH6ZCIEN	CH5ZCIEN	CH4ZCIEN	CH3ZCIEN	CH2ZCIEN	CH1ZCIEN	CH0ZCIEN
15	14	13	12	11	10	9	8
Reserved					TXTHIEN	TXOVFIEN	TXUDFIEN
7	6	5	4	3	2	1	0
Reserved					RXTHIEN	RXOVFIEN	RXUDFIEN

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23]	**CH7ZCIEN**	**Channel7 Zero-cross Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note 1:** Interrupt occurs if this bit is set to 1 and channel7 zero-cross. **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x3.
[22]	**CH6ZCIEN**	**Channel6 Zero-cross Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note 1:** Interrupt occurs if this bit is set to 1 and channel6 zero-cross. **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x3.
[21]	**CH5ZCIEN**	**Channel5 Zero-cross Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note 1:** Interrupt occurs if this bit is set to 1 and channel5 zero-cross. **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x2, 0x3.
[20]	**CH4ZCIEN**	**Channel4 Zero-cross Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note 1:** Interrupt occurs if this bit is set to 1 and channel4 zero-cross. **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x2, 0x3.

[19]	**CH3ZCIEN**	**Channel3 Zero-cross Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note 1:** Interrupt occurs if this bit is set to 1 and channel3 zero-cross. **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x1, 0x2, 0x3.
[18]	**CH2ZCIEN**	**Channel2 Zero-cross Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note 1:** Interrupt occurs if this bit is set to 1 and channel2 zero-cross. **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x1, 0x2, 0x3.
[17]	**CH1ZCIEN**	**Channel1 Zero-cross Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note 1:** Interrupt occurs if this bit is set to 1 and channel1 zero-cross. **Note 2:** Channel1 also means right audio channel while I^2S (FORMAT[2]=0) or 2-channel PCM mode.
[16]	**CH0ZCIEN**	**Channel0 Zero-cross Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note 1:** Interrupt occurs if this bit is set to 1 and channel0 zero-cross. **Note 2:** Channel0 also means left audio channel while I^2S (FORMAT[2]=0) or 2-channel PCM mode.
[15:11]	**Reserved**	Reserved.
[10]	**TXTHIEN**	**Transmit FIFO Threshold Level Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note:** Interrupt occurs if this bit is set to 1 and data words in transmit FIFO is less than or equal to TXTH (I2S_CTL1[11:8]).
[9]	**TXOVFIEN**	**Transmit FIFO Overflow Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note:** Interrupt occurs if this bit is set to 1 and TXOVIF (I2S_STATUS0[17]) flag is set to 1.
[8]	**TXUDFIEN**	**Transmit FIFO Underflow Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note:** Interrupt occurs if this bit is set to 1 and TXUDIF (I2S_STATUS0[16]) flag is set to 1.
[7:3]	**Reserved**	Reserved.
[2]	**RXTHIEN**	**Receive FIFO Threshold Level Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note:** Interrupt occurs if this bit is set to 1 and data words in receive FIFO is larger than RXTH (I2S_CTL1[19:16]).

[1]	**RXOVFIEN**	**Receive FIFO Overflow Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note:** Interrupt occurs if this bit is set to 1 and RXOVIF (I2S_STATUS0[9]) flag is set to 1.
[0]	**RXUDFIEN**	**Receive FIFO Underflow Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note:** Interrupt occurs if this bit is set to 1 and RXUDIF (I2S_STATUS0[8]) flag is set to 1.

I²S Status Register 0 (I2S_STATUS0)

Register	Offset	R/W	Description	Reset Value
I2S_STATUS0	I2S_BA+0x0C	R/W	I²S Status Register 0	0x0014_1038

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		TXBUSY	TXEMPTY	TXFULL	TXTHIF	TXOVIF	TXUDIF
15	14	13	12	11	10	9	8
Reserved			RXEMPTY	RXFULL	RXTHIF	RXOVIF	RXUDIF
7	6	5	4	3	2	1	0
Reserved		DATACH			I2STXINT	I2SRXINT	I2SINT

Bits	Description	
[31:22]	**Reserved**	Reserved.
[21]	**TXBUSY**	**Transmit Busy (Read Only)** 0 = Transmit shift buffer is empty. 1 = Transmit shift buffer is busy. **Note:** This bit is cleared to 0 when all data in transmit FIFO and shift buffer is shifted out. And set to 1 when 1st data is load to shift buffer.
[20]	**TXEMPTY**	**Transmit FIFO Empty (Read Only)** 0 = Not empty. 1 = Empty. **Note:** This bit reflects data words number in transmit FIFO is 0.
[19]	**TXFULL**	**Transmit FIFO Full (Read Only)** 0 = Not full. 1 = Full. **Note:** This bit reflects data words number in transmit FIFO is 16.
[18]	**TXTHIF**	**Transmit FIFO Threshold Interrupt Flag (Read Only)** 0 = Data word(s) in FIFO is larger than threshold level. 1 = Data word(s) in FIFO is less than or equal to threshold level. **Note:** When data word(s) in transmit FIFO is less than or equal to threshold value set in TXTH (I2S_CTL1[11:8]) the TXTHIF bit becomes to 1. It keeps at 1 till TXCNT (I2S_STATUS1[12:8]) is larger than TXTH (I2S_CTL1[11:8]) after software write TXFIFO register.
[17]	**TXOVIF**	**Transmit FIFO Overflow Interrupt Flag** 0 = No overflow. 1 = Overflow. **Note 1:** Write data to transmit FIFO when it is full and this bit set to 1. **Note 2:** Write 1 to clear this bit to 0.

[16]	**TXUDIF**	**Transmit FIFO Underflow Interrupt Flag** 0 = No underflow. 1 = Underflow. **Note 1:** This bit will be set to 1 when shift logic hardware read data from transmitting FIFO and the filling data level in transmitting FIFO is not enough for one audio frame. **Note 2:** Write 1 to clear this bit to 0.
[15:13]	**Reserved**	Reserved.
[12]	**RXEMPTY**	**Receive FIFO Empty (Read Only)** 0 = Not empty. 1 = Empty. **Note:** This bit reflects data words number in receive FIFO is 0.
[11]	**RXFULL**	**Receive FIFO Full (Read Only)** 0 = Not full. 1 = Full. **Note:** This bit reflects data words number in receive FIFO is 16.
[10]	**RXTHIF**	**Receive FIFO Threshold Interrupt Flag (Read Only)** 0 = Data word(s) in FIFO is less than or equal to threshold level. 1 = Data word(s) in FIFO is larger than threshold level. **Note:** When data word(s) in receive FIFO is larger than threshold value set in RXTH (I2S_CTL1[19:16]) the RXTHIF bit becomes to 1. It keeps at 1 till RXCNT (I2S_STATUS1[20:16]) is less than or equal to RXTH (I2S_CTL1[19:16]) after software read RXFIFO register.
[9]	**RXOVIF**	**Receive FIFO Overflow Interrupt Flag** 0 = No overflow occur. 1 = Overflow occur. **Note 1:** When receive FIFO is full and receive hardware attempt to write data into receive FIFO then this bit is set to 1, data in 1st buffer is overwritten. **Note 2:** Write 1 to clear this bit to 0.
[8]	**RXUDIF**	**Receive FIFO Underflow Interrupt Flag** 0 = No underflow occur. 1 = Underflow occur. **Note 1:** When receive FIFO is empty, and software reads the receive FIFO again. This bit will be set to 1, and it indicates underflow situation occurs. **Note 2:** Write 1 to clear this bit to 0
[7:6]	**Reserved**	Reserved.
[5:3]	**DATACH**	**Transmission Data Channel (Read Only)** This bit fields are used to indicate which audio channel is current transmit data belong. 000 = channel0 (means left channel while 2-channel I2S/PCM mode). 001 = channel1 (means right channel while 2-channel I2S/PCM mode). 010 = channel2 (available while 4-channel TDM PCM mode). 011 = channel3 (available while 4-channel TDM PCM mode). 100 = channel4 (available while 6-channel TDM PCM mode). 101 = channel5 (available while 6-channel TDM PCM mode). 110 = channel6 (available while 8-channel TDM PCM mode). 111 = channel7 (available while 8-channel TDM PCM mode).

[2]	**I2STXINT**	**I²S Transmit Interrupt (Read Only)** 0 = No transmit interrupt. 1 = Transmit interrupt.
[1]	**I2SRXINT**	**I²S Receive Interrupt (Read Only)** 0 = No receive interrupt. 1 = Receive interrupt.
[0]	**I2SINT**	**I²S Interrupt Flag (Read Only)** 0 = No I²S interrupt. 1 = I²S interrupt. **Note:** It is wire-OR of I2STXINT and I2SRXINT bits.

I²S Status Register 1 (I2S_STATUS1)

Register	Offset	R/W	Description	Reset Value
I2S_STATUS1	I2S_BA+0x24	R/W	I²S Status Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved			RXCNT				
15	14	13	12	11	10	9	8
Reserved			TXCNT				
7	6	5	4	3	2	1	0
CH7ZCIF	CH6ZCIF	CH5ZCIF	CH4ZCIF	CH3ZCIF	CH2ZCIF	CH1ZCIF	CH0ZCIF

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20:16]	**RXCNT**	**Receive FIFO Level (Read Only)** These bits indicate the number of available entries in receive FIFO. 00000 = No data. 00001 = 1 word in receive FIFO. 00010 = 2 words in receive FIFO. …. 01110 = 14 words in receive FIFO. 01111 = 15 words in receive FIFO. 10000 = 16 words in receive FIFO. Others are reserved.
[15:13]	**Reserved**	Reserved.
[12:8]	**TXCNT**	**Transmit FIFO Level (Read Only)** These bits indicate the number of available entries in transmit FIFO. 00000 = No data. 00001 = 1 word in transmit FIFO. 00010 = 2 words in transmit FIFO. …. 01110 = 14 words in transmit FIFO. 01111 = 15 words in transmit FIFO. 10000 = 16 words in transmit FIFO. Others are reserved.

[7]	**CH7ZCIF**	**Channel7 Zero-cross Interrupt Flag** It indicates channel7 next sample data sign bit is changed or all data bits are 0. 0 = No zero-cross in channel7. 1 = Channel7 zero-cross is detected. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x3.
[6]	**CH6ZCIF**	**Channel6 Zero-cross Interrupt Flag** It indicates channel6 next sample data sign bit is changed or all data bits are 0. 0 = No zero-cross in channel6. 1 = Channel6 zero-cross is detected. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x3.
[5]	**CH5ZCIF**	**Channel5 Zero-cross Interrupt Flag** It indicates channel5 next sample data sign bit is changed or all data bits are 0. 0 = No zero-cross in channel5. 1 = Channel5 zero-cross is detected. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x2, 0x3.
[4]	**CH4ZCIF**	**Channel4 Zero-cross Interrupt Flag** It indicates channel4 next sample data sign bit is changed or all data bits are 0. 0 = No zero-cross in channel4. 1 = Channel4 zero-cross is detected. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x2, 0x3.
[3]	**CH3ZCIF**	**Channel3 Zero-cross Interrupt Flag** It indicates channel3 next sample data sign bit is changed or all data bits are 0. 0 = No zero-cross in channel3. 1 = Channel3 zero-cross is detected. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x1, 0x2, 0x3.
[2]	**CH2ZCIF**	**Channel2 Zero-cross Interrupt Flag** It indicates channel2 next sample data sign bit is changed or all data bits are 0. 0 = No zero-cross in channel2. 1 = Channel2 zero-cross is detected. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x1, 0x2, 0x3.
[1]	**CH1ZCIF**	**Channel1 Zero-cross Interrupt Flag** It indicates channel1 next sample data sign bit is changed or all data bits are 0. 0 = No zero-cross in channel1. 1 = Channel1 zero-cross is detected. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** Channel1 also means right audio channel while I^2S (FORMAT[2]=0) or 2-channel PCM mode.

[0]	**CH0ZCIF**	**Channel0 Zero-cross Interrupt Flag** It indicates channel0 next sample data sign bit is changed or all data bits are 0. 0 = No zero-cross in channel0. 1 = Channel0 zero-cross is detected. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** Channel0 also means left audio channel while I^2S (FORMAT[2]=0) or 2-channel PCM mode.

I²S Transmit FIFO (I2S_TXFIFO)

Register	Offset	R/W	Description	Reset Value
I2S_TXFIFO	I2S_BA+0x10	W	I²S Transmit FIFO Register	0x0000_0000

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

Bits	Description	
[31:0]	**TXFIFO**	**Transmit FIFO Bits** The I²S contains 16 words (16x32 bits) data buffer for data transmit. Write data to this register to prepare data for transmit. The remaining word number is indicated by TXCNT (I2S_STATUS1[12:8]).

I²S Receive FIFO (I2S_RXFIFO)

Register	Offset	R/W	Description	Reset Value
I2S_RXFIFO	I2S_BA+0x14	R	I²S Receive FIFO Register	0x0000_0000

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

Bits	Description	
[31:0]	**RXFIFO**	**Receive FIFO Bits** I²S contains 16 words (16x32 bits) data buffer for data receive. Read this register to get data in FIFO. The remaining data word number is indicated by RXCNT (I2S_STATUS1[20:16]).

6.22 Serial Peripheral Interface (SPI)

6.22.1 Overview

The Serial Peripheral Interface (SPI) applies to synchronous serial data communication and allows full duplex transfer. Devices communicate in Master/Slave mode with the 4-wire bi-direction interface. The M2354 series contains up to four sets of SPI controllers performing a serial-to-parallel conversion on data received from a peripheral device, and a parallel-to-serial conversion on data transmitted to a peripheral device. Each SPI controller can be configured as a master or a slave device and supports the PDMA function to access the data buffer. Each SPI controller also supports I^2S mode to connect external audio CODEC.

6.22.2 Features

- SPI Mode
  - Up to four sets of SPI controllers
  - Supports Master or Slave mode operation
  - Configurable bit length of a transaction word from 8 to 32-bit
  - Provides separate 4-level depth transmit and receive FIFO buffers
  - Supports MSB first or LSB first transfer sequence
  - Supports Byte Reorder function
  - Supports Byte or Word Suspend mode
  - Supports PDMA transfer
  - Supports 3-Wire, no slave selection signal, bi-direction interface
  - Supports one data channel half-duplex transfer
  - Supports receive-only mode
- I^2S Mode
  - Supports Master or Slave
  - Capable of handling 8-, 16-, 24- and 32-bit word sizes
  - Each provides two 4-level FIFO data buffers, one for transmitting and the other for receiving
  - Supports monaural and stereo audio data
  - Supports PCM mode A, PCM mode B, I^2S and MSB justified data format
  - Supports two PDMA requests, one for transmitting and the other for receiving

6.22.3 Block Diagram

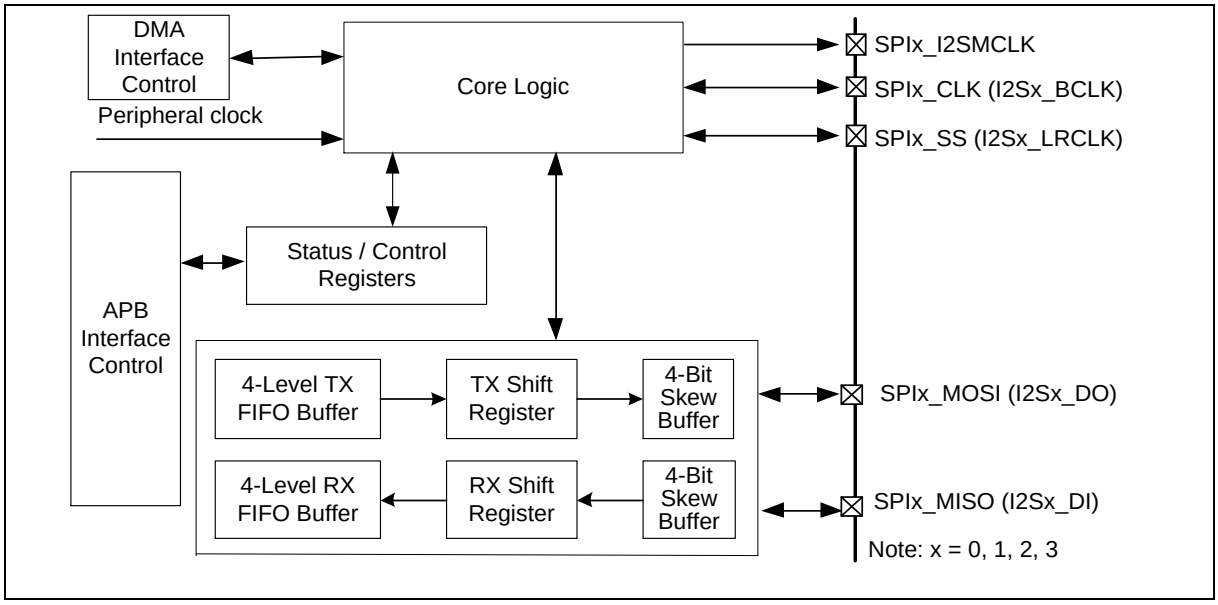


Figure 6.22-1 SPI Block Diagram

TX FIFO Buffer:

The transmit FIFO buffer is a 4-level depth, 32-bit wide, first-in, first-out register buffer. The data can be written to the transmit FIFO buffer in advance through software by writing the SPIx_TX register. In SPI mode, the transmit FIFO will be configured as 8-level while data length is set as 8~16 bits.

RX FIFO Buffer:

The receive FIFO buffer is also a 4-level depth, 32-bit wide, first-in, first-out register buffer. The receive control logic will store the receive data to this buffer. The FIFO buffer data can be read from SPIx_RX register by software. In SPI mode, the receive FIFO will be configured as 8-level while data length is set as 8~16 bits.

TX Shift Register:

The transmit shift register is a 32-bit wide register buffer. The transmit data is loaded from the TX FIFO buffer and shifted out bit-by-bit to the skew buffer.

RX Shift Register:

The receive shift register is also a 32-bit wide register buffer. The receive data is shift in bit-by-bit from the skew buffer and is loaded into RX FIFO buffer when a transaction done.

Skew Buffer:

The skew buffer is a 4-level 1-bit buffer. There are two skew buffers in transmitting and received side. In received side, it is used to shift bits into RX shift register from SPI bus. In transmitting side, it is used to shift bits into SPI bus from TX shift register.

6.22.4 Basic Configuration

6.22.4.1 SPI0 Basic Configuration

- Clock source Configuration
  - Select the source of SPI0 peripheral clock on SPI0SEL (CLK_CLKSEL2[5:4]).

- Enable SPI0 peripheral clock in SPI0CKEN (CLK_APBCLK0[13]).

- Reset Configuration
  - Reset SPI0 controller in SPI0RST (SYS_IPRST1[13]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
SPI0	SPI0_CLK	PA.2, PB.14, PD.2	MFP4
		PF.8	MFP5
	SPI0_I2SMCLK	PA.4	MFP4
		PF.10	MFP5
		PD.14	MFP6
		PB.0	MFP8
		PB.11	MFP9
	SPI0_MISO	PA.1, PB.13, PD.1	MFP4
		PF.7	MFP5
	SPI0_MOSI	PA.0, PB.12, PD.0	MFP4
		PF.6	MFP5
	SPI0_SS	PA.3, PB.15, PD.3	MFP4
		PF.9	MFP5

6.22.4.2 SPI1 Basic Configuration

- Clock source Configuration
  - Select the source of SPI1 peripheral clock on SPI1SEL (CLK_CLKSEL2[7:6]).
  - Enable SPI1 peripheral clock in SPI1CKEN (CLK_APBCLK0[14]).
- Reset Configuration
  - Reset SPI1 controller in SPI1RST (SYS_IPRST1[14]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
SPI1	SPI1_CLK	PH.6	MFP3
		PA.7	MFP4
		PB.3, PD.5	MFP5
		PH.8	MFP6
		PC.1	MFP7
	SPI1_I2SMCLK	PA.5	MFP4
		PB.1	MFP5
		PH.10	MFP6

		PC.4	MFP7
	SPI1_MISO	PH.4	MFP3
		PC.7	MFP4
		PB.5, PD.7	MFP5
		PE.1	MFP6
		PC.3	MFP7
	SPI1_MOSI	PH.5	MFP3
		PC.6	MFP4
		PB.4, PD.6	MFP5
		PE.0	MFP6
		PC.2	MFP7
	SPI1_SS	PH.7	MFP3
		PA.6	MFP4
		PB.2, PD.4	MFP5
		PH.9	MFP6
		PC.0	MFP7

6.22.4.3 SPI2 Basic Configuration

- Clock source Configuration
  - Select the source of SPI2 peripheral clock on SPI2SEL (CLK_CLKSEL2[11:10]).
  - Enable SPI2 peripheral clock in SPI2CKEN (CLK_APBCLK0[15]).
- Reset Configuration
  - Reset SPI2 controller in SPI2RST (SYS_IPRST1[15]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
SPI2	SPI2_CLK	PG.3	MFP3
		PA.10	MFP4
		PA.13, PE.8	MFP5
	SPI2_I2SMCLK	PC.13	MFP4
		PE.12	MFP5
	SPI2_MISO	PG.4	MFP3
		PA.9	MFP4
		PA.14, PE.9	MFP5
	SPI2_MOSI	PF.11	MFP3
		PA.8	MFP4

		PA.15, PE.10	MFP5
	SPI2_SS	PG.2	MFP3
		PA.11	MFP4
		PA.12, PE.11	MFP5

6.22.4.4 SPI3 Basic Configuration

- Clock source Configuration
  - Select the source of SPI3 peripheral clock on SPI3SEL (CLK_CLKSEL2[13:12]).
  - Enable SPI3 peripheral clock in SPI3CKEN (CLK_APBCLK1[6]).
- Reset Configuration
  - Reset SPI3 controller in SPI3RST (SYS_IPRST2[6]).
- Pin configuration

Group	Pin Name	GPIO	MFP
SPI3	SPI3_CLK	PE.4	MFP5
		PC.10	MFP6
		PB.11	MFP11
	SPI3_I2SMCLK	PD.14	MFP3
		PE.6	MFP5
		PB.1	MFP6
	SPI3_MISO	PE.3	MFP5
		PC.12	MFP6
		PB.9	MFP11
	SPI3_MOSI	PE.2	MFP5
		PC.11	MFP6
		PB.8	MFP11
	SPI3_SS	PE.5	MFP5
		PC.9	MFP6
		PB.10	MFP11

SPI/I2S (SPI0~SPI3) Interface Controller Pin description is shown as follows:

Pin	SPI Mode	I^2S Mode
SPIx_SS	SPI slave selection pin	I^2S left/right channel synchronization clock pin (I2Sx_LRCLK)
SPIx_CLK	SPI clock pin	I^2S bit clock pin (I2Sx_BCLK)
SPIx_MISO	SPI master input or slave output pin	I^2S data input pin (I2Sx_DI)
SPIx_MOSI	SPI master output or slave input pin	I^2S data output pin (I2Sx_DO)

SPIx_I2SMCLK	Not available	I²S Master clock output pin

Table 6.22-1 SPI/I²S Interface Controller Pin Description (SPI0~SPI3)

6.22.5 Functional Description

6.22.5.1 Terminology

SPI Peripheral Clock and SPI Bus Clock

The SPI controller needs the peripheral clock to drive the SPI logic unit to perform the data transfer. The peripheral clock rate is determined by the settings of clock divisor (SPIx_CLKDIV) and the clock source which can be HXT, PLL, PCLK or HIRC. SPIxSEL of CLK_CLKSEL2 register determines the clock source of the peripheral clock. The DIVIDER (SPIx_CLKDIV[8:0]) setting determines the divisor of the clock rate calculation.

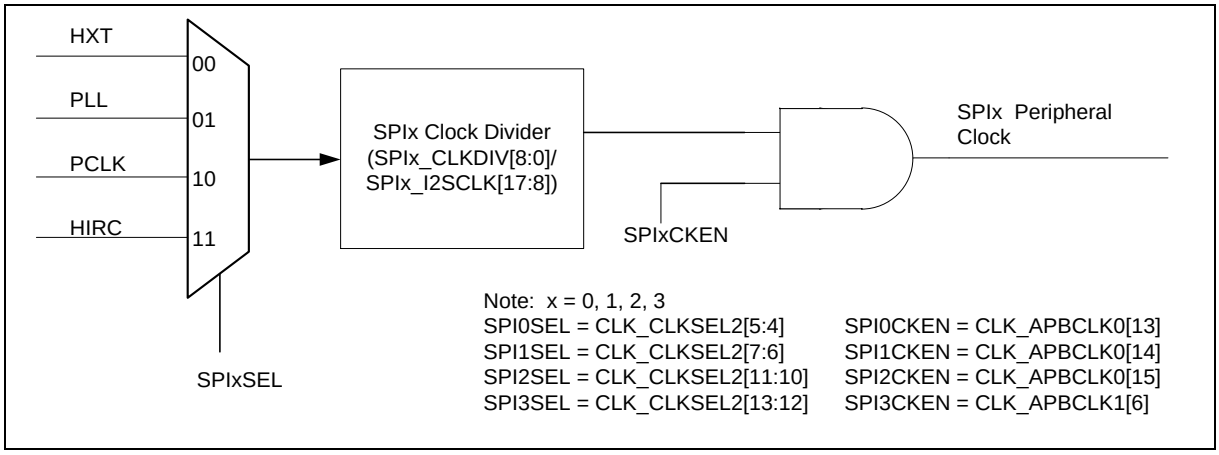


Figure 6.22-2 SPI Peripheral Clock

In Master mode, the frequency of the SPI bus clock is equal to the peripheral clock rate. In general, the SPI bus clock is denoted as SPI clock. In Slave mode, the SPI bus clock is provided by a master device. The frequency of SPI peripheral clock cannot be faster than the system clock rate regardless of Master or Slave mode. If the clock source of peripheral clock is not system clock, the frequency of SPI peripheral clock shall be slower than the system clock frequency regardless of Master or Slave mode.

In I²S mode, the peripheral clock rate is equal to I²S bit clock rate determined by SPIx_I2SCLK register.

Master/Slave mode

The SPI controllers can be set as Master or Slave mode by setting the SLAVE (SPIx_CTL[18]) to communicate with the off-chip SPI slave or master device. The HALFDPX (SPIx_CTL[14]) can be used to select the full-duplex or half-duplex in SPI transmission. The application block diagrams in Master and Slave mode are shown below.

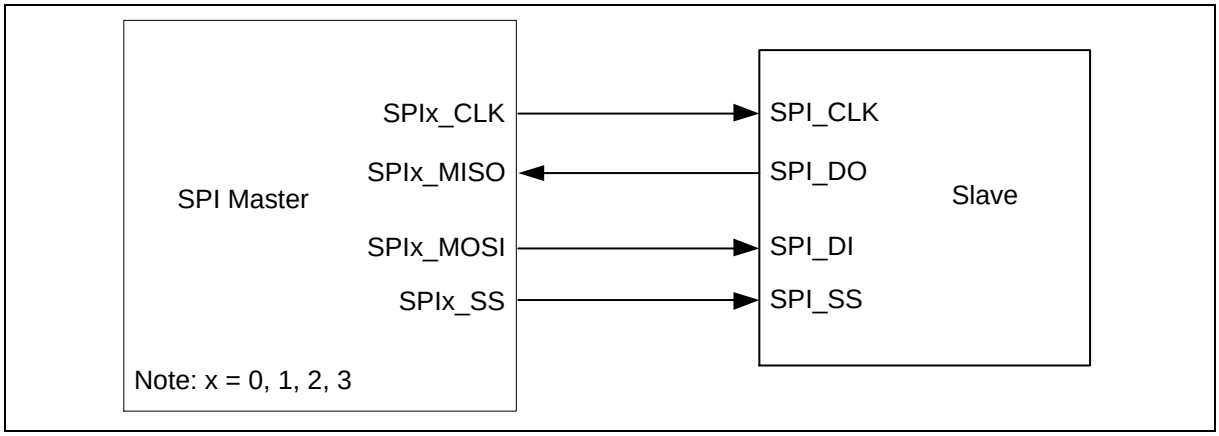


Figure 6.22-3 SPI Full-Duplex Master Mode Application Block Diagram

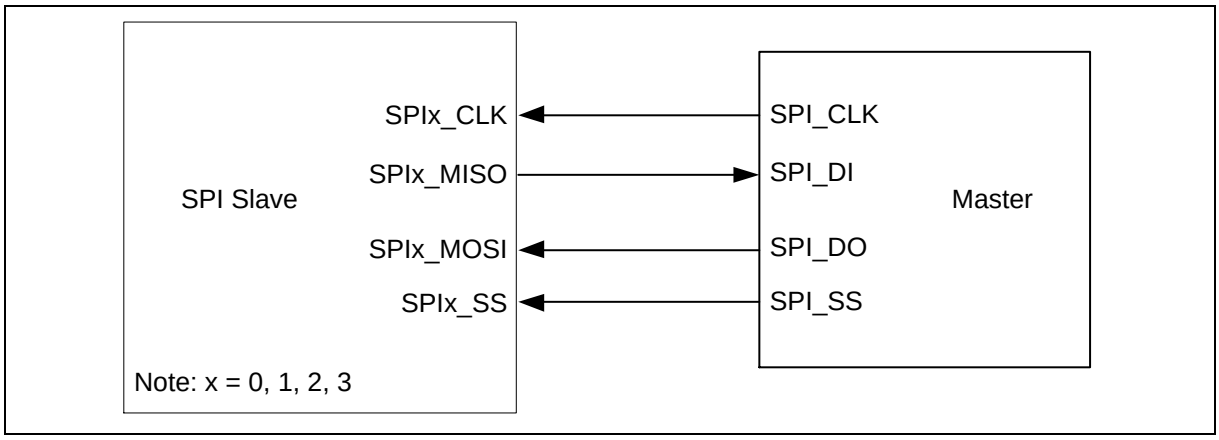


Figure 6.22-4 SPI Full-Duplex Slave Mode Application Block Diagram

Slave Selection

In Master mode, the SPI controller can drive off-chip slave device through the slave select output pin SPIx_SS. In Slave mode, the off-chip master device drives the slave selection signal from the SPIx_SS input port to this SPI controller. The duration between the slave select active edge and the first SPI clock input shall over 3 SPI peripheral clock cycles of slave.

In Master/Slave mode, the active state of slave selection signal can be programmed to low or high active in SSACTPOL (SPIx_SSCTL[2]). The selection of slave select conditions depends on what type of device is connected. In Slave mode, to recognize the inactive state of the slave selection signal, the inactive period of the slave selection signal must be larger than or equal to 3 peripheral clock cycles between two successive transactions.

In Slave mode, the SPIx_MISO signal type is controlled by the SPIx_SS pin. If the SPIx_SS pin is inactive state, the SPIx_MISO is set to tri-state to avoid interference with other Slave devices. If the SPIx_SS pin is active state, the SPIx_MISO is set to output mode for data transfer.

Timing Condition

The CLKPOL (SPIx_CTL[3]) defines the SPI clock idle state. If CLKPOL = 1, the output SPI clock is idle at high state; if CLKPOL = 0, it is idle at low state.

TXNEG (SPIx_CTL[2]) defines the data transmitted out either on negative edge or on positive edge of SPI clock. RXNEG (SPIx_CTL[1]) defines the data received either on negative edge or on positive edge of SPI clock.

Note: The settings of TXNEG and RXNEG are mutually exclusive. In other words, do not transmit and receive data at the same clock edge.

Transmit/Receive Bit Length

The bit length of a transaction word is defined in DWIDTH (SPIx_CTL[12:8]) and can be configured up to 32-bit length in a transaction word for transmitting and receiving.

When SPI controller finishes a transaction, i.e. receives or transmits a specific count of bits defined in DWIDTH (SPIx_CTL[12:8]), the unit transfer interrupt flag UNITIF (SPIx_STATUS[1]) will be set to 1.

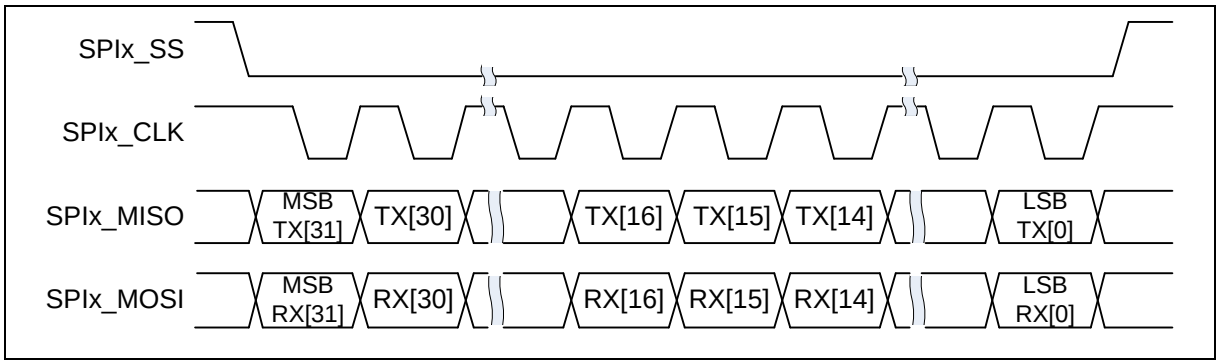


Figure 6.22-5 32-bit in One Transaction

LSB/MSB First

LSB (SPIx_CTL[13]) defines the bit transfer sequence in a transaction. If the LSB (SPIx_CTL[13]) is set to 1, the transfer sequence is LSB first. The bit 0 will be transferred firstly. If the LSB (SPIx_CTL[13]) is cleared to 0, the transfer sequence is MSB first.

Suspend Interval

SUSPITV (SPIx_CTL[7:4]) provides a configurable suspend interval, 0.5 ~ 15.5 SPI clock periods, between two successive transaction words in Master mode. The definition of the suspend interval is the interval between the last clock edge of the preceding transaction word and the first clock edge of the following transaction word. The default value of SUSPITV is 0x3 (3.5 SPI clock cycles).

6.22.5.2 Automatic Slave Selection

In Master mode, if AUTOSS (SPIx_SSCTL[3]) is set, the slave selection signal will be generated automatically and output to the SPIx_SS pin according to whether SS (SPIx_SSCTL[0]) is enabled or not. The slave selection signal will be set to active state by the SPI controller when the SPI data transfer is started by writing to FIFO. It will be set to inactive state when SPI bus is idle. If SPI bus is not idle, i.e. TX FIFO, TX shift register or TX skew buffer is not empty, the slave selection signal will be set to inactive state between transactions if the value of SUSPITV (SPIx_CTL[7:4]) is greater than or equal to 3.

In Master mode, if the value of SUSPITV is less than 3 and the AUTOSS is set as 1, the slave selection signal will be kept at active state between two successive transactions.

If the AUTOSS bit is cleared, the slave selection output signal will be determined by the SS setting. The active state of the slave selection output signal is specified in SSACTPOL (SPIx_SSCTL[2]).

The duration between the slave selection signal active edge and the first SPI bus clock edge is 1 SPI bus clock cycle and the duration between the last SPI bus clock and the slave selection signal inactive edge is 1.5 SPI bus clock cycle.

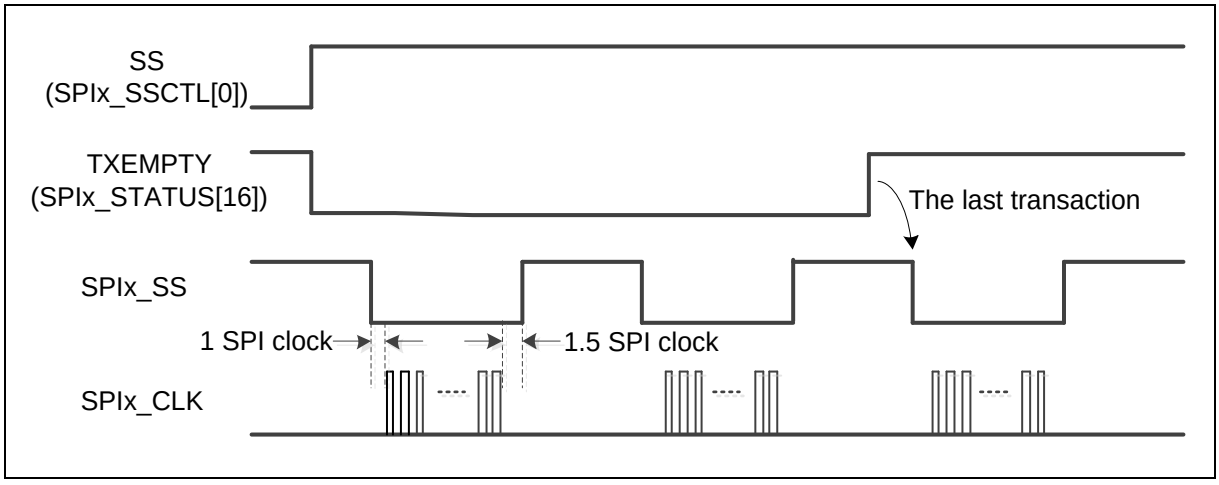


Figure 6.22-6 Automatic Slave Selection (SSACTPOL = 0, SUSPITV > 0x2)

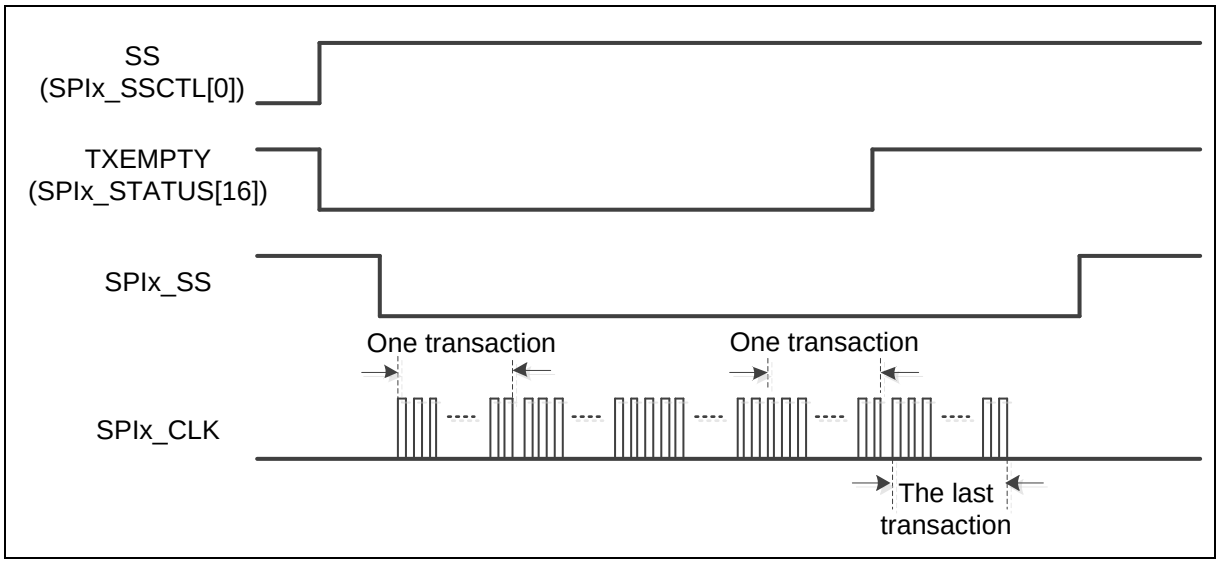


Figure 6.22-7 Automatic Slave Selection (SSACTPOL = 0, SUSPITV < 0x3)

6.22.5.3 Byte Reorder and Suspend Function

When the transfer is set as MSB first (LSB = 0) and the REORDER (SPIx_CTL[19]) is set to 1, the data stored in the TX buffer and RX buffer will be rearranged in the order as [Byte0, Byte1, Byte2, Byte3] in 32-bit transfer (DWIDTH = 0). The sequence of transmitted/received data will be Byte0, Byte1, Byte2, and then Byte3. If the DWIDTH is set as 24-bit transfer mode, the data in TX buffer and RX buffer will be rearranged as [unknown byte, Byte0, Byte1, Byte2]. The SPI controller will transmit/receive data with the sequence of Byte0, Byte1 and then Byte2. Each byte will be transmitted/received with MSB first. The rule of 16-bit mode is the same as above. Byte Reorder function is only available when DWIDTH is configured as 16, 24, and 32 bits.

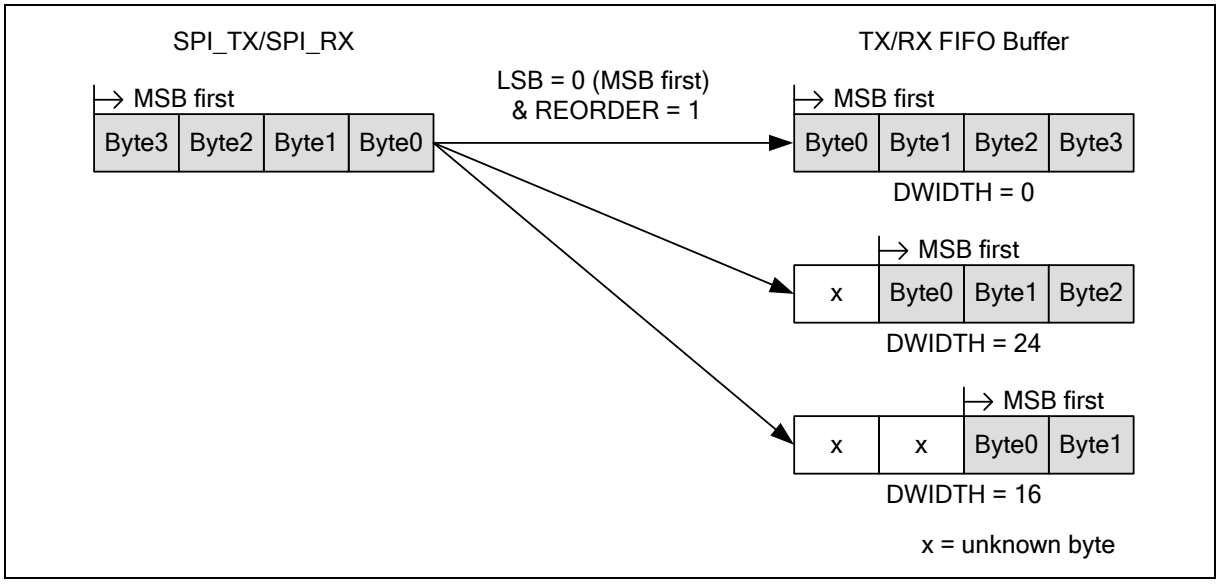


Figure 6.22-8 Byte Reorder Function

In Master mode, if REORDER (SPIx_CTL[19]) is set to 1, a suspend interval of 0.5 ~ 15.5 SPI clock periods will be inserted by hardware between two successive bytes in a transaction word. The suspend interval is configured in SUSPITV (SPIx_CTL[7:4]).

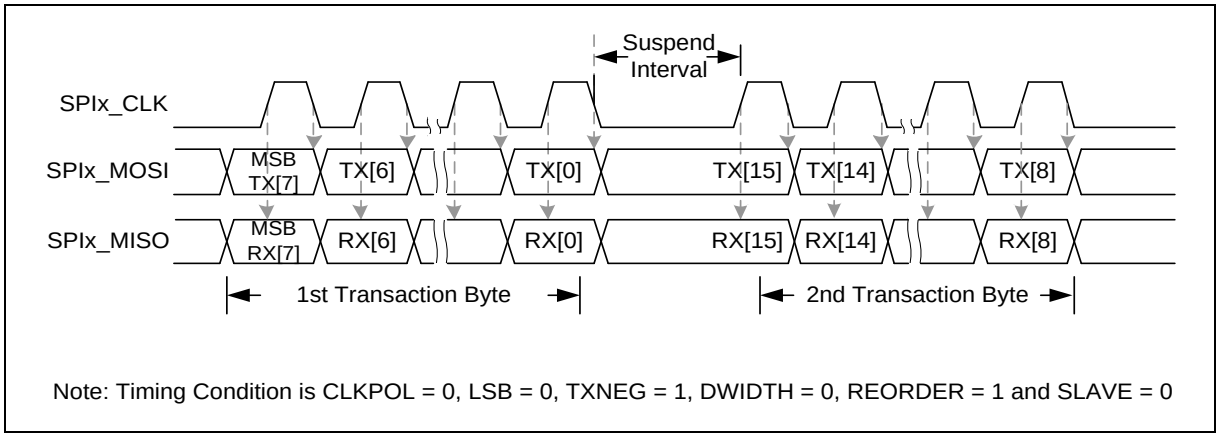


Figure 6.22-9 Timing Waveform for Byte Suspend

6.22.5.4 Half-Duplex Communication

The SPI controller can communicate in half-duplex mode by setting HALFDPX (SPIx_CTL[14]) bit. In half-duplex mode, there is only one data line for receiving or transmitting data direction which is defined by DATDIR (SPIx_CTL[20]). In half-duplex configuration, the SPIx_MISO pin is free for other applications and it can be configured as GPIO. Enabling or disabling the control bit HALFDPX (SPIx_CTL[14]) will produce TXFBCLR (SPIx_FIFOCTL[9]) and RXFBCLR (SPIx_FIFOCTL[8]) at the same time automatically.

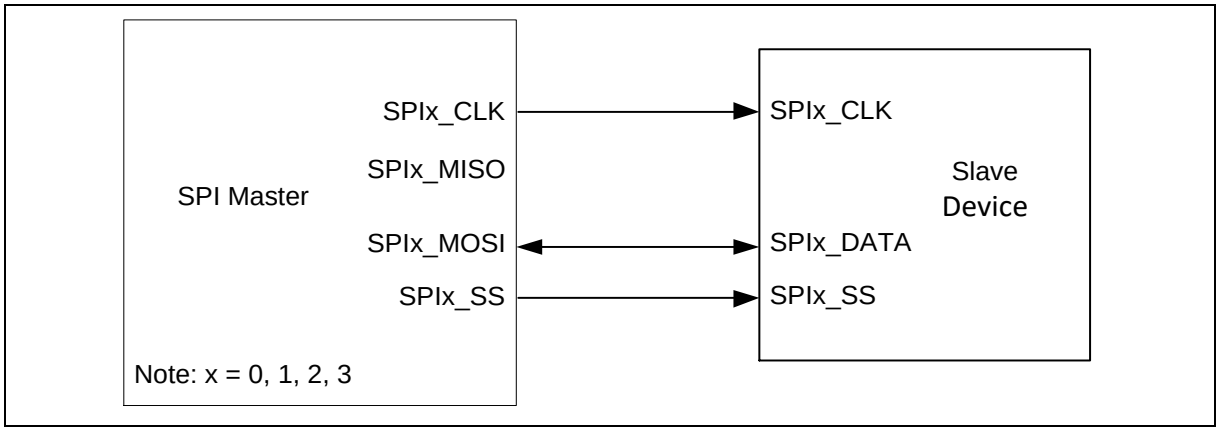


Figure 6.22-10 SPI Half-Duplex Master Mode Application Block Diagram

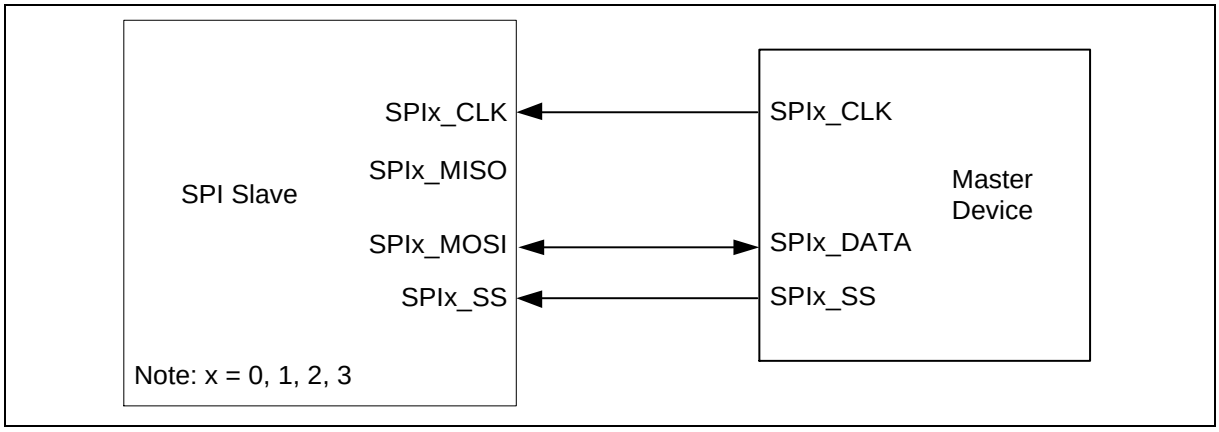


Figure 6.22-11 SPI Half-Duplex Slave Mode Application Block Diagram

6.22.5.5 Receive-Only Mode

In SPI Master device, it can communicate in receive-only mode by setting RXONLY (SPIx_CTL[15]). In this configuration, the SPI Master device will generate SPI bus clock continuously as long as the receive-only mode is enabled for receiving data bit from SPI slave device. If AUTOSS (SPIx_SSCTL[3]) is enabled in receive-only mode, SPI Master will keep activating the slave select signal.

The remaining SPIx_MOSI pin of SPI Master device is not used for communication and can be configured as GPIO. The status BUSY (SPIx_STATUS[0]) will be asserted in receive-only mode due to the generation of SPI bus clock. Entering this mode will produce the TXFBCLR (SPIx_FIFOCTL[9]) and RXFBCLR (SPIx_FIFOCTL[8]) at the same time automatically. When user enables this mode, the output SPI bus clock will be sent out after 6 peripheral clock cycles. In this mode, the data which has been written into transmit FIFO will be loaded into transmit shift register and sent out.

When user sets RXONLY (SPIx_CTL[15]) to enable, SPI RX data with data bit width of DWIDTH (SPIx_CTL[12:8]) will be received into RX FIFO and SPI clock will be sent to SPI slave device until RX FIFO is full.

For data bit width of 8~16 bits, the SPI master will send SPI output clock to SPI slave and receive RX data when RX FIFO counter RXCNT (SPIx_STATUS[27:24]) is less than or equal to 6.

For data bit width of 17~32 bits, the SPI master will send SPI output clock to SPI slave and receive RX data when RX FIFO counter RXCNT (SPIx_STATUS[27:24]) is less than or equal to 2.

6.22.5.6 Slave 3-Wire Mode

When SLV3WIRE (SPIx_SSCTL[4]) is set by software to enable the Slave 3-Wire mode, the SPI controller can work with no slave selection signal in Slave mode. The SLV3WIRE (SPIx_SSCTL[4]) only takes effect in Slave mode. Only three pins, SPIx_CLK, SPIx_MISO, and SPIx_MOSI, are required to communicate with a SPI master. The SPIx_SS pin can be configured as a GPIO. When the SLV3WIRE (SPIx_SSCTL[4]) is set to 1, the SPI slave will be ready to transmit/receive data after the SPIEN (SPIx_CTL[0]) is set to 1.

6.22.5.7 PDMA Transfer Function

SPI controller supports PDMA transfer function.

When TXPDMAEN (SPIx_PDMACTL[0]) is set to 1, the controller will issue request to PDMA controller to start the PDMA transmission process automatically.

When RXPDMAEN (SPIx_PDMACTL[1]) is set to 1, the controller will start the PDMA reception process. SPI controller will issue request to PDMA controller automatically when there is data in the RX FIFO buffer.

Note: SPI supports single request PDMA (Read/Write) only; burst request PDMA is not supported.

6.22.5.8 FIFO Buffer Operation

The SPI controllers are equipped with four 32-bit wide transmit and receive FIFO buffers. The data stored in the transmit FIFO buffer will be read and sent out by the transmission control logic. If the transmit FIFO buffer is full, the TXFULL (SPIx_STATUS[17]) will be set to 1. When the SPI transmission logic unit draws out the last datum of the transmit FIFO buffer, so that the transmit FIFO buffer is empty, the TXEMPTY (SPIx_STATUS[16]) will be set to 1. Note that the TXEMPTY (SPIx_STATUS[16]) flag is set to 1 while the last transaction is still in progress. In Master mode, the BUSY (SPIx_STATUS[0]) is set to 1 when the FIFO buffer is written any data or there is any transaction on the SPI bus. (e.g. the slave selection signal is active and the SPI controller is receiving data in Slave mode). It will set to 0 when the transmit FIFO is empty and the current transaction has done. Thus, the status of BUSY (SPIx_STATUS[0]) should be checked by software to make sure whether the SPI is in idle or not.

The receive control logic will store the SPI input data into the receive FIFO buffer. There are FIFO related status bits, like RXEMPTY (SPIx_STATUS[8]) and RXFULL (SPIx_STATUS[9]), to indicate the current status of RX FIFO buffer.

The transmitting and receiving threshold can be configured by setting TXTH (SPIx_FIFOCTL[30:28]) and RXTH (SPIx_FIFOCTL[26:24]). When the count of valid data stored in transmit FIFO buffer is less than or equal to TXTH (SPIx_FIFOCTL[30:28]) setting, TXTHIF (SPIx_STATUS[18]) will be set to 1. When the count of valid data stored in receive FIFO buffer is larger than RXTH (SPIx_FIFOCTL[26:24]) setting, RXTHIF (SPIx_STATUS[10]) will be set to 1.

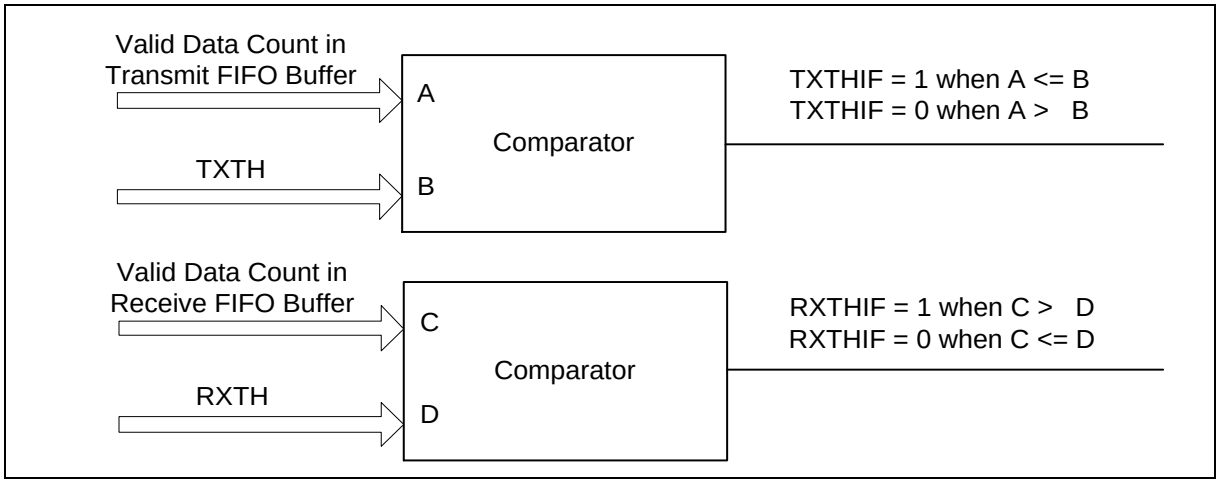


Figure 6.22-12 FIFO Threshold Comparator

In Master mode, when the first datum is written to the SPIx_TX register, the TXEMPTY flag (SPIx_STATUS[16]) will be cleared to 0. The transmission will start after 1 PCLK clock cycles and 6 peripheral clock cycles. User can write the next data into SPIx_TX register immediately. The SPI controller will insert a suspend interval between two successive transactions. The period of suspend interval is decided by the setting of SUSPITV (SPIx_CTL[7:4]). If the SUSPITV (SPIx_CTL[7:4]) equals 0, SPI controller can perform continuous transfer. User can write data into SPIx_TX register as long as the TXFULL (SPIx_STATUS[17]) is 0.

In the Example 1 of Figure 6.22-13, it indicates the updated condition of TXEMPTY (SPIx_STATUS[16]) and the relationship among the FIFO buffer, shift register and the skew buffer for 8~16 bits of data length. The TXEMPTY (SPIx_STATUS[16]) is set to 0 when the Data0 is written into the FIFO buffer. The Data0 will be loaded into the shift register by the core logic and the TXEMPTY (SPIx_STATUS[16]) will be to 1. The Data0 in shift register will be shift into skew buffer by bit for transmission until the transfer is done.

In the Example 2 of Figure 6.22-13, it indicates the updated condition of TXFULL (SPIx_STATUS[17]) when there are 8 data in the FIFO buffer and the next data of Data9 is not written into the FIFO buffer when the TXFULL = 1.

In the Example 1 of Figure 6.22-14, it indicates the updated condition of TXEMPTY (SPIx_STATUS[16]) and the relationship among the FIFO buffer, shift register and the skew buffer for 17~32 bits of data length.

In the Example 2 of Figure 6.22-14, it indicates the updated condition of TXFULL (SPIx_STATUS[17]) when there are 4 data in the FIFO buffer and the next data of Data5 is not written into the FIFO buffer when the TXFULL = 1.

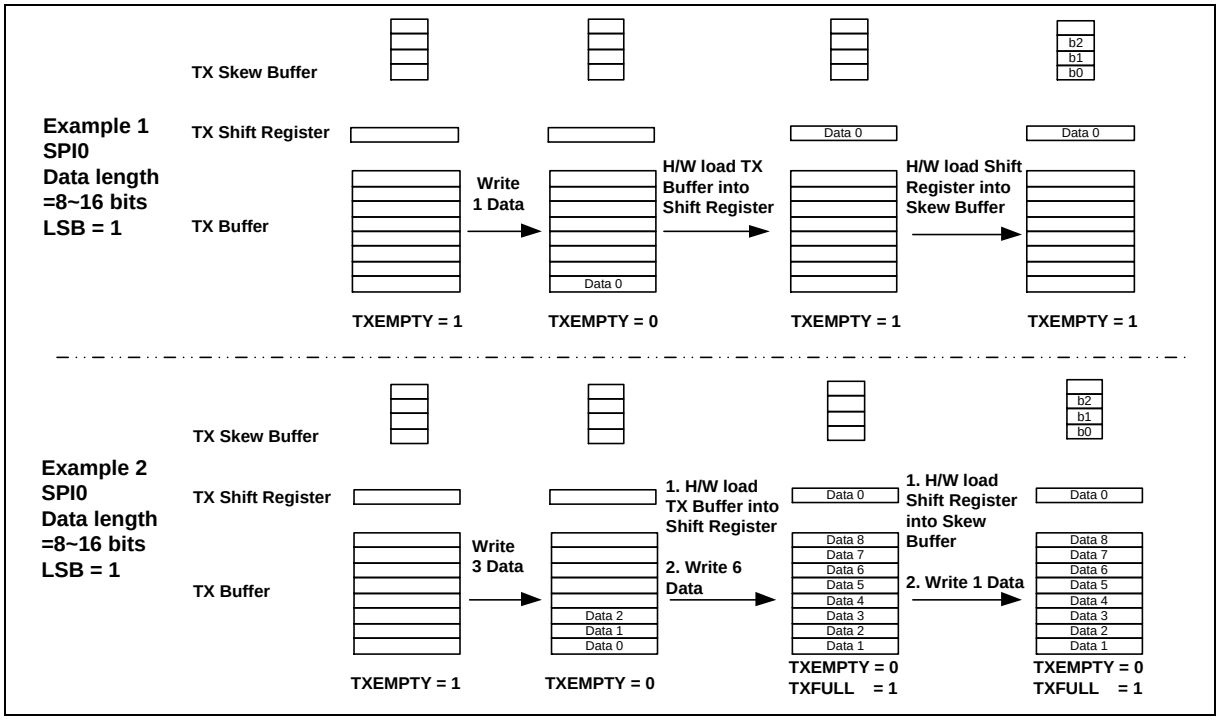


Figure 6.22-13 Transmit FIFO Buffer Example for 8~16 bits of data length

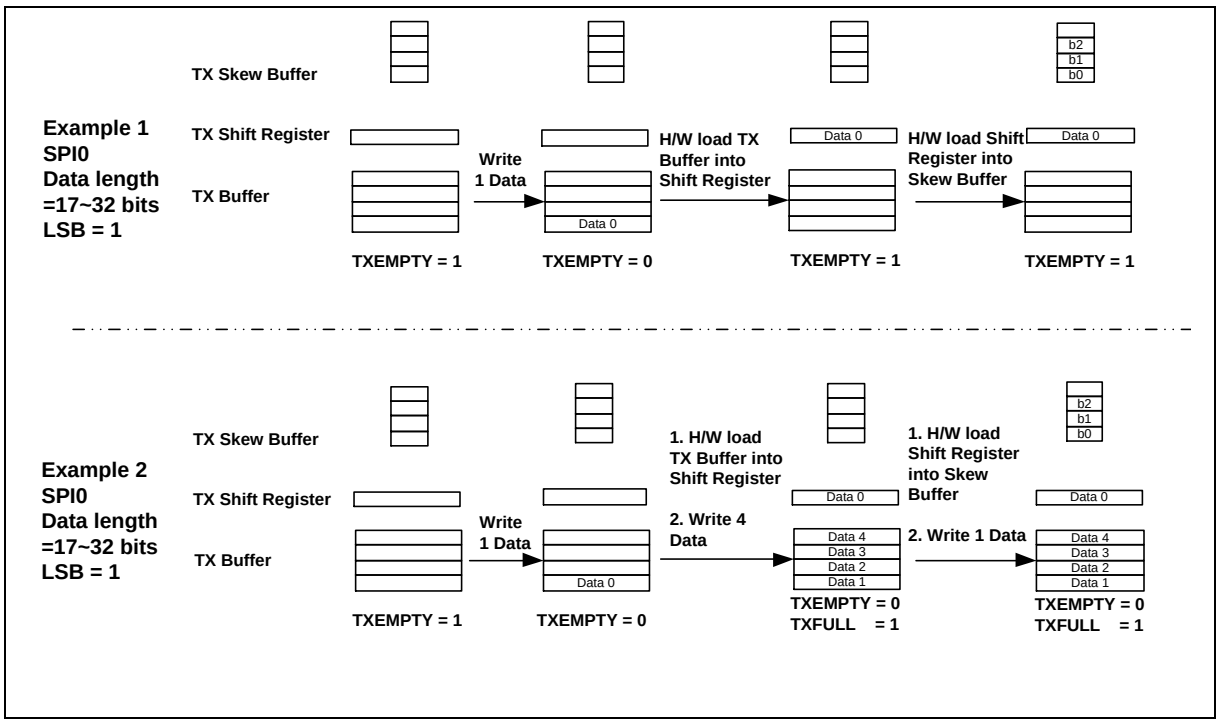


Figure 6.22-14 Transmit FIFO Buffer Example for 17~32 bits of data length

The subsequent transactions will be triggered automatically if the transmitted data are updated in time. If the SPIx_TX register is not updated after all data transfers are done, the transfer will stop.

In Master mode, during receiving operation, the serial data are received from SPIx_MISO pin and

stored to receive FIFO buffer.

The received data (Data0's b0, b1, …b31) is stored into skew buffer first according the serial clock (SPIx_CLK) and then it is shift into the shift register bit by bit. The core logic will load the data in shift register into FIFO buffer when the received data bit count reach the value of DWIDTH (SPIx_CTL[12:8]).

The RXEMPTY (SPIx_STATUS[8]) will be cleared to 0 while the receive FIFO buffer contains unread data (see the Example 1 of Receive FIFO Buffer Example for 16 bits of Data Length in Figure 6.22-15). The received data can be read by software from SPIx_RX register as long as the RXEMPTY (SPIx_STATUS[8]) is 0. If the receive FIFO buffer contains 8 unread data, the RXFULL (SPIx_STATUS[9]) will be set to 1 (see the Example 2 of Receive FIFO Buffer Example for 16 bits of Data Length in Figure 6.22-15).

The RXEMPTY (SPIx_STATUS[8]) will be cleared to 0 while the receive FIFO buffer contains unread data (see the Example 1 of Receive FIFO Buffer Example for 32 bits of Data Length in Figure 6.22-16). The received data can be read by software from SPIx_RX register as long as the RXEMPTY (SPIx_STATUS[8]) is 0. If the receive FIFO buffer contains 4 unread data, the RXFULL (SPIx_STATUS[9]) will be set to 1 (see the Example 2 of Receive FIFO Buffer Example for 32 bits of Data Length in Figure 6.22-16).

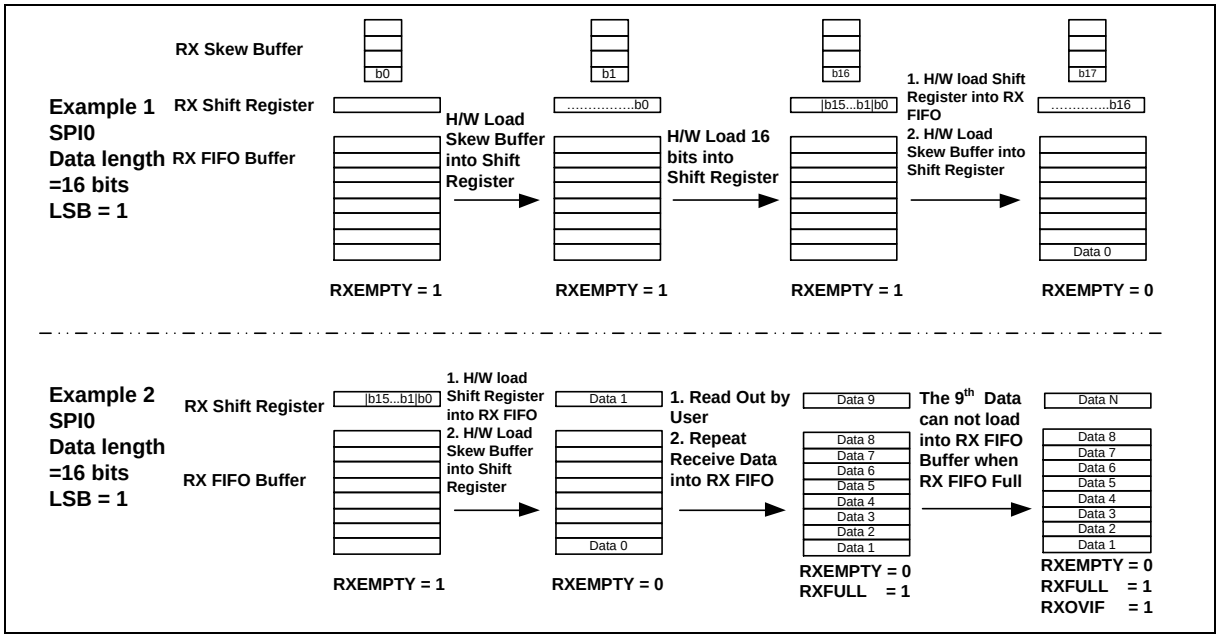


Figure 6.22-15 Receive FIFO Buffer Example for 16 bits of Data Length

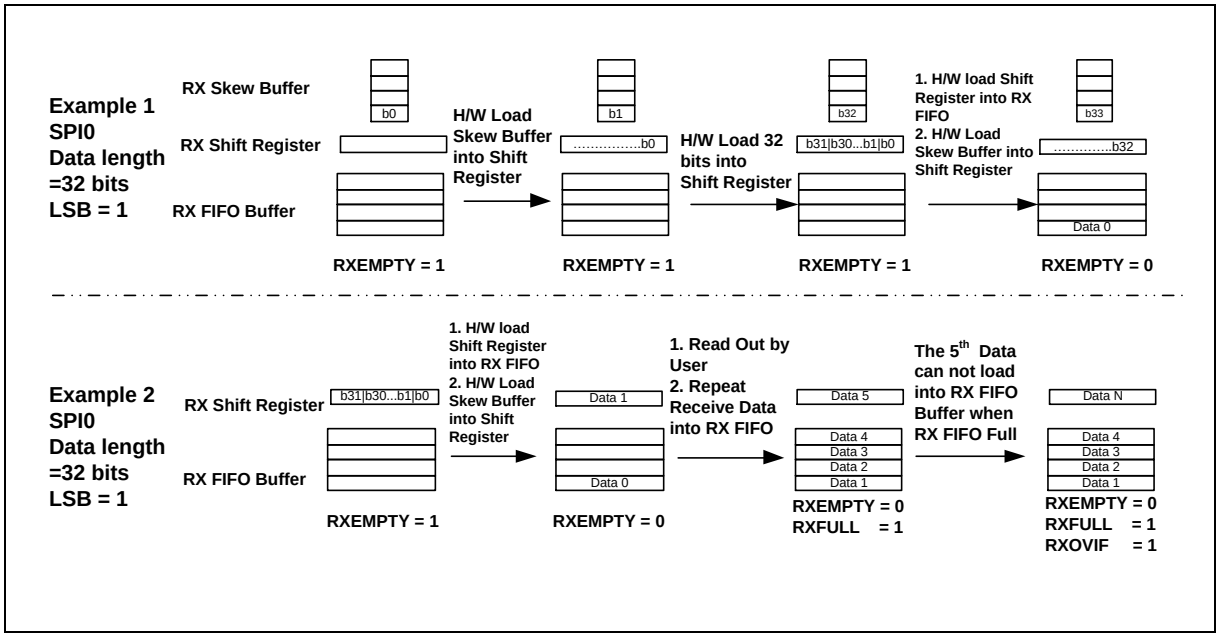


Figure 6.22-16 Receive FIFO Buffer Example for 32 bits of Data Length

In Slave mode, during transmission operation, when data is written to the SPIx_TX register by software, the data will be loaded into transmit FIFO buffer and the TXEMPTY (SPIx_STATUS[16]) will be set to 0. The transmission will start when the slave device receives clock signal from master. Data can be written to SPIx_TX register as long as the TXFULL (SPIx_STATUS[17]) is 0. After all data have been drawn out by the SPI transmission logic unit and the SPIx_TX register is not updated by software, the TXEMPTY (SPIx_STATUS[16]) will be set to 1.

If there is no any data written to the SPIx_TX register, the transmit underflow interrupt flag, TXUFIF (SPIx_STATUS[19]) will be set to 1 when the slave selection signal is active. The output data will be held by TXUFPOL (SPIx_FIFOCTL[6]) setting during this transfer until the slave selection signal goes to inactive state. When the transmit underflow event occurs, the slave under run interrupt flag, SLVURIF (SPIx_STATUS[7]), will be set to 1 as SPIx_SS goes to inactive state.

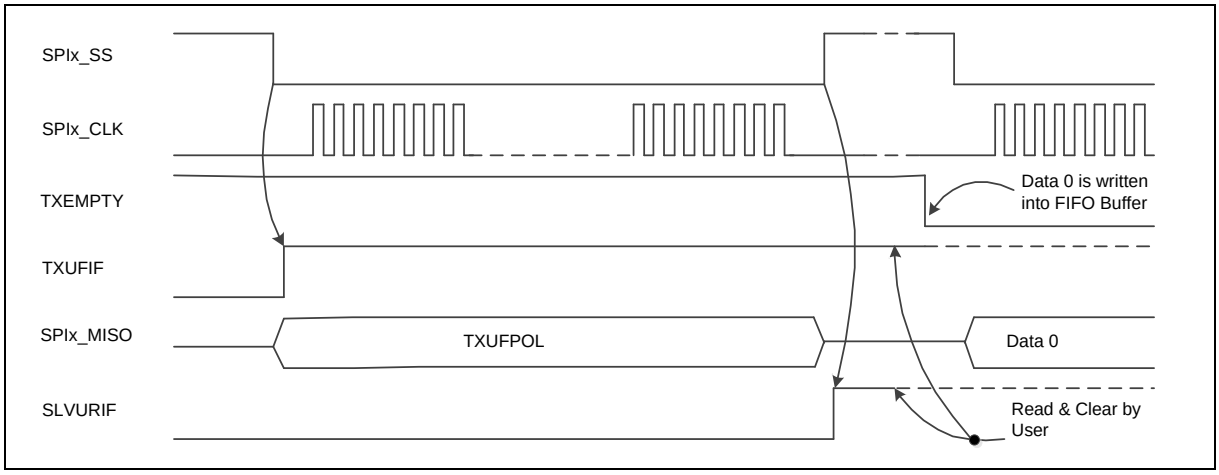


Figure 6.22-17 TX Underflow Event and Slave Under Run Event

In SPI Slave 3-Wire mode, the first 2-bit data is un-predicted (kept on the level of last bit in previously transfer) if the data is written into TX FIFO among 3 peripheral clock cycles before the SPI bus clock is

presented. The other bits are held by TXUFPOL (SPIx_FIFOCTL[6]) because there is TX underflow event. The written data will be transmitted in the next transfer.

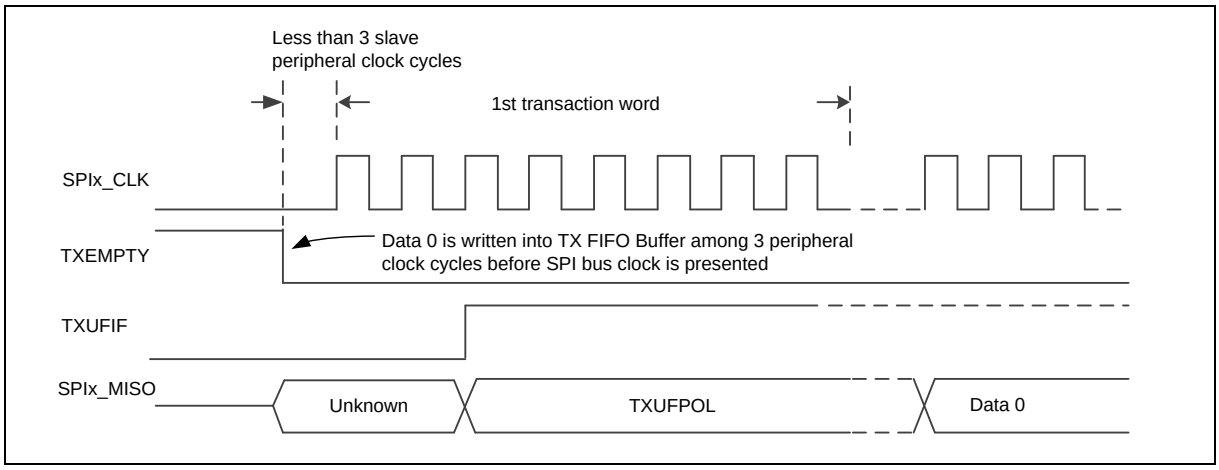


Figure 6.22-18 TX Underflow Event (SPI Slave 3-Wire Mode Enabled)

In Slave mode, during receiving operation, the serial data is received from SPIx_MOSI pin and stored to SPIx_RX register. The reception mechanism is similar to Master mode reception operation. If the receive FIFO buffer contains 4 unread data, the RXFULL (SPIx_STATUS[9]) will be set to 1 and the RXOVIF (SPIx_STATUS[11]) will be set to 1 if there is more serial data received from SPIx_MOSI and follow-up data will be dropped (refer to the Receive FIFO Buffer Example figure).

If the receive bit count mismatches with the DWIDTH (SPIx_CTL[12:8]) when the slave selection line goes to inactive state, the SLVBEIF (SPIx_STATUS[6]) will be set to 1 in SPI Slave mode. RX data will be written into RX FIFO and SLVBENUM (SPIx_STATUS2[29:24]) will be updated when SLVBERX (SPIx_FIFOCTL[10]) is enabled and SPI slave bit count error event happened. RX data will be dropped and SLVBENUM (SPIx_STATUS2[29:24]) will be fixed to 0x0 when the control register SLVBERX (SPIx_FIFOCTL[10]) is disabled and SPI slave bit count error event happened. The status register SLVBENUM (SPIx_STATUS2[29:24]) indicates that effective bit number of uncompleted RX data when SPI slave bit count error happened. In Figure 6.22-19, the timing diagram of SPI slave bit count error and SLVBENUM is shown, and SLVBENUM will be updated to 0x8 when SPI slave device receives 8 bits data and DWIDTH (SPIx_CTL[12:8]) is set to 16 bits at SPI bit count error event. The SPI slave bit count error event (SLVBEIF) will generate RX FIFO write operation pulse to write RX data from RX shift register to RX FIFO, and RX FIFO count (RXCNT) will be updated to one.

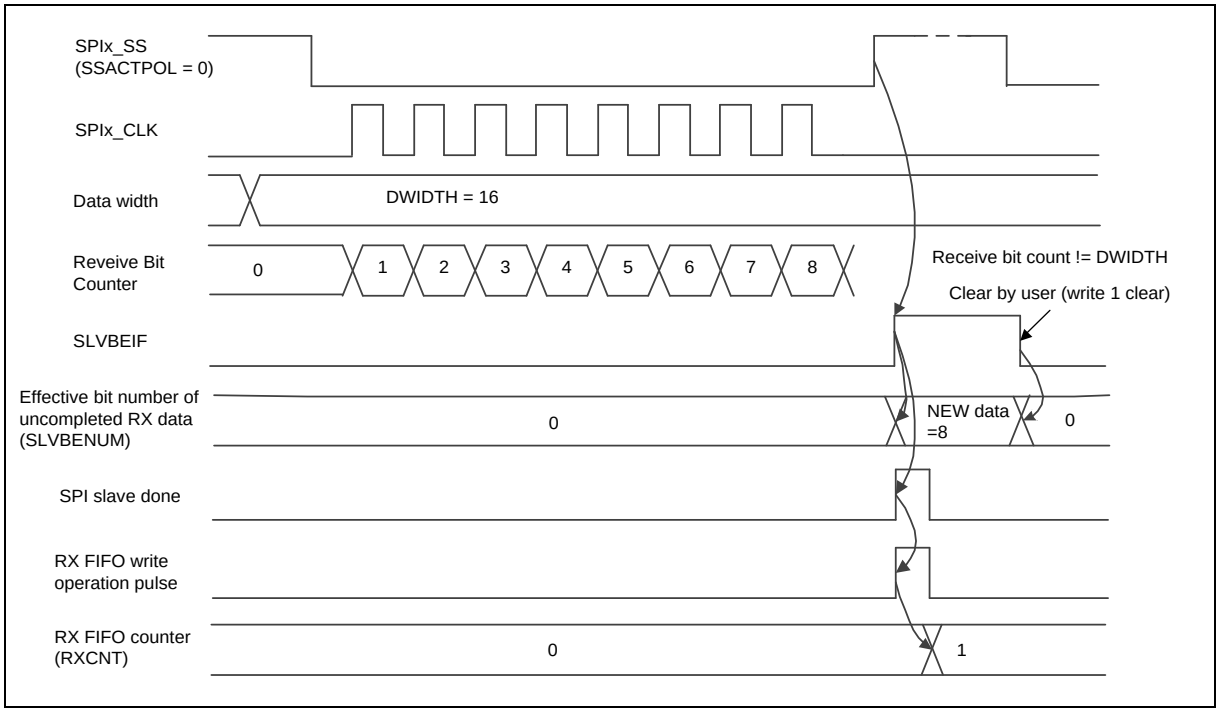


Figure 6.22-19 Slave Mode Bit Count Error and Effective Bit Number of Uncompleted RX Data

A receive time-out function is built-in in this controller. When the receive FIFO is not empty and no read operation in receive FIFO over 64 SPI peripheral clock periods in Master mode or over 576 SPI peripheral clock periods in Slave mode, the receive time-out occurs and the RXTOIF (SPIx_STATUS[12]) will be set to 1. When the receive FIFO is read by user, the time-out status will be cleared automatically.

6.22.5.9 Interrupt

- SPI unit transfer interrupt

  As the SPI controller finishes a unit transfer, the unit transfer interrupt flag UNITIF (SPIx_STATUS[1]) will be set to 1. The unit transfer interrupt event will generate an interrupt to CPU if the unit transfer interrupt enable bit UNITIEN (SPIx_CTL[17]) is set. The unit transfer interrupt flag can be cleared only by writing 1 to it.

- SPI slave selection active/inactive interrupt

  In Slave mode, the slave selection active/inactive interrupt flag, SSACTIF (SPIx_STATUS[2]) and SSINAIF (SPIx_STATUS[3]), will be set to 1 when the SPIEN (SPIx_CTL[0]) and SLAVE (SPIx_CTL[18]) are set to 1 and the slave selection signal goes to active/inactive state. The SPI controller will issue an interrupt if the SSACTIEN (SPIx_SSCTL[12]) or SSINAIEN (SPIx_SSCTL[13]), is set to 1.

- Slave bit count error interrupt

  In Slave mode, if the transmit/receive bit count mismatch with the DWIDTH (SPIx_CTL[12:8]) when the slave selection line goes to inactive state, the SLVBEIF (SPIx_STATUS[6]) will be set to 1. The uncompleted transaction will be dropped from TX shift register. When the control register SLVBERX (SPIx_FIFOCTL[10]) is disabled and SPI slave bit count error event happened, the uncompleted transaction data will be dropped from RX shift registers. When control register SLVBERX (SPIx_FIFOCTL[10]) is enabled and SPI slave bit count error event happened, the uncompleted transaction data will be written from RX shift registers into RX FIFO. The SPI controller will issue an

interrupt if the SLVBEIEN (SPIx_SSCTL[8]) is set to 1.

Note: If the slave selection signal is active but there is no any serial clock input, the SLVBEIF (SPIx_STATUS[6]) will be set to 1 when the slave selection signal goes to inactive state.

- TX underflow interrupt

  In SPI Slave mode, if there is no any data is written to the SPIx_TX register, the TXUFIF (SPIx_STATUS[19]) will be set to 1 when the slave selection signal is active. The SPI controller will issue a TX underflow interrupt if the TXUFIEN (SPIx_FIFOCTL[7]) is set to 1.

  Note: If underflow event occurs in SPI Slave mode, there are two conditions which make SPI Slave mode return to idle state and then goes for next transfer: (1) set TXRST to 1 (2) slave select signal is changed to inactive state while SLV3WIRE=0.

- Slave TX under run interrupt

  If the TX underflow event occurs, the SLVURIF (SPIx_STATUS[7]) will be set to 1 when SPIx_SS goes to inactive state. The SPI controller will issue a TX under run interrupt if the SLVURIEN (SPIx_SSCTL[9]) is set to 1.

  Note: In Slave 3-Wire mode, the slave selection signal is considered active all the time so that user shall poll the TXUFIF (SPIx_STATUS[19]) to know if there is TX underflow event or not.

- Receive Overrun interrupt

  In Slave mode, if the receive FIFO buffer contains 4 unread data, the RXFULL (SPIx_STATUS[9]) will be set to 1 and the RXOVIF (SPIx_STATUS[11]) will be set to 1 if there is more serial data received from SPI bus and follow-up data will be dropped. The SPI controller will issue an interrupt if the RXOVIEN (SPIx_FIFOCTL[5]) is set to 1.

- Receive FIFO time-out interrupt

  If there is a received data in the FIFO buffer and it is not read by software over 64 SPI peripheral clock periods in Master mode or over 576 SPI peripheral clock periods in Slave mode, it will send a RX time-out interrupt to the system if the RX time-out interrupt enable bit, RXTOIEN (SPIx_FIFOCTL[4]), is set to 1.

- Transmit FIFO interrupt

  In FIFO mode, if the valid data count of the transmit FIFO buffer is less than or equal to the setting value of TXTH (SPIx_FIFOCTL[30:28]), the transmit FIFO interrupt flag TXTHIF (SPIx_STATUS[18]) will be set to 1. The SPI controller will generate a transmit FIFO interrupt to the system if the transmit FIFO interrupt enable bit, TXTHIEN (SPIx_FIFOCTL[3]), is set to 1.

- Receive FIFO interrupt

  In FIFO mode, if the valid data count of the receive FIFO buffer is larger than the setting value of RXTH (SPIx_FIFOCTL[26:24]), the receive FIFO interrupt flag RXTHIF (SPIx_STATUS[10]) will be set to 1. The SPI controller will generate a receive FIFO interrupt to the system if the receive FIFO interrupt enable bit, RXTHIEN (SPIx_FIFOCTL[2]), is set to 1.

6.22.5.10 I^2S Mode

The SPI0~SPI3 controllers support I^2S mode with PCM mode A, PCM mode B, MSB justified and I^2S data format. The bit count of an audio channel is determined by WDWIDTH (SPIx_I2SCTL[5:4]). The transfer sequence is always first from the most significant bit, MSB. Data are read on rising clock edge and are driven on falling clock edge.

In I²S data format, the MSB is sent and latched on the second clock of an audio channel. The I2Sx_LRCLK signal indicates which audio channel is in transferring.

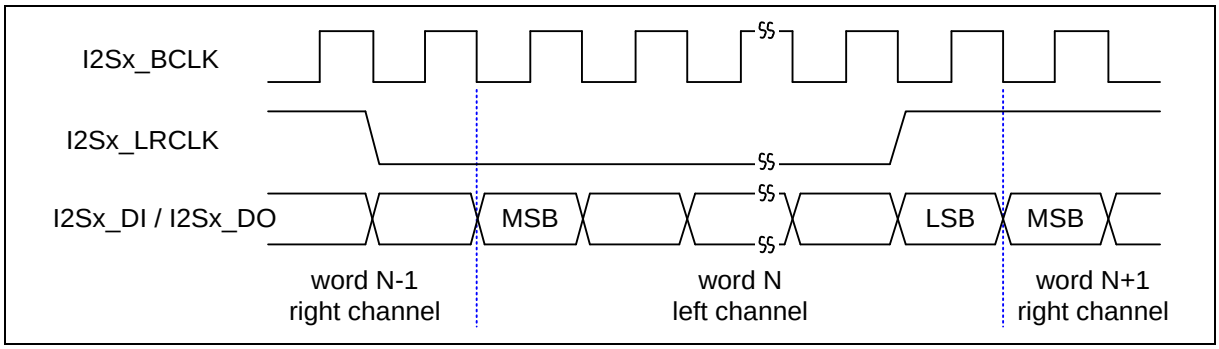


Figure 6.22-20 I²S Data Format Timing Diagram

In MSB justified data format, the MSB is sent and latched on the first clock of an audio channel.

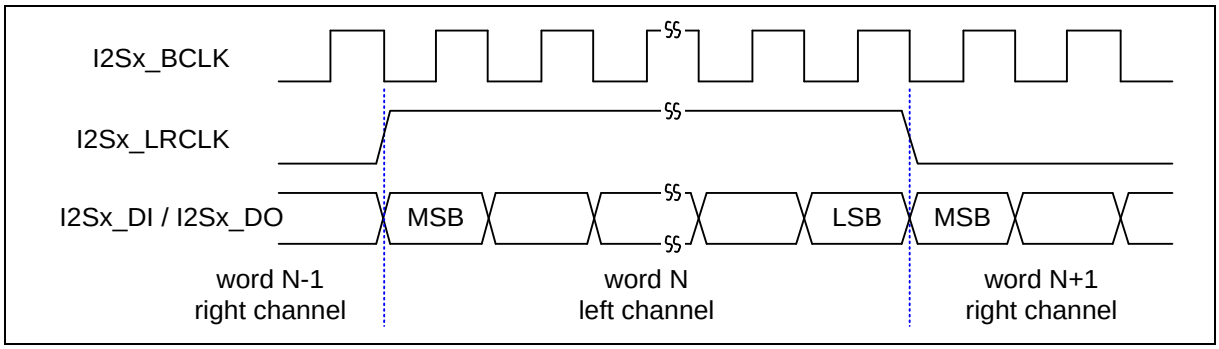


Figure 6.22-21 MSB Justified Data Format Timing Diagram

The I2Sx_LRCLK signal also supports PCM mode A and PCM mode B. The I2Sx_LRCLK signal in PCM mode indicates the beginning of an audio frame.

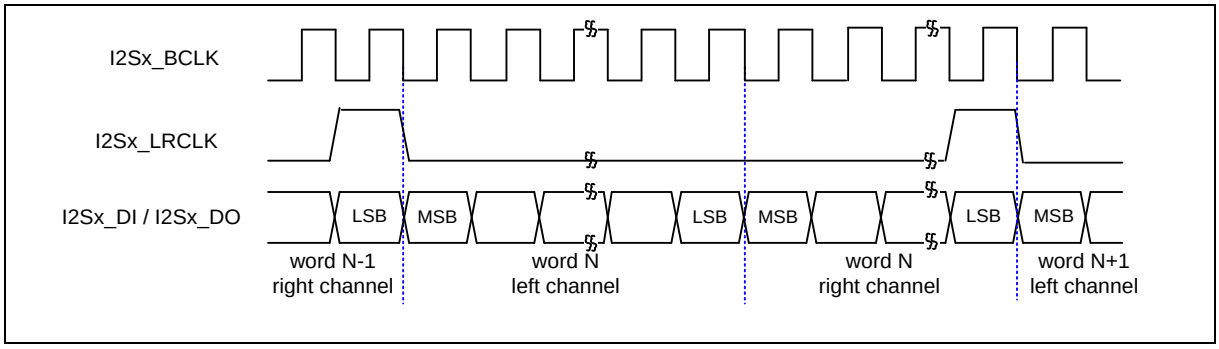


Figure 6.22-22 PCM Mode A Timing Diagram

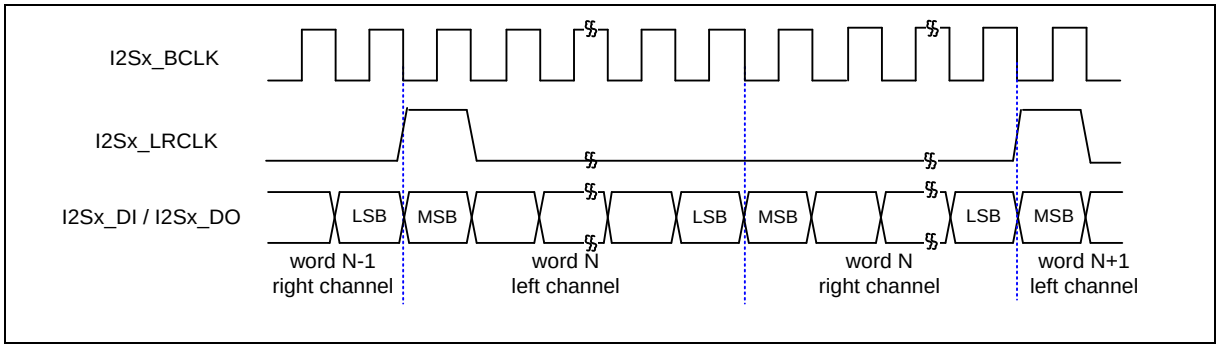


Figure 6.22-23 PCM Mode B Timing Diagram

6.22.5.11 I²S Mode FIFO Operation

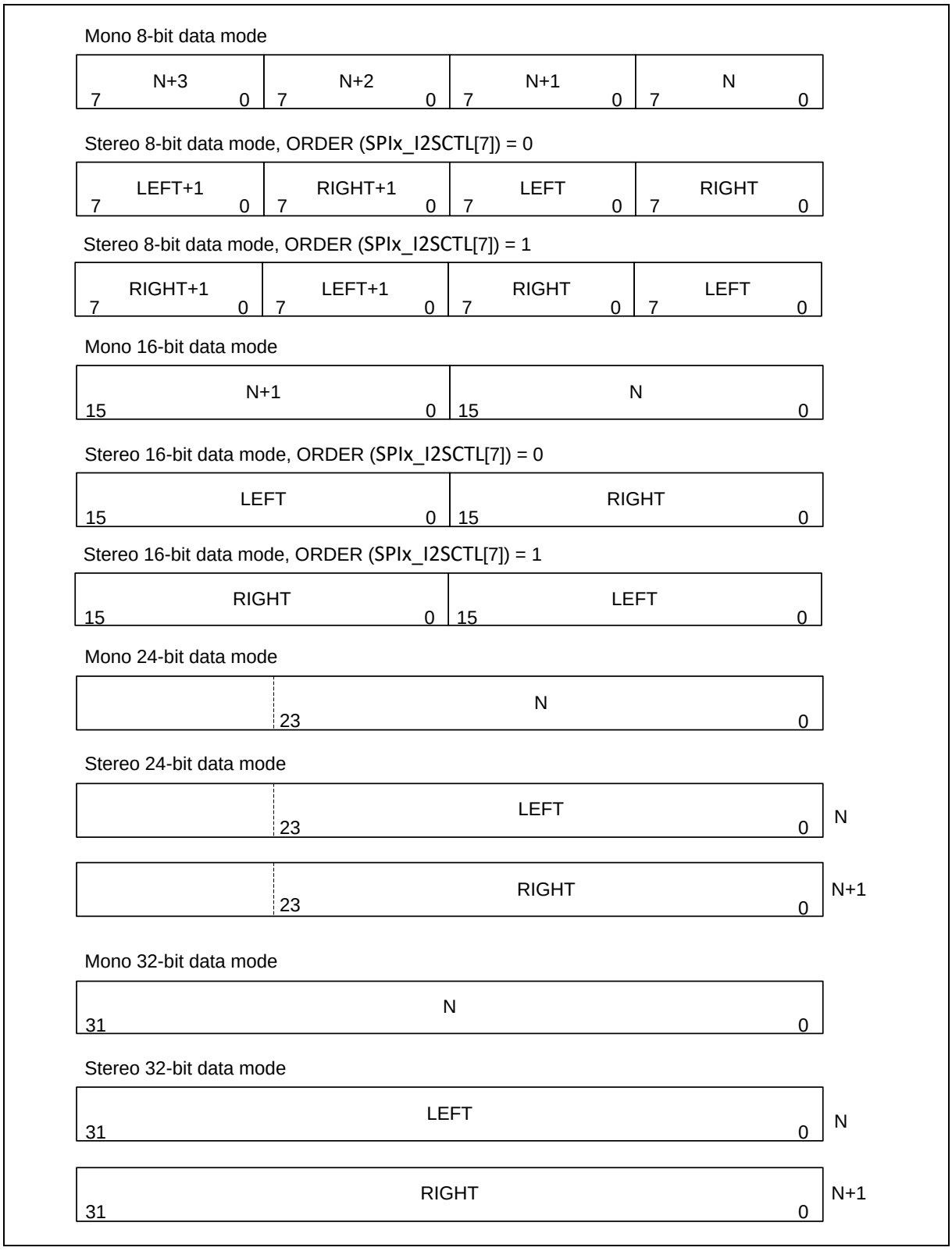


Figure 6.22-24 FIFO Contents for Various I²S Modes

6.22.5.12 Dummy Data Number for I²S / PCM Master mode and Monaural Mode

Before I²S / PCM master starts to send TX data to external slave device, set control registers I2SEN (SPIx_I2SCTL[0]) enable, TXEN (SPIx_I2SCTL[1]) enable, and write TX data to TX FIFO. After master sends dummy data (data with zero value) to external slave device, master will send TX FIFO data to external slave device. Table 6.22-2 shows number of dummy data for I²S / PCM master monaural mode, and the unit of dummy data number is L channel + R channel.

Data Width (SPIx_I2SCTL[5:4])	Dummy Data Number (Unit = L Channel + R Channel)
8 bits	0
16 bits	0
24 bits	1
32 bits	1

Table 6.22-2 Dummy Data Number for I²S / PCM Master Mode and Monaural Mode

6.22.5.13 Dummy Data Number for I²S / PCM Master mode and Stereo Mode

Before I²S / PCM master starts to send TX data to external slave device, set control registers I2SEN (SPIx_I2SCTL[0]) enable, TXEN (SPIx_I2SCTL[1]) enable, and write TX data to TX FIFO. After master sends dummy data (data with zero value) to external slave device, master will send TX FIFO data to external slave device. Table 6.22-3 shows number of dummy data for I²S / PCM master stereo mode, and the unit of dummy data number is L channel + R channel.

Data Width (SPIx_I2SCTL[5:4])	Dummy Data Number (Unit = L Channel + R Channel)
8 bits	0
16 bits	0
24 bits	1
32 bits	1

Table 6.22-3 Dummy Data Number for I²S / PCM Master Mode and Stereo Mode

6.22.5.14 Dummy Data Number for I²S Slave mode and Monaural Mode

Before I²S slave starts to send TX data to external master device, set control registers I2SEN (SPIx_I2SCTL[0]) enable, TXEN (SPIx_I2SCTL[1]) enable, SLAVE mode (SPIx_I2SCTL[8]), and write TX data to TX FIFO. After slave sends dummy data (data with zero value) to external master device, slave will send TX FIFO data to external master device. Table 6.22-4 shows number of dummy data for I²S slave monaural mode, and the unit of dummy data number is L channel + R channel.

Data Width (SPIx_I2SCTL[5:4])	Dummy Data Number (Unit = L Channel + R Channel)
8 bits	3
16 bits	2
24 bits	2
32 bits	2

Table 6.22-4 Dummy Data Number for I²S Slave Mode and Monaural Mode

6.22.5.15 Dummy Data Number for PCM Slave mode and Monaural Mode

Before PCM slave starts to send TX data to external master device, set control registers I2SEN (SPIx_I2SCTL[0]) enable, TXEN (SPIx_I2SCTL[1]) enable, SLAVE mode (SPIx_I2SCTL[8]), and write TX data to TX FIFO. After slave sends dummy data (data with zero value) to external master device, slave will send TX FIFO data to external master device. Table 6.22-5 shows number of dummy data for PCM slave monaural mode, and the unit of dummy data number is L channel + R channel.

Data Width (SPIx_I2SCTL[5:4])	Dummy Data Number (Unit = L Channel + R Channel)
8 bits	2
16 bits	1
24 bits	1
32 bits	1

Table 6.22-5 Dummy Data Number for PCM Slave Mode and Monaural Mode

6.22.5.16 Dummy Data Number for I^2S Slave mode and Stereo Mode

Before I^2S slave starts to send TX data to external master device, set control registers I2SEN (SPIx_I2SCTL[0]) enable, TXEN (SPIx_I2SCTL[1]) enable, SLAVE mode (SPIx_I2SCTL[8]), and write TX data to TX FIFO. After slave sends dummy data (data with zero value) to external master device, slave will send TX FIFO data to external master device. Table 6.22-6 shows number of dummy data for I^2S slave stereo mode, and the unit of dummy data number is L channel + R channel.

Data Width (SPIx_I2SCTL[5:4])	Dummy Data Number (Unit = L Channel + R Channel)
8 bits	3
16 bits	2
24 bits	2
32 bits	2

Table 6.22-6 Dummy Data Number for I^2S Slave Mode and Stereo Mode

6.22.5.17 Dummy Data Number for PCM Slave mode and Stereo Mode

Before PCM slave starts to send TX data to external master device, set control registers I2SEN (SPIx_I2SCTL[0]) enable, TXEN (SPIx_I2SCTL[1]) enable, SLAVE mode (SPIx_I2SCTL[8]), and write TX data to TX FIFO. After slave sends dummy data (data with zero value) to external master device, slave will send TX FIFO data to external master device. Table 6.22-7 shows number of dummy data for PCM slave stereo mode, and the unit of dummy data number is L channel + R channel.

Data Width (SPIx_I2SCTL[5:4])	Dummy Data Number (Unit = L Channel + R Channel)
8 bits	2
16 bits	1
24 bits	1
32 bits	1

Table 6.22-7 Dummy Data Number for PCM Slave Mode and Stereo Mode

6.22.6 Timing Diagram

The active state of slave selection signal can be defined by setting the SSACTPOL (SPIx_SSCTL[2]). The SPI clock which is in idle state can be configured as high or low state by setting the CLKPOL (SPIx_CTL[3]). It also provides the bit length of a transaction word in DWIDTH (SPIx_CTL[12:8]), and transmitting/receiving data from MSB or LSB first in LSB (SPIx_CTL[13]). User can also select which edge of SPI clock to transmit/receive data in TXNEG/RXNEG (SPIx_CTL[2:1]). Four SPI timing diagrams for master/slave operations and the related settings are shown below.

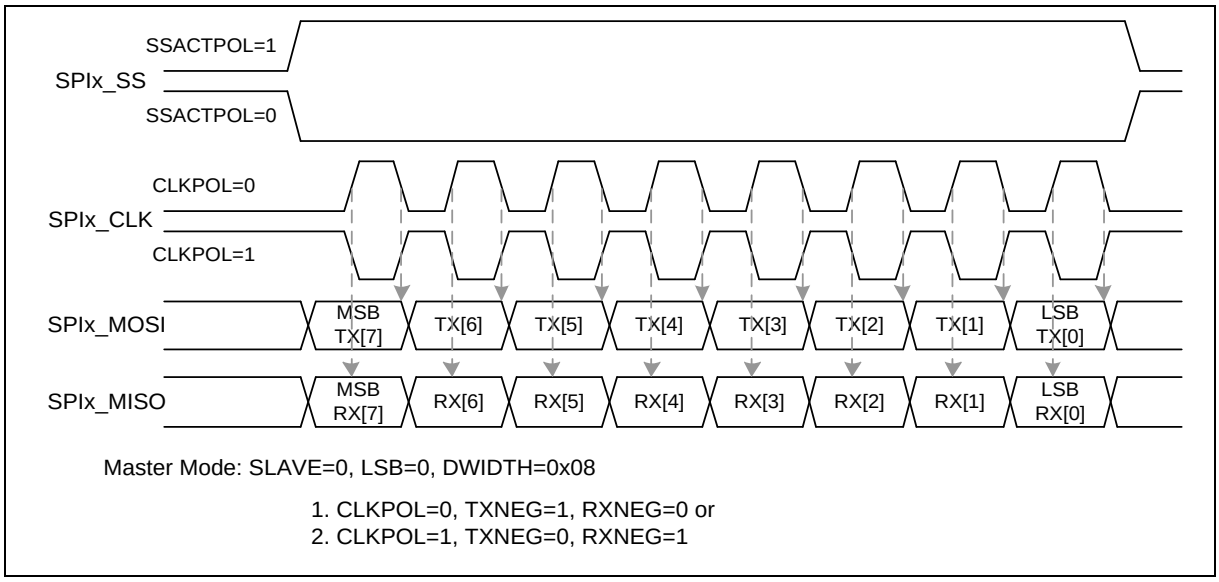


Figure 6.22-25 SPI Timing in Master Mode

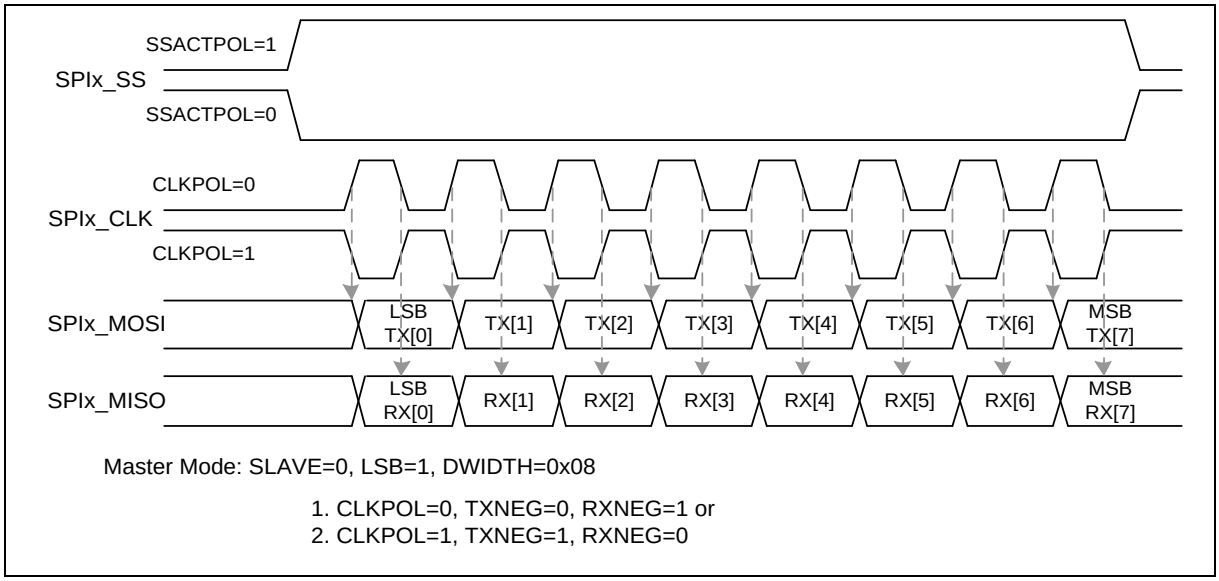


Figure 6.22-26 SPI Timing in Master Mode (Alternate Phase of SPIx_CLK)

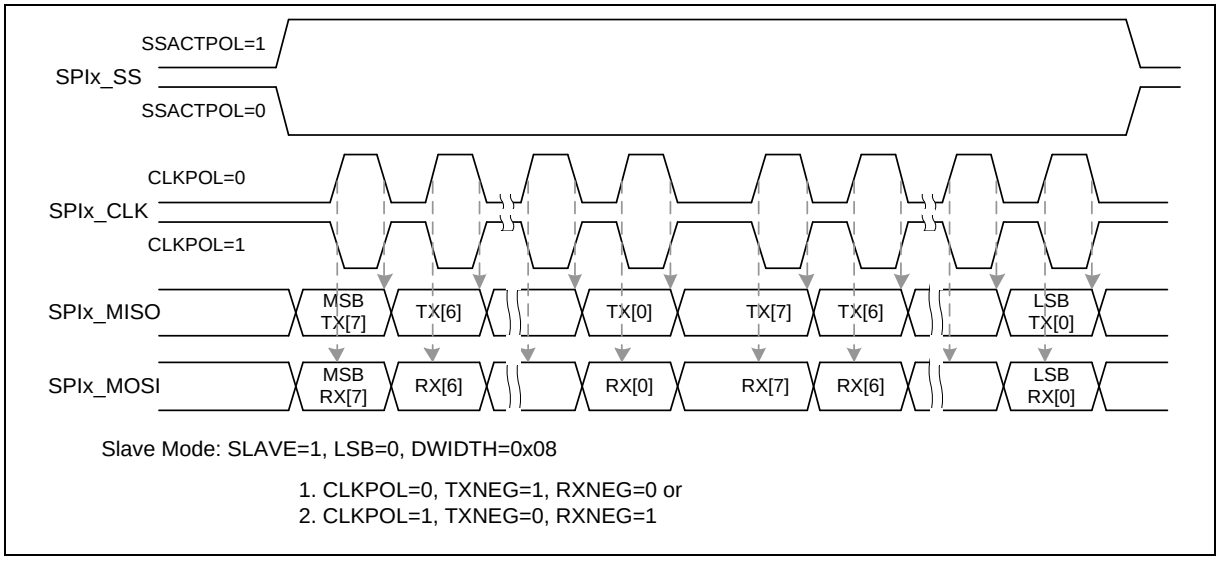


Figure 6.22-27 SPI Timing in Slave Mode

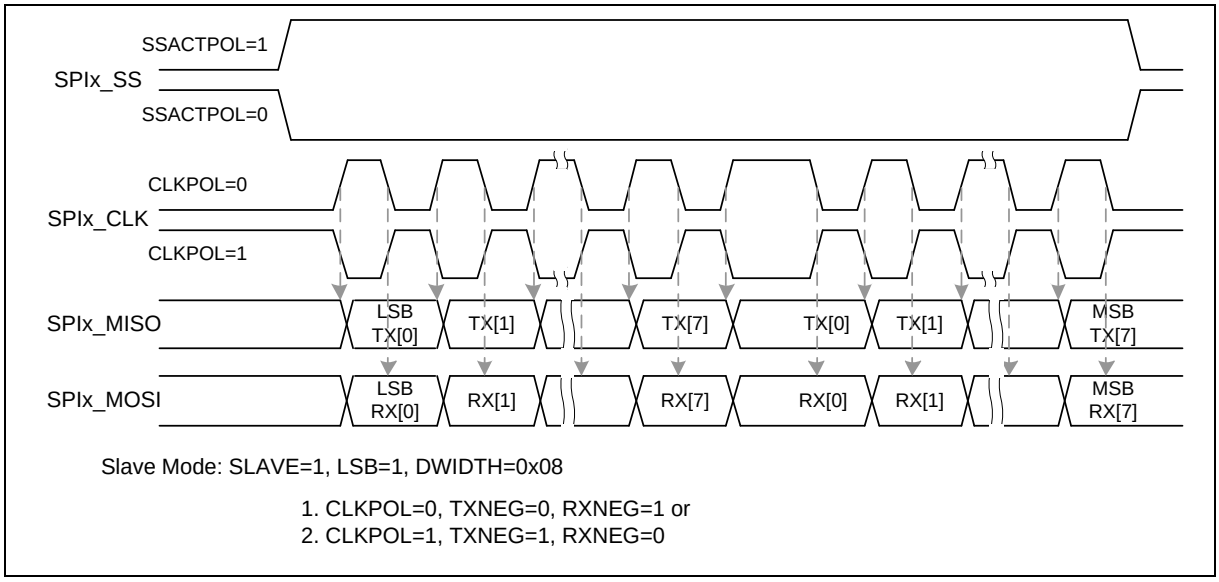


Figure 6.22-28 SPI Timing in Slave Mode (Alternate Phase of SPIx_CLK)

6.22.7 Programming Examples

Example 1:

The SPI controller is set as a full-duplex master to access an off-chip slave device with the following specifications:

- Data bit is latched on positive edge of SPI bus clock.
- Data bit is driven on negative edge of SPI bus clock.
- Data is transferred from MSB first.
- SPI bus clock is idle at low state.
- Only one byte of data to be transmitted/received in a transaction.

- Uses the SPI slave select pin to connect with an off-chip slave device. The slave selection signal is active low.

The operation flow is as follows:

1. Set DIVIDER (SPIx_CLKDIV[8:0]) to determine the output frequency of SPI clock.
2. Write the SPIx_SSCTL register a proper value for the related settings of Master mode:
   1) Clear AUTOSS (SPIx_SSCTL[3]) to 0 to disable the Automatic Slave Selection function.
   2) Configure slave selection signal as active low by clearing SSACTPOL (SPIx_SSCTL[2]) to 0.
   3) Enable slave selection signal by setting SS (SPIx_SSCTL[0]) to 1 to activate the off-chip slave device.
3. Write the related settings into the SPIx_CTL register to control the SPI master actions.
   1) Configure this SPI controller as master device by setting SLAVE (SPIx_CTL[18]) to 0.
   2) Force the SPI clock idle state at low by clearing CLKPOL (SPIx_CTL[3]) to 0.
   3) Select data transmitted on negative edge of SPI bus clock by setting TXNEG (SPIx_CTL[2]) to 1.
   4) Select data latched on positive edge of SPI bus clock by clearing RXNEG (SPIx_CTL[1]) to 0.
   5) Set the bit length of a transaction as 8-bit in DWIDTH bit field (SPIx_CTL[12:8] = 0x08).
   6) Set MSB transfer first by clearing LSB (SPIx_CTL[13]) to 0.
4. Set SPIEN (SPIx_CTL[0]) to 1 to enable the data transfer with the SPI interface.
5. If this SPI master attempts to transmit (write) one byte data to the off-chip slave device, write the byte data that will be transmitted into the SPIx_TX register.
6. Waiting for SPI interrupt if the UNITIEN (SPIx_CTL[17]) is set to 1, or just polling the unit transfer interrupt flag UNITIF (SPIx_STATUS[1]).
7. Read out the received one byte data from SPIx_RX register.
8. Go to 5) to continue another data transfer or set SS (SPIx_SSCTL[0]) to 0 to inactivate the off-chip slave device.

Example 2:

The SPI controller is set as a full-duplex slave device and connects with an off-chip master device. The off-chip master device communicates with the on-chip SPI slave controller through the SPI interface with the following specifications:

- Data bit is latched on positive edge of SPI bus clock.
- Data bit is driven on negative edge of SPI bus clock.
- Data is transferred from LSB first.
- SPI bus clock is idle at high state.
- Only one byte of data to be transmitted/received in a transaction.
- Slave selection signal is active high.

The operation flow is as follows:

1. Write the SPIx_SSCTL register a proper value for the related settings of Slave mode.
2. Select high level for the input of slave selection signal by setting SSACTPOL (SPIx_SSCTL[2])

to 1.

3. Write the related settings into the SPIx_CTL register to control this SPI slave actions
   1) Set the SPI controller as slave device by setting SLAVE (SPIx_CTL[18]) to 1.
   2) Select the SPI clock idle state at high by setting CLKPOL (SPIx_CTL[3]) to 1.
   3) Select data transmitted on negative edge of SPI bus clock by setting TXNEG (SPIx_CTL[2]) to 1.
   4) Select data latched on positive edge of SPI bus clock by clearing RXNEG (SPIx_CTL[1]) to 0.
   5) Set the bit length of a transaction as 8-bit in DWIDTH bit field (SPIx_CTL[12:8] = 0x08).
4. Set LSB transfer first by setting LSB (SPIx_CTL[13]) to 1.
5. Set the SPIEN (SPIx_CTL[0]) to 1. Wait for the slave select trigger input and SPI clock input from the off-chip master device to start the data transfer.
6. If this SPI slave attempts to transmit (be read) one byte data to the off-chip master device, write the byte data that will be transmitted into the SPIx_TX register.
7. If this SPI slave just only attempts to receive (be written) one byte data from the off-chip master device and does not care what data will be transmitted, the SPIx_TX register does not need to be updated by software.
8. Waiting for SPI interrupt if the UNITIEN (SPIx_CTL[17]) is set to 1, or just polling the unit transfer interrupt flag UNITIF (SPIx_STATUS[1]).
9. Read out the received one byte data from SPIx_RX register.
10. Go to 6 to continue another data transfer or stop data transfer.

6.22.8 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
SPI Base Address: **SPIx_BA = 0x4006_1000 + (0x0000_1000 * x)** **x=0, 1, 2, 3** **SPIx non-secure base address is SPIx_BA + 0x1000_0000.**				
SPIx_CTL	SPIx_BA+0x00	R/W	SPI Control Register	0x0000_0034
SPIx_CLKDIV	SPIx_BA+0x04	R/W	SPI Clock Divider Register	0x0000_0000
SPIx_SSCTL	SPIx_BA+0x08	R/W	SPI Slave Select Control Register	0x0000_0000
SPIx_PDMACTL	SPIx_BA+0x0C	R/W	SPI PDMA Control Register	0x0000_0000
SPIx_FIFOCTL	SPIx_BA+0x10	R/W	SPI FIFO Control Register	0x2200_0000
SPIx_STATUS	SPIx_BA+0x14	R/W	SPI Status Register	0x0005_0110
SPIx_STATUS2	SPIx_BA+0x18	R	SPI Status2 Register	0x0000_0000
SPIx_TX	SPIx_BA+0x20	W	SPI Data Transmit Register	0x0000_0000
SPIx_RX	SPIx_BA+0x30	R	SPI Data Receive Register	0x0000_0000
SPIx_I2SCTL	SPIx_BA+0x60	R/W	I^2S Control Register	0x0000_0000
SPIx_I2SCLK	SPIx_BA+0x64	R/W	I^2S Clock Divider Control Register	0x0000_0000
SPIx_I2SSTS	SPIx_BA+0x68	R/W	I^2S Status Register	0x0005_0100

6.22.9 Register Description

SPI Control Register (SPIx_CTL)

Register	Offset	R/W	Description	Reset Value
SPIx_CTL	SPIx_BA+0x00	R/W	SPI Control Register	0x0000_0034

Note: Not supported in I^2S mode.

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved			DATDIR	REORDER	SLAVE	UNITIEN	Reserved
15	14	13	12	11	10	9	8
RXONLY	HALFDPX	LSB	DWIDTH				
7	6	5	4	3	2	1	0
SUSPITV				CLKPOL	TXNEG	RXNEG	SPIEN

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20]	**DATDIR**	**Data Port Direction Control** This bit is used to select the data input/output direction in half-duplex transfer and Dual/Quad transfer 0 = SPI data is input direction. 1 = SPI data is output direction.
[19]	**REORDER**	**Byte Reorder Function Enable Bit** 0 = Byte Reorder function Disabled. 1 = Byte Reorder function Enabled. A byte suspend interval will be inserted among each byte. The period of the byte suspend interval depends on the setting of SUSPITV. **Note:** Byte Reorder function is only available if DWIDTH is defined as 16, 24, and 32 bits.
[18]	**SLAVE**	**Slave Mode Control** 0 = Master mode. 1 = Slave mode.
[17]	**UNITIEN**	**Unit Transfer Interrupt Enable Bit** 0 = SPI unit transfer interrupt Disabled. 1 = SPI unit transfer interrupt Enabled.
[16]	**Reserved**	Reserved.
[15]	**RXONLY**	**Receive-only Mode Enable Bit** This bit field is only available in Master mode. In receive-only mode, SPI Master will generate SPI bus clock continuously for receiving data bit from SPI slave device and assert the BUSY status. 0 = Receive-only mode Disabled. 1 = Receive-only mode Enabled.
[14]	**HALFDPX**	**SPI Half-duplex Transfer Enable Bit** This bit is used to select full-duplex or half-duplex for SPI transfer. The bit field DATDIR (SPIx_CTL[20])

		can be used to set the data direction in half-duplex transfer. 0 = SPI operates in full-duplex transfer. 1 = SPI operates in half-duplex transfer.
[13]	**LSB**	**Send LSB First** 0 = The MSB, which bit of transmit/receive register depends on the setting of DWIDTH, is transmitted/received first. 1 = The LSB, bit 0 of the SPI TX register, is sent first to the SPI data output pin, and the first bit received from the SPI data input pin will be put in the LSB position of the RX register (bit 0 of SPI_RX).
[12:8]	**DWIDTH**	**Data Width** This field specifies how many bits can be transmitted/received in one transaction. The minimum bit length is 8 bits and can up to 32 bits. DWIDTH = 0x08 …. 8 bits. DWIDTH = 0x09 …. 9 bits. …… DWIDTH = 0x1F …. 31 bits. DWIDTH = 0x00 …. 32 bits. **Note:** This bit field will decide the depth of TX/RX FIFO configuration in SPI mode**.** Therefore, changing this bit field will clear TX/RX FIFO by hardware automatically.
[7:4]	**SUSPITV**	**Suspend Interval** The four bits provide configurable suspend interval between two successive transmit/receive transaction in a Master transfer. The definition of the suspend interval is the interval between the last clock edge of the preceding transaction word and the first clock edge of the following transaction word. The default value is 0x3. The period of the suspend interval is obtained according to the following equation. (SUSPITV[3:0] + 0.5) * period of SPI_CLK clock cycle Example: SUSPITV = 0x0 …. 0.5 SPI_CLK clock cycle. SUSPITV = 0x1 …. 1.5 SPI_CLK clock cycle. …… SUSPITV = 0xE …. 14.5 SPI_CLK clock cycle. SUSPITV = 0xF …. 15.5 SPI_CLK clock cycle.
[3]	**CLKPOL**	**Clock Polarity** 0 = SPI bus clock is idle low. 1 = SPI bus clock is idle high.
[2]	**TXNEG**	**Transmit on Negative Edge** 0 = Transmitted data output signal is changed on the rising edge of SPI bus clock. 1 = Transmitted data output signal is changed on the falling edge of SPI bus clock.
[1]	**RXNEG**	**Receive on Negative Edge** 0 = Received data input signal is latched on the rising edge of SPI bus clock. 1 = Received data input signal is latched on the falling edge of SPI bus clock.
[0]	**SPIEN**	**SPI Transfer Control Enable Bit** In Master mode, the transfer will start when there is data in the FIFO buffer after this bit is set to 1. In Slave mode, this device is ready to receive data when this bit is set to 1. 0 = Transfer control Disabled. 1 = Transfer control Enabled. **Note:** Before changing the configurations of SPIx_CTL, SPIx_CLKDIV, SPIx_SSCTL and SPIx_FIFOCTL registers, user shall clear the SPIEN (SPIx_CTL[0]) and confirm the SPIENSTS (SPIx_STATUS[15]) is 0.

SPI Clock Divider Register (SPIx_CLKDIV)

Register	Offset	R/W	Description	Reset Value
SPIx_CLKDIV	SPIx_BA+0x04	R/W	SPI Clock Divider Register	0x0000_0000

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

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8:0]	**DIVIDER**	**Clock Divider** The value in this field is the frequency divider for generating the peripheral clock, f_{spi_eclk}, and the SPI bus clock of SPI Master. The frequency is obtained according to the following equation. $f_{spi\_eclk} = \frac{f_{spi\_clock\_src}}{(DIVIDER + 1)}$ where $f_{spi\_clock\_src}$ is the peripheral clock source, which is defined in the clock control register, CLK_CLKSEL2. **Note 1:** Not supported in I^2S mode. **Note 2:** The time interval must be larger than or equal 5 peripheral clock cycles between releasing SPI IP software reset and setting this clock divider register.

Note: DIVIDER should be set carefully because the peripheral clock frequency must be slower than or equal to system frequency.

SPI Slave Select Control Register (SPIx_SSCTL)

Register	Offset	R/W	Description	Reset Value
SPIx_SSCTL	SPIx_BA+0x08	R/W	SPI Slave Select Control Register	0x0000_0000

Note: Not supported in I^2S mode.

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		SSINAIEN	SSACTIEN	Reserved		SLVURIEN	SLVBEIEN
7	6	5	4	3	2	1	0
Reserved			SLV3WIRE	AUTOSS	SSACTPOL	Reserved	SS

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**SSINAIEN**	**Slave Select Inactive Interrupt Enable Bit** 0 = Slave select inactive interrupt Disabled. 1 = Slave select inactive interrupt Enabled.
[12]	**SSACTIEN**	**Slave Select Active Interrupt Enable Bit** 0 = Slave select active interrupt Disabled. 1 = Slave select active interrupt Enabled.
[11:10]	**Reserved**	Reserved.
[9]	**SLVURIEN**	**Slave Mode TX Under Run Interrupt Enable Bit** 0 = Slave mode TX under run interrupt Disabled. 1 = Slave mode TX under run interrupt Enabled.
[8]	**SLVBEIEN**	**Slave Mode Bit Count Error Interrupt Enable Bit** 0 = Slave mode bit count error interrupt Disabled. 1 = Slave mode bit count error interrupt Enabled.
[7:5]	**Reserved**	Reserved.
[4]	**SLV3WIRE**	**Slave 3-wire Mode Enable Bit** In Slave 3-wire mode, the SPI controller can work with 3-wire interface including SPIx_CLK, SPIx_MISO and SPIx_MOSI pins. 0 = 4-wire bi-direction interface. 1 = 3-wire bi-direction interface. **Note:** The value of this register equals to control register SLAVE (SPIx_I2SCTL[8]) when I^2S mode is enabled.
[3]	**AUTOSS**	**Automatic Slave Selection Function Enable Bit** 0 = Automatic slave selection function Disabled. Slave selection signal will be asserted/de-asserted according to SS (SPIx_SSCTL[0]).

		1 = Automatic slave selection function Enabled. **Note:** Master mode only.
[2]	**SSACTPOL**	**Slave Selection Active Polarity** This bit defines the active polarity of slave selection signal (SPIx_SS). 0 = The slave selection signal SPIx_SS is active low. 1 = The slave selection signal SPIx_SS is active high.
[1]	**Reserved**	Reserved.
[0]	**SS**	**Slave Selection Control** If AUTOSS bit is cleared to 0, 0 = set the SPIx_SS line to inactive state. 1 = set the SPIx_SS line to active state. If the AUTOSS bit is set to 1, 0 = Keep the SPIx_SS line at inactive state. 1 = SPIx_SS line will be automatically driven to active state for the duration of data transfer, and will be driven to inactive state for the rest of the time. The active state of SPIx_SS is specified in SSACTPOL (SPIx_SSCTL[2]). **Note:** Master mode only.

SPI PDMA Control Register (SPIx_PDMACTL)

Register	Offset	R/W	Description	Reset Value
SPIx_PDMACTL	SPIx_BA+0x0C	R/W	SPI PDMA Control Register	0x0000_0000

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

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**PDMARST**	**PDMA Reset** 0 = No effect. 1 = Reset the PDMA control logic of the SPI controller. This bit will be automatically cleared to 0.
[1]	**RXPDMAEN**	**Receive PDMA Enable Bit** 0 = Receive PDMA function Disabled. 1 = Receive PDMA function Enabled.
[0]	**TXPDMAEN**	**Transmit PDMA Enable Bit** 0 = Transmit PDMA function Disabled. 1 = Transmit PDMA function Enabled. **Note1:** In SPI Master mode with full duplex transfer, if both TX and RX PDMA functions are enabled, RX PDMA function cannot be enabled prior to TX PDMA function. User can enable TX PDMA function firstly or enable both functions simultaneously. **Note2:** In SPI Master mode with full duplex transfer, if both TX and RX PDMA functions are enabled, TX PDMA function cannot be disabled prior to RX PDMA function. User can disable RX PDMA function firstly or disable both functions simultaneously.

SPI FIFO Control Register (SPIx_FIFOCTL)

Register	Offset	R/W	Description	Reset Value
SPIx_FIFOCTL	SPIx_BA+0x10	R/W	SPI FIFO Control Register	0x2200_0000

31	30	29	28	27	26	25	24
Reserved	TXTH			Reserved	RXTH		
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					SLVBERX	TXFBCLR	RXFBCLR
7	6	5	4	3	2	1	0
TXUFIEN	TXUFPOL	RXOVIEN	RXTOIEN	TXTHIEN	RXTHIEN	TXRST	RXRST

Bits	Description	
[31]	**Reserved**	Reserved.
[30:28]	**TXTH**	**Transmit FIFO Threshold** If the valid data count of the transmit FIFO buffer is less than or equal to the TXTH setting, the TXTHIF bit will be set to 1, else the TXTHIF bit will be cleared to 0. The MSB of this bit field is only meaningful while SPI mode 8~16 bits of data length.
[27]	**Reserved**	Reserved.
[26:24]	**RXTH**	**Receive FIFO Threshold** If the valid data count of the receive FIFO buffer is larger than the RXTH setting, the RXTHIF bit will be set to 1, else the RXTHIF bit will be cleared to 0. The MSB of this bit field is only meaningful while SPI mode 8~16 bits of data length.
[23:11]	**Reserved**	Reserved.
[10]	**SLVBERX**	**RX FIFO Write Data Enable Bit When Slave Mode Bit Count Error** 0 = Uncompleted RX data will be dropped from RX FIFO when bit count error event happened in SPI Slave mode. 1 = Uncompleted RX data will be written into RX FIFO when bit count error event happened in SPI Slave mode. User can read SLVBENUM (SPIx_STATUS2[29:24]) to know the effective bit number of uncompleted RX data when SPI slave bit count error happened. **Note:** Slave mode only.
[9]	**TXFBCLR**	**Transmit FIFO Buffer Clear** 0 = No effect. 1 = Clear transmit FIFO pointer. The TXFULL bit will be cleared to 0 and the TXEMPTY bit will be set to 1. This bit will be cleared to 0 by hardware about 1 system clock after it is set to 1. **Note:** The TX shift register will not be cleared.
[8]	**RXFBCLR**	**Receive FIFO Buffer Clear** 0 = No effect. 1 = Clear receive FIFO pointer. The RXFULL bit will be cleared to 0 and the RXEMPTY bit will be set to 1. This bit will be cleared to 0 by hardware about 1 system clock after it is set to 1.

		Note: The RX shift register will not be cleared.
[7]	**TXUFIEN**	**TX Underflow Interrupt Enable Bit** When TX underflow event occurs in Slave mode, TXUFIF (SPIx_STATUS[19]) will be set to 1. This bit is used to enable the TX underflow interrupt. 0 = Slave TX underflow interrupt Disabled. 1 = Slave TX underflow interrupt Enabled.
[6]	**TXUFPOL**	**TX Underflow Data Polarity** 0 = The SPI data out is keep 0 if there is TX underflow event in Slave mode. 1 = The SPI data out is keep 1 if there is TX underflow event in Slave mode. **Note 1:** The TX underflow event occurs if there is no any data in TX FIFO when the slave selection signal is active. **Note 2:** This bit should be set as 0 in I^2S mode. **Note 3:** When TX underflow event occurs, SPIx_MISO pin state will be determined by this setting even though TX FIFO is not empty afterward. Data stored in TX FIFO will be sent through SPIx_MISO pin in the next transfer frame.
[5]	**RXOVIEN**	**Receive FIFO Overrun Interrupt Enable Bit** 0 = Receive FIFO overrun interrupt Disabled. 1 = Receive FIFO overrun interrupt Enabled.
[4]	**RXTOIEN**	**Receive Time-out Interrupt Enable Bit** 0 = Receive time-out interrupt Disabled. 1 = Receive time-out interrupt Enabled.
[3]	**TXTHIEN**	**Transmit FIFO Threshold Interrupt Enable Bit** 0 = TX FIFO threshold interrupt Disabled. 1 = TX FIFO threshold interrupt Enabled.
[2]	**RXTHIEN**	**Receive FIFO Threshold Interrupt Enable Bit** 0 = RX FIFO threshold interrupt Disabled. 1 = RX FIFO threshold interrupt Enabled.
[1]	**TXRST**	**Transmit Reset** 0 = No effect. 1 = Reset transmit FIFO pointer and transmit circuit. The TXFULL bit will be cleared to 0 and the TXEMPTY bit will be set to 1. This bit will be cleared to 0 by hardware about 3 system clock cycles + 2 peripheral clock cycles after it is set to 1. User can read TXRXRST (SPIx_STATUS[23]) to check if reset is accomplished or not. **Note:** If TX underflow event occurs in SPI Slave mode, this bit can be used to make SPI return to idle state.
[0]	**RXRST**	**Receive Reset** 0 = No effect. 1 = Reset receive FIFO pointer and receive circuit. The RXFULL bit will be cleared to 0 and the RXEMPTY bit will be set to 1. This bit will be cleared to 0 by hardware about 3 system clock cycles + 2 peripheral clock cycles after it is set to 1. User can read TXRXRST (SPIx_STATUS[23]) to check if reset is accomplished or not.

SPI Status Register (SPIx_STATUS)

Register	Offset	R/W	Description	Reset Value
SPIx_STATUS	SPIx_BA+0x14	R/W	SPI Status Register	0x0005_0110

Note: Not supported in I^2S mode.

31	30	29	28	27	26	25	24
TXCNT				RXCNT			
23	22	21	20	19	18	17	16
TXRXRST	Reserved			TXUFIF	TXTHIF	TXFULL	TXEMPTY
15	14	13	12	11	10	9	8
SPIENSTS	Reserved		RXTOIF	RXOVIF	RXTHIF	RXFULL	RXEMPTY
7	6	5	4	3	2	1	0
SLVURIF	SLVBEIF	Reserved	SSLINE	SSINAIF	SSACTIF	UNITIF	BUSY

Bits	Description	
[31:28]	**TXCNT**	**Transmit FIFO Data Count (Read Only)** This bit field indicates the valid data count of transmit FIFO buffer.
[27:24]	**RXCNT**	**Receive FIFO Data Count (Read Only)** This bit field indicates the valid data count of receive FIFO buffer.
[23]	**TXRXRST**	**TX or RX Reset Status (Read Only)** 0 = The reset function of TXRST or RXRST is done. 1 = Doing the reset function of TXRST or RXRST. **Note:** Both the reset operations of TXRST and RXRST need 3 system clock cycles + 2 peripheral clock cycles. User can check the status of this bit to monitor the reset function is doing or done.
[22:20]	**Reserved**	Reserved.
[19]	**TXUFIF**	**TX Underflow Interrupt Flag** When the TX underflow event occurs, this bit will be set to 1, the state of data output pin depends on the setting of TXUFPOL. 0 = No effect. 1 = No data in Transmit FIFO and TX shift register when the slave selection signal is active. **Note 1:** This bit will be cleared by writing 1 to it. **Note 2:** If reset slave's transmission circuit when slave selection signal is active, this flag will be set to 1 after 2 peripheral clock cycles + 3 system clock cycles since the reset operation is done.
[18]	**TXTHIF**	**Transmit FIFO Threshold Interrupt Flag (Read Only)** 0 = The valid data count within the transmit FIFO buffer is larger than the setting value of TXTH. 1 = The valid data count within the transmit FIFO buffer is less than or equal to the setting value of TXTH.
[17]	**TXFULL**	**Transmit FIFO Buffer Full Indicator (Read Only)** 0 = Transmit FIFO buffer is not full. 1 = Transmit FIFO buffer is full.
[16]	**TXEMPTY**	**Transmit FIFO Buffer Empty Indicator (Read Only)**

		0 = Transmit FIFO buffer is not empty. 1 = Transmit FIFO buffer is empty.
[15]	**SPIENSTS**	**SPI Enable Status (Read Only)** 0 = SPI controller Disabled. 1 = SPI controller Enabled. **Note:** The SPI peripheral clock is asynchronous with the system clock. In order to make sure the SPI control logic is disabled, this bit indicates the real status of SPI controller.
[14:13]	**Reserved**	Reserved.
[12]	**RXTOIF**	**Receive Time-out Interrupt Flag** 0 = No receive FIFO time-out event. 1 = Receive FIFO buffer is not empty and no read operation on receive FIFO buffer over 64 SPI peripheral clock periods in Master mode or over 576 SPI peripheral clock periods in Slave mode. When the received FIFO buffer is read by software, the time-out status will be cleared automatically. **Note:** This bit will be cleared by writing 1 to it.
[11]	**RXOVIF**	**Receive FIFO Overrun Interrupt Flag** When the receive FIFO buffer is full, the follow-up data will be dropped and this bit will be set to 1. 0 = No FIFO is overrun. 1 = Receive FIFO is overrun. **Note:** This bit will be cleared by writing 1 to it.
[10]	**RXTHIF**	**Receive FIFO Threshold Interrupt Flag (Read Only)** 0 = The valid data count within the receive FIFO buffer is smaller than or equal to the setting value of RXTH. 1 = The valid data count within the receive FIFO buffer is larger than the setting value of RXTH.
[9]	**RXFULL**	**Receive FIFO Buffer Full Indicator (Read Only)** 0 = Receive FIFO buffer is not full. 1 = Receive FIFO buffer is full.
[8]	**RXEMPTY**	**Receive FIFO Buffer Empty Indicator (Read Only)** 0 = Receive FIFO buffer is not empty. 1 = Receive FIFO buffer is empty.
[7]	**SLVURIF**	**Slave Mode TX Under Run Interrupt Flag** In Slave mode, if TX underflow event occurs and the slave select line goes to inactive state, this interrupt flag will be set to 1. 0 = No Slave TX under run event. 1 = Slave TX under run event occurred. **Note:** This bit will be cleared by writing 1 to it.
[6]	**SLVBEIF**	**Slave Mode Bit Count Error Interrupt Flag** In Slave mode, when the slave select line goes to inactive state, if bit counter is mismatch with DWIDTH, this interrupt flag will be set to 1. 0 = No Slave mode bit count error event. 1 = Slave mode bit count error event occurred. **Note:** If the slave select active but there is no any bus clock input, the SLVBEIF also active when the slave select goes to inactive state. This bit will be cleared by writing 1 to it.
[5]	**Reserved**	Reserved.
[4]	**SSLINE**	**Slave Select Line Bus Status (Read Only)** 0 = The slave select line status is 0. 1 = The slave select line status is 1.

		Note: This bit is only available in Slave mode. If SSACTPOL (SPIx_SSCTL[2]) is set 0, and the SSLINE is 1, the SPI slave select is in inactive status.
[3]	**SSINAIF**	**Slave Select Inactive Interrupt Flag** 0 = Slave select inactive interrupt was cleared or not occurred. 1 = Slave select inactive interrupt event occurred. **Note:** Only available in Slave mode. This bit will be cleared by writing 1 to it.
[2]	**SSACTIF**	**Slave Select Active Interrupt Flag** 0 = Slave select active interrupt was cleared or not occurred. 1 = Slave select active interrupt event occurred. **Note:** Only available in Slave mode. This bit will be cleared by writing 1 to it.
[1]	**UNITIF**	**Unit Transfer Interrupt Flag** 0 = No transaction has been finished since this bit was cleared to 0. 1 = SPI controller has finished one unit transfer. **Note:** This bit will be cleared by writing 1 to it.
[0]	**BUSY**	**Busy Status (Read Only)** 0 = SPI controller is in idle state. 1 = SPI controller is in busy state. The following lists the bus busy conditions: a. SPIEN (SPIx_CTL[0]) = 1 and TXEMPTY = 0. b. For SPI Master mode, SPIEN (SPIx_CTL[0]) = 1 and TXEMPTY = 1 but the current transaction is not finished yet. c. For SPI Master mode, SPIEN (SPIx_CTL[0]) = 1 and RXONLY = 1. d. For SPI Slave mode, SPIEN (SPIx_CTL[0]) = 1 and there is serial clock input into the SPI core logic when slave select is active. e. For SPI Slave mode, SPIEN (SPIx_CTL[0]) = 1 and the transmit buffer or transmit shift register is not empty even if the slave select is inactive. **Note:** By applications, this SPI busy flag should be used with other status registers in SPIx_STATUS such as TXCNT, RXCNT, TXTHIF, TXFULL, TXEMPTY, RXTHIF, RXFULL, RXEMPTY, and UNITIF. Therefore the SPI transfer done events of TX/RX operations can be obtained at correct timing point.

SPI Status2 Register (SPIx_STATUS2)

Register	Offset	R/W	Description	Reset Value
SPIx_STATUS2	SPIx_BA+0x18	R	SPI Status2 Register	0x0000_0000

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

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:24]	**SLVBENUM**	**Effective Bit Number of Uncompleted RX Data** This status register indicates that effective bit number of uncompleted RX data when SLVBERX (SPIx_FIFOCTL[10]) is enabled and RX bit count error event happen in SPI Slave mode. This status register will be fixed to 0x0 when SLVBERX (SPIx_FIFOCTL[10]) is disabled. **Note 1:** This register will be cleared to 0x0 when user writes 0x1 to SLVBEIF (SPIx_STATUS[6]). **Note 2:** Slave mode only.
[23:0]	**Reserved**	Reserved.

SPI Data Transmit Register (SPIx_TX)

Register	Offset	R/W	Description	Reset Value
SPIx_TX	SPIx_BA+0x20	W	SPI Data Transmit Register	0x0000_0000

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

Bits		Description
[31:0]	**TX**	**Data Transmit Register** The data transmit registers pass through the transmitted data into the 4-level transmit FIFO buffers. The number of valid bits depends on the setting of DWIDTH (SPIx_CTL[12:8]) in SPI mode or WDWIDTH (SPIx_I2SCTL[5:4]) in I^{2}S mode. In SPI mode, if DWIDTH is set to 0x08, the bits TX[7:0] will be transmitted. If DWIDTH is set to 0x00, the SPI controller will perform a 32-bit transfer. In I^{2}S mode, if WDWIDTH (SPIx_I2SCTL[5:4]) is set to 0x2, the data width of audio channel is 24-bit and corresponding to TX[23:0]. If WDWIDTH is set as 0x0, 0x1, or 0x3, all bits of this field are valid and referred to the data arrangement in I^{2}S mode FIFO operation section **Note:** In Master mode, SPI controller will start to transfer the SPI bus clock after 1 APB clock and 6 peripheral clock cycles after user writes to this register.

SPI Data Receive Register (SPIx_RX)

Register	Offset	R/W	Description	Reset Value
SPIx_RX	SPIx_BA+0x30	R	SPI Data Receive Register	0x0000_0000

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

Bits	Description	
[31:0]	**RX**	**Data Receive Register (Read Only)** There are 4-level FIFO buffers in this controller. The data receive register holds the data received from SPI data input pin. If the RXEMPTY (SPIx_STATUS[8] or SPIx_I2SSTS[8]) is not set to 1, the receive FIFO buffers can be accessed through software by reading this register.

I²S Control Register (SPIx_I2SCTL)

Register	Offset	R/W	Description	Reset Value
SPIx_I2SCTL	SPIx_BA+0x60	R/W	I²S Control Register	0x0000_0000

Note: Not supported in SPI mode.

31	30	29	28	27	26	25	24
SLVERRIEN	Reserved	FORMAT		Reserved		LZCIEN	RZCIEN
23	22	21	20	19	18	17	16
RXLCH	Reserved					LZCEN	RZCEN
15	14	13	12	11	10	9	8
MCLKEN	Reserved						SLAVE
7	6	5	4	3	2	1	0
ORDER	MONO	WDWIDTH		MUTE	RXEN	TXEN	I2SEN

Bits	Description	
[31]	**SLVERRIEN**	**Bit Number Error Interrupt Enable Bit for Slave Mode** Interrupt occurs if this bit is set to 1 and bit number error event occurs. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[30]	**Reserved**	Reserved.
[29:28]	**FORMAT**	**Data Format Selection** 00 = I²S data format. 01 = MSB justified data format. 10 = PCM mode A. 11 = PCM mode B.
[27:26]	**Reserved**	Reserved.
[25]	**LZCIEN**	**Left Channel Zero Cross Interrupt Enable Bit** Interrupt occurs if this bit is set to 1 and left channel zero cross event occurs. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[24]	**RZCIEN**	**Right Channel Zero Cross Interrupt Enable Bit** Interrupt occurs if this bit is set to 1 and right channel zero cross event occurs. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[23]	**RXLCH**	**Receive Left Channel Enable Bit** When monaural format is selected (MONO = 1), I²S controller will receive right channel data if RXLCH is set to 0, and receive left channel data if RXLCH is set to 1. 0 = Receive right channel data in Mono mode. 1 = Receive left channel data in Mono mode.
[22:18]	**Reserved**	Reserved.

[17]	**LZCEN**	**Left Channel Zero Cross Detection Enable Bit** If this bit is set to 1, when left channel data sign bit changes or next shift data bits are all 0 then LZCIF flag in SPIx_I2SSTS register is set to 1. This function is only available in transmit operation. 0 = Left channel zero cross detection Disabled. 1 = Left channel zero cross detection Enabled.
[16]	**RZCEN**	**Right Channel Zero Cross Detection Enable Bit** If this bit is set to 1, when right channel data sign bit change or next shift data bits are all 0 then RZCIF flag in SPIx_I2SSTS register is set to 1. This function is only available in transmit operation. 0 = Right channel zero cross detection Disabled. 1 = Right channel zero cross detection Enabled.
[15]	**MCLKEN**	**Master Clock Enable Bit** If MCLKEN is set to 1, I^2S controller will generate master clock on SPIx_I2SMCLK pin for external audio devices. 0 = Master clock Disabled. 1 = Master clock Enabled.
[14:9]	**Reserved**	Reserved.
[8]	**SLAVE**	**Slave Mode** I^2S can operate as master or slave. For Master mode, I2Sx_BCLK and I2Sx_LRCLK pins are output mode and send bit clock from this chip to audio CODEC chip. In Slave mode, I2Sx_BCLK and I2Sx_LRCLK pins are input mode and I2Sx_BCLK and I2Sx_LRCLK signals are received from outer audio CODEC chip. 0 = Master mode. 1 = Slave mode.
[7]	**ORDER**	**Stereo Data Order in FIFO** 0 = Left channel data at high byte. 1 = Left channel data at low byte.
[6]	**MONO**	**Monaural Data** 0 = Data is stereo format. 1 = Data is monaural format.
[5:4]	**WDWIDTH**	**Word Width** 00 = data size is 8-bit. 01 = data size is 16-bit. 10 = data size is 24-bit. 11 = data size is 32-bit.
[3]	**MUTE**	**Transmit Mute Enable Bit** 0 = Transmit data is shifted from buffer. 1 = Transmit channel zero.
[2]	**RXEN**	**Receive Enable Bit** 0 = Data receive Disabled. 1 = Data receive Enabled.
[1]	**TXEN**	**Transmit Enable Bit** 0 = Data transmit Disabled. 1 = Data transmit Enabled.

[0]	**I2SEN**	**I²S Controller Enable Bit** 0 = I²S mode Disabled. 1 = I²S mode Enabled. **Note 1:** If enabling this bit, I2Sx_BCLK will start to output in Master mode. **Note 2:** Before changing the configurations of SPIx_I2SCTL, SPIx_I2SCLK, and SPIx_FIFOCTL registers, user shall clear the I2SEN (SPIx_I2SCTL[0]) and confirm the I2SENSTS (SPIx_I2SSTS[15]) is 0.

I²S Clock Divider Control Register (SPIx_I2SCLK)

Register	Offset	R/W	Description	Reset Value
SPIx_I2SCLK	SPIx_BA+0x64	R/W	I²S Clock Divider Control Register	0x0000_0000

Note: Not supported in SPI mode.

31	30	29	28	27	26	25	24
Reserved						I2SSLAVE	I2SMODE
23	22	21	20	19	18	17	16
Reserved						BCLKDIV	
15	14	13	12	11	10	9	8
BCLKDIV							
7	6	5	4	3	2	1	0
Reserved	MCLKDIV						

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25]	**I2SSLAVE**	**I²S Clock Divider Number Selection for I²S Slave Mode and I²S Master Mode** User sets I2SSLAVE to set frequency of peripheral clock of I²S Master mode and I²S Slave mode when BCLKDIV (SPIx_I2SCLK[17:8]) is set. I2SSLAVE needs to set before I2SEN (SPIx_I2SCTL[0]) is enabled. 0 = The frequency of peripheral clock is set to I²S Master mode. 1 = The frequency of peripheral clock is set to I²S Slave mode.
[24]	**I2SMODE**	**I²S Clock Divider Number Selection for I²S Mode and SPI Mode** User sets I2SMODE to set frequency of peripheral clock of I²S mode or SPI mode when BCLKDIV (SPIx_I2SCLK[17:8]) or DIVIDER (SPIx_CLKDIV[8:0]) is set. User needs to set I2SMODE before I2SEN (SPIx_I2SCTL[0]) or SPIEN (SPIx_CTL[0]) is enabled. 0 = The frequency of peripheral clock is set to SPI mode. 1 = The frequency of peripheral clock is set to I²S mode.
[23:18]	**Reserved**	Reserved.

[17:8]	**BCLKDIV**	**Bit Clock Divider** The I²S controller will generate bit clock in Master mode. The clock frequency of bit clock , f_{BCLK}, is determined by the following expression: $f_{BCLK} = \frac{f_{i2s\_clock\_src}}{2 \times (\text{BCLKDIV} + 1)}$ where $f_{i2s\_clock\_src}$ is the frequency of I²S peripheral clock source, which is defined in the clock control register CLK_CLKSEL2. In I²S Slave mode, this field is used to define the frequency of peripheral clock and it's determined by $f_{i2s\_clock\_src} \div \left(\frac{\text{BCLKDIV}}{2} + 1\right)$. The peripheral clock frequency in I²S Slave mode must be equal to or faster than 6 times of input bit clock. **Note:** The time interval must be larger than or equal 5 peripheral clock cycles between releasing SPI IP software reset and setting this clock divider register.
[7]	**Reserved**	Reserved.
[6:0]	**MCLKDIV**	**Master Clock Divider** If MCLKEN is set to 1, I²S controller will generate master clock for external audio devices. The frequency of master clock, f_{MCLK}, is determined by the following expressions: If MCLKDIV >= 1,. $f_{MCLK} = \frac{f_{i2s\_clock\_src}}{2 \times \text{MCLKDIV}}$ If MCLKDIV = 0,. $f_{MCLK} = f_{i2s\_clock\_src}$ where $f_{i2s\_clock\_src}$ is the frequency of I²S peripheral clock source, which is defined in the clock control register CLK_CLKSEL2. In general, the master clock rate is 256 times sampling clock rate.

Note: BCLKDIV should be set carefully because the peripheral clock frequency must be slower than or equal to system frequency.

I²S Status Register (SPIx_I2SSTS)

Register	Offset	R/W	Description	Reset Value
SPIx_I2SSTS	SPIx_BA+0x68	R/W	I²S Status Register	0x0005_0100

Note: Not supported in SPI mode.

31	30	29	28	27	26	25	24
Reserved	TXCNT			Reserved	RXCNT		
23	22	21	20	19	18	17	16
TXRXRST	SLVERRIF	LZCIF	RZCIF	TXUFIF	TXTHIF	TXFULL	TXEMPTY
15	14	13	12	11	10	9	8
I2SENSTS	Reserved		RXTOIF	RXOVIF	RXTHIF	RXFULL	RXEMPTY
7	6	5	4	3	2	1	0
Reserved			RIGHT	Reserved			

Bits	Description	
[31]	**Reserved**	Reserved.
[30:28]	**TXCNT**	**Transmit FIFO Data Count (Read Only)** This bit field indicates the valid data count of transmit FIFO buffer.
[27]	**Reserved**	Reserved.
[26:24]	**RXCNT**	**Receive FIFO Data Count (Read Only)** This bit field indicates the valid data count of receive FIFO buffer.
[23]	**TXRXRST**	**TX or RX Reset Status (Read Only)** 0 = The reset function of TXRST or RXRST is done. 1 = Doing the reset function of TXRST or RXRST. **Note:** Both the reset operations of TXRST and RXRST need 3 system clock cycles + 2 peripheral clock cycles. User can check the status of this bit to monitor the reset function is doing or done.
[22]	**SLVERRIF**	**Bit Number Error Interrupt Flag for Slave Mode** 0 = No bit number error event occurred. 1 = Bit number error event occurred. **Note:** This bit will be cleared by writing 1 to it.
[21]	**LZCIF**	**Left Channel Zero Cross Interrupt Flag** 0 = No zero cross event occurred on left channel. 1 = Zero cross event occurred on left channel.
[20]	**RZCIF**	**Right Channel Zero Cross Interrupt Flag** 0 = No zero cross event occurred on right channel. 1 = Zero cross event occurred on right channel.
[19]	**TXUFIF**	**Transmit FIFO Underflow Interrupt Flag** When the transmit FIFO buffer is empty and there is no datum written into the FIFO buffer, if there is more bus clock input, this bit will be set to 1. **Note:** This bit will be cleared by writing 1 to it.

[18]	**TXTHIF**	**Transmit FIFO Threshold Interrupt Flag (Read Only)** 0 = The valid data count within the transmit FIFO buffer is larger than the setting value of TXTH. 1 = The valid data count within the transmit FIFO buffer is less than or equal to the setting value of TXTH. **Note:** If TXTHIEN = 1 and TXTHIF = 1, the SPI/I2S controller will generate a SPI interrupt request.
[17]	**TXFULL**	**Transmit FIFO Buffer Full Indicator (Read Only)** 0 = Transmit FIFO buffer is not full. 1 = Transmit FIFO buffer is full.
[16]	**TXEMPTY**	**Transmit FIFO Buffer Empty Indicator (Read Only)** 0 = Transmit FIFO buffer is not empty. 1 = Transmit FIFO buffer is empty.
[15]	**I2SENSTS**	**I^2S Enable Status (Read Only)** 0 = The SPI/I^2S control logic is disabled. 1 = The SPI/I^2S control logic is enabled. **Note:** The SPI peripheral clock is asynchronous with the system clock. In order to make sure the SPI/I^2S control logic is disabled, this bit indicates the real status of SPI/I^2S control logic for user.
[14:13]	**Reserved**	Reserved.
[12]	**RXTOIF**	**Receive Time-out Interrupt Flag** 0 = No receive FIFO time-out event. 1 = Receive FIFO buffer is not empty and no read operation on receive FIFO buffer over 64 SPI peripheral clock period in Master mode or over 576 SPI peripheral clock period in Slave mode. When the received FIFO buffer is read by software, the time-out status will be cleared automatically. **Note:** This bit will be cleared by writing 1 to it.
[11]	**RXOVIF**	**Receive FIFO Overrun Interrupt Flag** When the receive FIFO buffer is full, the follow-up data will be dropped and this bit will be set to 1. **Note:** This bit will be cleared by writing 1 to it.
[10]	**RXTHIF**	**Receive FIFO Threshold Interrupt Flag (Read Only)** 0 = The valid data count within the receive FIFO buffer is smaller than or equal to the setting value of RXTH. 1 = The valid data count within the receive FIFO buffer is larger than the setting value of RXTH. **Note:** If RXTHIEN = 1 and RXTHIF = 1, the SPI/I^2S controller will generate a SPI interrupt request.
[9]	**RXFULL**	**Receive FIFO Buffer Full Indicator (Read Only)** 0 = Receive FIFO buffer is not full. 1 = Receive FIFO buffer is full.
[8]	**RXEMPTY**	**Receive FIFO Buffer Empty Indicator (Read Only)** 0 = Receive FIFO buffer is not empty. 1 = Receive FIFO buffer is empty.
[7:5]	**Reserved**	Reserved.
[4]	**RIGHT**	**Right Channel (Read Only)** This bit indicates the current transmit data is belong to which channel. 0 = Left channel. 1 = Right channel.
[3:0]	**Reserved**	Reserved.

6.23 Quad Serial Peripheral Interface (QSPI)

6.23.1 Overview

The Quad Serial Peripheral Interface (QSPI) applies to synchronous serial data communication and allows full duplex transfer. Devices communicate in Master/Slave mode with the 4-wire bi-direction interface. The M2354 series contains one QSPI controller performing a serial-to-parallel conversion on data received from a peripheral device, and a parallel-to-serial conversion on data transmitted to a peripheral device.

The QSPI controller supports 2-bit transfer mode to perform full-duplex 2-bit data transfer and also supports Dual and Quad I/O transfer mode and the controller supports the PDMA function to access the data buffer.

6.23.2 Features

- Supports Master or Slave mode operation
- Supports 2-bit transfer mode
- Supports Dual and Quad I/O transfer mode
- Configurable bit length of a transaction word from 8 to 32-bit
- Provides separate 8-level depth transmit and receive FIFO buffers
- Supports MSB first or LSB first transfer sequence
- Supports Byte Reorder function
- Supports Byte or Word Suspend mode
- Supports PDMA transfer
- Supports 3-Wire, no slave selection signal, bi-direction interface
- Supports one data channel half-duplex transfer
- Supports Transmit Double Transfer Rate Mode (TX DTR mode)
- Supports receive-only mode

6.23.3 Block Diagram

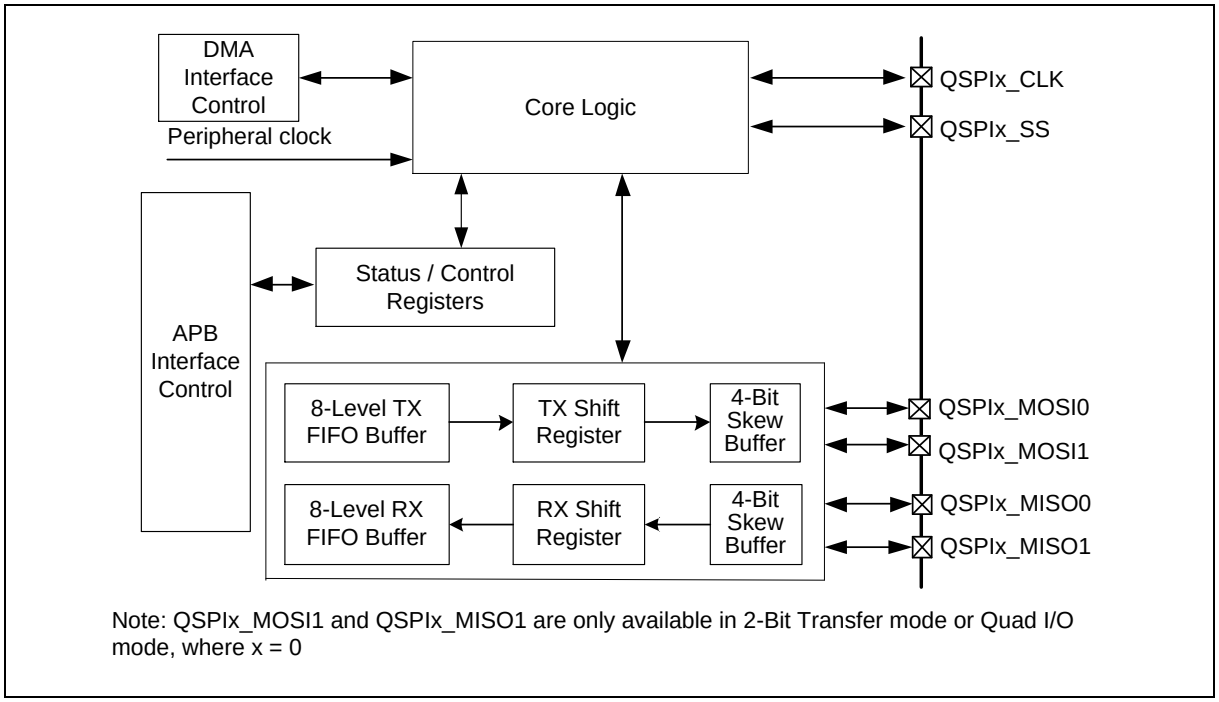


Figure 6.23-1 QSPI Block Diagram

TX FIFO Buffer:

The transmit FIFO buffer is a 8-level depth, 32-bit wide, first-in, first-out register buffer. The data can be written to the transmit FIFO buffer in advance through software by writing the QSPIx_TX register.

RX FIFO Buffer:

The receive FIFO buffer is also a 8-level depth, 32-bit wide, first-in, first-out register buffer. The receive control logic will store the receive data to this buffer. The FIFO buffer data can be read from QSPIx_RX register by software.

TX Shift Register:

The transmit shift register is a 32-bit wide register buffer. The transmit data is loaded from the TX FIFO buffer and shifted out bit-by-bit to the skew buffer.

RX Shift Register:

The receive shift register is also a 32-bit wide register buffer. The receive data is shift in bit-by-bit from the skew buffer and is loaded into RX FIFO buffer when a transaction done.

Skew Buffer:

The skew buffer is a 4-level 1-bit buffer. There are two skew buffers in transmitting and received side. In received side, it is used to shift bits into RX shift register from QSPI bus. In transmitting side, it is used to shift bits into QSPI bus from TX shift register.

6.23.4 Basic Configuration

6.23.4.1 QSPI0 Basic Configuration

- Clock source Configuration
  - Select the source of QSPI0 peripheral clock on QSPI0SEL (CLK_CLKSEL2[3:2]).

- Enable QSPI0 peripheral clock in QSPI0CKEN (CLK_APBCLK0[12]).

- Reset Configuration
  - Reset QSPI0 controller in QSPI0RST (SYS_IPRST1[12]).

- Pin Configuration

Group	Pin Name	GPIO	MFP
QSPI0	QSPI0_CLK	PA.2, PH.8	MFP3
		PC.2	MFP4
		PF.2	MFP5
	QSPI0_MISO0	PA.1, PE.1	MFP3
		PC.1	MFP4
	QSPI0_MISO1	PA.5, PH.10	MFP3
		PC.5	MFP4
	QSPI0_MOSI0	PA.0, PE.0	MFP3
		PC.0	MFP4
	QSPI0_MOSI1	PA.4, PH.11	MFP3
		PC.4	MFP4
	QSPI0_SS	PA.3, PH.9	MFP3
		PC.3	MFP4

6.23.5 Functional Description

6.23.5.1 Terminology

QSPI Peripheral Clock and QSPI Bus Clock

The QSPI controller needs the peripheral clock to drive the QSPI logic unit to perform the data transfer. The peripheral clock rate is determined by the settings of clock divisor (QSPIx_CLKDIV) and the clock source which can be HXT, PLL, PCLK or HIRC. QSPIxSEL of CLK_CLKSEL2 register determines the clock source of the peripheral clock. The DIVIDER (QSPIx_CLKDIV[8:0]) setting determines the divisor of the clock rate calculation.

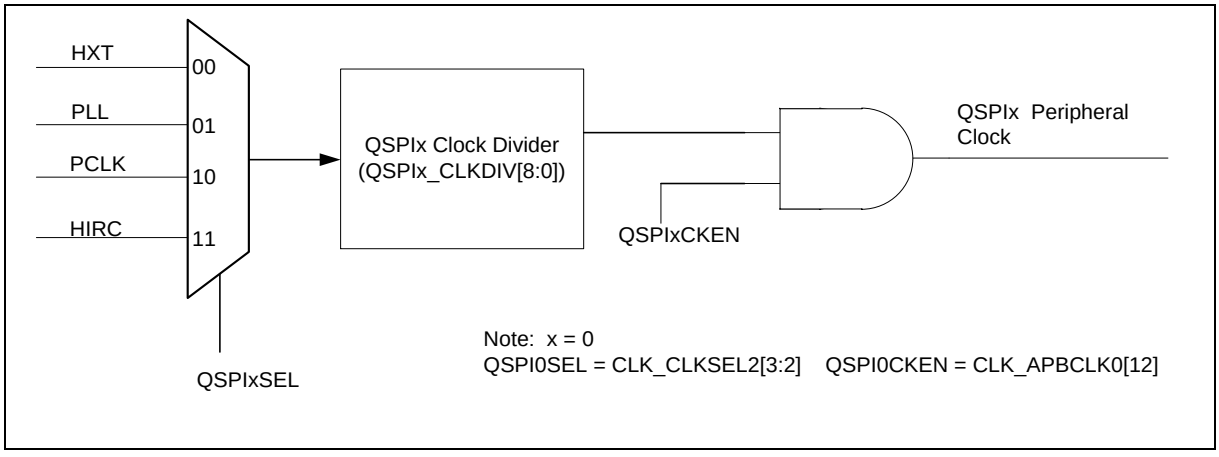


Figure 6.23-2 QSPI Peripheral Clock

In Master mode, the frequency of the QSPI bus clock is equal to the peripheral clock rate. In general, the QSPI bus clock is denoted as QSPI clock. In Slave mode, the QSPI bus clock is provided by a master device. The frequency of QSPI peripheral clock cannot be faster than the system clock rate regardless of Master or Slave mode. If the clock source of peripheral clock is not system clock, the frequency of QSPI peripheral clock shall be slower than the system clock frequency regardless of Master or Slave mode.

Master/Slave mode

The QSPI controllers can be set as Master or Slave mode by setting the SLAVE (QSPIx_CTL[18]) to communicate with the off-chip SPI slave or master device. The HALFDPX (QSPIx_CTL[14]) can be used to select the full-duplex or half-duplex in QSPI transmission. The application block diagrams in Master and Slave mode are shown below.

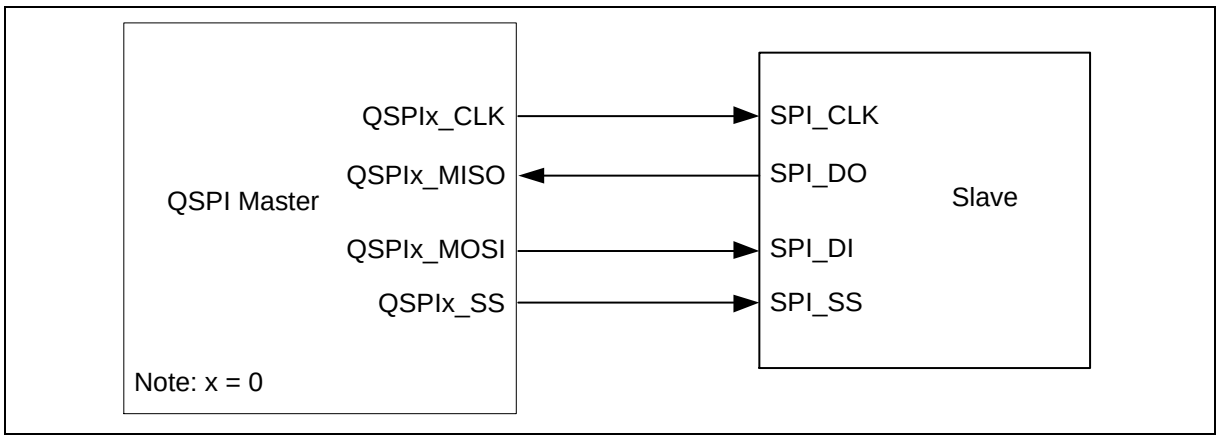


Figure 6.23-3 QSPI Full-Duplex Master Mode Application Block Diagram

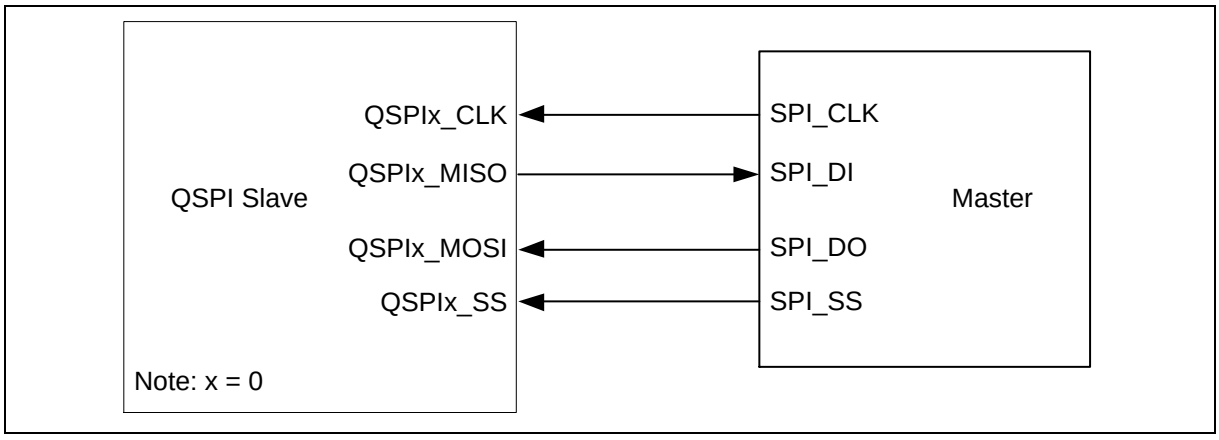


Figure 6.23-4 QSPI Full-Duplex Slave Mode Application Block Diagram

Slave Selection

In Master mode, the QSPI controller can drive off-chip slave device through the slave select output pin QSPIx_SS. In Slave mode, the off-chip master device drives the slave selection signal from the QSPIx_SS input port to this QSPI controller. The duration between the slave select active edge and the first QSPI clock input shall over 3 QSPI peripheral clock cycles of slave.

In Master/Slave mode, the active state of slave selection signal can be programmed to low or high

active in SSACTPOL (QSPIx_SSCTL[2]). The selection of slave select conditions depends on what type of device is connected. In Slave mode, to recognize the inactive state of the slave selection signal, the inactive period of the slave selection signal must be larger than or equal to 3 peripheral clock cycles between two successive transactions.

In Slave mode, the QSPIx_MISO signal type is controlled by the QSPIx_SS pin. If the QSPIx_SS pin is inactive state, the QSPIx_MISO is set to tri-state to avoid interference with other Slave devices. If the QSPIx_SS pin is active state, the QSPIx_MISO is set to output mode for data transfer.

Timing Condition

The CLKPOL (QSPIx_CTL[3]) defines the QSPI clock idle state. If CLKPOL = 1, the output QSPI clock is idle at high state; if CLKPOL = 0, it is idle at low state.

TXNEG (QSPIx_CTL[2]) defines the data transmitted out either on negative edge or on positive edge of QSPI clock. RXNEG (QSPIx_CTL[1]) defines the data received either on negative edge or on positive edge of QSPI clock.

Note: The settings of TXNEG and RXNEG are mutually exclusive. In other words, do not transmit and receive data at the same clock edge.

Transmit/Receive Bit Length

The bit length of a transaction word is defined in DWIDTH (QSPIx_CTL[12:8]) and can be configured up to 32-bit length in a transaction word for transmitting and receiving.

When QSPI controller finishes a transaction, i.e. receives or transmits a specific count of bits defined in DWIDTH (QSPIx_CTL[12:8]), the unit transfer interrupt flag UNITIF (QSPIx_STATUS[1]) will be set to 1.

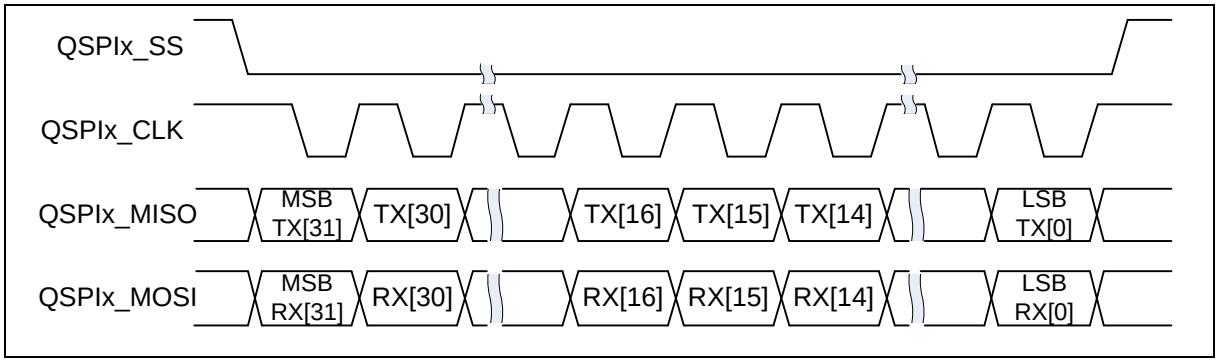


Figure 6.23-5 32-bit in One Transaction

LSB/MSB First

LSB (QSPIx_CTL[13]) defines the bit transfer sequence in a transaction. If the LSB (QSPIx_CTL[13]) is set to 1, the transfer sequence is LSB first. The bit 0 will be transferred firstly. If the LSB (QSPIx_CTL[13]) is cleared to 0, the transfer sequence is MSB first.

Suspend Interval

SUSPITV (QSPIx_CTL[7:4]) provides a configurable suspend interval, 0.5 ~ 15.5 QSPI clock periods, between two successive transaction words in Master mode. The definition of the suspend interval is the interval between the last clock edge of the preceding transaction word and the first clock edge of the following transaction word. The default value of SUSPITV is 0x3 (3.5 QSPI clock cycles).

6.23.5.2 Automatic Slave Selection

In Master mode, if AUTOSS (QSPIx_SSCTL[3]) is set, the slave selection signal will be generated automatically and output to the QSPIx_SS pin according to whether SS (QSPIx_SSCTL[0]) is enabled

or not. The slave selection signal will be set to active state by the QSPI controller when the QSPI data transfer is started by writing to FIFO. It will be set to inactive state when QSPI bus is idle. If QSPI bus is not idle, i.e. TX FIFO, TX shift register or TX skew buffer is not empty, the slave selection signal will be set to inactive state between transactions if the value of SUSPITV (QSPIx_CTL[7:4]) is greater than or equal to 3.

In Master mode, if the value of SUSPITV is less than 3 and the AUTOSS is set as 1, the slave selection signal will be kept at active state between two successive transactions.

If the AUTOSS bit is cleared, the slave selection output signal will be determined by the SS setting. The active state of the slave selection output signal is specified in SSACTPOL (QSPIx_SSCTL[2]).

The duration between the slave selection signal active edge and the first QSPI bus clock edge is 1 QSPI bus clock cycle and the duration between the last QSPI bus clock and the slave selection signal inactive edge is 1.5 QSPI bus clock cycle.

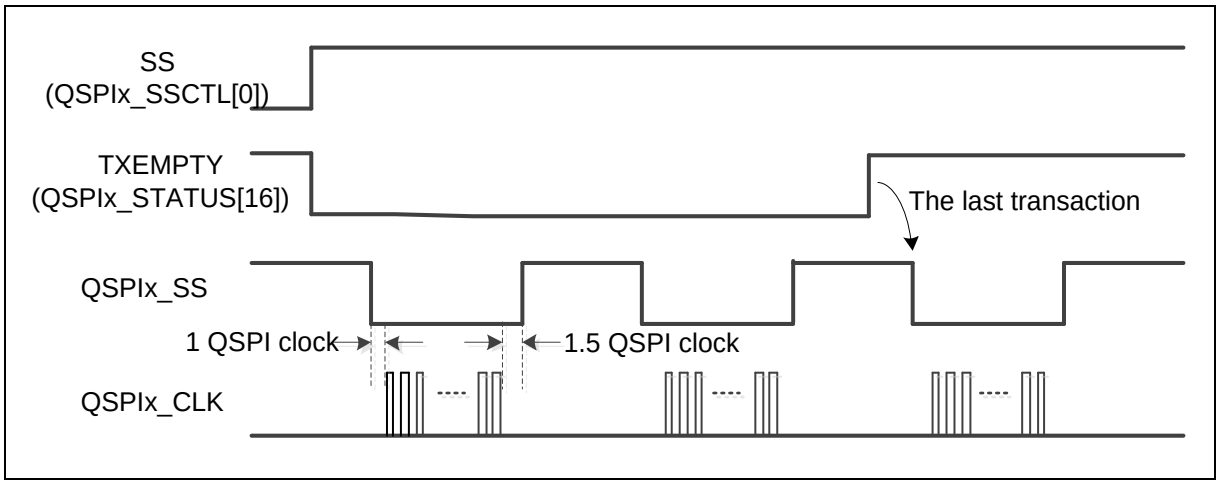


Figure 6.23-6 Automatic Slave Selection (SSACTPOL = 0, SUSPITV > 0x2)

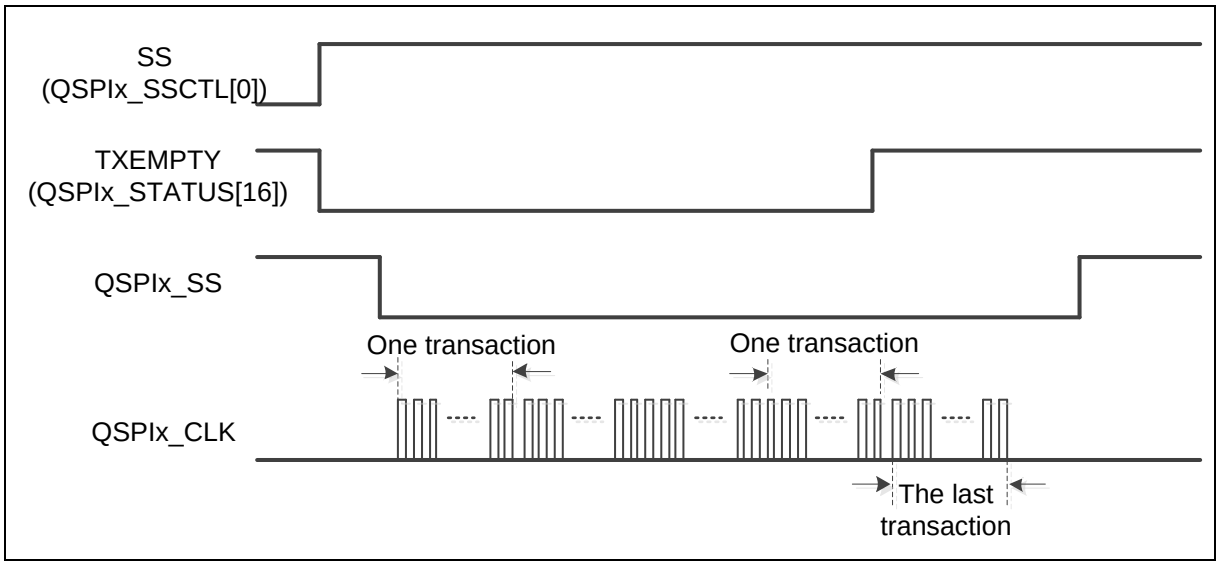


Figure 6.23-7 Automatic Slave Selection (SSACTPOL = 0, SUSPITV < 0x3)

6.23.5.3 Byte Reorder and Suspend Function

When the transfer is set as MSB first (LSB = 0) and the REORDER (QSPIx_CTL[19]) is set to 1, the

data stored in the TX buffer and RX buffer will be rearranged in the order as [Byte0, Byte1, Byte2, Byte3] in 32-bit transfer (DWIDTH = 0). The sequence of transmitted/received data will be Byte0, Byte1, Byte2, and then Byte3. If the DWIDTH is set as 24-bit transfer mode, the data in TX buffer and RX buffer will be rearranged as [unknown byte, Byte0, Byte1, Byte2]. The QSPI controller will transmit/receive data with the sequence of Byte0, Byte1 and then Byte2. Each byte will be transmitted/received with MSB first. The rule of 16-bit mode is the same as above. Byte Reorder function is only available when DWIDTH is configured as 16, 24, and 32 bits.

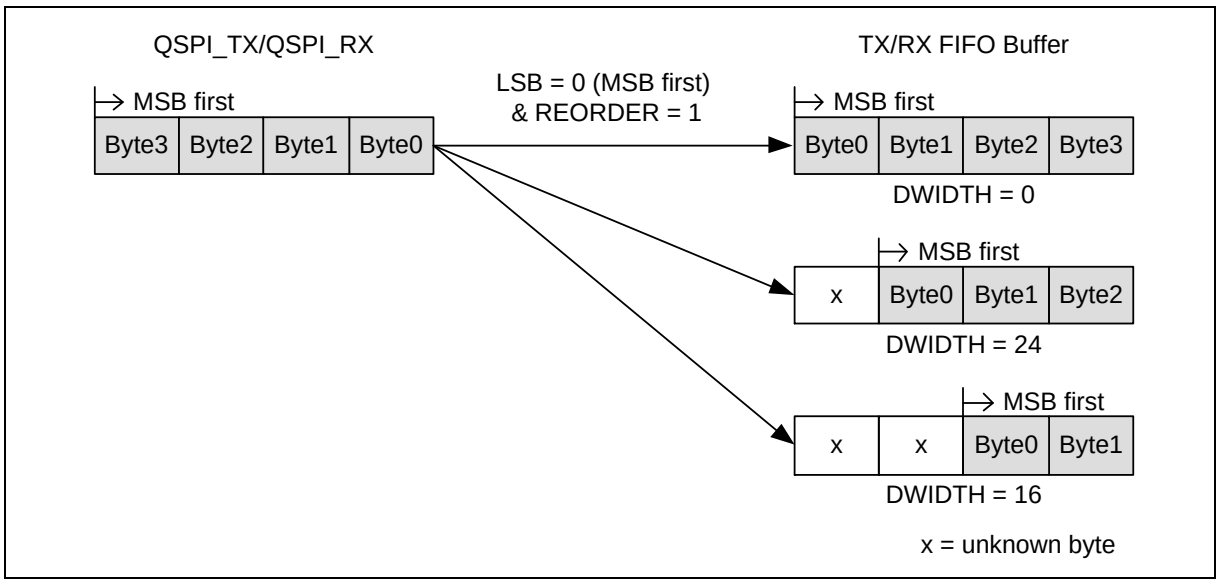


Figure 6.23-8 Byte Reorder Function

In Master mode, if REORDER (QSPIx_CTL[19]) is set to 1, a suspend interval of 0.5 ~ 15.5 QSPI clock periods will be inserted by hardware between two successive bytes in a transaction word. The suspend interval is configured in SUSPITV (QSPIx_CTL[7:4]).

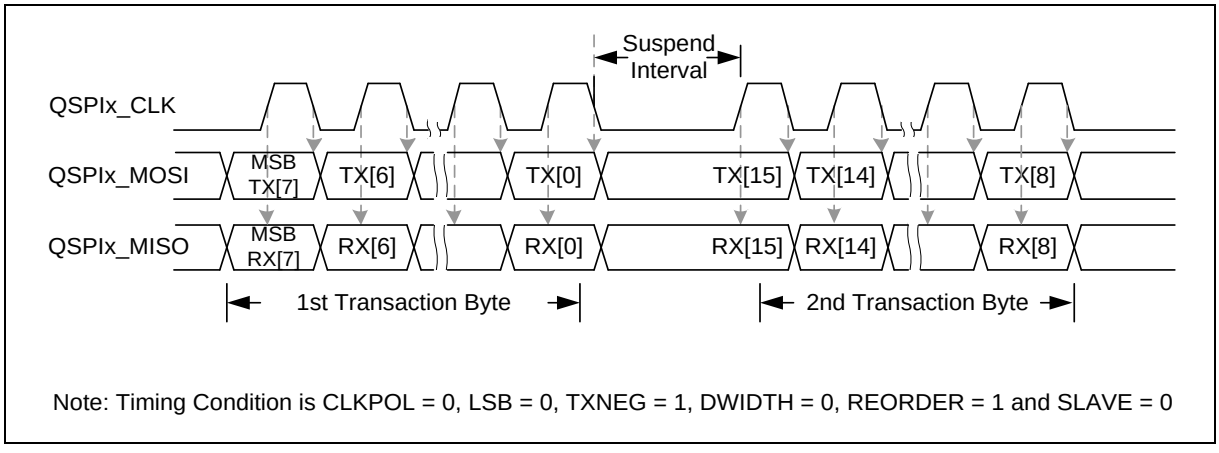


Figure 6.23-9 Timing Waveform for Byte Suspend

6.23.5.4 Half-Duplex Communication

The QSPI controller can communicate in half-duplex mode by setting HALFDPX (QSPIx_CTL[14]) bit. In half-duplex mode, there is only one data line for receiving or transmitting data direction which is defined by DATDIR (QSPIx_CTL[20]). In half-duplex configuration, the QSPIx_MISO pin is free for other applications and it can be configured as GPIO. Enabling or disabling the control bit HALFDPX

(QSPIx_CTL[14]) will produce TXFBCLR (QSPIx_FIFOCTL[9]) and RXFBCLR (QSPIx_FIFOCTL[8]) at the same time automatically.

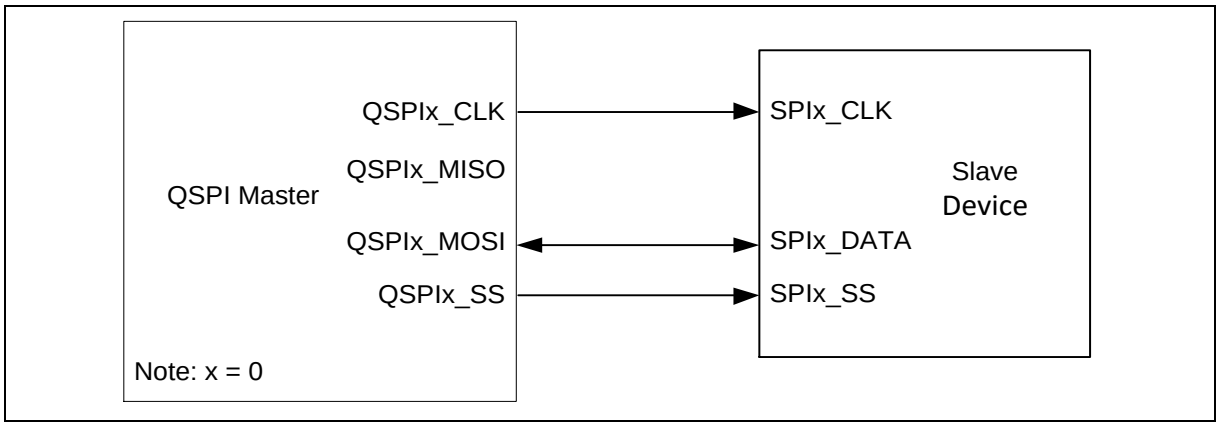


Figure 6.23-10 QSPI Half-Duplex Master Mode Application Block Diagram

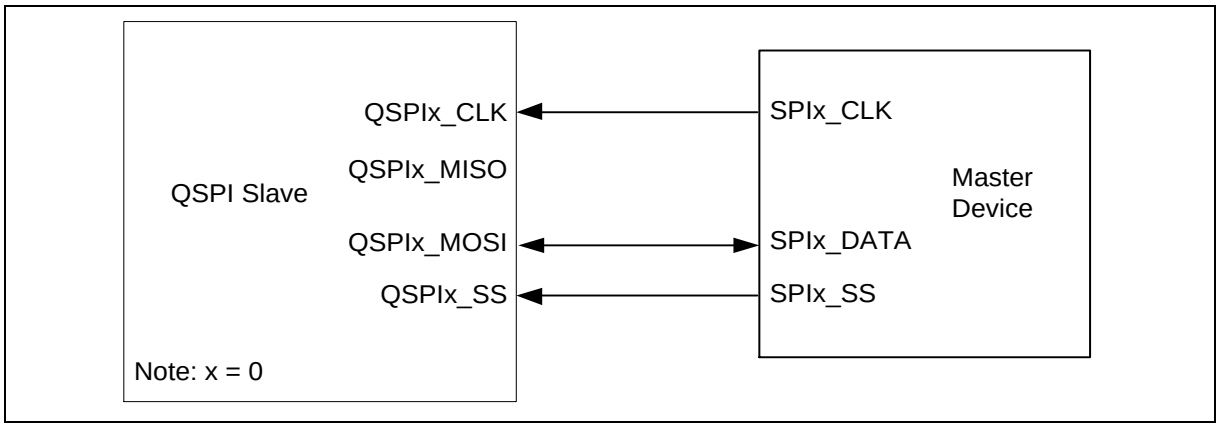


Figure 6.23-11 QSPI Half-Duplex Slave Mode Application Block Diagram

6.23.5.5 Receive-Only Mode

In QSPI Master device, it can communicate in receive-only mode by setting RXONLY (QSPIx_CTL[15]). In this configuration, the QSPI Master device will generate QSPI bus clock continuously as long as the receive-only mode is enabled for receiving data bit from SPI slave device. If AUTOSS (QSPIx_SSCTL[3]) is enabled in receive-only mode, QSPI Master will keep activating the slave select signal.

The remaining QSPIx_MOSI pin of QSPI Master device is not used for communication and can be configured as GPIO. The status BUSY (QSPIx_STATUS[0]) will be asserted in receive-only mode due to the generation of QSPI bus clock. Entering this mode will produce the TXFBCLR (QSPIx_FIFOCTL[9]) and RXFBCLR (QSPIx_FIFOCTL[8]) at the same time automatically. When user enables this mode, the output QSPI bus clock will be sent out after 6 peripheral clock cycles. In this mode, the data which has been written into transmit FIFO will be loaded into transmit shift register and sent out.

When user sets RXONLY (QSPIx_CTL[15]) to enable, QSPI RX data with data bit width of DWIDTH (QSPIx_CTL[12:8]) will be received into RX FIFO and QSPI clock will be sent to SPI slave device until RX FIFO is full.

For two-bit transfer mode TWOBIT (QSPIx_CTL[16]) is enabled, the QSPI master will send QSPI output clock to SPI slave and receive RX data until RX FIFO is full. After user reads RX data from RX

FIFO, QSPI master will send QSPI output clock to SPI slave and receive RX data again when RX FIFO counter RXCNT is less than or equal to 4.

6.23.5.6 Slave 3-Wire Mode

When SLV3WIRE (QSPIx_SSCTL[4]) is set by software to enable the Slave 3-Wire mode, the QSPI controller can work with no slave selection signal in Slave mode. The SLV3WIRE (QSPIx_SSCTL[4]) only takes effect in Slave mode. Only three pins, QSPIx_CLK, QSPIx_MISO, and QSPIx_MOSI, are required to communicate with a SPI master. The QSPIx_SS pin can be configured as a GPIO. When the SLV3WIRE (QSPIx_SSCTL[4]) is set to 1, the QSPI slave will be ready to transmit/receive data after the SPIEN (QSPIx_CTL[0]) is set to 1.

6.23.5.7 PDMA Transfer Function

QSPI controller supports PDMA transfer function.

When TXPDMAEN (QSPIx_PDMACTL[0]) is set to 1, the controller will issue request to PDMA controller to start the PDMA transmission process automatically.

When RXPDMAEN (QSPIx_PDMACTL[1]) is set to 1, the controller will start the PDMA reception process. QSPI controller will issue request to PDMA controller automatically when there is data in the RX FIFO buffer.

Note: QSPI supports single request PDMA (Read/Write) only, burst request PDMA is not supported.

6.23.5.8 Two-bit Transfer Mode

The QSPI controller also supports 2-bit transfer mode when setting TWOBIT (QSPIx_CTL[16]) to 1. In 2-bit transfer mode, the QSPI controller performs full duplex data transfer. In other words, the two serial data bits can be transmitted and received simultaneously.

For example, in Master mode, the even data (TX Data (n)) stored in the TX FIFO of QSPIx will be transmitted through the QSPIx_MOSI0 pin and the odd data (TX Data (n+1)) stored in the TX FIFO of QSPIx will be transmitted through the QSPIx_MOSI1 pin respectively. In the meanwhile, the even data received from QSPIx_MISO0 pin will be written to RX FIFO prior to the odd data received from QSPIx_MISO1 pin.

In Slave mode, the even and odd data stored in the TX FIFO of QSPIx will be transmitted through the QSPIx_MISO0 pin and QSPIx_MISO1 pin respectively. In the meanwhile, the RX FIFO of QSPIx will store the even data received from the QSPIx_MOSI0 pin and the odd data from QSPIx_MOSI1 pin respectively. The data sequence of FIFO buffers is the same as the Master mode.

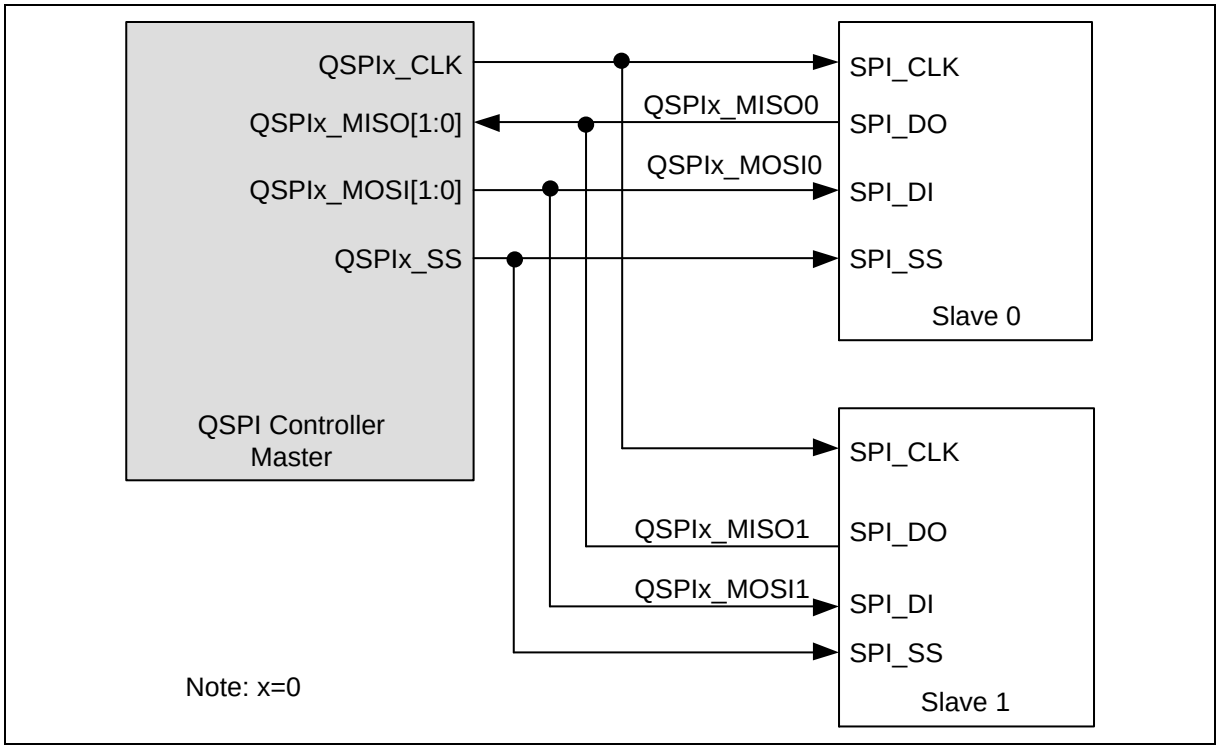


Figure 6.23-12 Two-bit Transfer Mode System Architecture

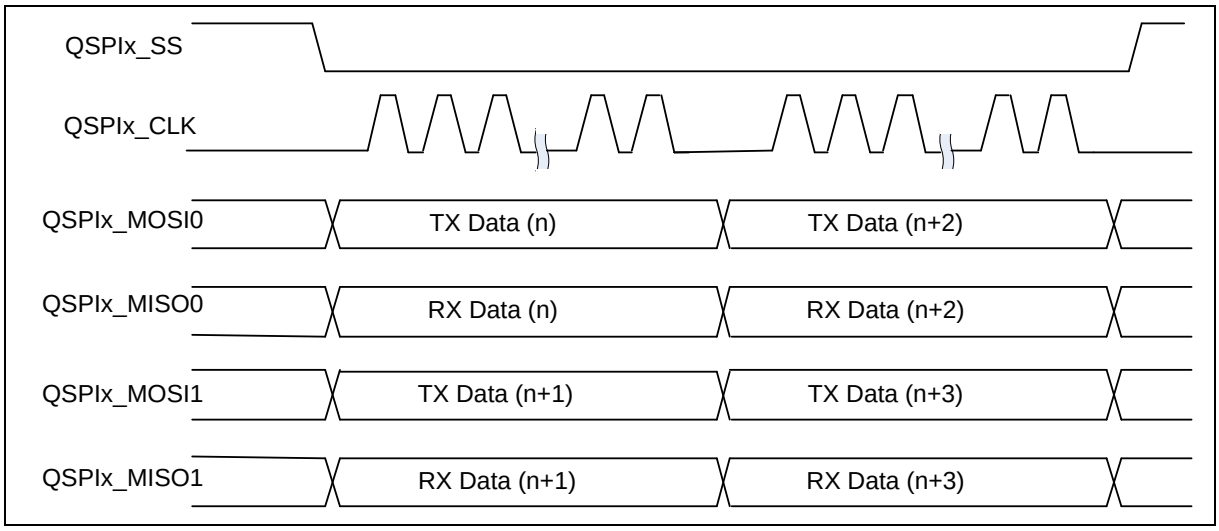


Figure 6.23-13 Two-bit Transfer Mode Timing (Master Mode)

6.23.5.9 Dual I/O Mode

The QSPI controller also supports Dual I/O transfer when setting the DUALIOEN (QSPIx_CTL[21]) to 1. Many general SPI Flashes support Dual I/O transfer. The DATDIR (QSPIx_CTL[20]) is used to define the direction of the transfer data. When the DATDIR bit is set to 1, the controller will send the data to external device. When the DATDIR bit is set to 0, the controller will read the data from the external device. This function supports 8, 16, 24, and 32 bits of length.

The Dual I/O mode is not supported when the Slave 3-Wire mode or the Byte Reorder function is enabled.

For Dual I/O mode, if both the DUALIOEN (QSPIx_CTL[21]) and DATDIR (QSPIx_CTL[20]) are set as 1, the QSPIx_MOSI0 is the even bit data output and the QSPIx_MISO0 will be set as the odd bit data output. If the DUALIOEN (QSPIx_CTL[21]) is set as 1 and DATDIR (QSPIx_CTL[20]) is set as 0, both the QSPIx_MISO0 and QSPIx_MOSI0 will be set as data input ports.

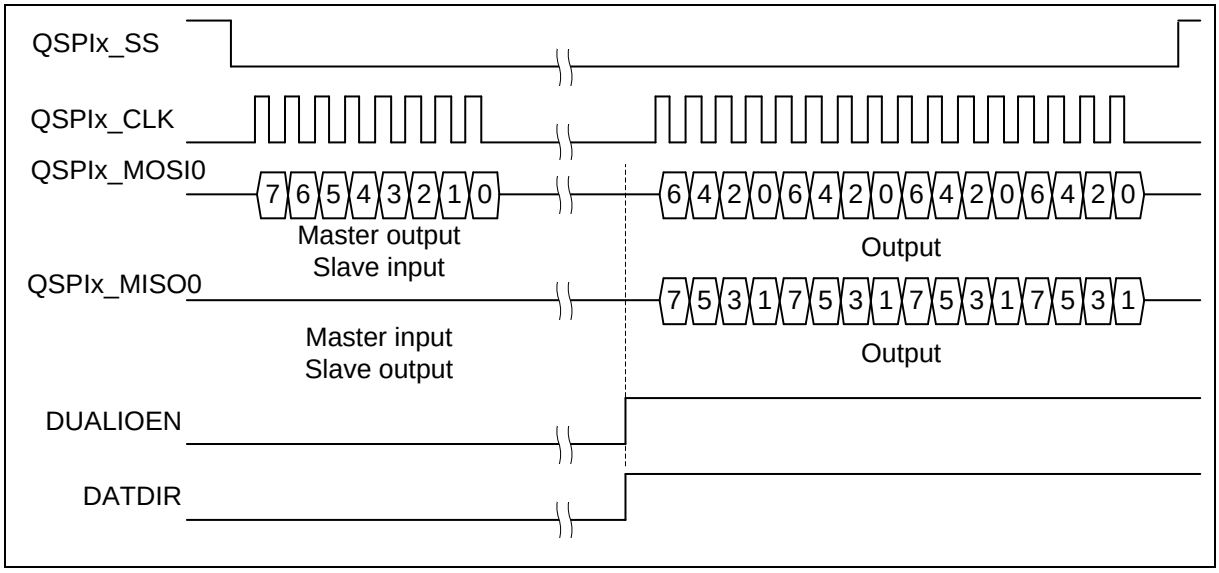


Figure 6.23-14 Bit Sequence of Dual Output Mode

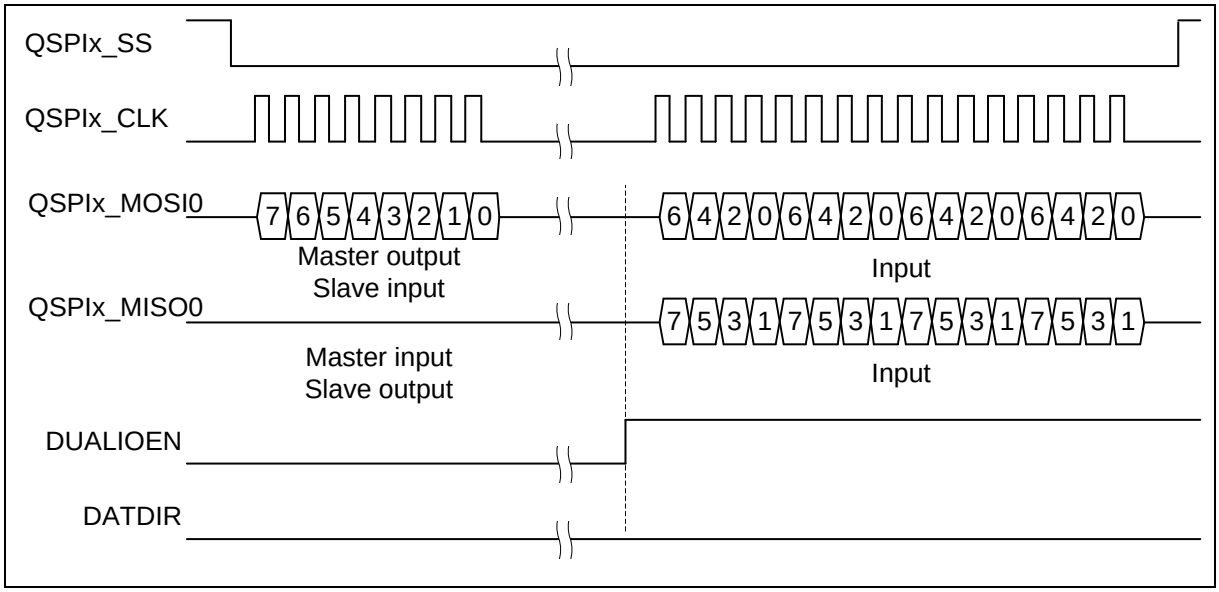


Figure 6.23-15 Bit Sequence of Dual Input Mode

6.23.5.10 Quad I/O Mode

The QSPI controller also supports Quad I/O transfer when setting the QUADIOEN (QSPIx_CTL[22]) to 1. Many general SPI Flashes support Quad I/O transfer. The DATDIR bit (QSPIx_CTL[20]) is used to define the direction of the transfer data. When the DATDIR (QSPIx_CTL[20]) is set to 1, the controller will send the data to external device. When the DATDIR (QSPIx_CTL[20]) is set to 0, the controller will read the data from the external device. This function supports 8, 16, 24, and 32 bits of length.

The Quad I/O mode is not supported when the Slave 3-Wire mode or the Byte Reorder function is

enabled. The DUALIOEN (QSPIx_CTL[21]) and QUADIOEN (QSPIx_CTL[22]) shall not be set to 1 simultaneously.

For Quad I/O mode, if both the QUADIOEN (QSPIx_CTL[22]) and DATDIR (QSPIx_CTL[20]) are set as 1, the QSPIx_MOSI0 and QSPIx_MOSI1 are the even bit data output and the QSPIx_MISO0 and QSPIx_MISO1 will be set as the odd bit data output. If the QUADIOEN (QSPIx_CTL[22]) is set as 1 and DATDIR (QSPIx_CTL[20]) is set as 0, all the QSPIx_MISO0, QSPIx_MISO1, QSPIx_MOSI0 and QSPIx_MOSI1 pins will be set as data input ports.

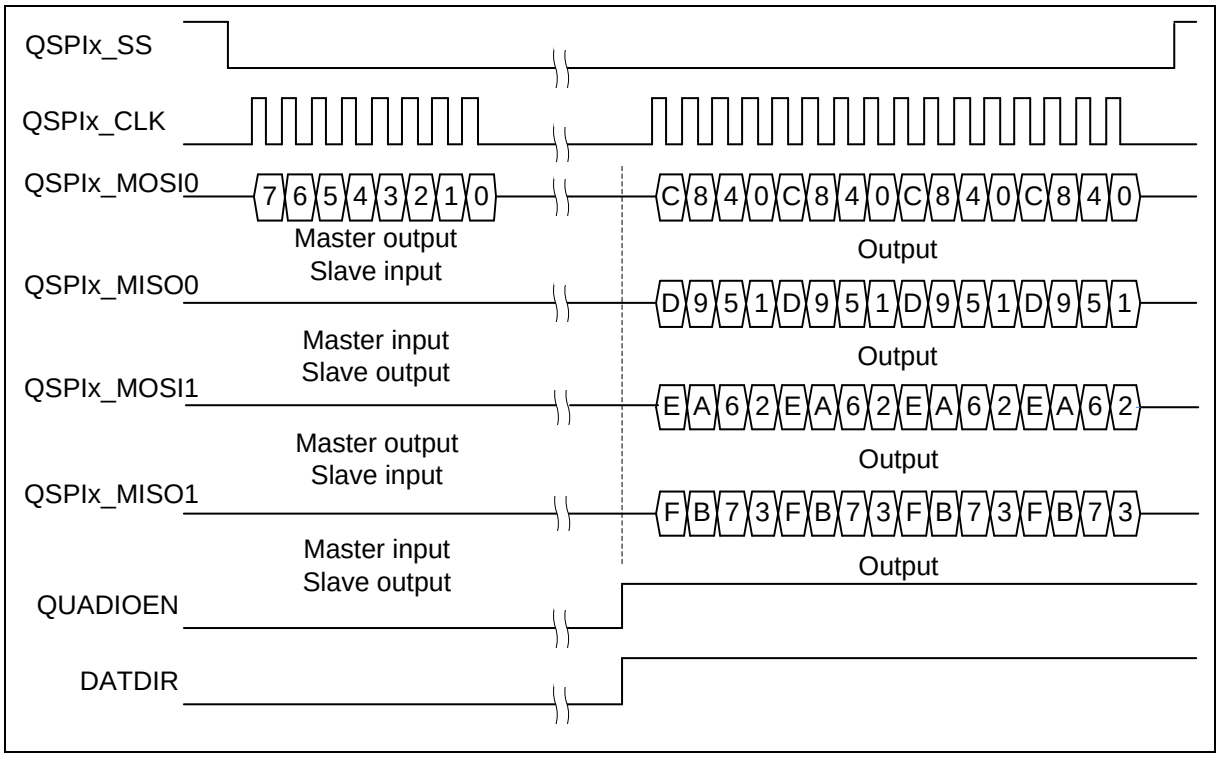


Figure 6.23-16 Bit Sequence of Quad Output Mode

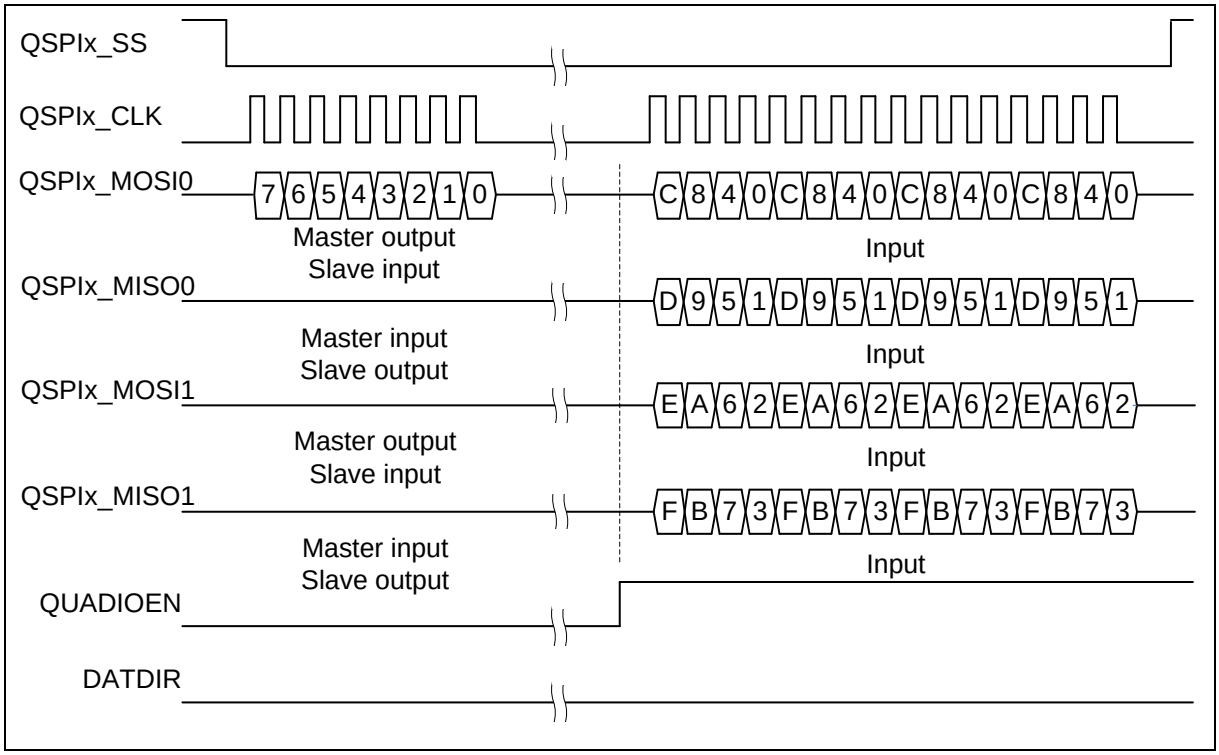


Figure 6.23-17 Bit Sequence of Quad Input Mode

6.23.5.11 Transmit Double Transfer Rate Mode

QSPI Master mode supports TX DTR mode (Transmit Double Transfer Rate Mode) when user sets control register TXDTREN (QSPIx_CTL[23]) to 1. The timing diagram of TX DTR mode is shown in Figure 6.23-18.

In TX DTR mode, the frequency of QSPI bus clock (QSPIx_CLK) equals to the half frequency of QSPI peripheral clock (QSPI_ECLK). TX output data of QSPI in TX DTR mode will be sent near rising edge and falling edge of QSPIx_CLK, where the rising edge and falling edge of QSPIx_CLK are the half period of QSPI peripheral clock (QSPI_ECLK) behind data driving edge of QSPI output data QSPIx_MOSI.

User can use control register CLKPOL (QSPIx_CTL[3]) to select idle polarity of QSPIx_CLK in TX DTR mode. When user sets CLKPOL to 0, the idle polarity of QSPIx_CLK is 0. When user sets CLKPOL to 1, the idle polarity of QSPIx_CLK is 1. In TX DTR mode, TXNEG (QSPIx_CTL[2]) equals to CLKPOL (QSPIx_CTL[3]), and SUSPITV (QSPIx_CTL[7:4]) equals to 0. In TX DTR mode of SPI master, user does not use RX data that is received from SPI slave device.

Note: QSPI Master mode supports TXDTR mode, and QSPI Slave mode does not support this mode.

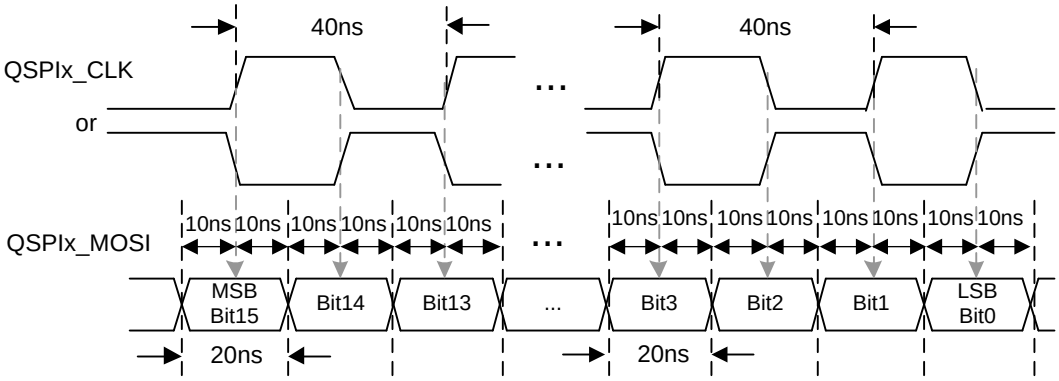


Figure 6.23-18 Timing Diagram of QSPI Output Data for Transmit Double Transfer Rate Mode

6.23.5.12 FIFO Buffer Operation

The QSPI controller is equipped with eight 32-bit wide transmit and receive FIFO buffers. The data stored in the transmit FIFO buffer will be read and sent out by the transmission control logic. If the transmit FIFO buffer is full, the TXFULL (QSPIx_STATUS[17]) will be set to 1. When the QSPI transmission logic unit draws out the last datum of the transmit FIFO buffer, so that the transmit FIFO buffer is empty, the TXEMPTY (QSPIx_STATUS[16]) will be set to 1. Note that the TXEMPTY (QSPIx_STATUS[16]) flag is set to 1 while the last transaction is still in progress. In Master mode, the BUSY (QSPIx_STATUS[0]) is set to 1 when the FIFO buffer is written any data or there is any transaction on the QSPI bus. (e.g. the slave selection signal is active and the QSPI controller is receiving data in Slave mode). It will set to 0 when the transmit FIFO is empty and the current transaction has done. Thus, the status of BUSY (QSPIx_STATUS[0]) should be checked by software to make sure whether the QSPI is in idle or not.

The receive control logic will store the QSPI input data into the receive FIFO buffer. There are FIFO related status bits, like RXEMPTY (QSPIx_STATUS[8]) and RXFULL (QSPIx_STATUS[9]), to indicate the current status of RX FIFO buffer.

The transmitting and receiving threshold can be configured by setting TXTH (QSPIx_FIFOCTL[30:28]) and RXTH (QSPIx_FIFOCTL[26:24]). When the count of valid data stored in transmit FIFO buffer is less than or equal to TXTH (QSPIx_FIFOCTL[30:28]) setting, TXTHIF (QSPIx_STATUS[18]) will be set to 1. When the count of valid data stored in receive FIFO buffer is larger than RXTH (QSPIx_FIFOCTL[26:24]) setting, RXTHIF (QSPIx_STATUS[10]) will be set to 1.

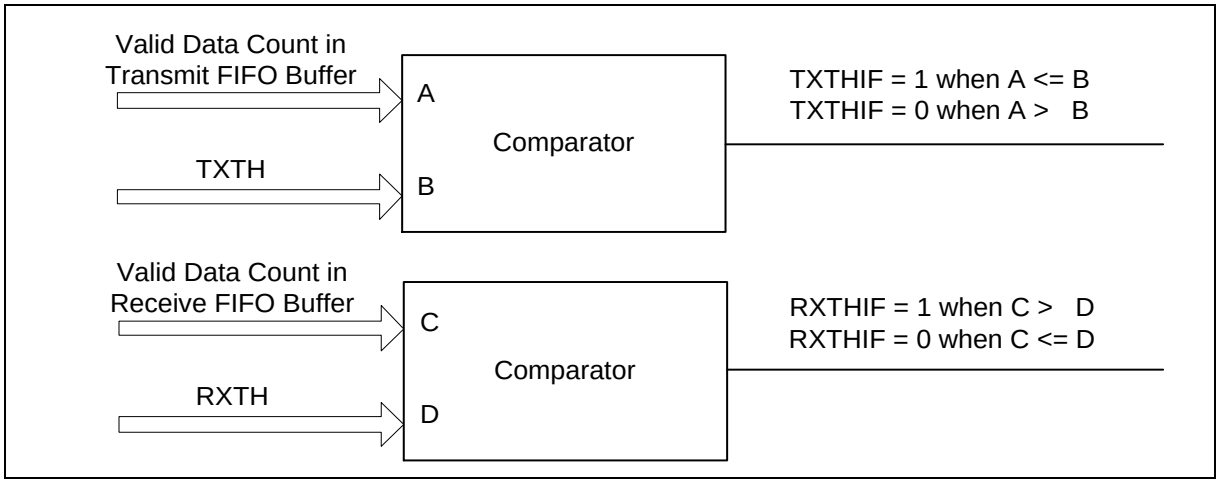


Figure 6.23-19 FIFO Threshold Comparator

In Master mode, when the first datum is written to the QSPIx_TX register, the TXEMPTY flag (QSPIx_STATUS[16]) will be cleared to 0. The transmission will start after 1 PCLK clock cycles and 6 peripheral clock cycles. User can write the next data into QSPIx_TX register immediately. The QSPI controller will insert a suspend interval between two successive transactions. The period of suspend interval is decided by the setting of SUSPITV (QSPIx_CTL[7:4]). If the SUSPITV (QSPIx_CTL[7:4]) equals 0, QSPI controller can perform continuous transfer. User can write data into QSPIx_TX register as long as the TXFULL (QSPIx_STATUS[17]) is 0.

In the Example 1 of Figure 6.23-20, it indicates the updated condition of TXEMPTY (QSPIx_STATUS[16]) and the relationship among the FIFO buffer, shift register and the skew buffer. The TXEMPTY (QSPIx_STATUS[16]) is set to 0 when the Data0 is written into the FIFO buffer. The Data0 will be loaded into the shift register by the core logic and the TXEMPTY (QSPIx_STATUS[16]) will be to 1. The Data0 in shift register will be shift into skew buffer by bit for transmission until the transfer is done.

In the Example 2 of Figure 6.23-20, it indicates the updated condition of TXFULL (QSPIx_STATUS[17]) when there are 8 data in the FIFO buffer and the next data of Data9 does not be written into the FIFO buffer when the TXFULL = 1.

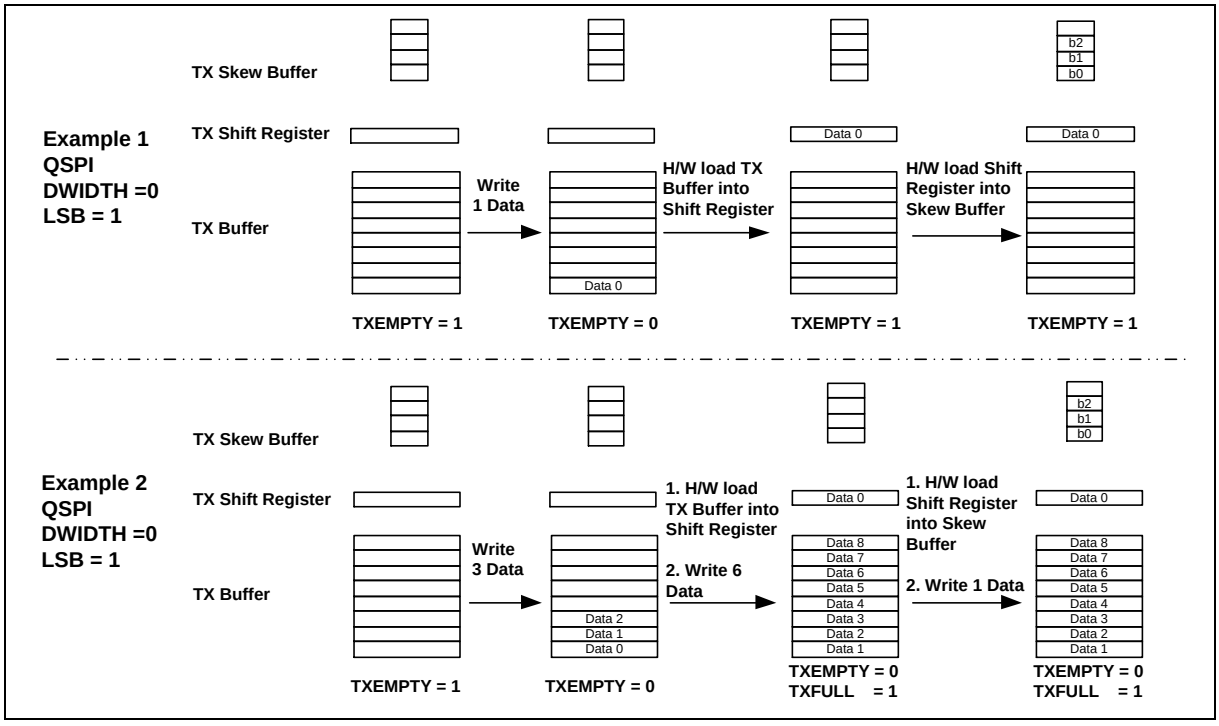


Figure 6.23-20 Transmit FIFO Buffer Example

The subsequent transactions will be triggered automatically if the transmitted data are updated in time. If the QSPIx_TX register is not updated after all data transfer are done, the transfer will stop.

In Master mode, during receiving operation, the serial data are received from QSPIx_MISO pin and stored to receive FIFO buffer.

The received data (Data0's b0, b1, …b31) is stored into skew buffer first according the serial clock (QSPIx_CLK) and then it is shift into the shift register bit by bit. The core logic will load the data in shift register into FIFO buffer when the received data bit count reach the value of DWIDTH (QSPIx_CTL[12:8]). The RXEMPTY (QSPIx_STATUS[8]) will be cleared to 0 while the receive FIFO buffer contains unread data (see the Example 1 of Receive FIFO Buffer Example in Figure 6.23-21). The received data can be read by software from QSPIx_RX register as long as the RXEMPTY (QSPIx_STATUS[8]) is 0. If the receive FIFO buffer contains 8 unread data, the RXFULL (QSPIx_STATUS[9]) will be set to 1 (see the Example 2 of Receive FIFO Buffer Example in Figure 6.23-21).

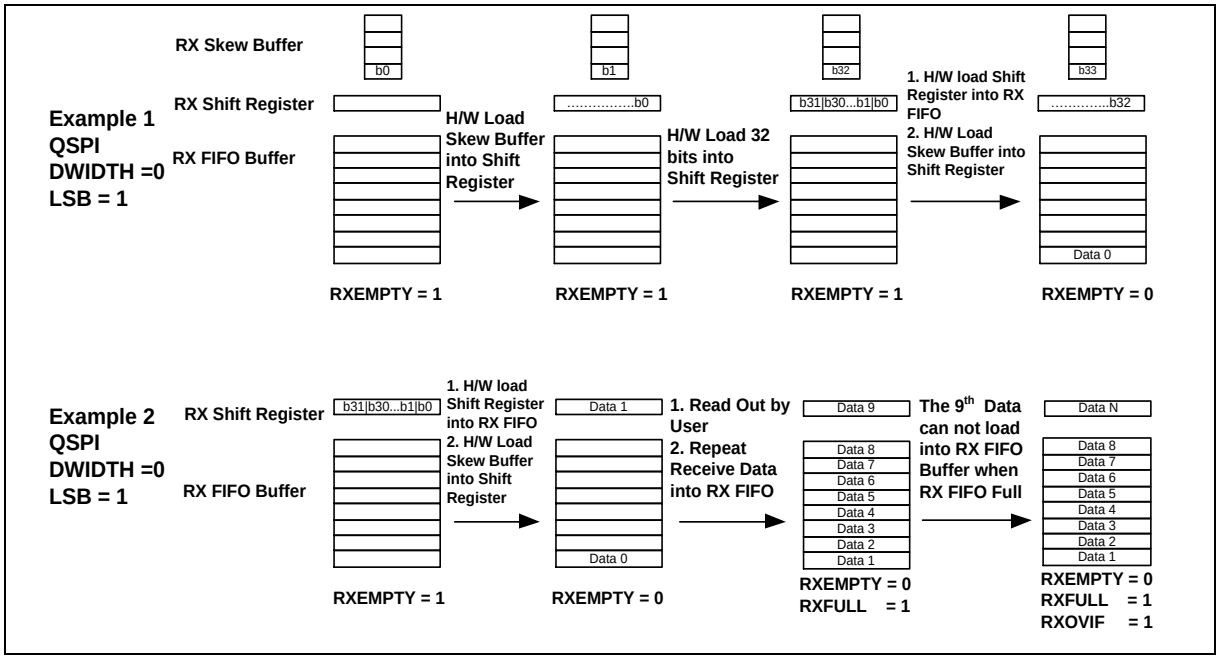


Figure 6.23-21 Receive FIFO Buffer Example

In Slave mode, during transmission operation, when data is written to the QSPIx_TX register by software, the data will be loaded into transmit FIFO buffer and the TXEMPTY (QSPIx_STATUS[16]) will be set to 0. The transmission will start when the slave device receives clock signal from master. Data can be written to QSPIx_TX register as long as the TXFULL (QSPIx_STATUS[17]) is 0. After all data have been drawn out by the QSPI transmission logic unit and the QSPIx_TX register is not updated by software, the TXEMPTY (QSPIx_STATUS[16]) will be set to 1.

If there is no any data written to the QSPIx_TX register, the transmit underflow interrupt flag, TXUFIF (QSPIx_STATUS[19]) will be set to 1 when the slave selection signal is active. The output data will be held by TXUFPOL (QSPIx_FIFOCTL[6]) setting during this transfer until the slave selection signal goes to inactive state. When the transmit underflow event occurs, the slave under run interrupt flag, SLVURIF (QSPIx_STATUS[7]), will be set to 1 as QSPIx_SS goes to inactive state.

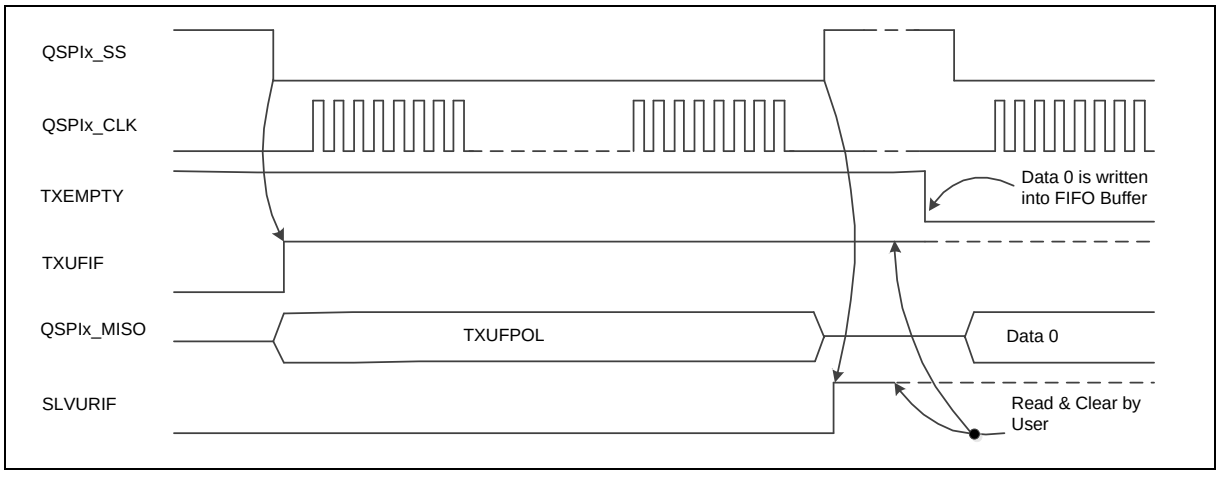


Figure 6.23-22 TX Underflow Event and Slave Under Run Event

In 2-bit transfer mode, the transmit data is loaded into shift register after 2 datum have been written

into the TX FIFO buffer. It uses two shift registers and two 4-level skew buffers concurrently. The detail timing of 2-bit transfer mode, please refer to the section of Two-bit Transfer Mode.

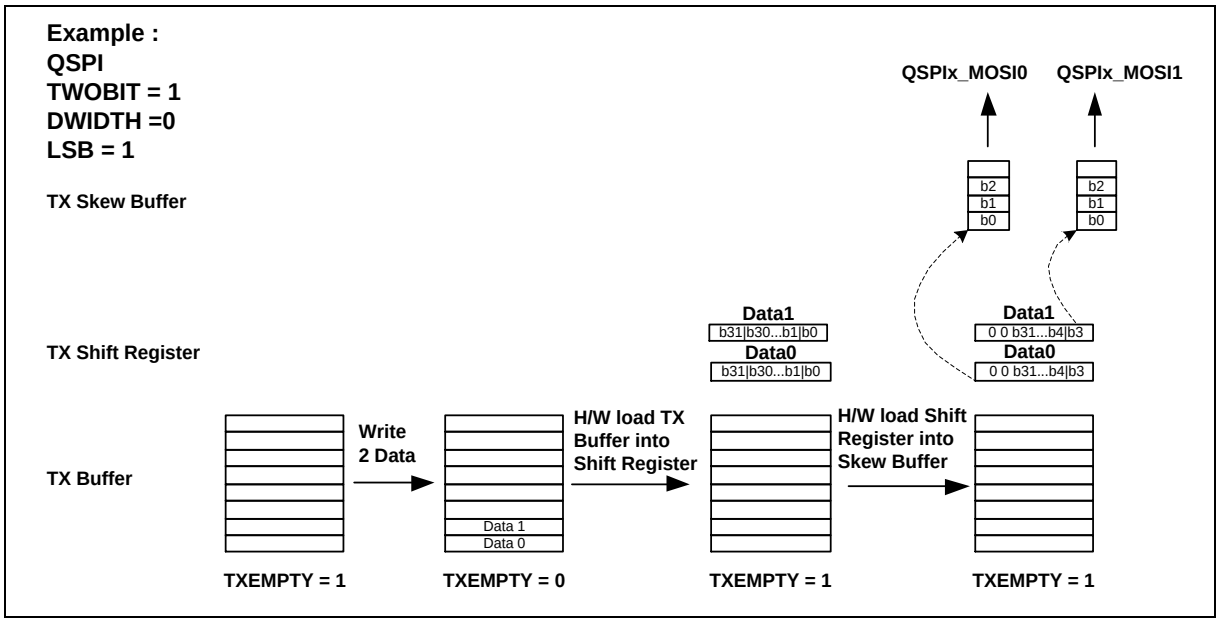


Figure 6.23-23 Two-bit Transfer Mode FIFO Buffer Example

In QSPI Slave 3-Wire mode, the first 2-bit data is un-predicted (keep on the level of last bit in previously transfer) if the data is written into TX FIFO among 3 peripheral clock cycles before the QSPI bus clock is presented. The other bits are held by TXUFPOL (QSPIx_FIFOCTL[6]) because there is TX underflow event. The written data will be transmitted in the next transfer.

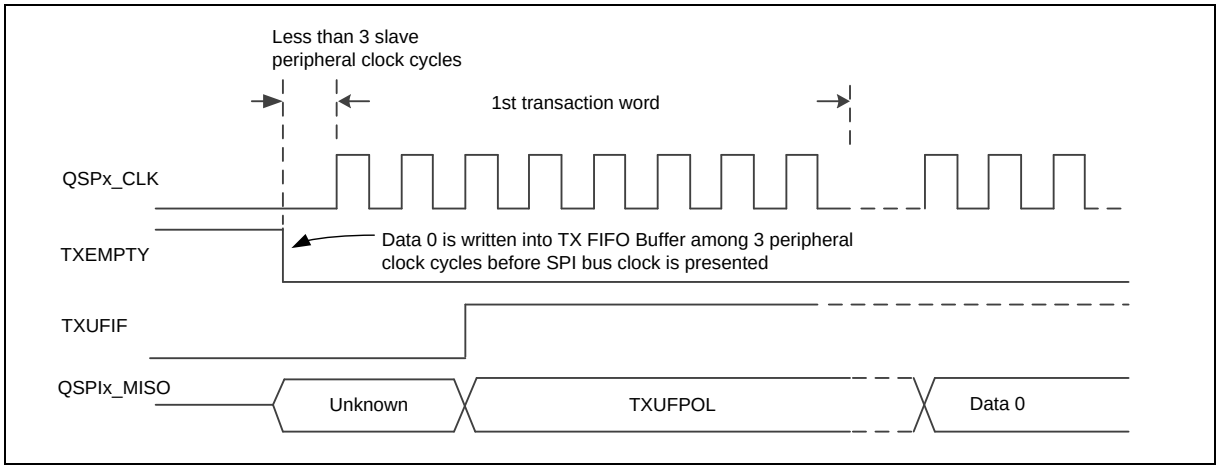


Figure 6.23-24 TX Underflow Event (QSPI0 Slave 3-Wire Mode Enabled)

In Slave mode, during receiving operation, the serial data is received from QSPIx_MOSI pin and stored to QSPIx_RX register. The reception mechanism is similar to Master mode reception operation. If the receive FIFO buffer contains 8 unread data, the RXFULL (QSPIx_STATUS[9]) will be set to 1 and the RXOVIF (QSPIx_STATUS[11]) will be set to 1 if there is more serial data received from QSPIx_MOSI and follow-up data will be dropped (refer to the Receive FIFO Buffer Example figure).

If the receive bit count mismatch with the DWIDTH (QSPIx_CTL[12:8]) when the slave selection line goes to inactive state, the SLVBEIF (QSPIx_STATUS[6]) will be set to 1 in QSPI Slave mode. RX data

will be written into RX FIFO and SLVBENUM (QSPIx_STATUS2[29:24]) will be updated when SLVBERX (QSPIx_FIFOCTL[10]) is enabled and QSPI slave bit count error event happened. RX data will be dropped and SLVBENUM (QSPIx_STATUS2[29:24]) will be fixed to 0x0 when the control register SLVBERX (QSPIx_FIFOCTL[10]) is disabled and QSPI slave bit count error event happened. The status register SLVBENUM (QSPIx_STATUS2[29:24]) indicates that effective bit number of uncompleted RX data when QSPI slave bit count error happened. In Figure 6.23-25, the timing diagram of QSPI slave bit count error and SLVBENUM is shown, and SLVBENUM will be updated to 0x8 when QSPI slave device receives 8 bits data and DWIDTH (QSPIx_CTL[12:8]) is set to 16 bits at QSPI bit count error event. The QSPI slave bit count error event (SLVBEIF) will generates RX FIFO write operation pulse to write RX data from RX shift register to RX FIFO, and RX FIFO count (RXCNT) will update to one.

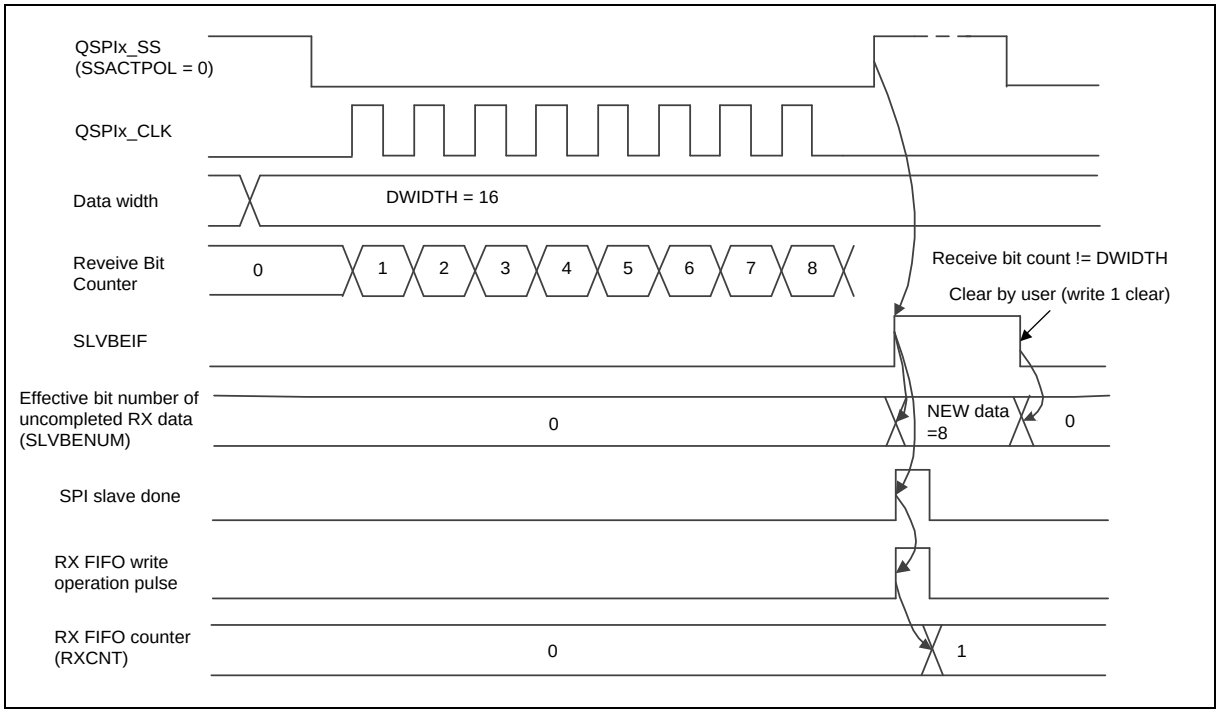


Figure 6.23-25 Slave Mode Bit Count Error and Effective Bit Number of Uncompleted RX Data

When the Slave select signal is active and the value of SLVTOCNT (QSPIx_SSCTL[31:16]) is not 0, the Slave time-out counter in the QSPI controller logic will start after the serial clock input. This counter will be cleared after one transaction done or the SLVTOCNT is set to 0. If the value of the time-out counter is equal to the value of SLVTOCNT before one transaction done, the slave time-out event occurs and the SLVTOIF (QSPIx_STATUS[5]) will be set to 1.

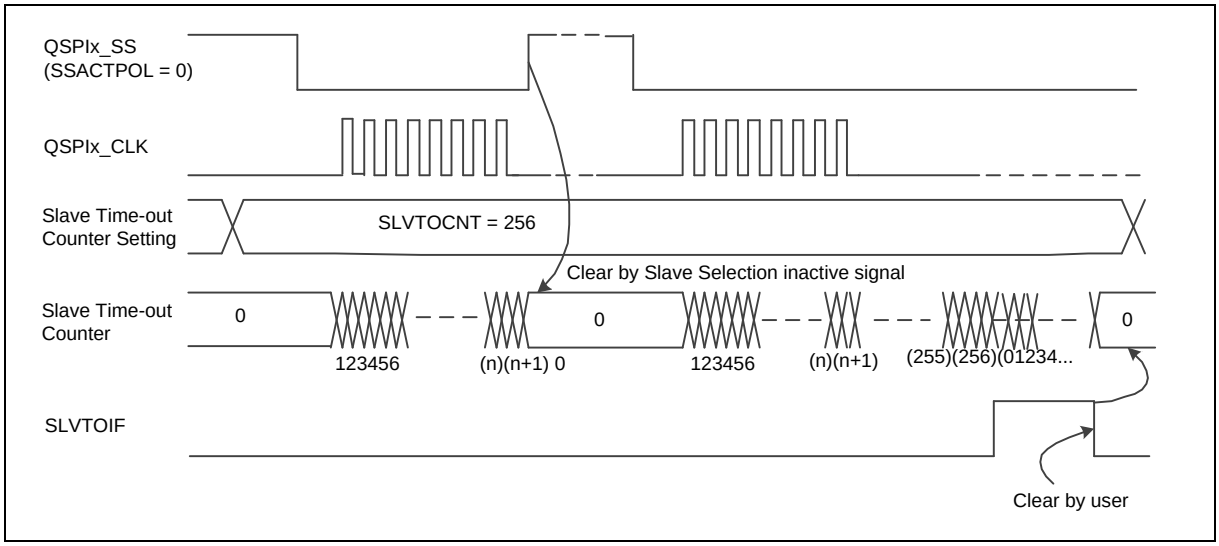


Figure 6.23-26 Slave Time-out Event

A receive time-out function is built-in in this controller. When the receive FIFO is not empty and no read operation in receive FIFO over 64 QSPI peripheral clock periods in Master mode or over 576 QSPI peripheral clock periods in Slave mode, the receive time-out occurs and the RXTOIF (QSPIx_STATUS[12]) will be set to 1. When the receive FIFO is read by user, the time-out status will be cleared automatically.

6.23.5.13 Interrupt

- QSPI unit transfer interrupt

  As the QSPI controller finishes a unit transfer, the unit transfer interrupt flag UNITIF (QSPIx_STATUS[1]) will be set to 1. The unit transfer interrupt event will generate an interrupt to CPU if the unit transfer interrupt enable bit UNITIEN (QSPIx_CTL[17]) is set. The unit transfer interrupt flag can be cleared only by writing 1 to it.

- QSPI slave selection active/inactive interrupt

  In Slave mode, the slave selection active/inactive interrupt flag, SSACTIF (QSPIx_STATUS[2]) and SSINAIF (QSPIx_STATUS[3]), will be set to 1 when the SPIEN (QSPIx_CTL[0]) and SLAVE (QSPIx_CTL[18]) are set to 1 and the slave selection signal goes to active/inactive state. The QSPI controller will issue an interrupt if the SSACTIEN (QSPIx_SSCTL[12]) or SSINAIEN (QSPIx_SSCTL[13]), is set to 1.

- Slave time-out interrupt

  In Slave mode, there is slave time-out function for user to know that there is serial clock input but one transaction is not finished over the period of SLVTOCNT (QSPIx_SSCTL[31:16]) basing on Slave peripheral clock.

  When the slave selection signal is active and the value of SLVTOCNT (QSPIx_SSCTL[31:16]) is not 0, the slave time-out counter in the QSPI controller logic will start after the serial clock input. This counter will be cleared after one transaction done or the SLVTOCNT (QSPIx_SSCTL[31:16]) is set to 0. If the value of the time-out counter is greater than or equal to the value of SLVTOCNT (QSPIx_SSCTL[31:16]) before one transaction done, the slave time-out event occurs and the SLVTOIF (QSPIx_STATUS[5]) will be set to 1. The QSPI controller will issue an interrupt if the SLVTOIEN (QSPIx_SSCTL[5]) is set to 1.

- Slave bit count error interrupt

In Slave mode, if the transmit/receive bit count mismatch with the DWIDTH (QSPIx_CTL[12:8]) when the slave selection line goes to inactive state, the SLVBEIF (QSPIx_STATUS[6]) will be set to 1. The uncompleted transaction will be dropped from TX shift register. When the control register SLVBERX (QSPIx_FIFOCTL[10]) is disabled and QSPI slave bit count error event happened, the uncompleted transaction data will be dropped from RX shift registers. When control register SLVBERX (QSPIx_FIFOCTL[10]) is enabled and QSPI slave bit count error event happened, the uncompleted transaction data will be written from RX shift registers into RX FIFO. The QSPI controller will issue an interrupt if the SLVBEIEN (QSPIx_SSCTL[8]) is set to 1.

Note: If the slave selection signal is active but there is no any serial clock input, the SLVBEIF (QSPIx_STATUS[6]) will be set to 1 when the slave selection signal goes to inactive state.

- TX underflow interrupt

  In QSPI Slave mode, if there is no any data is written to the QSPIx_TX register, the TXUFIF (QSPIx_STATUS[19]) will be set to 1 when the slave selection signal is active. The QSPI controller will issue a TX underflow interrupt if the TXUFIEN (QSPIx_FIFOCTL[7]) is set to 1.

  Note: If underflow event occurs in QSPI Slave mode, there are two conditions which make QSPI Slave mode return to idle state and then goes for next transfer: (1) set TXRST to 1 (2) slave select signal is changed to inactive state while SLV3WIRE=0.

- Slave TX under run interrupt

  If the TX underflow event occurs, the SLVURIF (QSPIx_STATUS[7]) will be set to 1 when QSPIx_SS goes to inactive state. The QSPI controller will issue a TX under run interrupt if the SLVURIEN (QSPIx_SSCTL[9]) is set to 1.

  Note: In Slave 3-Wire mode, the slave selection signal is considered active all the time so that user shall poll the TXUFIF (QSPIx_STATUS[19]) to know if there is TX underflow event or not.

- Receive Overrun interrupt

  In Slave mode, if the receive FIFO buffer contains 8 unread data, the RXFULL (QSPIx_STATUS[9]) will be set to 1 and the RXOVIF (QSPIx_STATUS[11]) will be set to 1 if there is more serial data received from QSPI bus and follow-up data will be dropped. The QSPI controller will issue an interrupt if the RXOVIEN (QSPIx_FIFOCTL[5]) is set to 1.

- Receive FIFO time-out interrupt

  If there is a received data in the FIFO buffer and it is not read by software over 64 QSPI peripheral clock periods in Master mode or over 576 QSPI peripheral clock periods in Slave mode, it will send a RX time-out interrupt to the system if the RX time-out interrupt enable bit, RXTOIEN (QSPIx_FIFOCTL[4]), is set to 1.

- Transmit FIFO interrupt

  In FIFO mode, if the valid data count of the transmit FIFO buffer is less than or equal to the setting value of TXTH (QSPIx_FIFOCTL[30:28]), the transmit FIFO interrupt flag TXTHIF (QSPIx_STATUS[18]) will be set to 1. The QSPI controller will generate a transmit FIFO interrupt to the system if the transmit FIFO interrupt enable bit, TXTHIEN (QSPIx_FIFOCTL[3]), is set to 1.

- Receive FIFO interrupt

  In FIFO mode, if the valid data count of the receive FIFO buffer is larger than the setting value of RXTH (QSPIx_FIFOCTL[26:24]), the receive FIFO interrupt flag RXTHIF

(QSPIx_STATUS[10]) will be set to 1. The QSPI controller will generate a receive FIFO interrupt to the system if the receive FIFO interrupt enable bit, RXTHIEN (QSPIx_FIFOCTL[2]), is set to 1.

6.23.6 Timing Diagram

The active state of slave selection signal can be defined by setting the SSACTPOL (QSPIx_SSCTL[2]). The QSPI clock which is in idle state can be configured as high or low state by setting the CLKPOL (QSPIx_CTL[3]). It also provides the bit length of a transaction word in DWIDTH (QSPIx_CTL[12:8]), and transmitting/receiving data from MSB or LSB first in LSB (QSPIx_CTL[13]). User can also select which edge of QSPI clock to transmit/receive data in TXNEG/RXNEG (QSPIx_CTL[2:1]). Four QSPI timing diagrams for master/slave operations and the related settings are shown below.

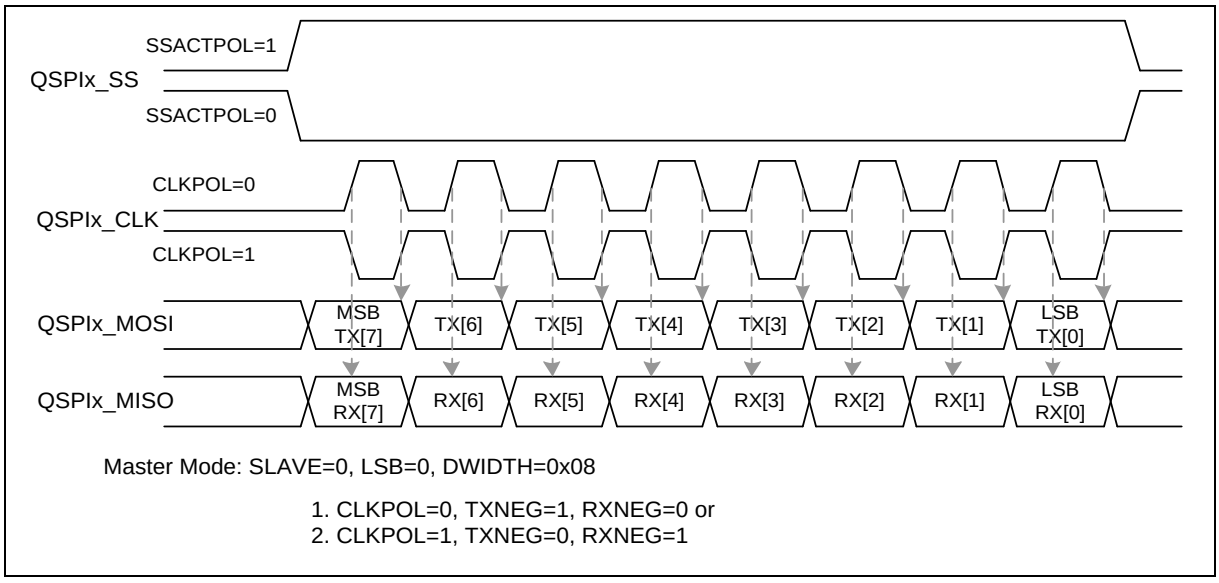


Figure 6.23-27 QSPI Timing in Master Mode

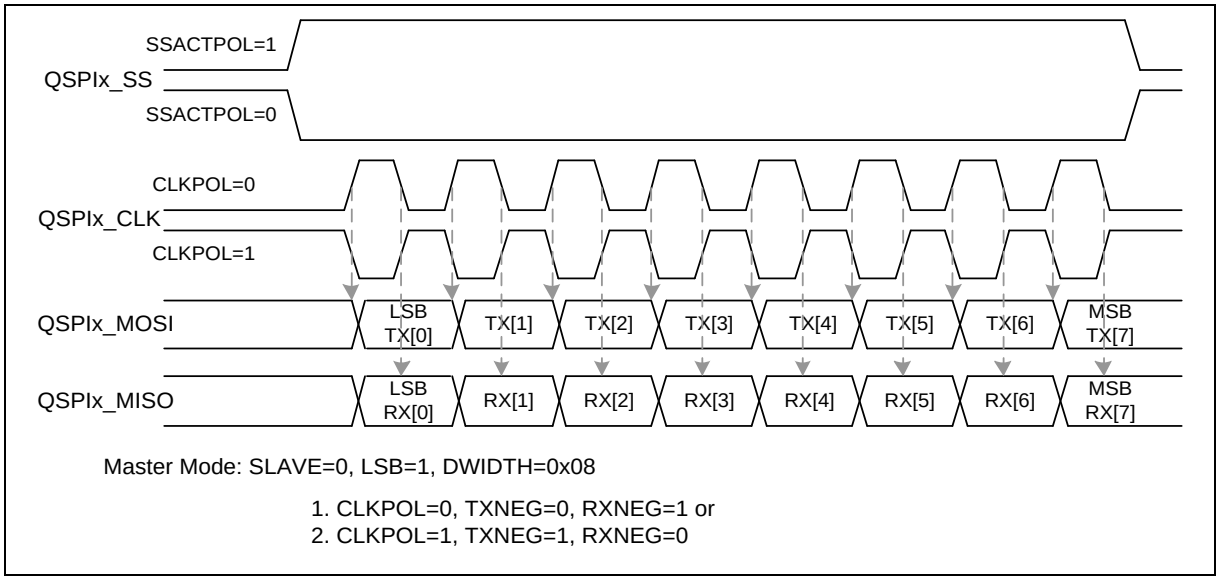


Figure 6.23-28 QSPI Timing in Master Mode (Alternate Phase of QSPIx_CLK)

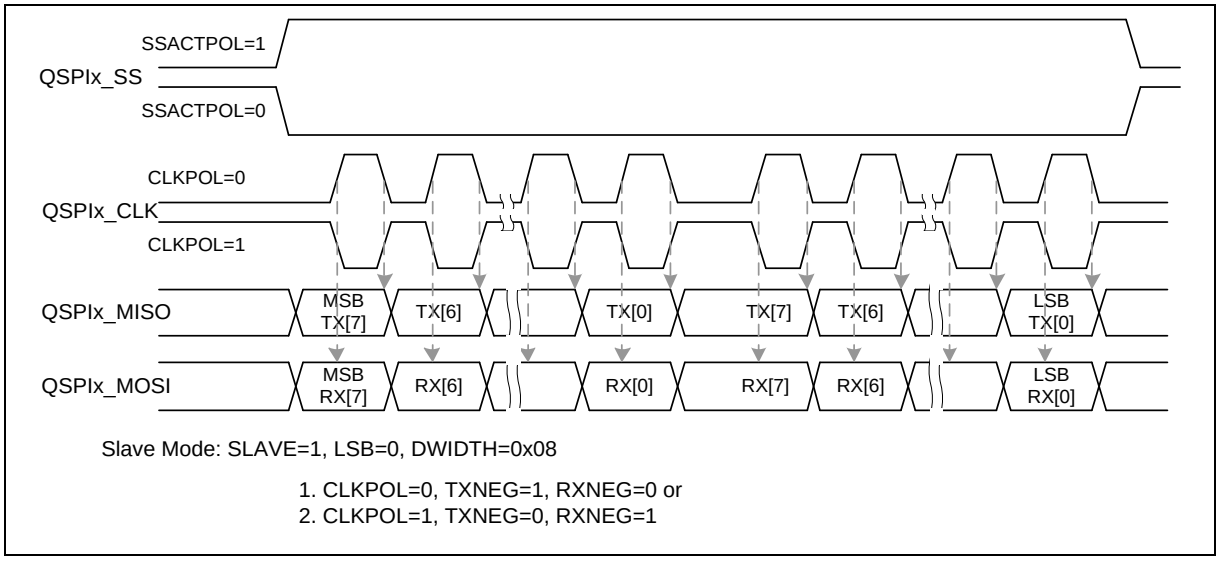


Figure 6.23-29 QSPI Timing in Slave Mode

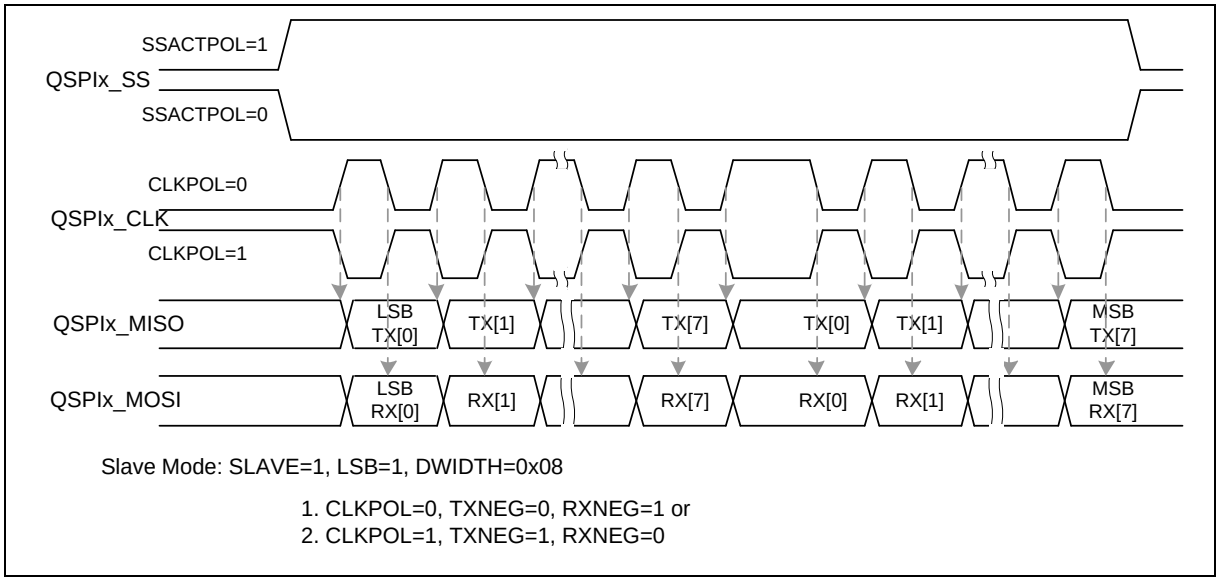


Figure 6.23-30 QSPI Timing in Slave Mode (Alternate Phase of QSPIx_CLK)

6.23.7 Programming Examples

Example 1:

The QSPI controller is set as a full-duplex master to access an off-chip slave device with the following specifications:

- Data bit is latched on positive edge of QSPI bus clock.
- Data bit is driven on negative edge of QSPI bus clock.
- Data is transferred from MSB first.
- QSPI bus clock is idle at low state.
- Only one byte of data to be transmitted/received in a transaction.

- Uses the QSPI slave select pin to connect with an off-chip slave device. The slave selection signal is active low.

The operation flow is as follows:

1. Set DIVIDER (QSPIx_CLKDIV[8:0]) to determine the output frequency of QSPI clock.
2. Write the QSPIx_SSCTL register a proper value for the related settings of Master mode:
   1) Clear AUTOSS (QSPIx_SSCTL[3]) to 0 to disable the Automatic Slave Selection function.
   2) Configure slave selection signal as active low by clearing SSACTPOL (QSPIx_SSCTL[2]) to 0.
   3) Enable slave selection signal by setting SS (QSPIx_SSCTL[0]) to 1 to activate the off-chip slave device.
3. Write the related settings into the QSPIx_CTL register to control the QSPI master actions.
   1) Configure this QSPI controller as master device by setting SLAVE (QSPIx_CTL[18]) to 0.
   2) Force the QSPI clock idle state at low by clearing CLKPOL (QSPIx_CTL[3]) to 0.
   3) Select data transmitted on negative edge of QSPI bus clock by setting TXNEG (QSPIx_CTL[2]) to 1.
   4) Select data latched on positive edge of QSPI bus clock by clearing RXNEG (QSPIx_CTL[1]) to 0.
   5) Set the bit length of a transaction as 8-bit in DWIDTH bit field (QSPIx_CTL[12:8] = 0x08).
   6) Set MSB transfer first by clearing LSB (QSPIx_CTL[13]) to 0.
4. Set SPIEN (QSPIx_CTL[0]) to 1 to enable the data transfer with the QSPI interface.
5. If this QSPI master attempts to transmit (write) one byte data to the off-chip slave device, write the byte data that will be transmitted into the QSPIx_TX register.
6. Waiting for QSPI interrupt if the UNITIEN (QSPIx_CTL[17]) is set to 1, or just polling the unit transfer interrupt flag UNITIF (QSPIx_STATUS[1]).
7. Read out the received one byte data from QSPIx_RX register.
8. Go to 5) to continue another data transfer or set SS (QSPIx_SSCTL[0]) to 0 to inactivate the off-chip slave device.

Example 2:

The QSPI controller is set as a full-duplex slave device and connects with an off-chip master device. The off-chip master device communicates with the on-chip QSPI slave controller through the QSPI interface with the following specifications:

- Data bit is latched on positive edge of QSPI bus clock.
- Data bit is driven on negative edge of QSPI bus clock.
- Data is transferred from LSB first.
- QSPI bus clock is idle at high state.
- Only one byte of data to be transmitted/received in a transaction.
- Slave selection signal is active high.

The operation flow is as follows:

1. Write the QSPIx_SSCTL register a proper value for the related settings of Slave mode.
2. Select high level for the input of slave selection signal by setting SSACTPOL

(QSPIx_SSCTL[2]) to 1.

3. Write the related settings into the QSPIx_CTL register to control this QSPI slave actions
   1) Set the QSPI controller as slave device by setting SLAVE (QSPIx_CTL[18]) to 1.
   2) Select the QSPI clock idle state at high by setting CLKPOL (QSPIx_CTL[3]) to 1.
   3) Select data transmitted on negative edge of QSPI bus clock by setting TXNEG (QSPIx_CTL[2]) to 1.
   4) Select data latched on positive edge of QSPI bus clock by clearing RXNEG (QSPIx_CTL[1]) to 0.
   5) Set the bit length of a transaction as 8-bit in DWIDTH bit field (QSPIx_CTL[12:8] = 0x08).
4. Set LSB transfer first by setting LSB (QSPIx_CTL[13]) to 1.
5. Set the SPIEN (QSPIx_CTL[0]) to 1. Wait for the slave select trigger input and QSPI clock input from the off-chip master device to start the data transfer.
6. If this QSPI slave attempts to transmit (be read) one byte data to the off-chip master device, write the byte data that will be transmitted into the QSPIx_TX register.
7. If this QSPI slave just only attempts to receive (be written) one byte data from the off-chip master device and does not care what data will be transmitted, the QSPIx_TX register does not need to be updated by software.
8. Waiting for QSPI interrupt if the UNITIEN (QSPIx_CTL[17]) is set to 1, or just polling the unit transfer interrupt flag UNITIF (QSPIx_STATUS[1]).
9. Read out the received one byte data from QSPIx_RX register.
10. Go to 6 to continue with another data transfer or stop data transfer.

6.23.8 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
QSPI Base Address: **QSPI0_BA = 0x4006_0000** **QSPIx non-secure base address is QSPIx_BA + 0x1000_0000.**				
QSPIx_CTL	QSPI0_BA+0x00	R/W	QSPI Control Register	0x0000_0034
QSPIx_CLKDIV	QSPI0_BA+0x04	R/W	QSPI Clock Divider Register	0x0000_0000
QSPIx_SSCTL	QSPI0_BA+0x08	R/W	QSPI Slave Select Control Register	0x0000_0000
QSPIx_PDMACTL	QSPI0_BA+0x0C	R/W	QSPI PDMA Control Register	0x0000_0000
QSPIx_FIFOCTL	QSPI0_BA+0x10	R/W	QSPI FIFO Control Register	0x4400_0000
QSPIx_STATUS	QSPI0_BA+0x14	R/W	QSPI Status Register	0x0005_0110
QSPIx_STATUS2	QSPI0_BA+0x18	R	QSPI Status2 Register	0x0000_0000
QSPIx_TX	QSPI0_BA+0x20	W	QSPI Data Transmit Register	0x0000_0000
QSPIx_RX	QSPI0_BA+0x30	R	QSPI Data Receive Register	0x0000_0000

6.23.9 Register Description

QSPI Control Register (QSPIx_CTL)

Register	Offset	R/W	Description	Reset Value
QSPIx_CTL	QSPI0_BA+0x00	R/W	QSPI Control Register	0x0000_0034

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
TXDTREN	QUADIOEN	DUALIOEN	DATDIR	REORDER	SLAVE	UNITIEN	TWOBIT
15	14	13	12	11	10	9	8
RXONLY	HALFDPX	LSB	DWIDTH				
7	6	5	4	3	2	1	0
SUSPITV				CLKPOL	TXNEG	RXNEG	SPIEN

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23]	**TXDTREN**	**Transmit Double Transfer Rate Mode Enable Bit** 0 = TX DTR mode Disabled. 1 = TX DTR mode Enabled. **Note:** QSPI Master mode supports TXDTR (Transmit Double Transfer Rate) mode, and QSPI Slave mode does not support this mode.
[22]	**QUADIOEN**	**Quad I/O Mode Enable Bit** 0 = Quad I/O mode Disabled. 1 = Quad I/O mode Enabled.
[21]	**DUALIOEN**	**Dual I/O Mode Enable Bit** 0 = Dual I/O mode Disabled. 1 = Dual I/O mode Enabled.
[20]	**DATDIR**	**Data Port Direction Control** This bit is used to select the data input/output direction in half-duplex transfer and Dual/Quad transfer 0 = QSPI data is input direction. 1 = QSPI data is output direction.
[19]	**REORDER**	**Byte Reorder Function Enable Bit** 0 = Byte Reorder function Disabled. 1 = Byte Reorder function Enabled. A byte suspend interval will be inserted among each byte. The period of the byte suspend interval depends on the setting of SUSPITV. **Note:** Byte Reorder function is only available if DWIDTH is defined as 16, 24, and 32 bits.
[18]	**SLAVE**	**Slave Mode Control** 0 = Master mode. 1 = Slave mode.

[17]	**UNITIEN**	**Unit Transfer Interrupt Enable Bit** 0 = QSPI unit transfer interrupt Disabled. 1 = QSPI unit transfer interrupt Enabled.
[16]	**TWOBIT**	**2-bit Transfer Mode Enable Bit** 0 = 2-bit transfer mode Disabled. 1 = 2-bit transfer mode Enabled. **Note:** When 2-bit transfer mode is enabled, the first serial transmitted bit data is from the first FIFO buffer data, and the 2nd serial transmitted bit data is from the second FIFO buffer data. As the same as transmitted function, the first received bit data is stored into the first FIFO buffer and the 2nd received bit data is stored into the second FIFO buffer at the same time.
[15]	**RXONLY**	**Receive-only Mode Enable Bit** This bit field is only available in Master mode. In receive-only mode, QSPI Master will generate QSPI bus clock continuously for receiving data bit from SPI slave device and assert the BUSY status. 0 = Receive-only mode Disabled. 1 = Receive-only mode Enabled.
[14]	**HALFDPX**	**QSPI Half-duplex Transfer Enable Bit** This bit is used to select full-duplex or half-duplex for QSPI transfer. The bit field DATDIR (QSPIx_CTL[20]) can be used to set the data direction in half-duplex transfer. 0 = QSPI operates in full-duplex transfer. 1 = QSPI operates in half-duplex transfer.
[13]	**LSB**	**Send LSB First** 0 = The MSB, which bit of transmit/receive register depends on the setting of DWIDTH, is transmitted/received first. 1 = The LSB, bit 0 of the QSPIx TX register, is sent first to the QSPI data output pin, and the first bit received from the QSPI data input pin will be put in the LSB position of the RX register (bit 0 of QSPIx_RX).
[12:8]	**DWIDTH**	**Data Width** This field specifies how many bits can be transmitted/received in one transaction. The minimum bit length is 8 bits and can up to 32 bits. DWIDTH = 0x08 …. 8 bits. DWIDTH = 0x09 …. 9 bits. …… DWIDTH = 0x1F …. 31 bits. DWIDTH = 0x00 …. 32 bits.
[7:4]	**SUSPITV**	**Suspend Interval** The four bits provide configurable suspend interval between two successive transmit/receive transaction in a Master transfer. The definition of the suspend interval is the interval between the last clock edge of the preceding transaction word and the first clock edge of the following transaction word. The default value is 0x3. The period of the suspend interval is obtained according to the following equation. (SUSPITV[3:0] + 0.5) * period of QSPI_CLK clock cycle Example: SUSPITV = 0x0 …. 0.5 QSPI_CLK clock cycle. SUSPITV = 0x1 …. 1.5 QSPI_CLK clock cycle. …… SUSPITV = 0xE …. 14.5 QSPI_CLK clock cycle. SUSPITV = 0xF …. 15.5 QSPI_CLK clock cycle. **Note:** In TX DTR mode, SUSPITV equals to 0x0.
[3]	**CLKPOL**	**Clock Polarity**

		0 = QSPI bus clock is idle low. 1 = QSPI bus clock is idle high.
[2]	**TXNEG**	**Transmit on Negative Edge** 0 = Transmitted data output signal is changed on the rising edge of QSPI bus clock. 1 = Transmitted data output signal is changed on the falling edge of QSPI bus clock. **Note:** In TX DTR mode, TXNEG equals to CLKPOL (QSPIx_CTL[3]).
[1]	**RXNEG**	**Receive on Negative Edge** 0 = Received data input signal is latched on the rising edge of QSPI bus clock. 1 = Received data input signal is latched on the falling edge of QSPI bus clock.
[0]	**SPIEN**	**QSPI Transfer Control Enable Bit** In Master mode, the transfer will start when there is data in the FIFO buffer after this bit is set to 1. In Slave mode, this device is ready to receive data when this bit is set to 1. 0 = Transfer control Disabled. 1 = Transfer control Enabled. **Note:** Before changing the configurations of QSPIx_CTL, QSPIx_CLKDIV, QSPIx_SSCTL and QSPIx_FIFOCTL registers, user shall clear the SPIEN (QSPIx_CTL[0]) and confirm the SPIENSTS (QSPIx_STATUS[15]) is 0.

QSPI Clock Divider Register (QSPIx_CLKDIV)

Register	Offset	R/W	Description	Reset Value
QSPIx_CLKDIV	QSPI0_BA+0x04	R/W	QSPI Clock Divider Register	0x0000_0000

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

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8:0]	**DIVIDER**	**Clock Divider** The value in this field is the frequency divider for generating the peripheral clock, f_{spi_eclk}, and the QSPI bus clock of QSPI Master. The frequency is obtained according to the following equation. $f_{qspi\_eclk} = \frac{f_{qspi\_clock\_src}}{(DIVIDER + 1)}$ where $f_{qspi\_clock\_src}$ is the peripheral clock source, which is defined in the clock control register, CLK_CLKSEL2. **Note:** The time interval must be larger than or equal 5 peripheral clock cycles between releasing QSPI IP software reset and setting this clock divider register.

Note: DIVIDER should be set carefully because the peripheral clock frequency must be slower than or equal to system frequency.

QSPI Slave Select Control Register (QSPIx_SSCTL)

Register	Offset	R/W	Description	Reset Value
QSPIx_SSCTL	QSPI0_BA+0x08	R/W	QSPI Slave Select Control Register	0x0000_0000

31	30	29	28	27	26	25	24
SLVTOCNT							
23	22	21	20	19	18	17	16
SLVTOCNT							
15	14	13	12	11	10	9	8
Reserved		SSINAIEN	SSACTIEN	Reserved		SLVURIEN	SLVBEIEN
7	6	5	4	3	2	1	0
Reserved	SLVTORST	SLVTOIEN	SLV3WIRE	AUTOSS	SSACTPOL	Reserved	SS

Bits	Description	
[31:16]	**SLVTOCNT**	**Slave Mode Time-out Period** In Slave mode, these bits indicate the time-out period when there is bus clock input during slave select active. The clock source of the time-out counter is Slave peripheral clock. If the value is 0, it indicates the slave mode time-out function is disabled.
[15:14]	**Reserved**	Reserved.
[13]	**SSINAIEN**	**Slave Select Inactive Interrupt Enable Bit** 0 = Slave select inactive interrupt Disabled. 1 = Slave select inactive interrupt Enabled.
[12]	**SSACTIEN**	**Slave Select Active Interrupt Enable Bit** 0 = Slave select active interrupt Disabled. 1 = Slave select active interrupt Enabled.
[11:10]	**Reserved**	Reserved.
[9]	**SLVURIEN**	**Slave Mode TX Under Run Interrupt Enable Bit** 0 = Slave mode TX under run interrupt Disabled. 1 = Slave mode TX under run interrupt Enabled.
[8]	**SLVBEIEN**	**Slave Mode Bit Count Error Interrupt Enable Bit** 0 = Slave mode bit count error interrupt Disabled. 1 = Slave mode bit count error interrupt Enabled.
[7]	**Reserved**	Reserved.
[6]	**SLVTORST**	**Slave Mode Time-out Reset Control** 0 = When Slave mode time-out event occurs, the TX and RX control circuit will not be reset. 1 = When Slave mode time-out event occurs, the TX and RX control circuit will be reset by hardware.
[5]	**SLVTOIEN**	**Slave Mode Time-out Interrupt Enable Bit** 0 = Slave mode time-out interrupt Disabled.

		1 = Slave mode time-out interrupt Enabled.
[4]	**SLV3WIRE**	**Slave 3-wire Mode Enable Bit** In Slave 3-wire mode, the QSPI controller can work with 3-wire interface including QSPIx_CLK, QSPIx_MISO and QSPIx_MOSI pins. 0 = 4-wire bi-direction interface. 1 = 3-wire bi-direction interface.
[3]	**AUTOSS**	**Automatic Slave Selection Function Enable Bit** 0 = Automatic slave selection function Disabled. Slave selection signal will be asserted/de-asserted according to SS (QSPIx_SSCTL[0]). 1 = Automatic slave selection function Enabled. **Note:** Master mode only.
[2]	**SSACTPOL**	**Slave Selection Active Polarity** This bit defines the active polarity of slave selection signal (QSPIx_SS). 0 = The slave selection signal QSPIx_SS is active low. 1 = The slave selection signal QSPIx_SS is active high.
[1]	**Reserved**	Reserved.
[0]	**SS**	**Slave Selection Control** If AUTOSS bit is cleared to 0, 0 = Set the QSPIx_SS line to inactive state. 1 = Set the QSPIx_SS line to active state. If the AUTOSS bit is set to 1, 0 = Keep the QSPIx_SS line at inactive state. 1 = QSPIx_SS line will be automatically driven to active state for the duration of data transfer, and will be driven to inactive state for the rest of the time. The active state of QSPIx_SS is specified in SSACTPOL (QSPIx_SSCTL[2]). **Note:** Master mode only.

QSPI PDMA Control Register (QSPIx_PDMACTL)

Register	Offset	R/W	Description	Reset Value
QSPIx_PDMACTL	QSPI0_BA+0x0C	R/W	QSPI PDMA Control Register	0x0000_0000

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

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**PDMARST**	**PDMA Reset** 0 = No effect. 1 = Reset the PDMA control logic of the QSPI controller. This bit will be automatically cleared to 0.
[1]	**RXPDMAEN**	**Receive PDMA Enable Bit** 0 = Receive PDMA function Disabled. 1 = Receive PDMA function Enabled.
[0]	**TXPDMAEN**	**Transmit PDMA Enable Bit** 0 = Transmit PDMA function Disabled. 1 = Transmit PDMA function Enabled. **Note1:** In QSPI Master mode with full duplex transfer, if both TX and RX PDMA functions are enabled, RX PDMA function cannot be enabled prior to TX PDMA function. User can enable TX PDMA function firstly or enable both functions simultaneously. **Note2:** In QSPI Master mode with full duplex transfer, if both TX and RX PDMA functions are enabled, TX PDMA function cannot be disabled prior to RX PDMA function. User can disable RX PDMA function firstly or disable both functions simultaneously.

QSPI FIFO Control Register (QSPIx_FIFOCTL)

Register	Offset	R/W	Description	Reset Value
QSPIx_FIFOCTL	QSPI0_BA+0x10	R/W	QSPI FIFO Control Register	0x4400_0000

31	30	29	28	27	26	25	24
Reserved	TXTH			Reserved	RXTH		
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					SLVBERX	TXFBCLR	RXFBCLR
7	6	5	4	3	2	1	0
TXUFIEN	TXUFPOL	RXOVIEN	RXTOIEN	TXTHIEN	RXTHIEN	TXRST	RXRST

Bits	Description	
[31]	**Reserved**	Reserved.
[30:28]	**TXTH**	**Transmit FIFO Threshold** If the valid data count of the transmit FIFO buffer is less than or equal to the TXTH setting, the TXTHIF bit will be set to 1, else the TXTHIF bit will be cleared to 0.
[27]	**Reserved**	Reserved.
[26:24]	**RXTH**	**Receive FIFO Threshold** If the valid data count of the receive FIFO buffer is larger than the RXTH setting, the RXTHIF bit will be set to 1, else the RXTHIF bit will be cleared to 0.
[23:11]	**Reserved**	Reserved.
[10]	**SLVBERX**	**RX FIFO Write Data Enable Bit When Slave Mode Bit Count Error** 0 = Uncompleted RX data will be dropped from RX FIFO when bit count error event happen in QSPI Slave mode. 1 = Uncompleted RX data will be written into RX FIFO when bit count error event happen in QSPI Slave mode. User can read SLVBENUM (QSPIx_STATUS2[29:24]) to know that the effective bit number of uncompleted RX data when SPI slave bit count error happened. **Note:** Slave mode only.
[9]	**TXFBCLR**	**Transmit FIFO Buffer Clear** 0 = No effect. 1 = Clear transmit FIFO pointer. The TXFULL bit will be cleared to 0 and the TXEMPTY bit will be set to 1. This bit will be cleared to 0 by hardware about 1 system clock after it is set to 1. **Note:** The TX shift register will not be cleared.
[8]	**RXFBCLR**	**Receive FIFO Buffer Clear** 0 = No effect. 1 = Clear receive FIFO pointer. The RXFULL bit will be cleared to 0 and the RXEMPTY bit will be set to 1. This bit will be cleared to 0 by hardware about 1 system clock after it is set to 1. **Note:** The RX shift register will not be cleared.

[7]	**TXUFIEN**	**TX Underflow Interrupt Enable Bit** When TX underflow event occurs in Slave mode, TXUFIF (QSPIx_STATUS[19]) will be set to 1. This bit is used to enable the TX underflow interrupt. 0 = Slave TX underflow interrupt Disabled. 1 = Slave TX underflow interrupt Enabled.
[6]	**TXUFPOL**	**TX Underflow Data Polarity** 0 = The QSPI data out is kept 0 if there is TX underflow event in Slave mode. 1 = The QSPI data out is kept 1 if there is TX underflow event in Slave mode. **Note 1:** The TX underflow event occurs if there is no any data in TX FIFO when the slave selection signal is active. **Note 2:** When TX underflow event occurs, QSPIx_MISO pin state will be determined by this setting even though TX FIFO is not empty afterward. Data stored in TX FIFO will be sent through QSPIx_MISO pin in the next transfer frame.
[5]	**RXOVIEN**	**Receive FIFO Overrun Interrupt Enable Bit** 0 = Receive FIFO overrun interrupt Disabled. 1 = Receive FIFO overrun interrupt Enabled.
[4]	**RXTOIEN**	**Receive Time-out Interrupt Enable Bit** 0 = Receive time-out interrupt Disabled. 1 = Receive time-out interrupt Enabled.
[3]	**TXTHIEN**	**Transmit FIFO Threshold Interrupt Enable Bit** 0 = TX FIFO threshold interrupt Disabled. 1 = TX FIFO threshold interrupt Enabled.
[2]	**RXTHIEN**	**Receive FIFO Threshold Interrupt Enable Bit** 0 = RX FIFO threshold interrupt Disabled. 1 = RX FIFO threshold interrupt Enabled.
[1]	**TXRST**	**Transmit Reset** 0 = No effect. 1 = Reset transmit FIFO pointer and transmit circuit. The TXFULL bit will be cleared to 0 and the TXEMPTY bit will be set to 1. This bit will be cleared to 0 by hardware about 3 system clock cycles + 2 peripheral clock cycles after it is set to 1. User can read TXRXRST (QSPIx_STATUS[23]) to check if reset is accomplished or not. **Note:** If TX underflow event occurs in QSPI Slave mode, this bit can be used to make QSPI return to idle state.
[0]	**RXRST**	**Receive Reset** 0 = No effect. 1 = Reset receive FIFO pointer and receive circuit. The RXFULL bit will be cleared to 0 and the RXEMPTY bit will be set to 1. This bit will be cleared to 0 by hardware about 3 system clock cycles + 2 peripheral clock cycles after it is set to 1. User can read TXRXRST (QSPIx_STATUS[23]) to check if reset is accomplished or not.

QSPI Status Register (QSPIx_STATUS)

Register	Offset	R/W	Description	Reset Value
QSPIx_STATUS	QSPI0_BA+0x14	R/W	QSPI Status Register	0x0005_0110

31	30	29	28	27	26	25	24
TXCNT				RXCNT			
23	22	21	20	19	18	17	16
TXRXRST	Reserved			TXUFIF	TXTHIF	TXFULL	TXEMPTY
15	14	13	12	11	10	9	8
SPIENSTS	Reserved		RXTOIF	RXOVIF	RXTHIF	RXFULL	RXEMPTY
7	6	5	4	3	2	1	0
SLVURIF	SLVBEIF	SLVTOIF	SSLINE	SSINAIF	SSACTIF	UNITIF	BUSY

Bits	Description	
[31:28]	**TXCNT**	**Transmit FIFO Data Count (Read Only)** This bit field indicates the valid data count of transmit FIFO buffer.
[27:24]	**RXCNT**	**Receive FIFO Data Count (Read Only)** This bit field indicates the valid data count of receive FIFO buffer.
[23]	**TXRXRST**	**TX or RX Reset Status (Read Only)** 0 = The reset function of TXRST or RXRST is done. 1 = Doing the reset function of TXRST or RXRST. **Note:** Both the reset operations of TXRST and RXRST need 3 system clock cycles + 2 peripheral clock cycles. User can check the status of this bit to monitor the reset function is doing or done.
[22:20]	**Reserved**	Reserved.
[19]	**TXUFIF**	**TX Underflow Interrupt Flag** When the TX underflow event occurs, this bit will be set to 1, the state of data output pin depends on the setting of TXUFPOL. 0 = No effect. 1 = No data in Transmit FIFO and TX shift register when the slave selection signal is active. **Note 1:** This bit will be cleared by writing 1 to it. **Note 2:** If reset slave's transmission circuit when slave selection signal is active, this flag will be set to 1 after 2 peripheral clock cycles + 3 system clock cycles since the reset operation is done.
[18]	**TXTHIF**	**Transmit FIFO Threshold Interrupt Flag (Read Only)** 0 = The valid data count within the transmit FIFO buffer is larger than the setting value of TXTH. 1 = The valid data count within the transmit FIFO buffer is less than or equal to the setting value of TXTH.
[17]	**TXFULL**	**Transmit FIFO Buffer Full Indicator (Read Only)** 0 = Transmit FIFO buffer is not full. 1 = Transmit FIFO buffer is full.
[16]	**TXEMPTY**	**Transmit FIFO Buffer Empty Indicator (Read Only)**

		0 = Transmit FIFO buffer is not empty. 1 = Transmit FIFO buffer is empty.
[15]	**SPIENSTS**	**QSPI Enable Status (Read Only)** 0 = QSPI controller Disabled. 1 = QSPI controller Enabled. **Note:** The QSPI peripheral clock is asynchronous with the system clock. In order to make sure the QSPI control logic is disabled, this bit indicates the real status of QSPI controller.
[14:13]	**Reserved**	Reserved.
[12]	**RXTOIF**	**Receive Time-out Interrupt Flag** 0 = No receive FIFO time-out event. 1 = Receive FIFO buffer is not empty and no read operation on receive FIFO buffer over 64 QSPI peripheral clock periods in Master mode or over 576 QSPI peripheral clock periods in Slave mode. When the received FIFO buffer is read by software, the time-out status will be cleared automatically. **Note:** This bit will be cleared by writing 1 to it.
[11]	**RXOVIF**	**Receive FIFO Overrun Interrupt Flag** When the receive FIFO buffer is full, the follow-up data will be dropped and this bit will be set to 1. 0 = No FIFO is overrun. 1 = Receive FIFO is overrun. **Note:** This bit will be cleared by writing 1 to it.
[10]	**RXTHIF**	**Receive FIFO Threshold Interrupt Flag (Read Only)** 0 = The valid data count within the receive FIFO buffer is smaller than or equal to the setting value of RXTH. 1 = The valid data count within the receive FIFO buffer is larger than the setting value of RXTH.
[9]	**RXFULL**	**Receive FIFO Buffer Full Indicator (Read Only)** 0 = Receive FIFO buffer is not full. 1 = Receive FIFO buffer is full.
[8]	**RXEMPTY**	**Receive FIFO Buffer Empty Indicator (Read Only)** 0 = Receive FIFO buffer is not empty. 1 = Receive FIFO buffer is empty.
[7]	**SLVURIF**	**Slave Mode TX Under Run Interrupt Flag** In Slave mode, if TX underflow event occurs and the slave select line goes to inactive state, this interrupt flag will be set to 1. 0 = No Slave TX under run event. 1 = Slave TX under run event occurred. **Note:** This bit will be cleared by writing 1 to it.
[6]	**SLVBEIF**	**Slave Mode Bit Count Error Interrupt Flag** In Slave mode, when the slave select line goes to inactive state, if bit counter is mismatch with DWIDTH, this interrupt flag will be set to 1. 0 = No Slave mode bit count error event. 1 = Slave mode bit count error event occurred. **Note:** If the slave select active but there is no any bus clock input, the SLVBEIF also active when the slave select goes to inactive state. This bit will be cleared by writing 1 to it.
[5]	**SLVTOIF**	**Slave Time-out Interrupt Flag** When the slave select is active and the value of SLVTOCNT is not 0, if the bus clock is detected, the slave time-out counter in QSPI controller logic will be started. When the value of time-out counter is greater than or equal to the value of SLVTOCNT (QSPIx_SSCTL[31:16]) before one transaction is done, the slave time-out interrupt event will be asserted.

		0 = Slave time-out is not active. 1 = Slave time-out is active. **Note:** This bit will be cleared by writing 1 to it.
[4]	**SSLINE**	**Slave Select Line Bus Status (Read Only)** 0 = The slave select line status is 0. 1 = The slave select line status is 1. **Note:** This bit is only available in Slave mode. If SSACTPOL (QSPIx_SSCTL[2]) is set 0, and the SSLINE is 1, the QSPI slave select is in inactive status.
[3]	**SSINAIF**	**Slave Select Inactive Interrupt Flag** 0 = Slave select inactive interrupt was cleared or not occurred. 1 = Slave select inactive interrupt event occurred. **Note:** Only available in Slave mode. This bit will be cleared by writing 1 to it.
[2]	**SSACTIF**	**Slave Select Active Interrupt Flag** 0 = Slave select active interrupt was cleared or not occurred. 1 = Slave select active interrupt event occurred. **Note:** Only available in Slave mode. This bit will be cleared by writing 1 to it.
[1]	**UNITIF**	**Unit Transfer Interrupt Flag** 0 = No transaction has been finished since this bit was cleared to 0. 1 = QSPI controller has finished one unit transfer. **Note:** This bit will be cleared by writing 1 to it.
[0]	**BUSY**	**Busy Status (Read Only)** 0 = QSPI controller is in idle state. 1 = QSPI controller is in busy state. The following lists the bus busy conditions: f. SPIEN (QSPIx_CTL[0]) = 1 and TXEMPTY = 0. g. For QSPI Master mode, SPIEN (QSPIx_CTL[0]) = 1 and TXEMPTY = 1 but the current transaction is not finished yet. h. For QSPI Master mode, SPIEN (QSPIx_CTL[0]) = 1 and RXONLY = 1. i. For QSPI Slave mode, SPIEN (QSPIx_CTL[0]) = 1 and there is serial clock input into the QSPI core logic when slave select is active. j. For QSPI Slave mode, SPIEN (QSPIx_CTL[0]) = 1 and the transmit buffer or transmit shift register is not empty even if the slave select is inactive. **Note:** By applications, this QSPI busy flag should be used with other status registers in QSPIx_STATUS such as TXCNT, RXCNT, TXTHIF, TXFULL, TXEMPTY, RXTHIF, RXFULL, RXEMPTY, and UNITIF. Therefore the QSPI transfer done events of TX/RX operations can be obtained at correct timing point.

QSPI Status2 Register (QSPIx_STATUS2)

Register	Offset	R/W	Description	Reset Value
QSPIx_STATUS2	QSPI0_BA+0x18	R	QSPI Status2 Register	0x0000_0000

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

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:24]	**SLVBENUM**	**Effective Bit Number of Uncompleted RX Data** This status register indicates that effective bit number of uncompleted RX data when SLVBERX (QSPIx_FIFOCTL[10]) is enabled and RX bit count error event happen in QSPI Slave mode. This status register will be fixed to 0x0 when SLVBERX (QSPIx_FIFOCTL[10]) is disabled. **Note 1:** This register will be cleared to 0x0 when user write 0x1 to SLVBEIF (QSPIx_STATUS[6]). **Note 2:** Slave mode only.
[23:0]	**Reserved**	Reserved.

QSPI Data Transmit Register (QSPIx_TX)

Register	Offset	R/W	Description	Reset Value
QSPIx_TX	QSPI0_BA+0x20	W	QSPI Data Transmit Register	0x0000_0000

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

Bits		Description
[31:0]	**TX**	**Data Transmit Register** The data transmit registers pass through the transmitted data into the 8-level transmit FIFO buffers. The number of valid bits depends on the setting of DWIDTH (QSPIx_CTL[12:8]). If DWIDTH is set to 0x08, the bits TX[7:0] will be transmitted. If DWIDTH is set to 0x00, the QSPI controller will perform a 32-bit transfer. **Note:** In Master mode, QSPI controller will start to transfer the QSPI bus clock after 1 APB clock and 6 peripheral clock cycles after user writes to this register.

QSPI Data Receive Register (QSPIx_RX)

Register	Offset	R/W	Description	Reset Value
QSPIx_RX	QSPI0_BA+0x30	R	QSPI Data Receive Register	0x0000_0000

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

Bits	Description	
[31:0]	**RX**	**Data Receive Register (Read Only)** There are 8-level FIFO buffers in this controller. The data receive register holds the data received from QSPI data input pin. If the RXEMPTY (QSPIx_STATUS[8) is not set to 1, the receive FIFO buffers can be accessed through software by reading this register.

6.24 USCI - Universal Serial Control Interface Controller (USCI)

6.24.1 Overview

The Universal Serial Control Interface (USCI) is a flexible interface module covering several serial communication protocols. The user can configure this controller as UART, SPI, or I^2C functional protocol.

6.24.2 Features

The controller can be individually configured to match the application needs. The following protocols are supported:

- UART
- SPI
- I^2C

6.24.3 Block Diagram

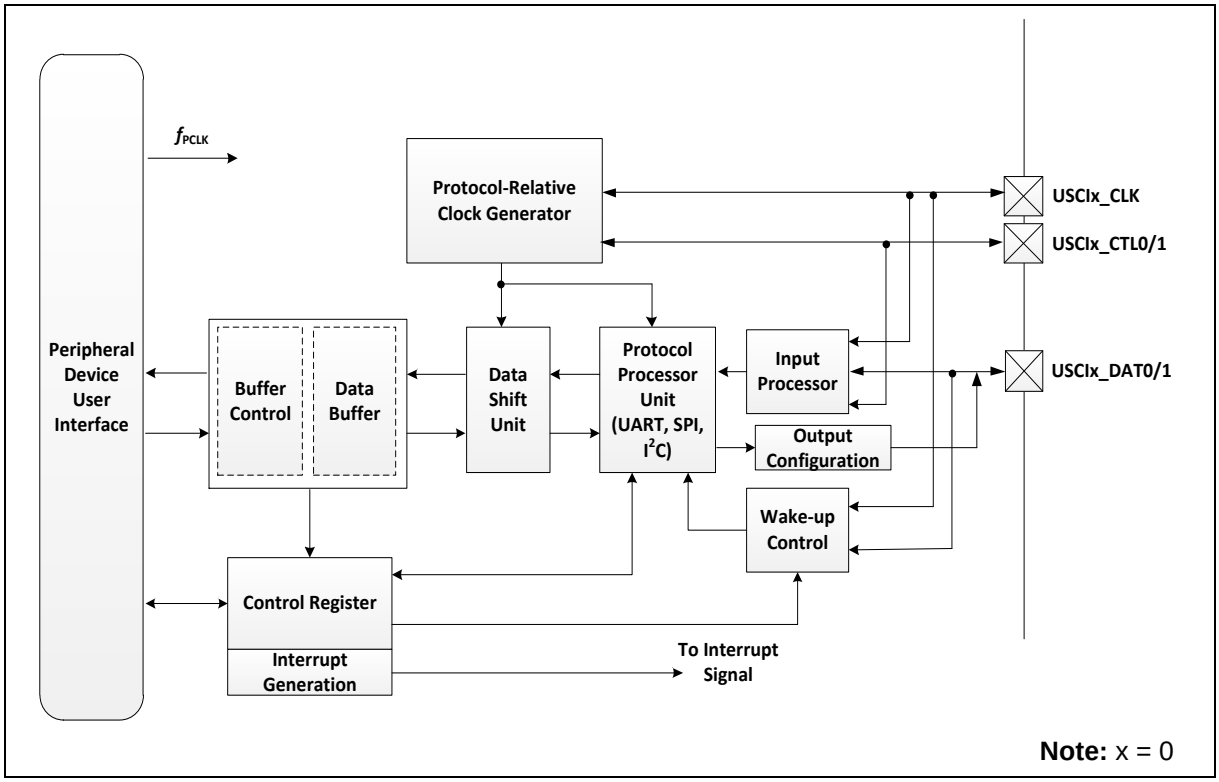


Figure 6.24-1 USCI Block Diagram

6.24.4 Functional Description

The structure of the Universal Serial Control Interface (USCI) controller is shown in Figure 6.24-1 USCI Block Diagram. The input signal is implemented in input processor. The data buffers and the data shift unit support the data transfers. Each protocol-specific function is handled by the protocol processor unit. The timing and time event control signals of the specific protocol are handled by the protocol-relative clock generator. All the protocol-specific events are processed in the interrupt generation unit. The wake-up function of the specific protocol is implemented in the wake-up control unit.

The USCI is equipped with three protocols including UART, SPI, and I²C. They can be selected by FUNMODE (USCI_CTL [2:0]). Note that the FUNMODE must be set to 0 before changing protocol.

6.24.4.1 I/O Processer

Input Signal

All input stages offer the similar feature set. They are used for all protocols.

Table 6.24-1 lists the relative input signals for each selected protocol. Each input signal is handled by an input processor for signal conditioning, such as signal inverse selection control, or a digital input filter.

Selected Protocol		UART	SPI	I²C
Serial Bus Clock Input	USCIx_CLK	-	SPI_CLK	SCL
Control Input	USCIx_CTL0	nCTS	SPI_SS	-
	USCIx_CTL1	-	-	-
Data Input	USCIx_DAT0	RX	SPI_MOSI_0	SDA
	USCIx_DAT1	-	SPI_MISO_0	-

Table 6.24-1 Input Signals for Different Protocols

The description of protocol-specific items are given in the related protocol chapters.

General Input Structure

The input structures of data and control signals include inverter, digital filter and edge detection (data signal only).

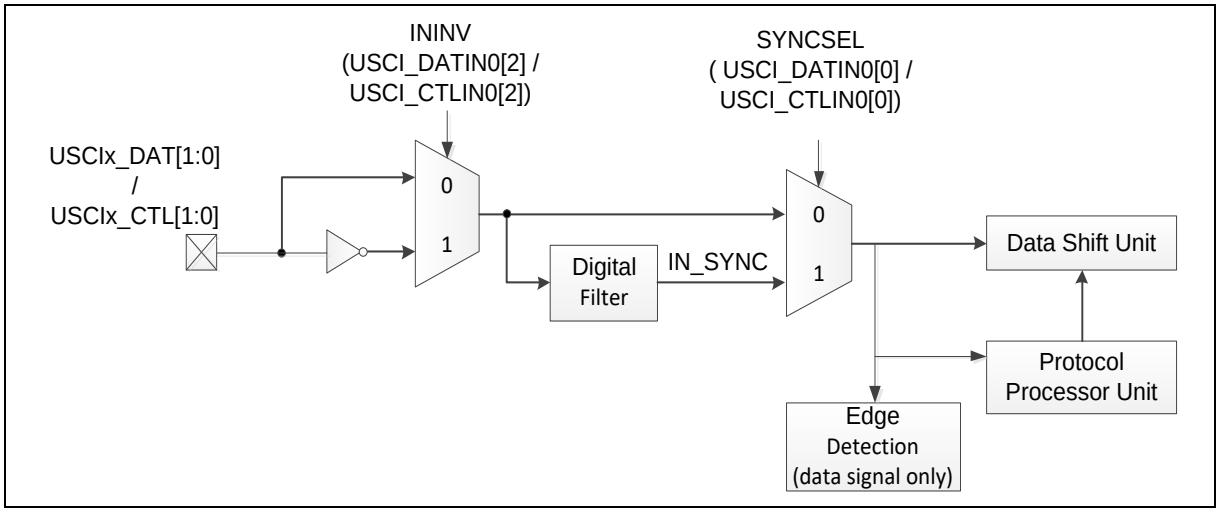


Figure 6.24-2 Input Conditioning for USCIx_DAT[1:0] and USCIx_CTL[1:0]

The input structure of USCIx_CLK is similar to USCIx_CTL[1:0] input structure, except it does not support inverse function.

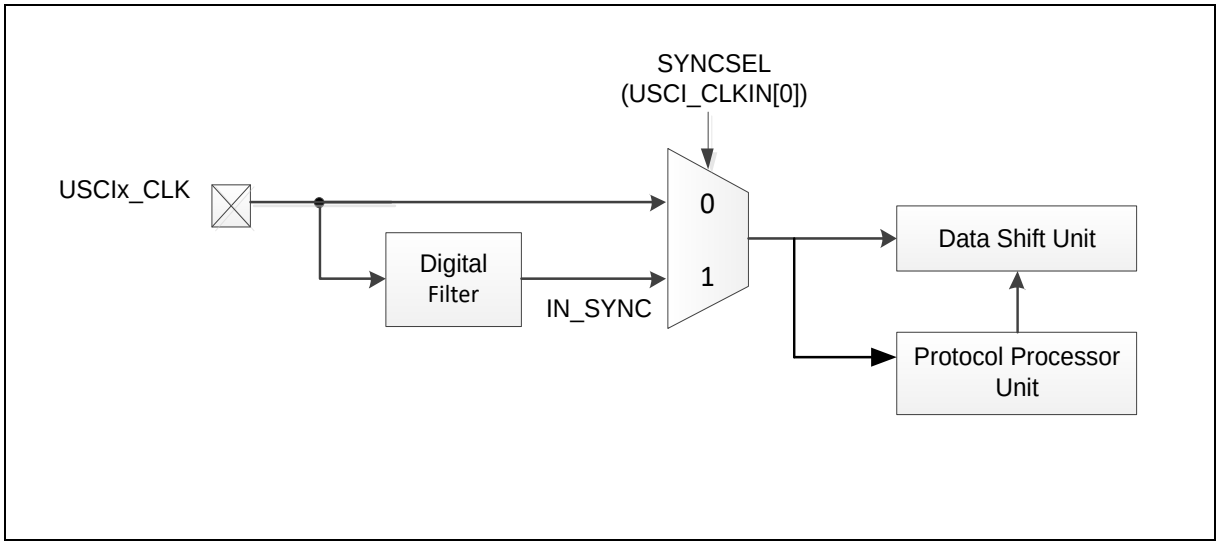


Figure 6.24-3 Input Conditioning for USCIx_CLK

All configurations of control, clock and data input structures are in USCI_CTLIN0, USCI_CLKIN and USCI_DATIN0 registers respectively. EDGEDET (USCI_DATIN0[4:3]) is used to select the edge detection condition. Note that the EDGEDET for USCI_DATIN0 must be set 2'b10 in UART mode. The programmable edge detection indicates that the desired event has occurred by activating the trigger signal.

ININV (USCI_DATIN0[2] / USCI_CTLIN0[2]) allows a polarity inversion of the selected input signal to adapt the input signal polarity to the internal polarity of the data shift unit and the protocol state machine.

If the SYNCSEL (USCI_DATIN0[0] / USCI_CTLIN0[0] / USCI_CLKIN[0]) is set to 0, the paths of input signals do not contain any delay due to synchronization or filtering. If there is noise on the input signals, there is the possibility to synchronize the input signal (signal IN_SYNC is synchronized to f_{PCLK}). The synchronization leads to a delay in the signal path of 2-3 times the period of f_{PCLK}.

Output Signals

Table 6.24-2 shows the relative output signals for each protocol. The number of actually used outputs depends on the selected protocol and they can be classified according to their meaning for the protocols.

Selected Protocol		UART	SPI	I^2C
Serial Bus Clock Output	USCIx_CLK	-	SPI_CLK	SCL
Control Output	USCIx_CTL0	-	SPI_SS	-
	USCIx_CTL1	nRTS	-	-
Data Output	USCIx_DAT0	-	SPI_MOSI_0	SDA
	USCIx_DAT1	TX	SPI_MISO_0	-

Table 6.24-2 Output Signals for Different Protocols

The description of protocol-specific items are given in the related protocol chapters.

6.24.4.2 Data Buffering

The data handling of the USCI controller is based on a Data Shift Unit (DSU) and a buffer structure. Both of the data shift and buffer registers are 16-bit wide. The inputs of Data Shift Unit include the shift data, the serial bus clock, and the shift control. The output pin of transmission can be USCIx_DAT0 pin or USCIx_DAT1 pin depends on what protocol is selected.

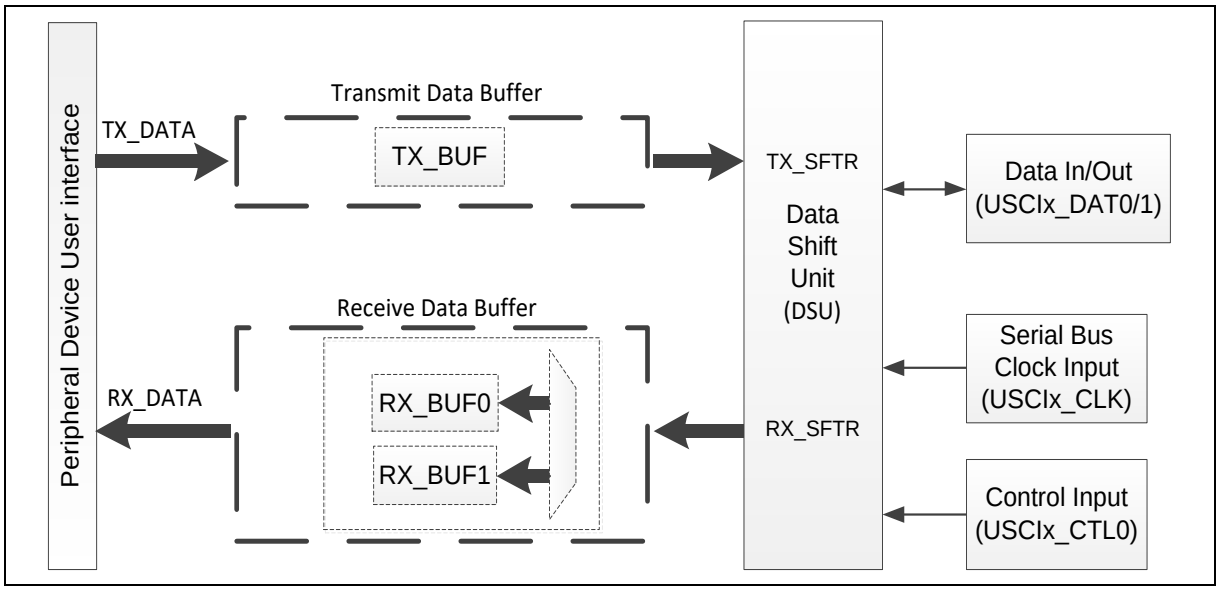


Figure 6.24-4 Block Diagram of Data Buffering

The operation of data handling includes:

- The peripheral device user interface (APB) is used to handle data, interrupts, status and control information.
- A transmitter includes transmit shift register (TX_SFTR) and a transmit data buffer (TX_BUF). The TXFULL / TXEMPTY (USCI_BUFSTS[9:8]) and TXENDIF (USCI_PROTSTS[2]) can indicate the status of transmitter.
- A receiver includes receive shift register (RX_SFTR) and a double receive buffer structure (RX_BUF0, RX_BUF1). In double buffer structure, user need not care about the reception sequence and two received data can be hold if user does not read the data of USCI_RXDAT register in time.

Data Access Structure

The Data Access Structure includes read access to received data and write access of data to be transmitted. The received data is stored in the receiver buffers including RX_BUF0 and RX_BUF1. User need not care about the reception sequence. The receive buffer can be accessed by reading USCI_RXDAT register. The first received data is read out first and the next received data becomes visible in USCI_RXDAT and can be read out next.

Transmit data can be loaded to TX_BUF by writing to the transmit register USCI_TXDAT.

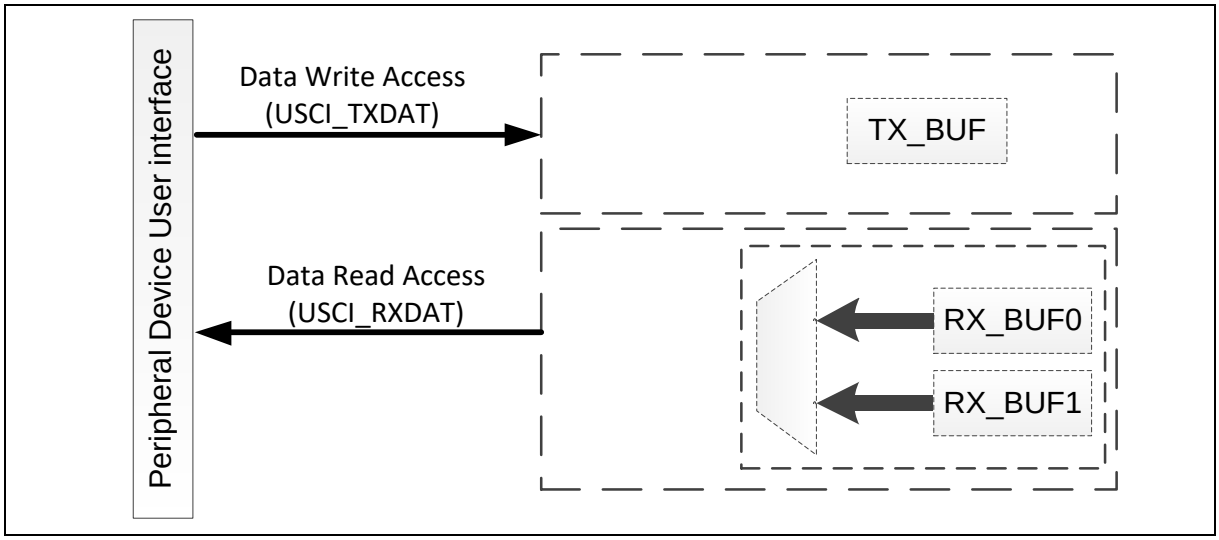


Figure 6.24-5 Data Access Structure

Transmit Data Path

The transmit data path is based on 16-bit wide transmit shift register (TX_SFTR) and transmit buffer TX_BUF. The data transfer parameters like data word length is controlled commonly for transmission and reception by the line control register USCI_LINECTL.

Transmit Buffering

The transmit shift register cannot be directly accessed by user. It is updated automatically with the value stored in the transmit buffer (TX_BUF) if a currently transmitted data is finished and new data is valid for transmission.

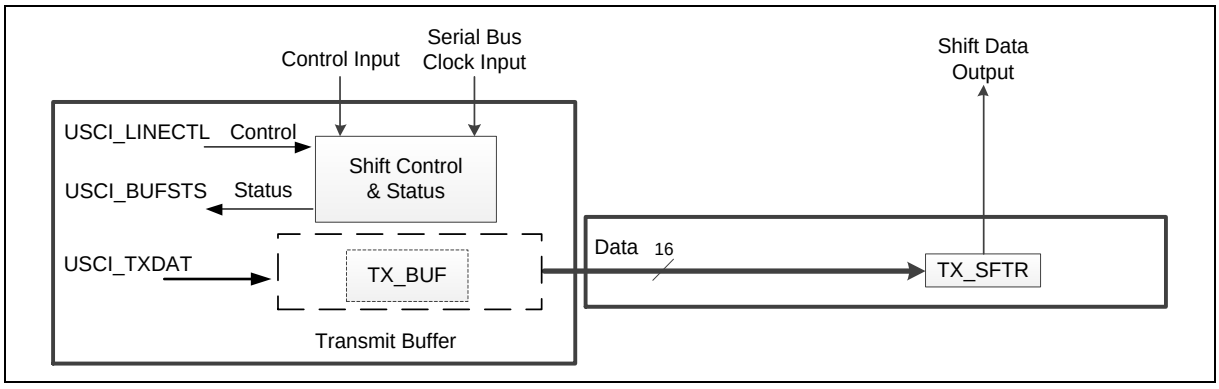


Figure 6.24-6 Transmit Data Path

Transmit Data Validation

The status of TXEMPTY (USCI_BUFSTS[8]) indicates the transmission data is valid or not in the transmit buffer (TX_BUF) and the TXSTIF (USCI_PROTSTS[1]) labels the start conditions for each data.

- If the USCI controller is a Master, the data transfer can only be started with valid data in the transmit buffer (TX_BUF). In this case, the transmit shift register is loaded with the content of transmit buffer.

  Note: Master defines the start of data transfer.

- If the USCI controller is a Slave, a data transfer requested by Master and it has to be started independently of the status in transmit buffer (TX_BUF). If a data transfer is requested and started by the Master, the transmit shift register is loaded from specific protocol control signal if it is valid for transmission.

  Note: Slave can not define the start itself, but has to react.

- The timing of loading data from transmit buffer to data shift unit depends on protocol configurations.
- **UART:** A transmission of the data word in transmit buffer can be started if TXEMPTY = 0 in normal operation. In auto flow control, A transmission of the data word in transmit buffer can be started while TXEMPTY = 0 and USCIx_CTL0 in active stage.
- **SPI:** In Master mode, data transmission will be started when TXEMPTY (USCI_BUFSTS[8]) is 0. In Slave mode, the data transmission can be started only when slave selection signal is at active state and clock is presented on USCIx_CLK pin.
- **I^2C:** A transmission of the data byte in transmit buffer can be started if TXEMPTY = 0.
- A transmission data which is located in transmit buffer can be started if the TXEMPTY (USCI_BUFSTS [8]) = 0. The content of the transmit buffer (in TX_BUF condition) should not be overwritten with new data while it is valid for transmission and a new transmission can start. If the content of TX_BUF has to be changed, user can set TXRST (USCI_BUFCTL [16]) to 1 to clear the content of TX_BUF before updating the data. Moreover, TXEMPTY (USCI_BUFSTS [8]) will be cleared automatically when transmit buffer (TX_BUF) is updated with new data. While a transmission is in progress, TX_BUF can be loaded with new data. User has to update the TX_BUF before a new transmission.

Receive Data Path

The receive data path is based on 16-bit wide receive shift register RX_SFTR and receive buffers RX_BUF0 and RX_BUF1. The data transfer parameters like data word length, or the shift direction are controlled commonly for transmission and reception by the line control register USCI_LINECTL. Register USCI_BUFSTS monitors the data validation of USCI_RXDAT.

Receive Buffering

The receive shift register cannot be directly accessed by user, but its content is automatically loaded into the receive buffer if a complete data word has been received or the frame is finished. The received data words in Receive Buffer can be read out automatically from register USCI_RXDAT.

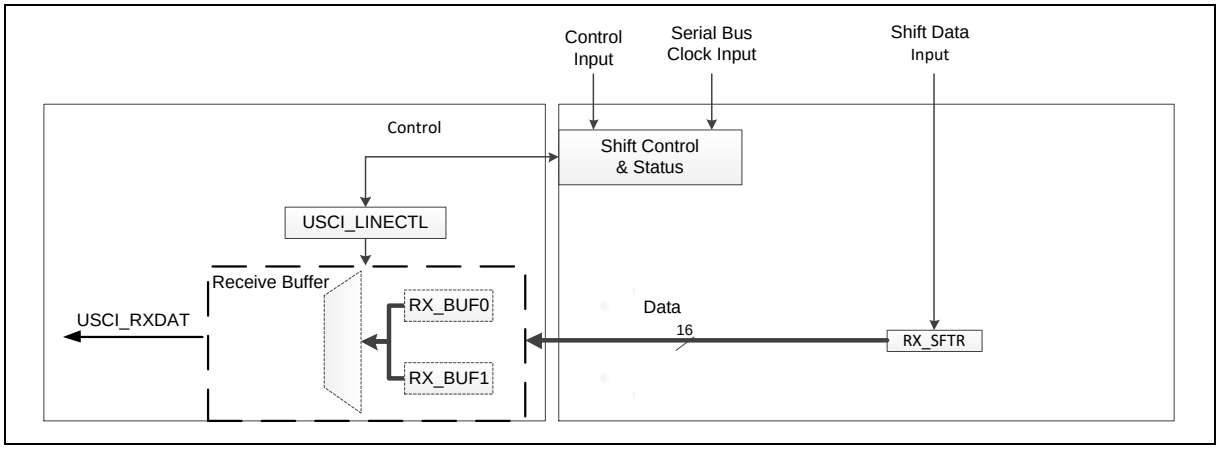


Figure 6.24-7 Receive Data Path

6.24.4.3 Port Direction Control

In SPI protocol with half-duplex configurations, the data port is bidirectional. Port direction control is intended to control the pin direction through a dedicated hardware interface.

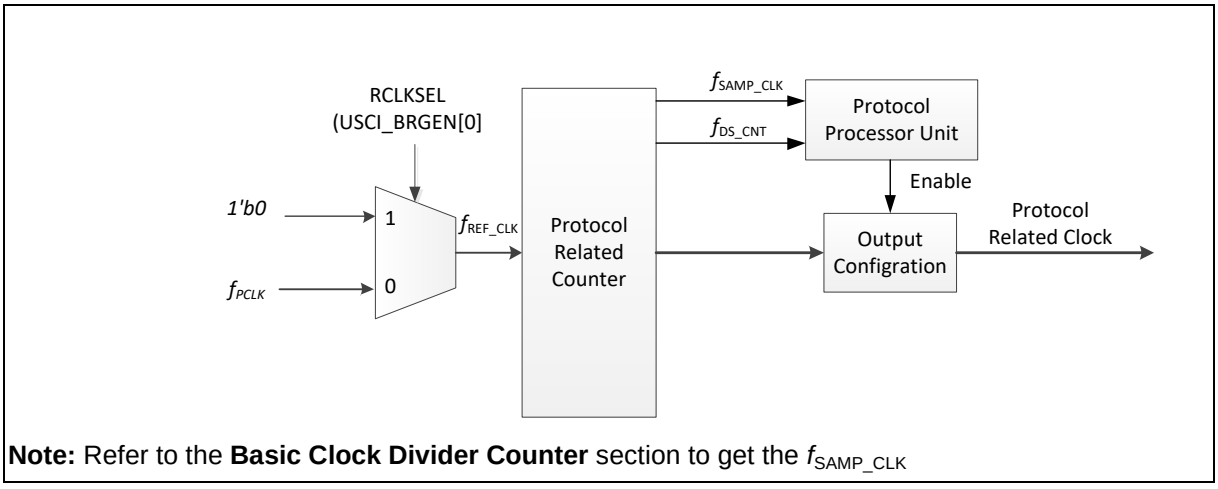
The direction of selected pin is controlled by PORTDIR (USCI_TXDAT[16]). When user writes USCI_TXDAT register, the transmit data and its port direction are settled simultaneously.

6.24.4.4 Protocol Control and Status

The protocol-related control and status information are located in the protocol control register USCI_PROTCTL and in the protocol status register USCI_PROTSTS. These registers are shared between the available protocols. As a consequence, the meaning of the bit positions in these registers is different within the protocols. Refer to each protocol's relative register for detail information.

6.24.4.5 Protocol-Relative Clock Generator

The USCI controller contains a protocol-relative clock generator and it is controlled by register USCI_BRGEN. It is reset when the USCI_BRGEN register is written. The structured of protocol-relative clock generator is shown below.



Note: Refer to the **Basic Clock Divider Counter** section to get the f_{SAMP_CLK}

Figure 6.24-8 Protocol-Relative Clock Generator

The protocol related counter contains basic clock divider counter and timing measurement counter. It is based on a divider stages, providing the frequencies needed for the different protocols. It contains:

- The basic clock divider counter provides the protocol relative clock signal and other protocol-related signals (f_{SAMP_CLK} and f_{DS_CLK}).
- The timing measurement counter for time interval measurement, e.g. baud rate detection on UART protocol.
- The output signals of protocol relative clock generator can be made available on pins (e.g USCIx_CLK for SPI).

Basic Clock Divider Counter

The basic clock divider counter is used for an integer division delivering f_{REF_CLK2}, f_{REF_CLK}, f_{DIV_CLK}, f_{SCLK} ,and f_{SAMP_CLK} . The frequencies of this divider are controlled by PTCLKSEL (USCI_BRGEN [1]), CLKDIV (USCI_BRGEN [25:16]), SPCLKSEL (USCI_BRGEN [3:2]).

The basic clock divider counter is used to generate the relative protocol timing signals.

$$f_{\mathrm{DIV\_CLK}} = f_{REF\_CLK} \times \frac{1}{\mathrm{CLKDIV} + 1} \text{ if PTCLKSEL} = 0$$

$$f_{\mathrm{DIV\_CLK}} = f_{REF\_CLK} \times \frac{1}{(\mathrm{CLKDIV} + 1) \times 2} \text{ if PTCLKSEL} = 1$$

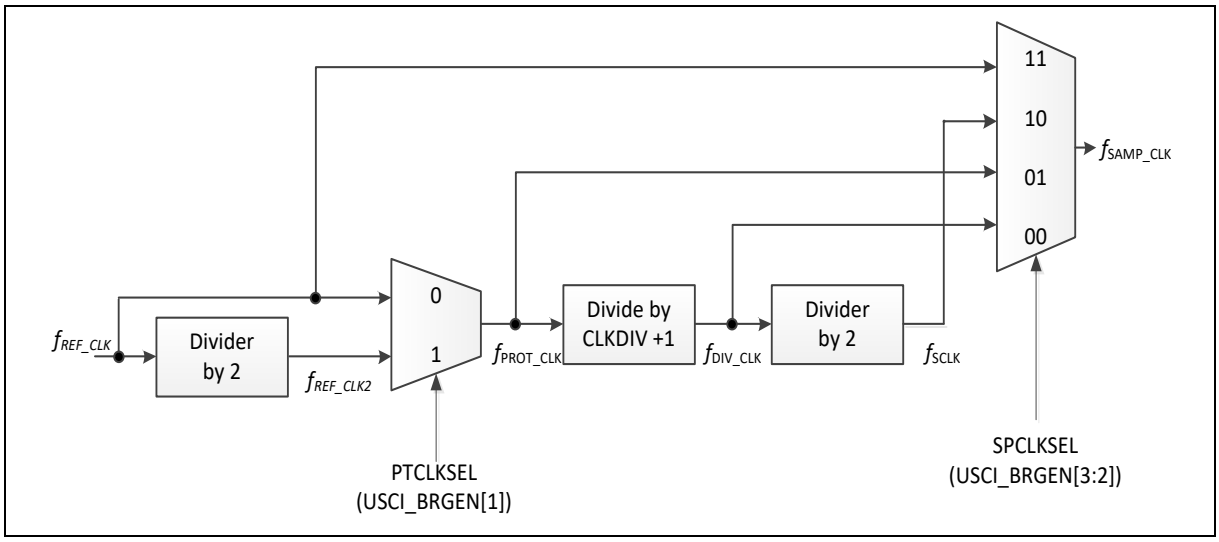


Figure 6.24-9 Basic Clock Divider Counter

Timing Measurement Counter

The timing measurement counter is used for time interval measurement and is enabled by TMCNTEN (USCI_BRGEN [4]) = 1. When TMCNTSRC (USCI_BRGEN [5]) is set to 1, the timer works on f_{DIV_CLK}, otherwise, the timer works independently from f_{PROT_CLK}. Therefore, any serial data reception or transmission can continue while the timer is performing timing measurements. The timer counts the length of protocol-related signals with f_{PROT_CLK} or f_{DIV_CLK}. It stops counting when it reaches the user-specified value.

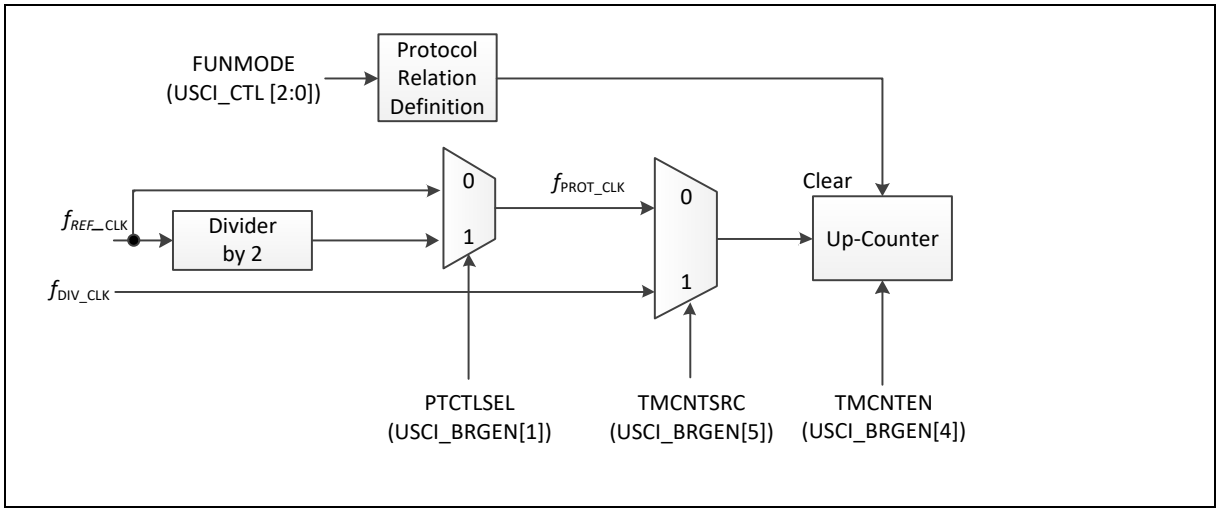


Figure 6.24-10 Block of Timing Measurement Counter

The timing measurement counter is used to perform time-out function or auto-baud rate mechanism. Its functionality depends on the selected protocol as shown below.

- UART: The timing measurement counter is used in auto baud rate detection.
- SPI: The timing measurement counter is used for counting the slave time-out period.

- I²C: The timing measurement counter indicates time-out clock cycle.

Sample Time Counter

A sample time counter associated to the protocol related counter defining protocolspecific timings, such shift control signals or bit timings, based on the input frequency f_{SAMP_CLK}. The sample time counter allows generating time intervals for protocol-specific purposes. The period of a sample frequency f_{PDS_CNT} is given by the selected input frequency f_{SAMP_CLK} and the programmed pre-divider value (PDSCNT (USCI_BRGEN [9:8])). The meaning of the sample time depends on the selected protocol. Please refer to the corresponding chapters for more protocol-specific information.

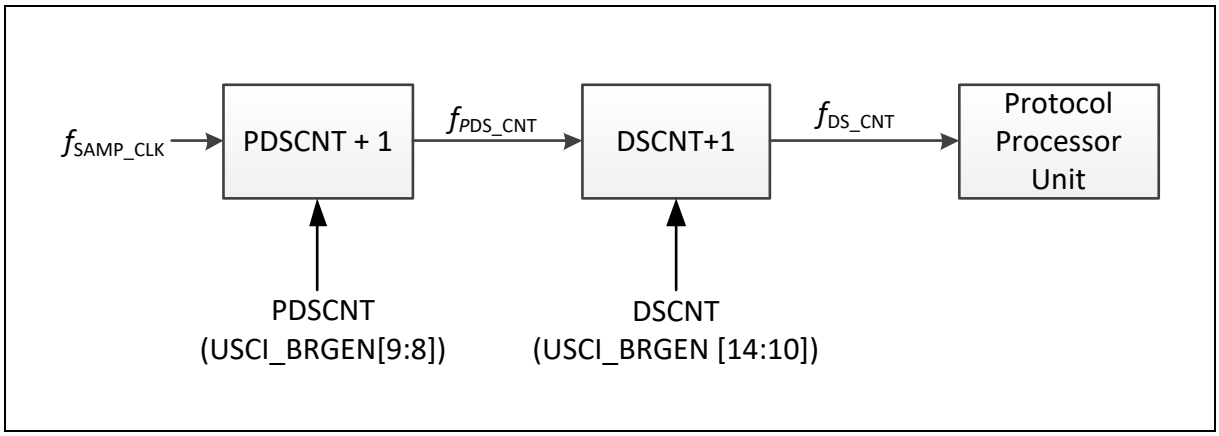


Figure 6.24-11 Sample Time Counter

6.24.4.6 Data Transfer Events and Interrupts

The data transfer events are based on the transmission or reception of a data word. The related indication flags are located in register USCI_PROTSTS. All events can be individually enabled for interrupt generation. If the FUNMODE (USCI_CTL [2:0]) is set to 0, the USCI is disabled. When FUNMODE (USCI_CTL [2:0]) is setting for a protocol port, the internal states will be controlled by logic hardware of the selected protocol.

- Transmit start interrupt event to indicate that a data word has been started:

  A transmit start interrupt event occurs when the data is loaded into transmitted shift register. It is indicated by flag TXSTIF (USCI_PROTSTS [1]) and, if enabled, leads to transmit start interrupt.

- Transmit end interrupt event to indicate that a data word transmission has been done:

  A transmit end interrupt event occurs when the current transmit data in shift register had been finished. It is indicated by flag TXENDIF (USCI_PROTSTS [2]) and, if enabled, leads to transmit end interrupt. This event also indicates when the shift control settings (word length, shift direction, etc.) are internally “frozen” for the current data word transmission. In UART and I²C mode, the transmit data valid is according to TXEMPTY (USCI_BUFSTS [8]) and protocol relative internal signal with the transmit end interrupt event.

- Receiver start event to indicate that a data word reception has started:

  When the receive clock edge that shifts in the first bit of a new data word is detected and reception is enabled, a receiver start event occurs. It is indicated by flag RXSTIF (USCI_PROTSTS [3]) and, if enabled, leads to receiver start interrupt.

- Receive event to indicate that a data word has been received:

  If a new received word becomes available in the receive buffer, a receive event occurs. It

is indicated by flag RXENDIF (USCI_PROTSTS [4]) and, if enabled, leads to receive interrupt.

- Data lost event to indicate a loss of the newest received data word:

  If the data word available in register USCI_RXDAT (oldest data word from RX_BUF0 or RX_BUF1) has not been read out and the receive buffer is FULL, the new incoming data will lose and this event occurs. It is indicated by flag RXOVIF (USCI_BUFSTS[3]) and, if enabled, leads to a protocol interrupt.

The general event and interrupt structure is shown in Figure 6.24-12.

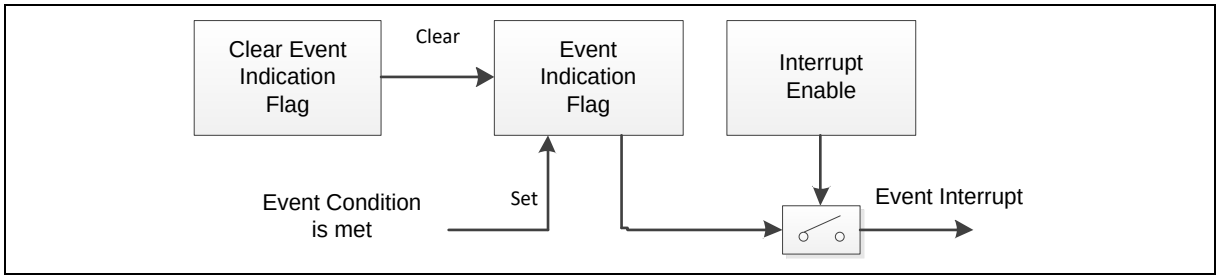


Figure 6.24-12 Event and Interrupt Structure

Each general interrupt enable can set by RXENDIEN, RXSTIEN, TXENDIEN, and TXSTIEN of USCI_INTEN [4:1]. The events are including receive end interrupt event, receive start interrupt event, transmit end interrupt event, and transmit start interrupt event. For protocol-specific interrupt, it is specified in each protocol interrupt enable register.

If a defined condition is met, an event is detected and an event indication flag becomes automatically set. The flag stays set until it is cleared by software. If enabled, an interrupt can be generated if an event is detected.

The registers, bits and bit fields indicate the data transfer events and control the general interrupts of a USCI are shown in Table 6.24-3.

Event	Indication Flag	Indication Cleared By	Interrupt Enabled By
Transmit start interrupt event	TXSTIF (USCI_PROTSTS [1])	It is cleared by software writes 1 to corresponding interrupt bit of USCI_PROTSTS.	TXSTIEN (USCI_INTEN [1])
Transmit end interrupt event	TXENDIF (USCI_PROTSTS [2])		TXENDIEN (USCI_INTEN [2])
Receive start interrupt event	RXSTIF (USCI_PROTSTS [3])		RXSTIEN (USCI_INTEN [3])
Receive end interrupt event	RXENDIF (USCI_PROTSTS [4])		RXENDIEN (USCI_INTEN [4])

Table 6.24-3 Data Transfer Events and Interrupt Handling

6.24.4.7 Protocol-specific Events and Interrupts

These events are related to protocol-specific actions that are described in the corresponding protocol chapters. The related indication flags are located in register USCI_PROTSTS. All events can be individually enabled for the generation of the common protocol interrupt.

Event	Indication Flag	Indication Cleared By	Interrupt Enabled By

Protocol-specific events in UART mode	USCI_PROTSTS [17:16] and USCI_PROTSTS [11:5]	It is cleared by software writes 1 to corresponding interrupt bit of USCI_PROTSTS.	USCI_PROTIEN[2:1]
Protocol-specific events in SPI mode	USCI_PROTSTS [9:8], USCI_PROTSTS [6:5]		USCI_PROTIEN [3:0]
Protocol-specific events in I^2C mode	USCI_PROTSTS [13:8], USCI_PROTSTS [5]		USCI_PROTIEN [6:0]

Table 6.24-4 Protocol-specific Events and Interrupt Handling

6.24.4.8 Wake-up

The protocol-related wake-up functional information is located in the Wake-up Control Register (USCI_WKCTL) and in the Wake-up Status Register (USCI_WKSTS). These registers are shared between the available protocols. As a consequence, the meaning of the bit positions in these registers is different within the protocols.

6.24.4.9 PDMA

The USCI supports PDMA transfer function. When PDMAEN (USCI_PDMACTL [3]) is set to 1, the PDMA function is enabled.

When TXPDMAEN (USCI_PDMACTL [1]) is set to 1, the controller will issue request to PDMA controller to start the PDMA transmission process automatically.

When RXPDMAEN (USCI_PDMACTL [2]) is set to 1, the controller will start the PDMA reception process. USCI will issue request to PDMA controller automatically when there is data in the receive FIFO buffer.

In UART function, the requirement of RXPDMAEN will be cleared and hold if there is any error condition events including frame error, parity error or break detection. The user shall read out the current data and then the requirement of RXPDMAEN will send to the PDMA module in the next data.

6.25 I²C Serial Interface Controller (I²C)

6.25.1 Overview

I²C is a two-wire, bi-directional serial bus that provides a simple and efficient method of data exchange between devices. The I²C standard is a true multi-master bus including collision detection and arbitration that prevents data corruption if two or more masters attempt to control the bus simultaneously.

There are three sets of I²C controllers which support Power-down wake-up function.

6.25.2 Features

The I²C bus uses two wires (SDA and SCL) to transfer information between devices connected to the bus. The main features of the I²C bus include:

- Supports up to three I²C ports
- Master/Slave mode
- Bidirectional data transfer between masters and slaves
- Multi-master bus (no central master)
- Supports Standard mode (100 kbps), Fast mode (400 kbps) and Fast mode plus (1 Mbps)
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
- Serial clock synchronization allow devices with different bit rates to communicate via one serial bus
- Serial clock synchronization used as a handshake mechanism to suspend and resume serial transfer
- Built-in 14-bit time-out counter requesting the I²C interrupt if the I²C bus hangs up and timer-out counter overflow
- Programmable clocks allow for versatile rate control
- Supports 7-bit addressing and 10-bit addressing mode
- Supports multiple address recognition (four slave address with mask option)
- Supports Power-down wake-up function
- Supports PDMA with one buffer capability
- Supports setup/hold time programmable
- Supports Bus Management (SM/PM compatible) function.

6.25.3 Block Diagram

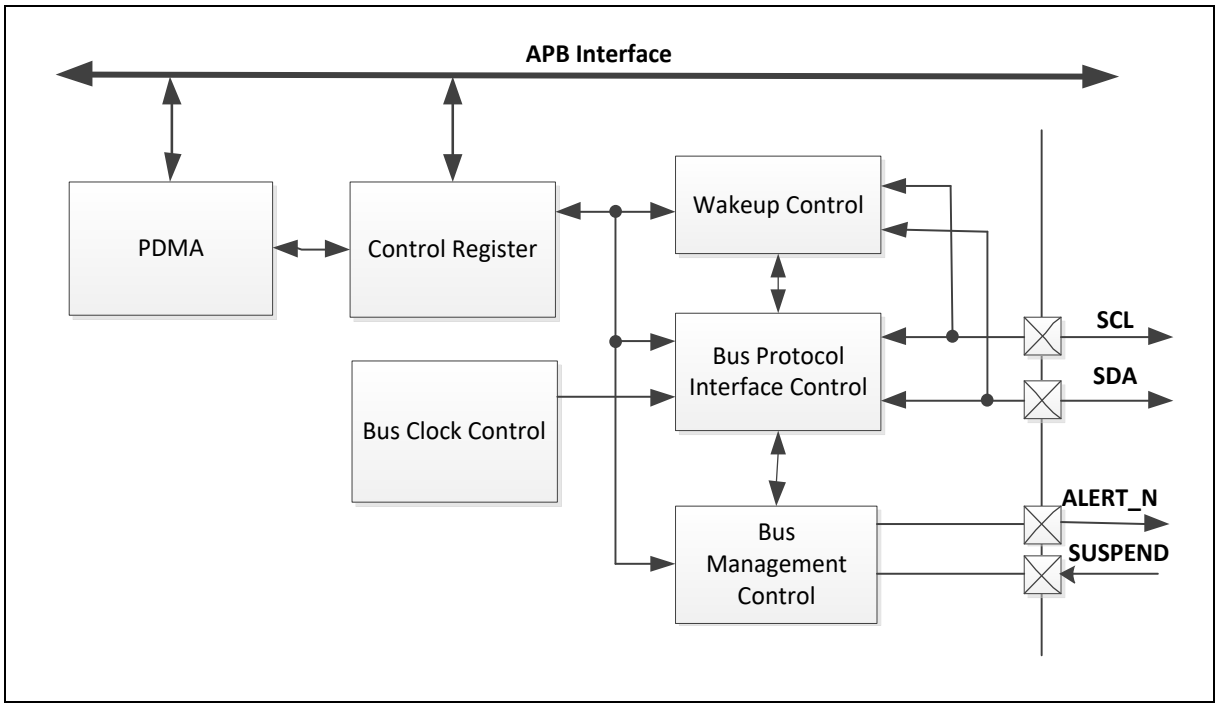


Figure 6.25-1 I²C Controller Block Diagram

6.25.4 Basic Configuration

6.25.4.1 I2C0 basic configurations

- Clock source Configuration
  - Enable I2C0 peripheral clock in I2C0CKEN (CLK_APBCLK0[8]).
- Reset Configuration
  - Reset I2C0 controller in I2C0RST (SYS_IPRST1[8]).
- Pin configuration

Group	Pin Name	GPIO	MFP
I2C0	I2C0_SCL	PC.12, PD.7, PE.13, PF.3	MFP4
		PB.5	MFP6
		PA.5, PC.1	MFP9
	I2C0_SDA	PC.8, PC.11, PD.6, PF.2	MFP4
		PB.4	MFP6
		PA.4, PC.0	MFP9
	I2C0_SMBAL	PG.2	MFP4
		PC.3	MFP9
	I2C0_SMBSUS	PG.3	MFP4
		PC.2	MFP9

6.25.4.2 I2C1 Basic Configurations

- Clock Source Configuration
  - Enable I2C1 peripheral clock in I2C1CKEN (CLK_APBCLK0[9]).
- Reset Configuration
  - Reset I2C1 controller in I2C1RST (SYS_IPRST1[9]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
I2C1	I2C1_SCL	PF.0	MFP3
		PA.12, PD.5	MFP4
		PG.2	MFP5
		PB.11	MFP7
		PA.7, PE.1	MFP8
		PA.3, PB.1, PC.5	MFP9
	I2C1_SDA	PF.1	MFP3
		PA.13, PD.4	MFP4
		PG.3	MFP5
		PB.10	MFP7
		PA.6, PE.0	MFP8
		PA.2, PB.0, PC.4	MFP9
	I2C1_SMBAL	PB.9	MFP7
		PC.7, PH.8	MFP8
	I2C1_SMBSUS	PB.8	MFP7
		PC.6, PH.9	MFP8

6.25.4.3 I2C2 Basic Configurations

- Clock source Configuration
  - Enable I2C2 peripheral clock in I2C2CKEN (CLK_APBCLK0[10]).
- Reset Configuration
  - Reset I2C2 controller in I2C2RST (SYS_IPRST1[10]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
I2C2	I2C2_SCL	PD.9	MFP3
		PA.14, PD.1	MFP6
		PA.11	MFP7
		PB.13	MFP8
		PA.1, PH.8	MFP9

	I2C2_SDA	PD.8	MFP3
		PA.15, PD.0	MFP6
		PA.10	MFP7
		PB.12	MFP8
		PA.0, PH.9	MFP9
	I2C2_SMBAL	PB.15	MFP8
	I2C2_SMBSUS	PB.14	MFP8

6.25.5 Functional Description

On I^{2}C bus, data is transferred between a Master and a Slave. Data bits transfer on the SCL and SDA lines are synchronously on a byte-by-byte basis. Each data byte is 8-bit long. There is one SCL clock pulse for each data bit with the MSB being transmitted first, and an acknowledge bit follows each transferred byte. Each bit is sampled during the high period of SCL; therefore, the SDA line may be changed only during the low period of SCL and must be held stable during the high period of SCL. A transition on the SDA line while SCL is high is interpreted as a command (START or STOP). Please refer to Figure 6.25-2 for more detailed I^{2}C BUS Timing.

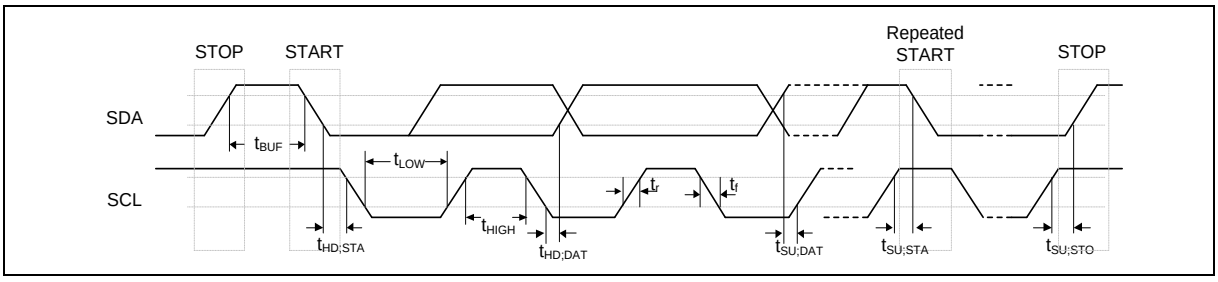


Figure 6.25-2 I^{2}C Bus Timing

The device's on-chip I^{2}C provides the serial interface that meets the I^{2}C bus standard mode specification. The I^{2}C port handles byte transfers autonomously. To enable this port, the bit I2CEN in I2C_CTL0 should be set to '1'. The I^{2}C hardware interfaces to the I^{2}C bus via two pins: SDA and SCL. When I/O pins are used as I^{2}C ports, user must set the pins function to I^{2}C in advance.

Note: Pull-up resistor is needed for I^{2}C operation as the SDA and SCL are open-drain pins.

6.25.5.1 I^{2}C Protocol

Figure 6.25-3 shows the typical I^{2}C protocol. Normally, a standard communication consists of four parts:

- START or Repeated START signal generation
- Slave address and R/W bit transfer
- Data transfer
- STOP signal generation

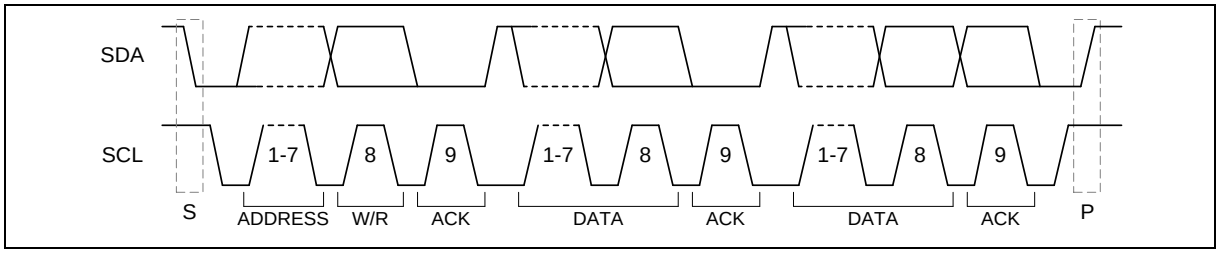


Figure 6.25-3 I^{2}C Protocol

- START or Repeated START signal

  When the bus is free/idle, which means no master device is engaging the bus (both SCL and SDA lines are high), a master can initiate a transfer by sending a START signal. A START signal, usually referred to as the “S” bit, is defined as a HIGH to LOW transition on the SDA line while SCL is HIGH. The START signal denotes the beginning of a new data transmission.

  After having sent the address byte (address and read/write bit), the master may send any number of bytes followed by a stop condition. Instead of sending the stop condition it is also allowed to send another start condition again followed by an address (and of course including a read/write bit) and more data. The start condition is called as Repeat START (Sr). This is defined recursively allowing any number of start conditions to be sent. The purpose of this is to allow combined write/read operations to one or more devices without releasing the bus and thus with the guarantee that the operation is not interrupted. The controller uses this method to communicate with another slave or the same slave in a different transfer direction (e.g. from writing to a device to reading from a device) without releasing the bus.

- STOP signal

  The master can terminate the communication by generating a STOP signal. A STOP signal, usually referred to as the “P” bit, is defined as a LOW to HIGH transition on the SDA line while SCL is HIGH.

  Figure 6.25-4 shows the waveform of START, Repeat START and STOP.

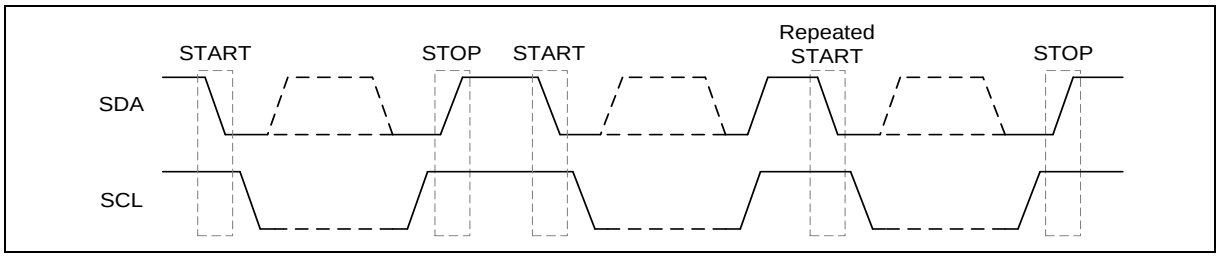


Figure 6.25-4 START and STOP Conditions

- Slave Address Transfer

  After a (Repeated) START condition, the master sends a slave address to identify the target device of the communication. The start address can comprise one or two address bytes (for 7-bit or for 10-bit addressing schemes). After an address byte, a slave sensitive to the transmitted address has to acknowledge the reception.

  Therefore, the slave’s address can be programmed in the device, where it is compared to the received address. In case of a match, the slave answers with an acknowledge (SDA = 0). Slaves that are not targeted answer with a non-acknowledge (SDA = 1). In addition to the match of the programmed address, another address byte value has to be answered

with an acknowledge if the slave is capable to handle the corresponding requests.

- Data Transfer

  When a slave receives a correct address with an R/W bit, the data will follow R/W bit specified to transfer. Each transferred byte is followed by an acknowledge bit on the 9th SCL clock cycle. If the slave signals a Not Acknowledge (NACK), the master can generate a STOP signal to abort the data transfer or generate a Repeated START signal and start a new transfer cycle.

  If the master, as a receiving device, does Not Acknowledge (NACK) the slave, the slave releases the SDA line for the master to generate a STOP or Repeated START signal. The Figure 6.25-5 and Figure 6.25-6 shows the waveform of bit transfer and acknowledge.

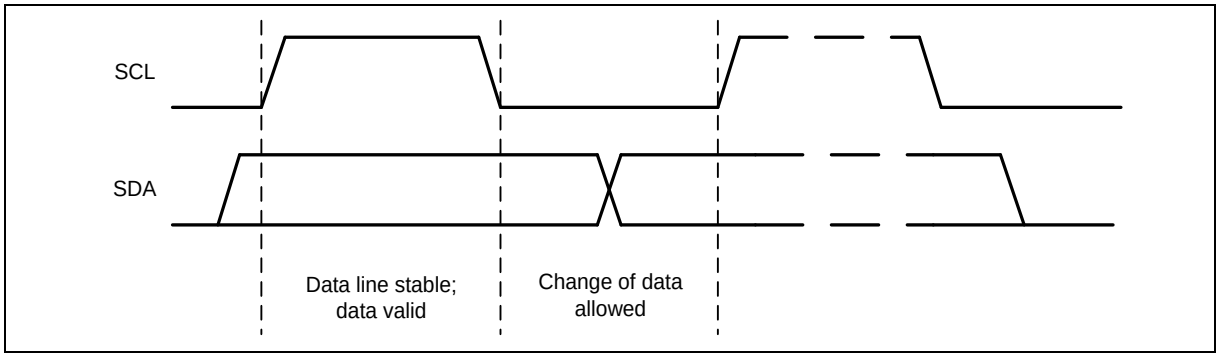


Figure 6.25-5 Bit Transfer on the I^{2}C Bus

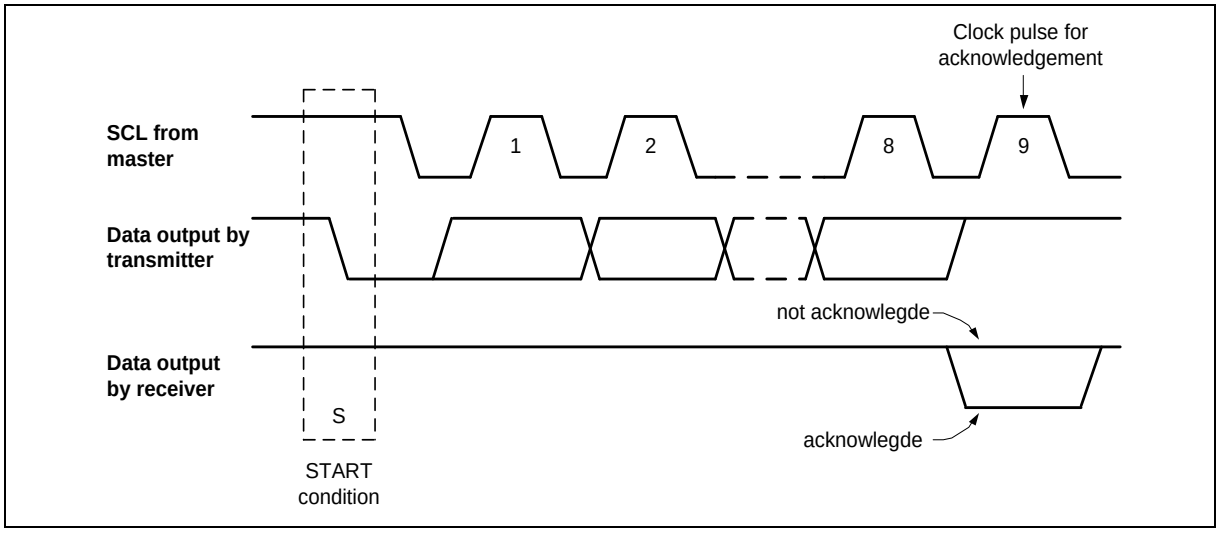


Figure 6.25-6 Acknowledge on the I^{2}C Bus

- Data transfer on I^{2}C bus

  Figure 6.25-7 shows a master transmits data to slave by 7-bit. A master addresses a slave with a 7-bit address and 1-bit write index to denote that the master wants to transmit data to the slave. The master keeps transmitting data after the slave returns acknowledge to the master.

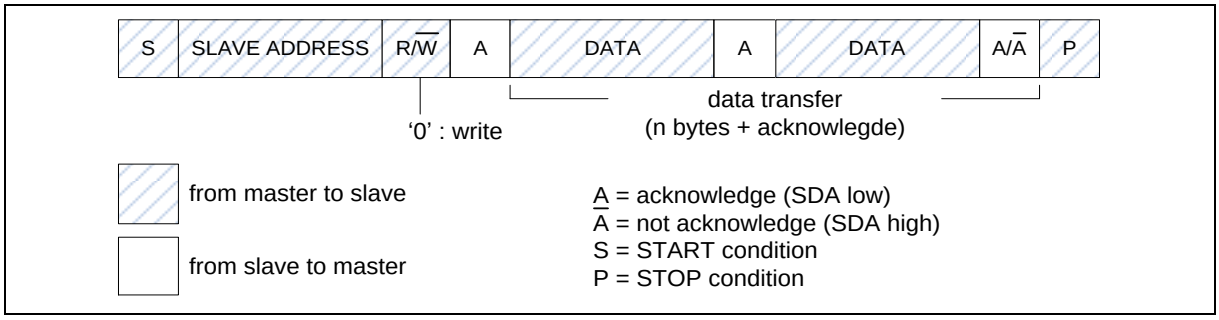


Figure 6.25-7 Master Transmits Data to Slave by 7-bit

Figure 6.25-8 shows a master read data from slave by 7-bit. A master addresses a slave with a 7-bit address and 1-bit read index to denote that the master wants to read data from the slave. The slave will start transmitting data after the slave returns acknowledge to the master.

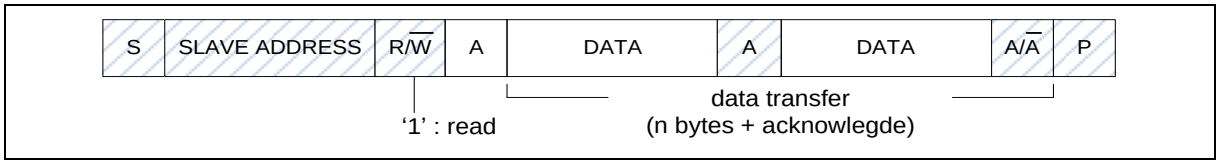


Figure 6.25-8 Master Reads Data from Slave by 7-bit

Figure 6.25-9 shows a master transmits data to slave by 10-bit. A master addresses a slave with a 10-bit address. First byte contains 10-bit address indicator (5'b11110) and 2-bit address with write index, second byte contains 8-bit address. The master keeps transmitting data after the second byte end. Note that 7-bit and 10-bit address device can work on the same bus.

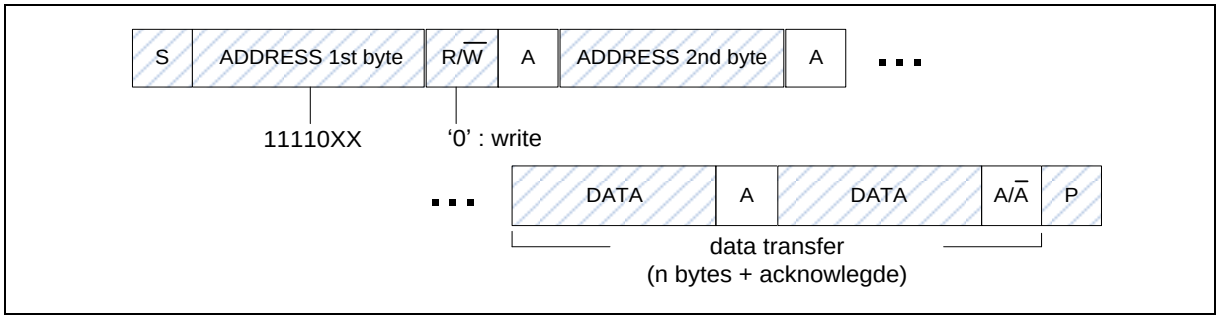


Figure 6.25-9 Master Transmits Data to Slave by 10-bit

Figure 6.25-10 shows a master read data from slave by 10-bit. A master addresses a slave with a 10-bit address. First mater transmits 10-bit address to slave, after that master transmits first byte with read index. The slave will start transmitting data after the first byte with read index.

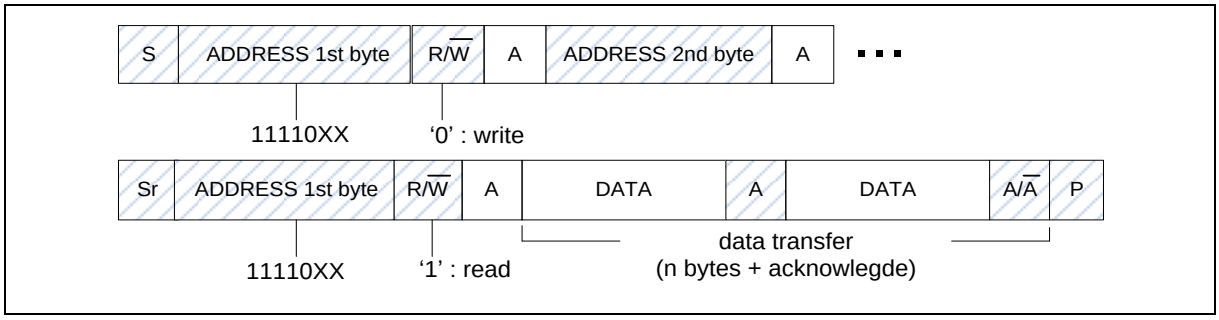


Figure 6.25-10 Master Reads Data from Slave by 10-bit

6.25.5.2 Operation Modes

The on-chip I²C ports support three operation modes, Master, Slave, and General Call Mode.

In a given application, I²C port may operate as a master or as a slave. In Slave mode, the I²C port hardware looks for its own slave address and the general call address. If one of these addresses is detected, and if the slave is willing to receive or transmit data from/to master(by setting the AA bit), acknowledge pulse will be transmitted out on the 9th clock, hence an interrupt is requested on both master and slave devices if interrupt is enabled. When the microcontroller wishes to become the bus master, hardware waits until the bus is free before entering Master mode so that a possible slave action is not be interrupted. If bus arbitration is lost in Master mode, I²C port switches to Slave mode immediately and can detect its own slave address in the same serial transfer.

To control the I²C bus transfer in each mode, user needs to set I2C_CTL0, I2C_DAT registers according to current status code of I2C_STATUS0 register. In other words, for each I²C bus action, user needs to check current status by I2C_STATUS0 register, and then set I2C_CTL0, I2C_DAT registers to take bus action. Finally, check the response status by I2C_STATUS0.

The bits, STA, STO and AA in I2C_CTL0 register are used to control the next state of the I²C hardware after SI flag of I2C_CTL0 [3] register is cleared. Upon completion of the new action, a new status code will be updated in I2C_STATUS0 register and the SI flag of I2C_CTL0 register will be set. But the SI flag will not be set when I²C STOP. If the I²C interrupt control bit INTEN (I2C_CTL0 [7]) is set, appropriate action or software branch of the new status code can be performed in the Interrupt service routine.

Figure 6.25-11 shows the current I²C status code is 0x08, and then set I2C_DATA=SLA+W and (STA,STO,SI,AA) = (0,0,1,x) to send the address to I²C bus. If a slave on the bus matches the address and response ACK, the I2C_STATUS0 will be updated by status code 0x18.

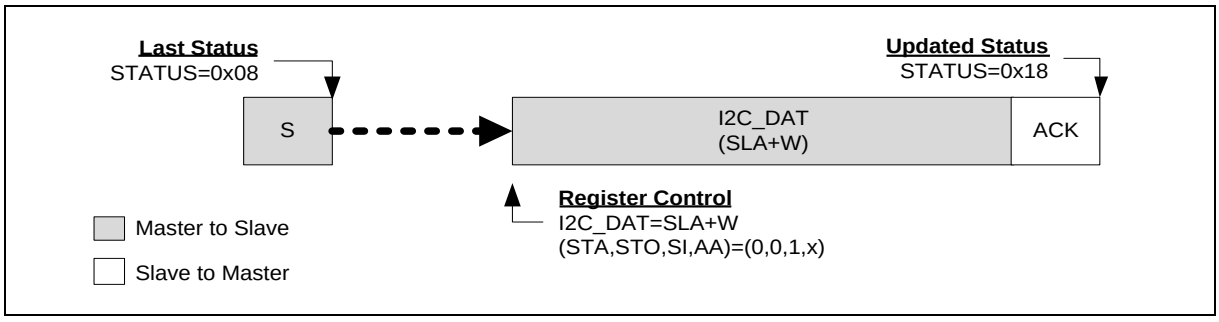


Figure 6.25-11 Control I²C Bus according to the Current I²C Status

Master Mode

In Figure 6.25-12 and Figure 6.25-13, all possible protocols for I²C master are shown. User needs to follow proper path of the flow to implement required I²C protocol.

In other words, user can send a START signal to bus and I^{2}C will be in Master Transmitter (MT) mode (Figure 6.25-12) or Master receiver (MR) mode (Figure 6.25-13) after START signal has been sent successfully and new status code would be 0x08. Followed by START signal, user can send slave address, read/write bit, data and Repeat START, STOP to perform I^{2}C protocol.

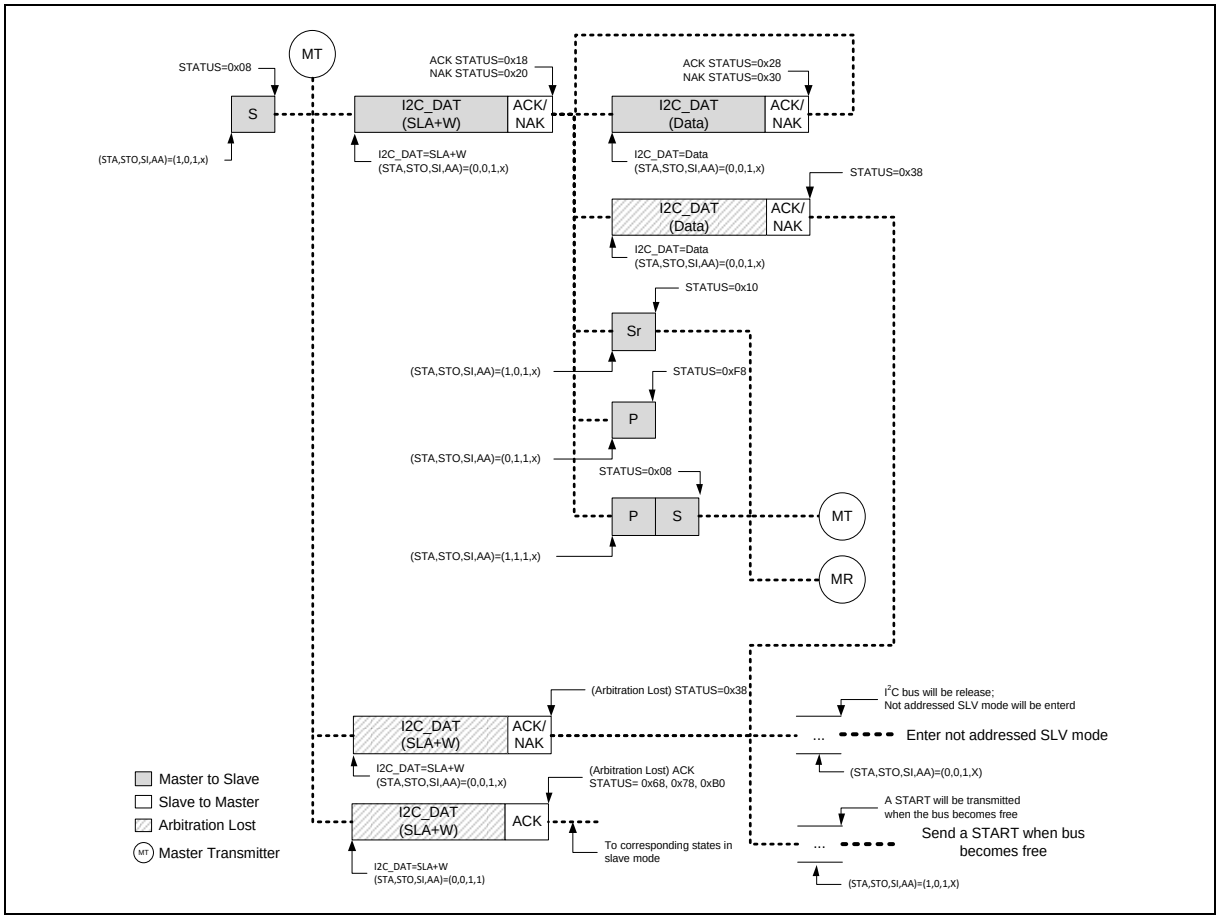


Figure 6.25-12 Master Transmitter Mode Control Flow

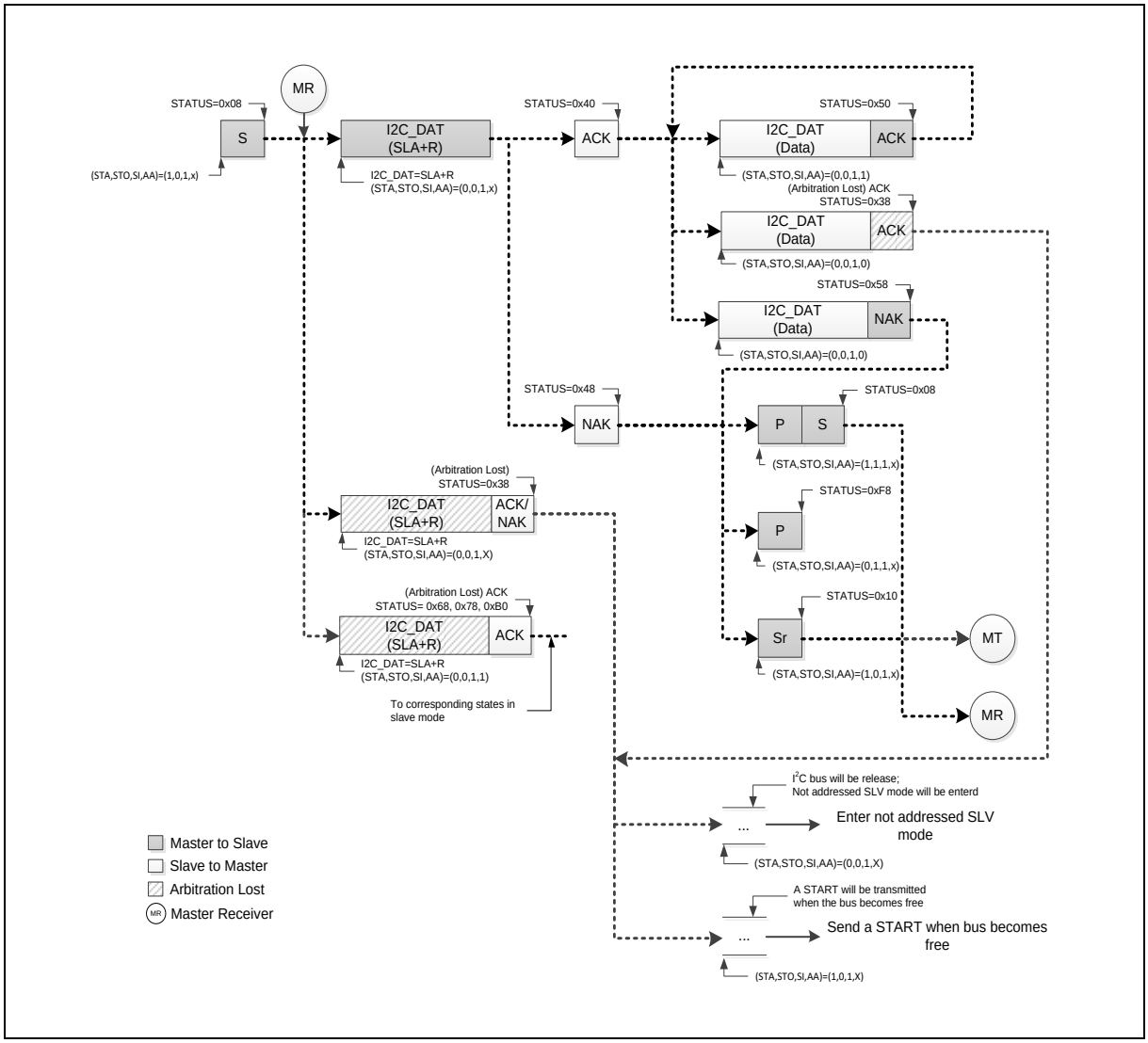


Figure 6.25-13 Master Receiver Mode Control Flow

If the I²C is in Master mode and gets arbitration lost, the status code will be 0x38. In status 0x38, user may set (STA, STO, SI, AA) = (1, 0, 1, X) to send START to re-start Master operation when bus become free. Otherwise, user may set (STA, STO, SI, AA) = (0, 0, 1, X) to release I²C bus and enter not addressed Slave mode.

Slave Mode

When reset default, I²C is not addressed and will not recognize the address on I²C bus. User can set slave address by I2C_ADDRn (n=0~3) and set (STA, STO, SI, AA) = (0, 0, 1, 1) to let I²C recognize the address sent by master. Figure 6.25-14 shows all the possible flow for I²C in Slave mode. Users need to follow a proper flow (as shown in Figure 6.25-14 to implement their own I²C protocol).

If bus arbitration is lost in Master mode, I²C port switches to Slave mode immediately and can detect its own slave address in the same serial transfer. If the detected address is SLA+W (Master want to write data to Slave) after arbitration lost, the status code is 0x68. If the detected address is SLA+R (Master want to read data from Slave) after arbitration lost, the status code is 0xB0.

Note: During I²C communication, the SCL clock will be released when writing '1' to clear SI flag in

Slave mode.

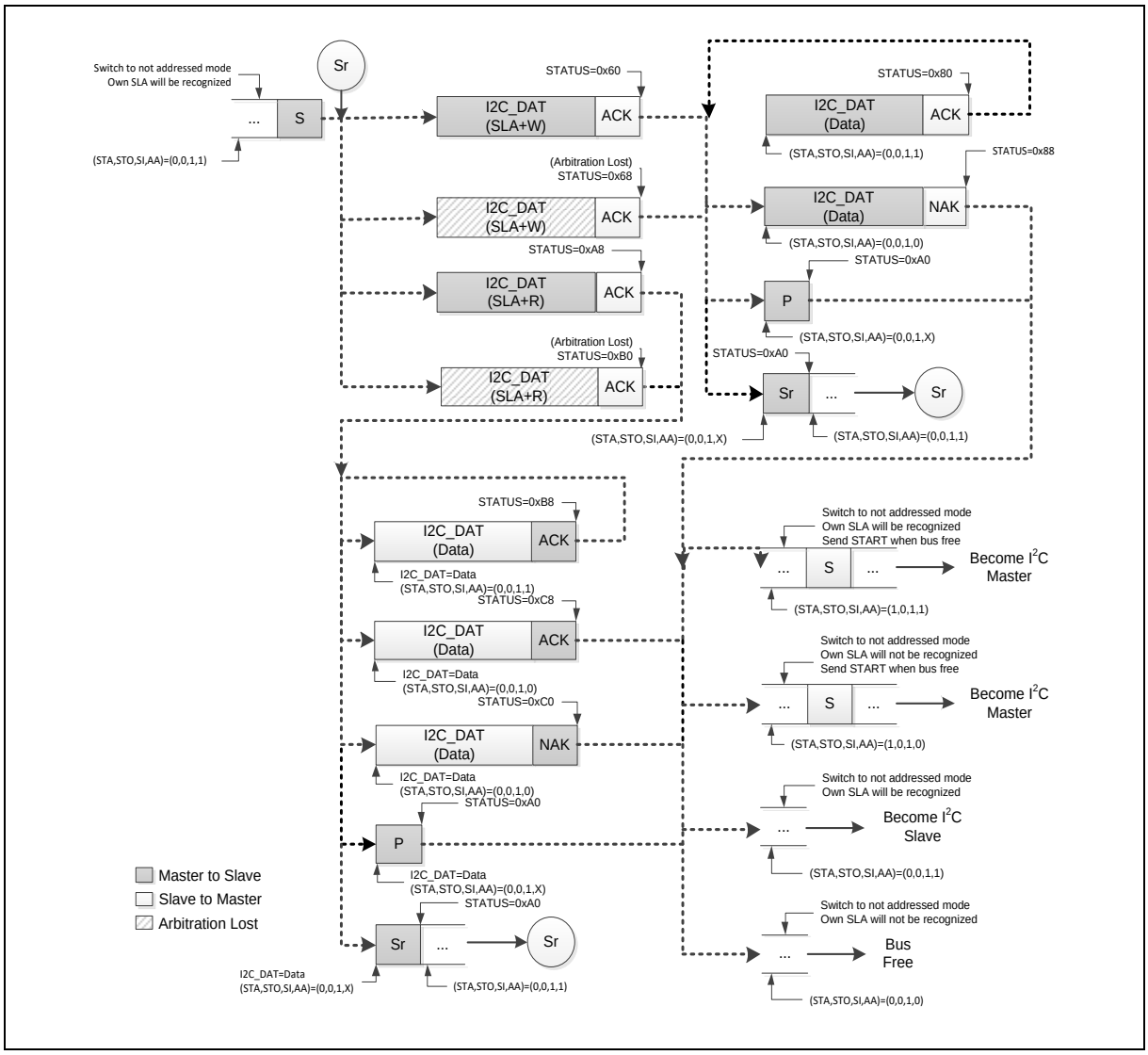


Figure 6.25-14 Slave Mode Control Flow

If I²C is still receiving data in addressed Slave mode but got a STOP or Repeat START, the status code will be 0xA0. User could follow the action for status code 0x88 as shown in the above figure when getting 0xA0 status.

If I²C is still transmitting data in addressed Slave mode but got a STOP or Repeat START, the status code will be 0xA0. User could follow the action for status code 0xC8 as shown in the above figure when getting 0xA0 status.

Note: After slave gets status of 0x88, 0xC8, 0xC0 and 0xA0, slave can switch to not address mode and own SLA will not be recognized. If entering this status, slave will not receive any I²C signal or address from master. At this status, I²C should enter idle mode.

General Call (GC) Mode

If the GC bit (I2C_ADDRn [0]) is set, the I²C port hardware will respond to General Call address (00H). User can clear GC bit to disable general call function. When the GC bit is set and the I²C in Slave

mode, it can receive the general call address by 0x00 after master send general call address to I^{2}C bus, then it will follow status of GC mode.

The GC mode can wake up when address matched. Note that the default address is 0x00, but user must set an address except for 0x00.

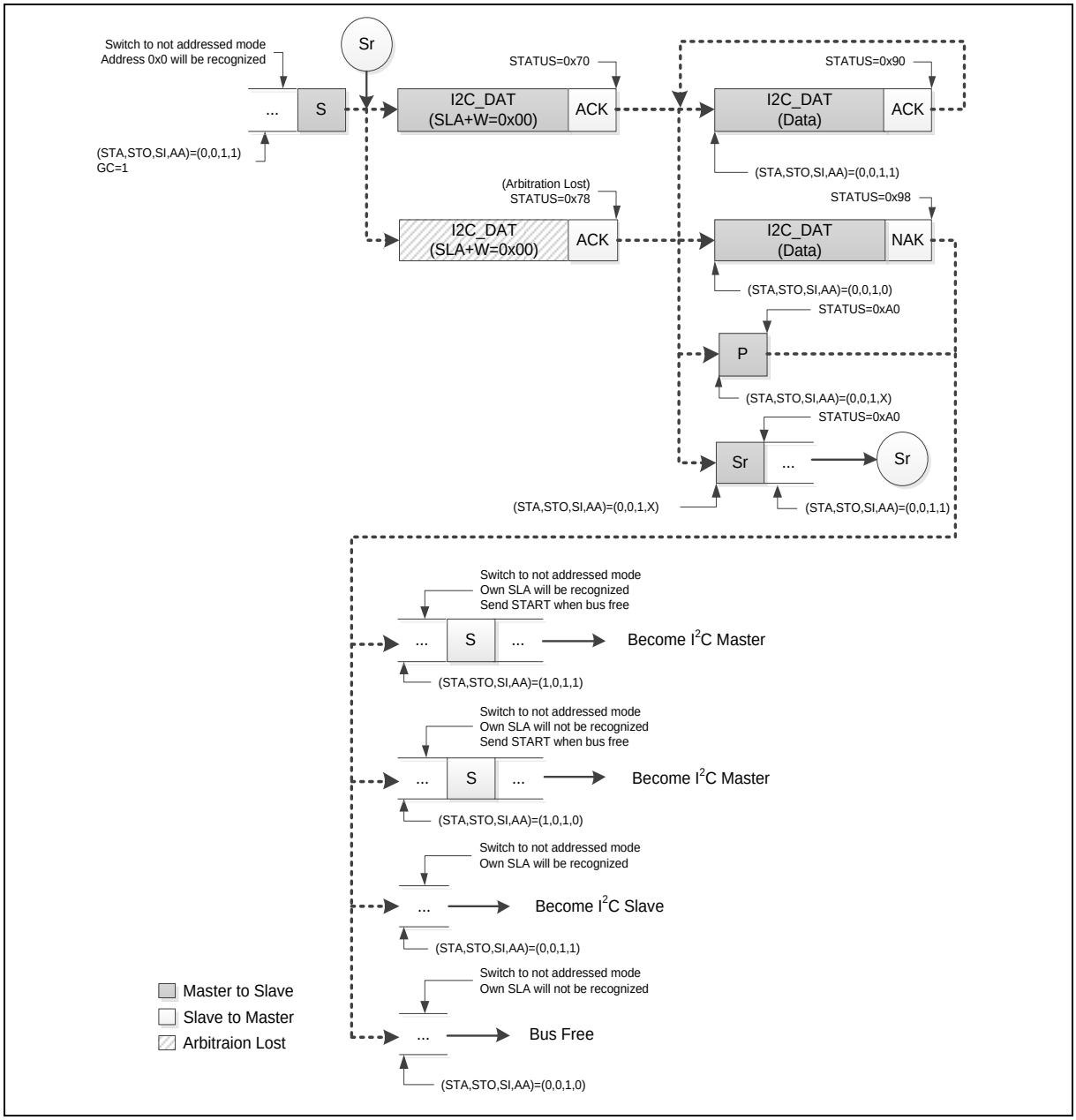


Figure 6.25-15 GC Mode

If I^{2}C is still receiving data in GC mode but got a STOP or Repeat START, the status code will be 0xA0. User could follow the action for status code 0x98 in above figure when getting 0xA0 status.

Note: After slave gets status of 0x98 and 0xA0, slave can switch to not address mode and own SLA will not be recognized. If entering this status, slave will not receive any I^{2}C signal or address from master. At this time, the I^{2}C controller should enter idle mode.

Multi-Master

In some applications, there are two or more masters on the same I^{2}C bus to access slaves, and the masters may transmit data simultaneously. The I^{2}C supports multi-master by including collision detection and arbitration to prevent data corruption.

If for some reason two masters initiate command at the same time, the arbitration procedure determines which master wins and can continue with the command. Arbitration is performed on the SDA signal while the SCL signal is high. Each master checks if the SDA signal on the bus corresponds to the generated SDA signal. If the SDA signal on the bus is low but it should be high, then this master has lost arbitration. The device that has lost arbitration can generate SCL pulses until the byte ends and must then release the bus and go into slave mode. The arbitration procedure can continue until all the data is transferred. This means that in multi-master system each master must monitor the bus for collisions and act accordingly.

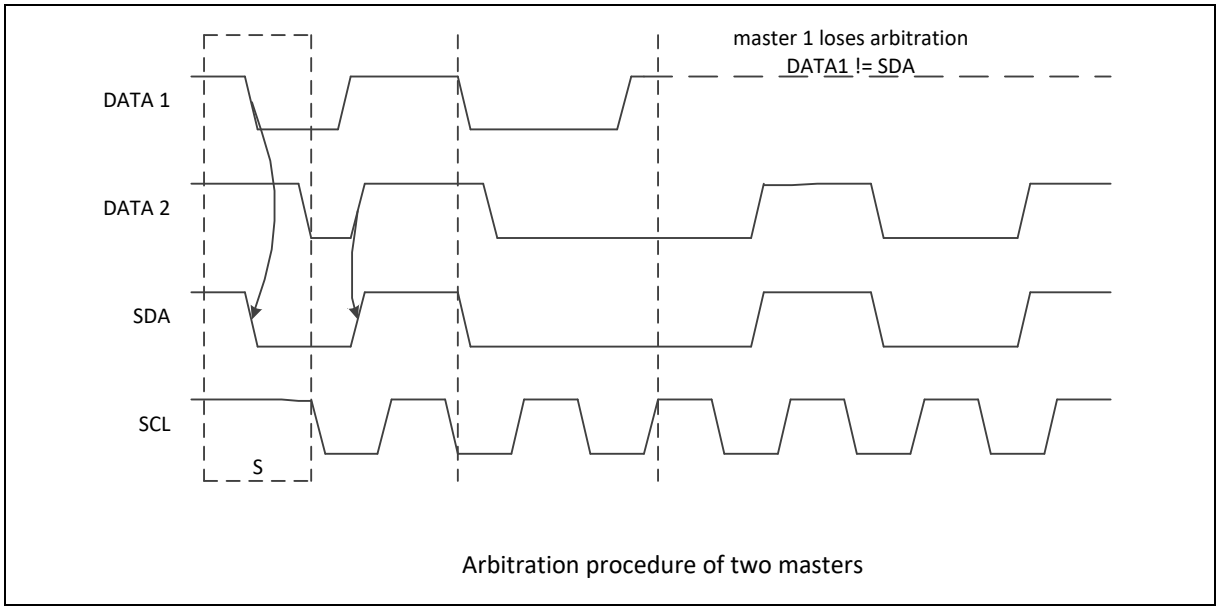


Figure 6.25-16 Arbitration Lost

- When I2C_STATUS0 = 0x38, an “Arbitration Lost” is received. Arbitration lost event maybe occur during the send START bit, data bits or STOP bit. User could set (STA, STO, SI, AA) = (1, 0, 1, X) to send START again when bus free, or set (STA, STO, SI, AA) = (0, 0, 1, X) to not addressed Slave mode. User can detect bus free by ONBUSY (I2C_STATUS1 [8]).
- When I2C_STATUS0 = 0x00, a “Bus Error” is received. To recover I^{2}C bus from a bus error, STO should be set and SI should be cleared, and then STO is cleared to release bus.
  - Set (STA, STO, SI, AA) = (0, 1, 1, X) to stop current transfer
  - Set (STA, STO, SI, AA) = (0, 0, 1, X) to release bus

Bus Management (SMBus/PMBus Compatible)

This section is relevant only when Bus Management feature is supported.

Introduction

The Bus Management is an I^{2}C interface through which various devices can communicate with each other and with the rest of the system. It is based on I^{2}C principles of operation. The Bus Management provides a control bus for system and power management related tasks.

This peripheral is compatible with the SMBUS specification rev 2.0 (http://smbus.org/specs/) and PMBUS specification rev 1.2 (http://pmbus.org/).

The System Management Bus Specification refers to three types of devices.

- A slave is a device that receives or responds to a command.
- A master is a device that issues commands, generates the clocks and terminates the transfer.
- A host is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

This Bus Management peripheral is based on I^2C specification Rev 2.1.

Device Identification – Slave Address

Any device that exists on the Bus Management as a slave has a unique address called the Slave Address. For reference, the following addresses are reserved and must not be used by or assign to any Bus Management device. (Refer to SMBus specification for detail information)

Slave Address Bits 7-1	R/W Bit Bit 0	Comment
0000 000	0	General Call Address
0000 000	1	START byte
0000 001	X	CBUS address
0000 010	X	Address reserved for different bus format
0000 011	X	Reserved for future use
0000 1XX	X	Reserved for future use
0101 000	X	Reserved for ACCESS.bus host
0110 111	X	Reserved for ACCESS.bus default address
1111 0XX	X	10-bit slave addressing
1111 1XX	X	Reserved for future use
0001 000	X	SMBus Host
0001 100	X	SMBus Alert Response Address
1100 001	X	SMBus Device Default Address

Table 6.25-1 Reserved SMBus Address

Bus Protocols

There are eleven possible command protocols for any given device. A device may use any or all of the eleven protocols to communicate. The protocols are Quick Command, Send Byte, Receive Byte, Write Byte, Write Word, Read Byte, Read Word, Process Call, Block Read, Block Write and Block Write-Block Read Process Call. These protocols should be implemented by the user software. (For more details of these protocols, refer to SMBus specification ver. 2.0)

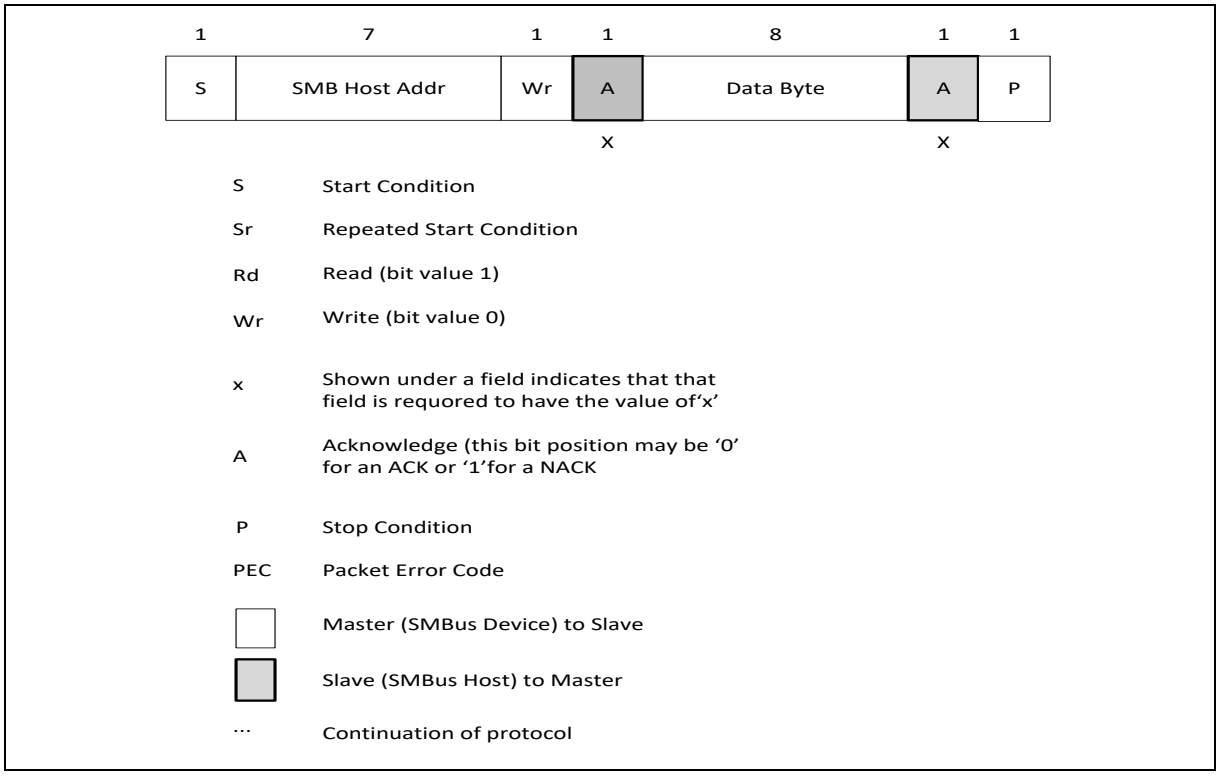


Figure 6.25-17 Bus Management Packet Protocol Diagram Element Key

Address Resolution Protocol (ARP)

Bus Management slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. In order to provide a mechanism to isolate each device for the purpose of address assignment each device must implement a unique device identifier (UDID). This 128-bit number is implemented by software.

This peripheral supports the Address Resolution Protocol (ARP). The Bus Management Device Default Address (0b1100 001) is enabled by setting BUSEN (I2C_BUSCTL[7]), BMDEN (I2C_BUSCTL[2]) and ALERTEN (I2C_BUSCTL[4]) bits. The ARP commands should be implemented by the user software. Arbitration is also performed in slave mode for ARP support.

Received Command and Data acknowledge control

A Bus Management receiver must be able to NACK each received command or data. In order to allow the ACK control in slave mode, the Slave Byte Control mode must be enabled by setting ACKMEN bit (I2C_BUSCTL[0]).

Host Notify Protocol

To prevent message coming to the Bus Management host controller from unknown devices in unknown formats only one method of communication is allowed, a modified form of the Write Word protocol. The standard Write Word protocol is modified by replacing the command code with the alerting device's address.

This peripheral supports the Host Notify protocol by setting the BUSEN (I2C_BUSCTL[7]), BMHEN (I2C_BUSCTL[3]) and ALERTEN (I2C_BUSCTL[4]). In this case the host will acknowledge the Bus Management Host address (0b0001000). This protocol is used when the device acts as a master and the host as a slave.

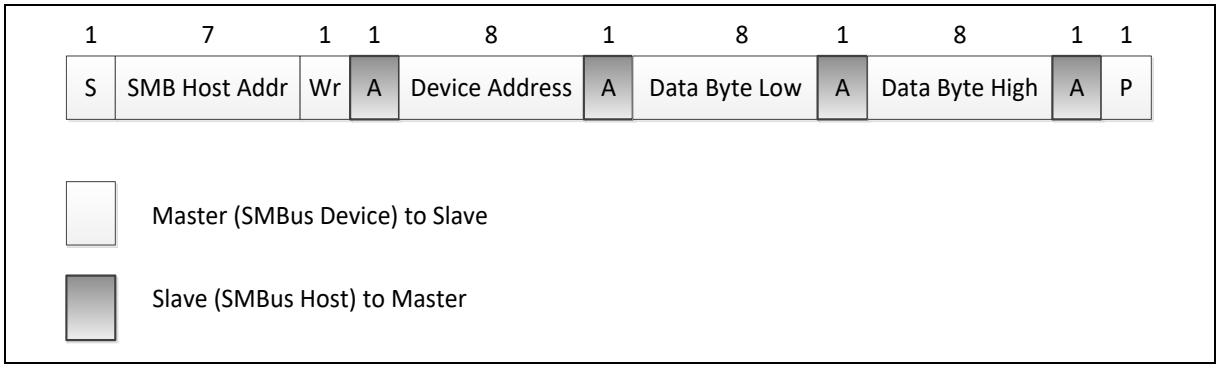


Figure 6.25-187-bit Addressable Device to Host Communication

Bus Management Alert

The Bus Management ALERT optional signal is supported. A slave-only device can signal the host through the Bus Management ALERT pin (GPA[14]/GPE[10]) that it wants to talk. The host processes the interrupt and simultaneously accesses all Bus Management ALERT pin's devices through the Alert Response Address (0b0001 100). Only the device(s) which pulled Bus Management ALERT pin low will acknowledge the Alert Response Address.

When configured as a slave device(BMHEN=0), the Bus Management ALERT pin is pulled low by setting the ALERTEN bit (I2C_BUSCTL[4]). The Alert Response Address (ARA) is enabled at the same time.

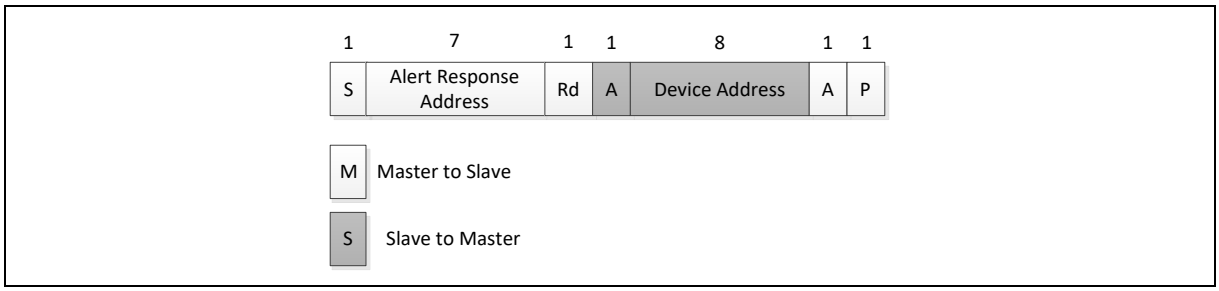


Figure 6.25-197-bit Addressable Device Responds to an ARA

When configured as a host (BMHEN=1), the ALERT flag (I2C_BUSSTS[3]) is set when a falling edge is detected on the Bus Management ALERT pin and ALERTEN=1. When ALERTEN=0, the ALERT line is considered high even if the external Bus Management ALERT pin is low. If the Bus Management ALERT pin is not needed, the Bus Management ALERT pin can be used as a standard GPIO if ALERTEN = 0;

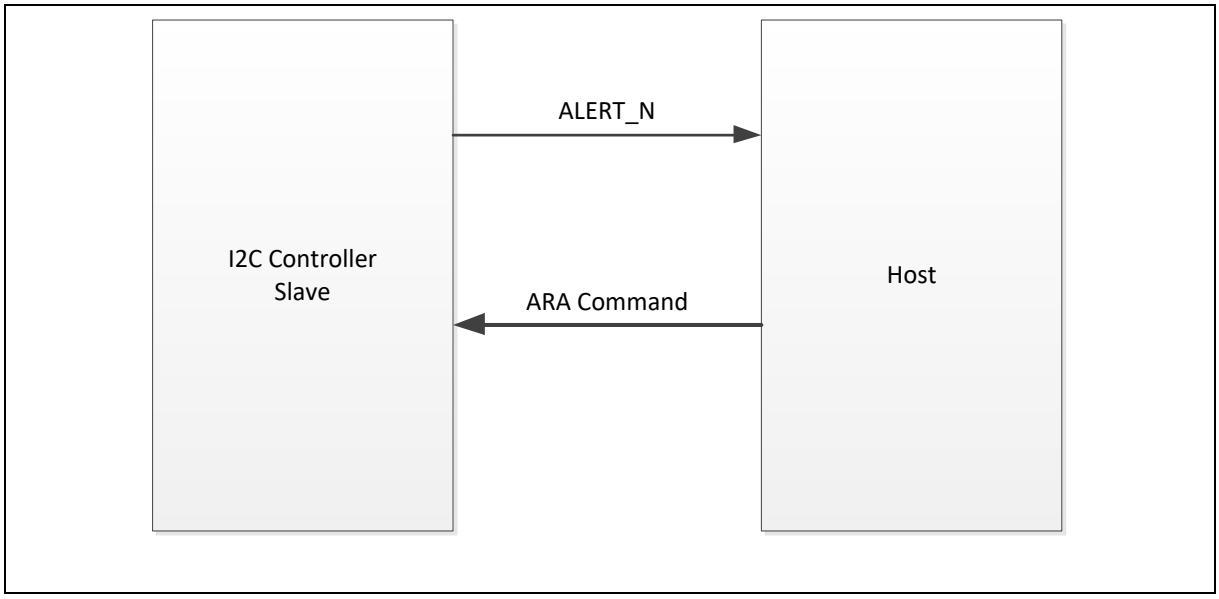


Figure 6.25-20 Bus Management ALERT function

Packet Error Checking

A packet error checking mechanism has been introduced in the SMBus specification to improve reliability and communication robustness. Packet Error Checking is implemented by appending a Packet Error Code (PEC) at the end of each message transfer. The PEC is calculated by using the C(x) = x8 + x2 + x + 1 CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The peripheral embeds a hardware PEC calculator when the PECEN bit (I2C_BUSCTL[1]) is set and allows to send a Not Acknowledge automatically when the received byte does not match with the hardware calculated PEC. The calculated value of PEC also can be read back on I2C_PKTCRC.

Time-out

This peripheral embeds hardware timers in order to be compliant with the 3 time-outs defined in SMBus specification ver. 2.0.

Bus Management Time-out:

The SCLK low time-out condition when bus no IDLE

$T_{Time\text{-}out}$ = (BUSTO(I2C_BUSTOUT[7:0]) +1) x 16x1024 (14-bit) x T_{PCLK} (if TOCDIV4 = 0).

= (BUSTO(I2C_BUSTOUT[7:0])+1) x 16x1024 (14-bit) x 4 x T_{PCLK} (if TOCDIV4 = 1)

The bus idle condition (both SCLK and SDA high) when bus IDLE

$T_{Time\text{-}out}$ = (BUSTO(I2C_BUSTOUT[7:0])+ 1) x 4 x T_{PCLK}.

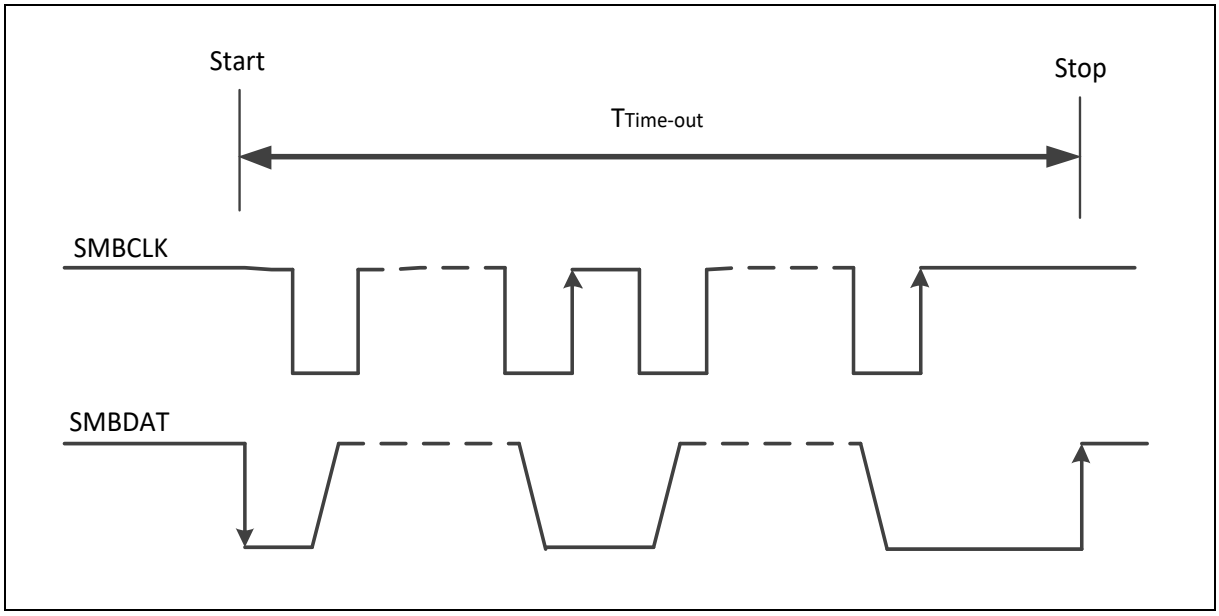


Figure 6.25-21 Bus Management Time Out Timing

Bus Clock Low Time-out:

In Master mode, the Master cumulative clock low extend time ($T_{LOW:MEXT}$) is detected

In Slave mode, the slave cumulative clock low extend time ($T_{LOW:SEXT}$) is detected

$T_{TLOW:EXT}$ = (CLKTO (I2C_CLKTOUT[7:0])+1) x 16x1024 (14-bit) x T_{PCLK} (if TOCDIV4= 0).

= (CLKTO (I2C_CLKTOUT[7:0])+1) x 16x1024 (14-bit) x 4 x T_{PCLK} (if TOCDIV4= 1)

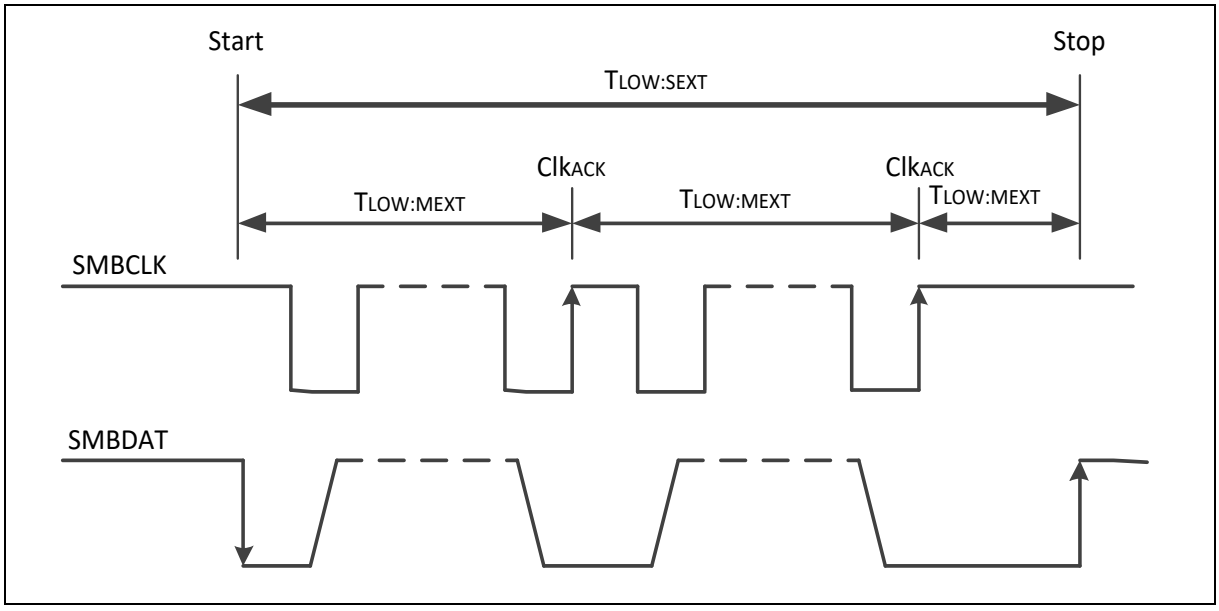


Figure 6.25-22 Bus Clock Low Time Out Timing

Bus Idle Detection

A master can assume that the bus is free if it detects that the clock and data signals have been high for T_{IDLE} greater than $T_{HIGH,MAX}$.

This timing parameter covers the condition where a master has been dynamically added to the bus and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the master must wait long enough to ensure that a transfer is not currently in progress. The peripheral supports a hardware bus idle detection.

6.25.5.3 PDMA Transfer Function

The I^2C controller supports PDMA transfer function. When TXPDMAEN (I2C_CTL1 [0]) is set to 1, the I^2C controller will issue request to PDMA controller to start the DMA transmission process automatically.

When RXPDMAEN (I2C_CTL1 [1]) is set to 1, the I^2C controller will start the receive PDMA process. The I^2C controller will issue the request to PDMA controller automatically when there is data written into the received BUFFER.

When I^2C enters PDMA mode, the mostly status interrupt will be masked. Let the interrupt not occur besides the bus error or NACK or STOP interrupt (0x20, 0x30, 0x38, 0x48, 0x58, 0x00, 0xA0, 0xC0, 0x88 and 0x98).

Set the PDMASTR (I2C_CTL1 [8]) only the I^2C controller in master TX mode. If PDMASTR is cleared to 0, I^2C will send STOP automatically after PDMA transfer done and buffer empty. If PDMASTR is set to 1, SI will be set to 1 and I^2C bus will be stretched by hardware after PDMA transfer done and buffer empty.

6.25.5.4 Programmable setup and hold times

To guarantee a correct data setup and hold time, the timing must be configured. By programming HTCTL (I2C_TMCTL[24:16]) to configure hold time and STCTL (I2C_TMCTL[8:0]) to configure setup time.

The delay timing refer peripheral clock (PCLK). When device stretch master clock, the setup and hold time configuration value will not affected by stretched.

User should focus the limitation of setup and hold time configuration, the timing setting must follow I^2C protocol. Once setup time configuration greater than design limitation, that means if setup time setting make SCL output less than three PCLKs, the I^2C controller can't work normally due to SCL must sample three times. And once hold time configuration greater than I^2C clock limitation, I^2C will occur bus error. It is suggested that user calculate suitable timing with baud rate and protocol before setting timing. Table 6.25-2 shows the relationship between I^2C baud rate and PCLK, the number of table represent one clock duty contain how many PCLKs. Setup and hold time configuration even can program some extreme values in the design, but user should follow I^2C protocol standard.

I^2C Baud Rate PCLK	**100k**	**200k**	**400k**	**800k**	**1200k**
12 MHz	120	60	30	15	10
24 MHz	240	120	60	30	20
48 MHz	480	240	120	60	40
72 MHz	720	360	180	90	60

Table 6.25-2 Relationship between I^2C Baud Rate and PCLK

For setup time wrong adjustment example, assuming one SCL cycle contains 5 PCLKs and set STCTL (I2C_TMCTL[8:0]) to 3 that stretch three PCLKs for setup time setting. The setup time maximum setting value: ST_{limit} = (I2C_CLKDIV[7:0]+1) X 2 - 6.

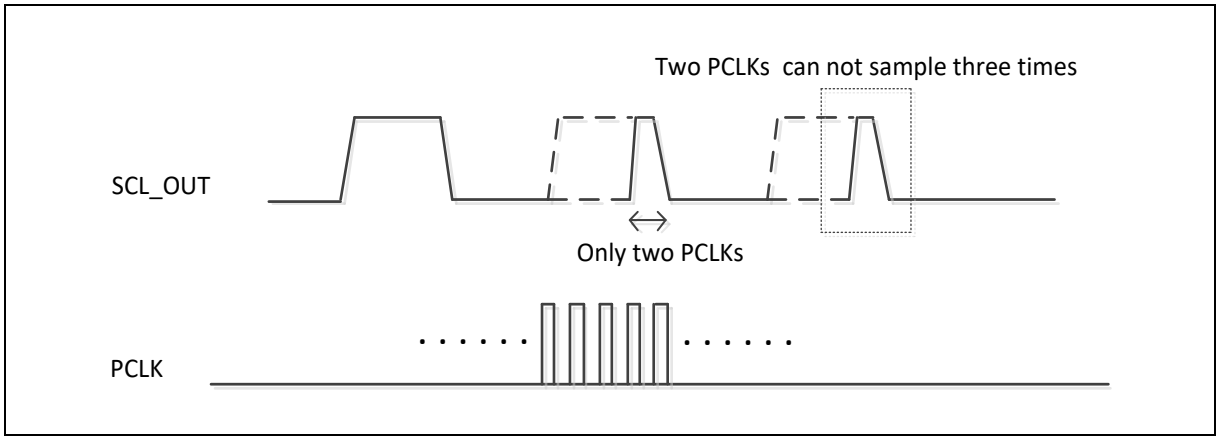


Figure 6.25-23 Setup Time Wrong Adjustment

For hold time wrong adjustment example, use I²C Baud Rate = 1200k and PCLK = 72 MHz, the SCL high/low duty = 60 PCLK. When HTCTL (I2C_TMCTL[24:16]) is set to 61 and STCTL (I2C_TMCTL[8:0]) is set to 0, then SDA output delay will over SCL high duty and cause bus error. The hold time maximum setting value: HT_{limit} = (I2C_CLKDIV[7:0]+1) X 2 - 9.

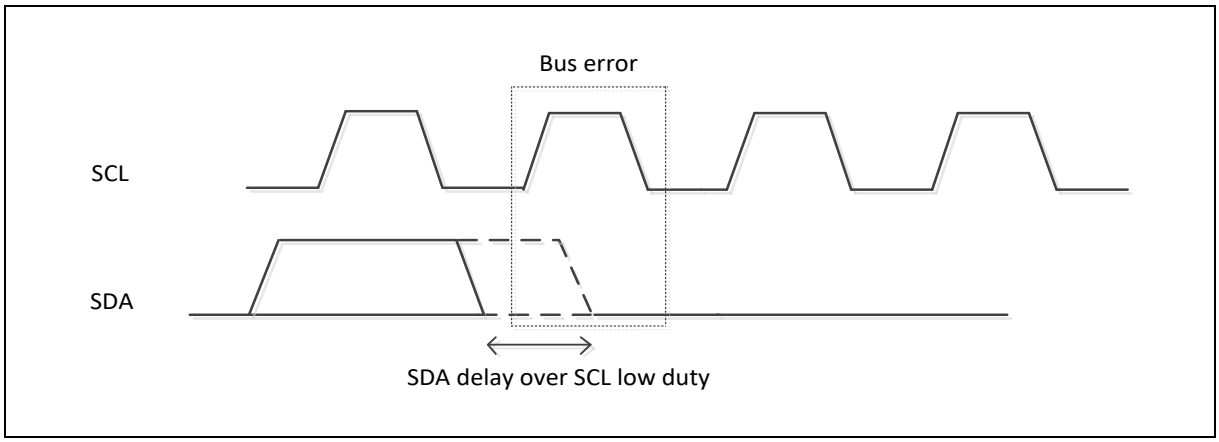


Figure 6.25-24 Hold Time Wrong Adjustment

6.25.5.5 I²C Protocol Registers

To control I²C port through the following fifteen special function registers: I2C_CTL0 (control register), I2C_STATUS0 (status register), I2C_DAT (data register), I2C_ADDRn (address registers, n=0~3), I2C_ADDRMSKn (address mask registers, n=0~3), I2C_CLKDIV (clock rate register), I2C_TOCTL (Time-out control register), I2C_WKCTL(wake up control register) and I2C_WKSTS(wake up status register).

Address Registers (I2C_ADDR)

The I²C port is equipped with four slave address registers, I2C_ADDRn (n=0~3). The contents of the register are irrelevant when I²C is in Master mode. In Slave mode, the bit field ADDR(I2C_ADDRn[7:1]) must be loaded with the chip's own slave address. The I²C hardware will react if the contents of I2C_ADDRn are matched with the received slave address.

The I²C ports support the "General Call" function. If the GC bit (I2C_ADDRn [0]) is set the I²C port hardware will respond to General Call address (00H). Clear GC bit to disable general call function.

When the GC bit is set and the I²C is in Slave mode, it can receive the general call address by 00H

after Master send general call address to I²C bus, then it will follow status of GC mode.

Slave Address Mask Registers (I2C_ADDRMSK)

The I²C bus controller supports multiple address recognition with four address mask registers I2C_ADDRMSKn (n=0~3). When the bit in the address mask register is set to 1, it means the received corresponding address bit is "Don't care". If the bit is set to 0, it means the received corresponding register bit should be exactly the same as address register.

Data Register (I2C_DAT)

This register contains a byte of serial data to be transmitted or a byte which just has been received. The CPU can be read from or written to the 8-bit (I2C_DAT [7:0]) directly while it is not in the process of shifting a byte. When I²C is in a defined state and the serial interrupt flag (SI) is set, data in I2C_DAT [7:0] remains stable. While data is being shifted out, data on the bus is simultaneously being shifted in; I2C_DAT [7:0] always contains the last data byte presented on the bus.

The acknowledge bit is controlled by the I²C hardware and cannot be accessed by the CPU. Serial data is shifted into I2C_DAT [7:0] on the rising edges of serial clock pulses on the SCL line. When a byte has been shifted into I2C_DAT [7:0], the serial data is available in I2C_DAT [7:0], and the acknowledge bit (ACK or NACK) is returned by the control logic during the ninth clock pulse. In order to monitor bus status while sending data, the bus data will be shifted to I2C_DAT[7:0] when sending I2C_DAT[7:0] to bus. In the case of sending data, serial data bits are shifted out from I2C_DAT [7:0] on the falling edge of SCL clocks, and is shifted to I2C_DAT [7:0] on the rising edge of SCL clocks. Figure 6.25-25 shows I²C Data Shifting Direction.

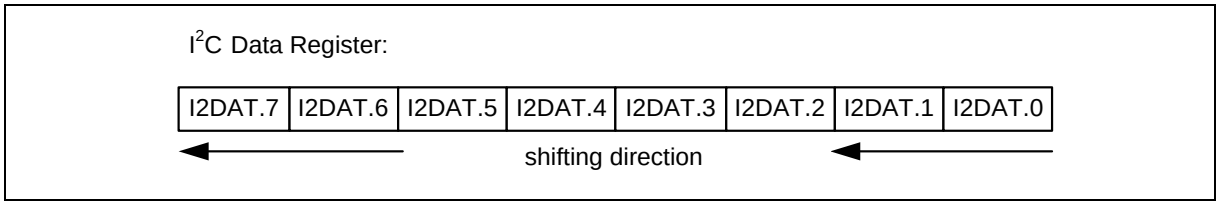


Figure 6.25-25 I²C Data Shifting Direction

Control Register (I2C_CTL0)

The CPU can be read from and written to I2C_CTL0 [7:0] directly. When the I²C port is enabled by setting I2CEN (I2C_CTL0 [6]) to high, the internal states will be controlled by I2C_CTL0 and I²C logic hardware.

There are two bits are affected by hardware: the SI bit is set when the I²C hardware requests a serial interrupt, and the STO bit is cleared when a STOP condition is present on the bus. The STO bit is also cleared when I2CEN = 0.

Once a new status code is generated and stored in I2C_STATUS0, the I²C Interrupt Flag bit SI (I2C_CTL0 [3]) will be set automatically. If the Enable Interrupt bit INTEN (I2C_CTL0 [7]) is set at this time, the I²C interrupt will be generated. The bit field I2C_STATUS0[7:0] stores the internal state code, the content keeps stable until SI is cleared by software.

Status Register (I2C_STATUS0)

I2C_STATUS0 [7:0] is an 8-bit read-only register. The bit field I2C_STATUS0 [7:0] contains the status code and there are 26 possible status codes. All states are listed in Table 6.25-3. When I2C_STATUS0 [7:0] is F8H, no serial interrupt is requested. All other I2C_STATUS0 [7:0] values correspond to the defined I²C states. When each of these states is entered, a status interrupt is requested (SI = 1). A valid status code is present in I2C_STATUS0[7:0] one cycle PCLK after SI set by hardware and is still present one cycle PCLK after SI reset by software.

In addition, the state 00H stands for a Bus Error, which occurs when a START or STOP condition is

present at an incorrect position in the I²C format frame. A Bus Error may occur during the serial transfer of an address byte, a data byte or an acknowledge bit. To recover I²C from bus error, STO should be set and SI should be cleared to enter Not Addressed Slave mode. Then STO is cleared to release bus and to wait for a new communication. The I²C bus cannot recognize stop condition during this action when a bus error occurs.

Master Mode		Slave Mode	
STATUS	Description	STATUS	Description
0x08[1]	Start	0xA0	Slave Transmit Repeat Start or Stop
0x10[1]	Master Repeat Start	0xA8[1]	Slave Transmit Address ACK
0x18[1]	Master Transmit Address ACK	0xB8[1]	Slave Transmit Data ACK
0x20	Master Transmit Address NACK	0xC0	Slave Transmit Data NACK
0x28[1]	Master Transmit Data ACK	0xC8[1]	Slave Transmit Last Data ACK
0x30	Master Transmit Data NACK	0x60[1]	Slave Receive Address ACK
0x38	Master Arbitration Lost	0x68[1]	Slave Receive Arbitration Lost
0x40[1]	Master Receive Address ACK	0x80[1]	Slave Receive Data ACK
0x48	Master Receive Address NACK	0x88	Slave Receive Data NACK
0x50[1]	Master Receive Data ACK	0x70[1]	GC mode Address ACK
0x58	Master Receive Data NACK	0x78[1]	GC mode Arbitration Lost
0x00	Bus error	0x90[1]	GC mode Data ACK
		0x98	GC mode Data NACK
		0xB0[1]	Address Transmit Arbitration Lost
0xF0	If the BMDEN =1 and the ACKMEN bit is enabled, the information of I2C_STATUS0 will be fixed as 0xF0 in slave receive condition.		
0xF8	Bus Released **Note:** Status "0xF8" exists in both master/slave modes, and it won't raise interrupt. Note [1]: No interrupt in PDMA mode		

Table 6.25-3 I²C Status Code Description

Clock Baud Rate Bits (I2C_CLKDIV)

The data baud rate of I²C is determined by DIVIDER(I2C_CLKDIV [7:0]) register when I²C is in Master mode, and it is not necessary in a Slave mode. In the Slave mode, I²C will automatically synchronize it with any clock frequency from master I²C device. In the slave mode, system clock frequency should be greater than I²C bus maximum clock 20 times.

The data baud rate of I²C setting is Data Baud Rate of I²C = (system clock) / (4x (I2C_CLKDIV [7:0] +1)). If system clock = 16 MHz, the I2C_CLKDIV [7:0] = 40 (28H), the data baud rate of I²C = 16 MHz/ (4x (40 +1)) = 97.5 Kbits/sec.

Time-out Control Register (I2C_TOCTL)

There is a 14-bit time-out counter which can be used to deal with the I²C bus hang-up. If the time-out counter is enabled, the counter starts up counting until it overflows (TOIF=1) and generates I²C interrupt to CPU or stops counting by clearing TOCEN to 0. When time-out counter is enabled, writing 1 to the SI flag will reset counter and re-start up counting after SI is cleared. If I²C bus hangs up, it

causes the I2C_STATUS0 and flag SI are not updated for a period, the 14-bit time-out counter may overflow and acknowledge CPU the I²C interrupt. Refer to Figure 6.25-26 for the 14-bit time-out counter. User may write 1 to clear TOIF to 0.

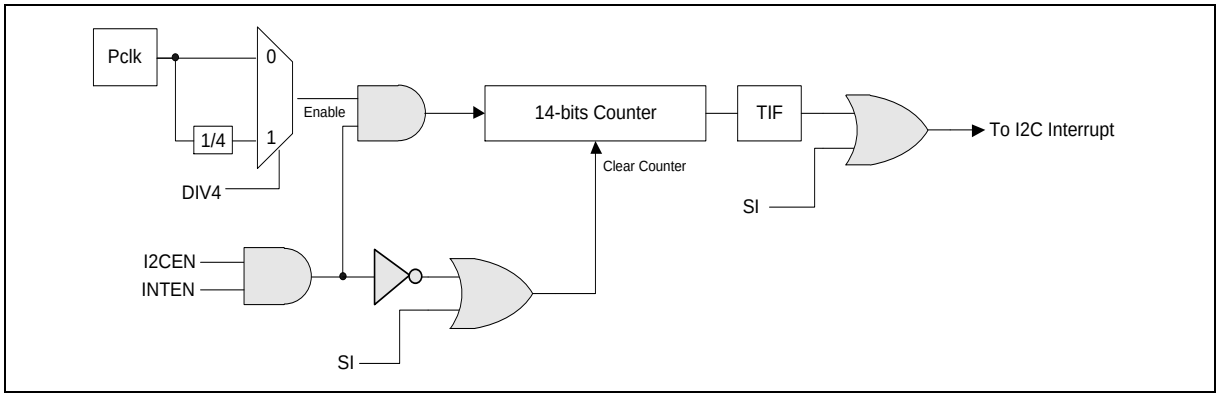


Figure 6.25-26 I²C Time-out Count Block Diagram

Wake-up Control Register (I2C_WKCTL)

When chip enters Power-down mode and sets WKEN (I2C_WKCTL [0]) to 1, other I²C master can wake up the chip by addressing the I²C device, user must configure the related setting before entering sleep mode. The ACK bit cycle of address match frame is done in power-down. The controller will stretch the SCL to low when the address is matched the device's address and the ACK cycle done, then the I²C controller will go ahead. If NHDBUSEN (I2C_WKCTL [7]) is set, the controller will don't stretch the SCL to low. Note that when the controller doesn't stretch the SCL to low, transmit or receive data will perform immediately. If data transmitted or received when SI event is not clear, user must reset the I²C controller and execute the original operation again.

Wake-up Status Register (I2C_WKSTS)

When system is woken up by other I²C master device, WKIF is set to indicate this event. User needs write "1" to clear this bit.

When the chip is woken-up by address match with one of the device address register (I2C_ADDRn), the user shall check the WKAKDONE (I2C_WKSTS [1]) bit is set to 1 to confirm the address byte has done. The WKAKDONE bit indicates that the ACK bit cycle of address byte is done in power-down. The controller will stretch the SCL to low when the address is matched the device's slave address and the ACK cycle done. The SCL is stretched until WKAKDONE is clear by user. If the frequency of SCL is low speed and the system has wakeup from address match frame, the user shall check WKAKDONE to confirm this frame has transaction done and then to do the wakeup procedure. Note that user can't release WKIF through clearing the WKAKDONE bit to 0.

The WRSTSWK (I2C_WKSTS [2]) bit records the Read/Write command before the I²C controller sends address. The user can read this bit's status to prepare the next transmitted data (WRSTSWK = 0) or to wait the incoming data (WRSTSWK = 1) can be stored in time after the system is woken up by the address match frame. Note that the WRSTSWK (I2C_WKSTS [2]) bit is cleared when writing one to the WKAKDONE (I2C_WKSTS [1]) bit.

When system is woken up by other I²C master device, WKIF is set to indicate this event. User needs to write "1" to clear this bit.

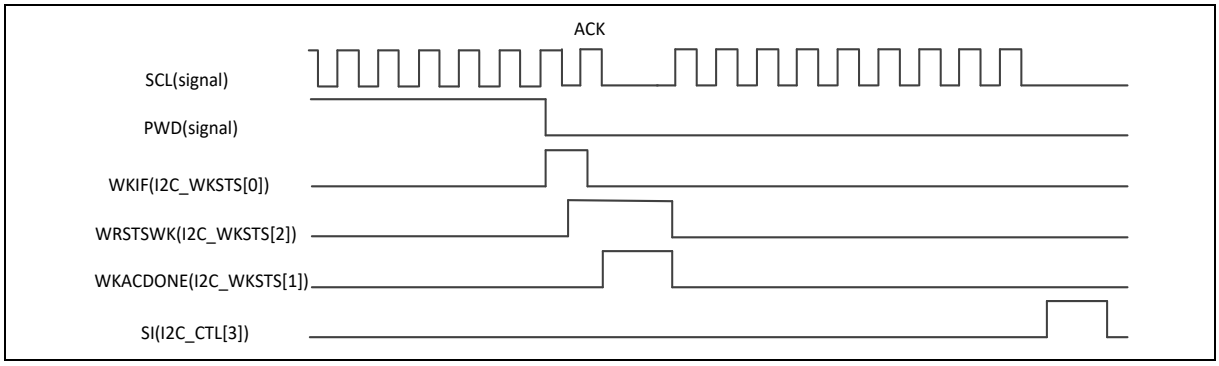


Figure 6.25-27 I²C Wake-Up Related Signals Waveform

I²C Control Register 1 (I2C_CTL1)

If enable 10-bit addressing mode ADDR10EN (I2C_CTL1 [9]) is set, the I²C will run in 10-bit mode.

For PDMA function, set TXPDMAEN (I2C_CTL1 [0]) and RXPDMAEN (I2C_CTL1 [1]) can be set to operate. And set PDMARST (I2C_CTL1 [2]) to reset the PDMA control logic.

I²C Status Register 1 (I2C_STATUS1)

The I²C controller supports four slave address flag registers, ADMAT0, ADMAT1, ADMAT2 and ADMAT3 (I2C_STATUS1[3:0]). Every control register represent which address is used and set 1 to inform software.

I²C Timing Configure Control Register (I2C_TMCTL)

In order to configure setup/hold time, the HTCTL (I2C_TMCTL[24:16]) and STCTL (I2C_TMCTL[8:0]) are set based on actual demand.

Bus Management Control Register (I2C_BUSCTL)

The SM bus management control events are defined in this register. It includes the Acknowledge Control by Manual (ACKMEN (I2C_BUSCTL[0])), Packet Error Checking Enable (PECEN (I2C_BUSCTL[1])), device (BMDEN(I2C_BUSCTL[2])) or host (BMHEN (I2C_BUSCTL[3])) enable in this peripheral device. Both the alert and the suspend function can be set in ALERTEN (I2C_BUSCTL[4]), SCTLOSTS (I2C_BUSCTL[5])) and SCTLOEN (I2C_BUSCTL[6]).

The system bus management enable control by BUSEN(I2CBUSCTL[7]) bit. The BUSTOUT(I2CBUSCTL[9]) is used to calculate the time-out of clock low in bus active and the idle period in bus Idle.

The calculated PEC (when the PECEN is set) value is transmitted or received can be controlled by PECTXEN bit (I2C_BUSCTL[8]).

There is a special bit of ACKM9SI (I2C_BUSCTL[11]). When the ACKMEN is set, there is SI interrupt in the 8th clock input and the user can read the data and status register. If the 8th clock bus is released when the SI interrupt is cleared, there is another SI interrupt event in the 9th clock cycle when this bit is set to 1 to know the bus status in this transaction frame done.

Set the PECDIEN (I2C_BUSCTL[13]), BCDIEN (I2C_BUSCTL[12]) or PECCLR (I2C_BUSCTL[10]) for PEC control flow.

I²C Bus Management Timer Control Register (I2C_BUSTCTL)

Set TORSTEN (I2C_BUSTCTL[4]), CLKTOIEN (I2C_BUSTCTL[3]), BUSTOIEN (I2C_BUSTCTL[2]), CLKTOEN (I2C_BUSTCTL[1]) and BUSTOEN (I2C_BUSTCTL[0]) for bus time-out or clock low time-out control flow.

I²C Bus Management Status Register (I2C_BUSSTS)

Monitor the PECDONE (I2C_BUSSTS[7]), BCDONE (I2C_BUSSTS[1]) or PECERR (I2C_BUSSTS[2]) for PEC control flow.

Monitor the SCTLDIN (I2C_BUSSTS[4]) for SUSCON input status.

I²C Byte Number Register (I2C_PKTSIZE)

When the PECEN bit (I2C_BUSCTL[1]) is set. The I²C controller will calculate the PEC value of the data on the bus. The PLDSIZE (I2C_PKTSIZE[8:0]) is used to define the data number in the bus. When the counter reach the value of PLDSIZE, the final PEC value will be transmitted or received automatically when the PECTXEN bit (I2C_BUSCTL[8]) is set.

I²C PEC VALUR Register (I2C_PKTCRC)

The register indicates the calculated PECCRC (I2C_PKTCRC[7:0]) value of data on the I²C bus. The detail of information is defined in the PEC section of SM Bus.

I²C Bus Management Timer and I²C CLock Low Timer Register (I2C_BUSTOUT/ I2C_CLKTOUT)

Both of the definitions of these registers are described in the time-out section of SM Bus.

6.25.5.6 Example for Random Read on EEPROM

The following steps are used to configure the I2C0 related registers when using I²C to read data from EEPROM.

1. Set I2C0 the multi-function pin as SCL and SDA pins. The muti-function configuration reference Basic Configuration.
2. Enable I2C0 APB clock. The clock configuration reference Basic Configuration.
3. Set I2C0RST=1 to reset I2C0 controller then set I2C0 controller to normal operation. The reset controller configuration reference Basic Configuration.
4. Set I2CEN=1 to enable I2C0 controller in the “I2C_CTL0” register.
5. Give I2C0 clock a divided register value for I²C clock rate in the “I2C_CLKDIV”.
6. Enable system I2C0 IRQ in system “NVIC” control register.
7. Set INTEN=1 to enable I2C0 Interrupt in the “I2C_CTL0” register.
8. Set I2C0 address registers “I2C_ADDR0 ~ I2C_ADDR3”.

Random read operation is one of the methods of access EEPROM. The method allows the master to access any address of EEPROM space. Figure 6.25-28 shows the EEPROM random read operation.

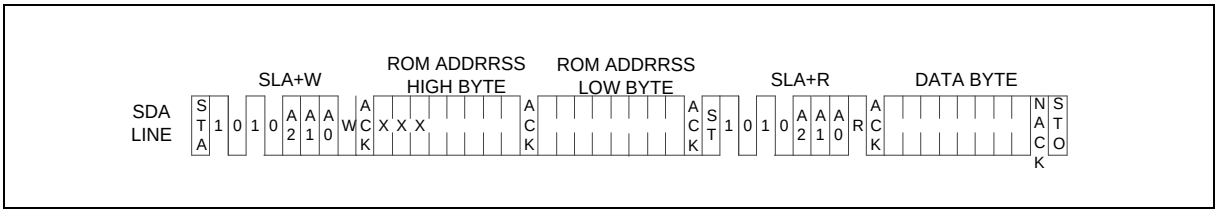


Figure 6.25-28 EEPROM Random Read

Figure 6.25-29 shows how to use the I²C controller to implement the protocol of EEPROM random read.

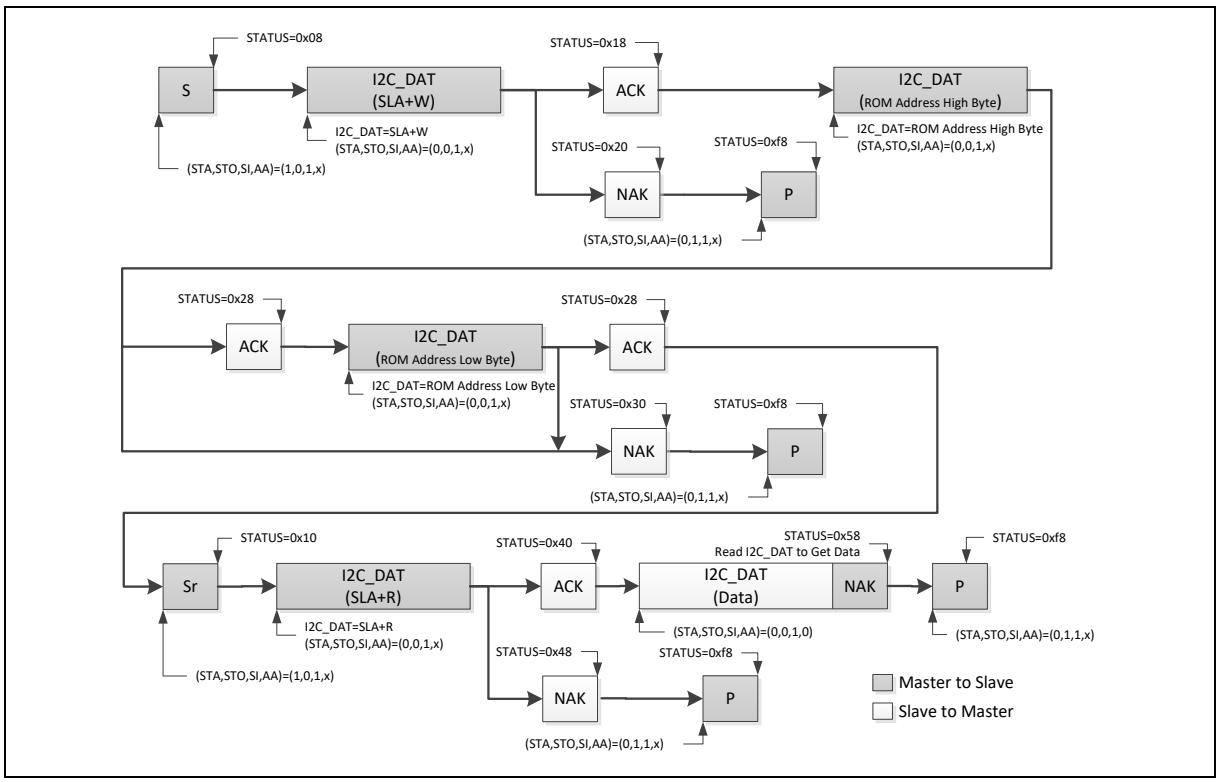


Figure 6.25-29 Protocol of EEPROM Random Read

The I^2C controller, which is a master, sends START to bus. Then, it sends a SLA+W (Slave address + Write bit) to EERPOM followed by two bytes data address to set the EEPROM address to read. Finally, a Repeat START followed by SLA+R is sent to read the data from EEPROM.

6.25.6 Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
I²C Base Address: **I2Cn_BA = 0x4008_0000 + (0x1000 *n)** **n= 0,1,2** **I²C non-secure base address is I2Cn_BA + 0x1000_0000.**				
I2C_CTL0	I2Cn_BA+0x00	R/W	I²C Control Register 0	0x0000_0000
I2C_ADDR0	I2Cn_BA+0x04	R/W	I²C Slave Address Register0	0x0000_0000
I2C_DAT	I2Cn_BA+0x08	R/W	I²C Data Register	0x0000_0000
I2C_STATUS0	I2Cn_BA+0x0C	R	I²C Status Register 0	0x0000_00F8
I2C_CLKDIV	I2Cn_BA+0x10	R/W	I²C Clock Divided Register	0x0000_0000
I2C_TOCTL	I2Cn_BA+0x14	R/W	I²C Time-out Control Register	0x0000_0000
I2C_ADDR1	I2Cn_BA+0x18	R/W	I²C Slave Address Register1	0x0000_0000
I2C_ADDR2	I2Cn_BA+0x1C	R/W	I²C Slave Address Register2	0x0000_0000
I2C_ADDR3	I2Cn_BA+0x20	R/W	I²C Slave Address Register3	0x0000_0000
I2C_ADDRMSK0	I2Cn_BA+0x24	R/W	I²C Slave Address Mask Register0	0x0000_0000
I2C_ADDRMSK1	I2Cn_BA+0x28	R/W	I²C Slave Address Mask Register1	0x0000_0000
I2C_ADDRMSK2	I2Cn_BA+0x2C	R/W	I²C Slave Address Mask Register2	0x0000_0000
I2C_ADDRMSK3	I2Cn_BA+0x30	R/W	I²C Slave Address Mask Register3	0x0000_0000
I2C_WKCTL	I2Cn_BA+0x3C	R/W	I²C Wake-up Control Register	0x0000_0000
I2C_WKSTS	I2Cn_BA+0x40	R/W	I²C Wake-up Status Register	0x0000_0000
I2C_CTL1	I2Cn_BA+0x44	R/W	I²C Control Register 1	0x0000_0000
I2C_STATUS1	I2Cn_BA+0x48	R/W	I²C Status Register 1	0x0000_0000
I2C_TMCTL	I2Cn_BA+0x4C	R/W	I²C Timing Configure Control Register	0x0000_0000
I2C_BUSCTL	I2Cn_BA+0x50	R/W	I²C Bus Management Control Register	0x0000_0000
I2C_BUSTCTL	I2Cn_BA+0x54	R/W	I²C Bus Management Timer Control Register	0x0000_0000
I2C_BUSSTS	I2Cn_BA+0x58	R/W	I²C Bus Management Status Register	0x0000_0000
I2C_PKTSIZE	I2Cn_BA+0x5C	R/W	I²C Packet Error Checking Byte Number Register	0x0000_0000
I2C_PKTCRC	I2Cn_BA+0x60	R	I²C Packet Error Checking Byte Value Register	0x0000_0000
I2C_BUSTOUT	I2Cn_BA+0x64	R/W	I²C Bus Management Timer Register	0x0000_0005
I2C_CLKTOUT	I2Cn_BA+0x68	R/W	I²C Bus Management Clock Low Timer Register	0x0000_0005

6.25.7 Register Description

I²C Control Register (I2C_CTL0)

Register	Offset	R/W	Description	Reset Value
I2C_CTL0	I2Cn_BA+0x00	R/W	I²C Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
INTEN	I2CEN	STA	STO	SI	AA	Reserved	

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**INTEN**	**Enable Interrupt** 0 = I²C interrupt Disabled. 1 = I²C interrupt Enabled.
[6]	**I2CEN**	**I²C Controller Enable Bit** Set to enable I²C serial function controller. When I2CEN=1 the I²C serial function enable. The multi-function pin function must set to SDA, and SCL of I²C function first. 0 = I²C controller Disabled. 1 = I²C controller Enabled.
[5]	**STA**	**I²C START Control** Setting STA to logic 1 to enter Master mode, the I²C hardware sends a START or Repeat START condition to bus when the bus is free.
[4]	**STO**	**I²C STOP Control** In Master mode, setting STO to transmit a STOP condition to bus then I²C controller will check the bus condition if a STOP condition is detected. This bit will be cleared by hardware automatically.
[3]	**SI**	**I²C Interrupt Flag** When a new I²C state is present in the I2C_STATUS0 register, the SI flag is set by hardware. If bit INTEN (I2C_CTL0 [7]) is set, the I²C interrupt is requested. SI must be cleared by software. Clear SI by writing 1 to this bit. For ACKMEN is set in slave read mode, the SI flag is set in 8th clock period for user to confirm the acknowledge bit and 9th clock period for user to read the data in the data buffer.
[2]	**AA**	**Assert Acknowledge Control** When AA =1 prior to address or data is received, an acknowledged (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line when 1.) A slave is acknowledging the address sent from master, 2.) The receiver devices are acknowledging the data sent by transmitter. When AA=0 prior to address or data received, a Not acknowledged (high level to SDA) will be returned during the acknowledge clock pulse on the SCL line.

[1:0]	**Reserved**	Reserved.

I²C Data Register (I2C_DAT)

Register	Offset	R/W	Description	Reset Value
I2C_DAT	I2Cn_BA+0x08	R/W	I²C Data Register	0x0000_0000

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**DAT**	**I²C Data** Bit [7:0] is located with the 8-bit transferred/received data of I²C serial port.

I²C Status Register (I2C_STATUS0)

Register	Offset	R/W	Description	Reset Value
I2C_STATUS0	I2Cn_BA+0x0C	R	I²C Status Register 0	0x0000_00F8

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**STATUS**	**I²C Status** The three least significant bits are always 0. The five most significant bits contain the status code. There are 28 possible status codes. When the content of I2C_STATUS0 is F8H, no serial interrupt is requested. Others I2C_STATUS0 values correspond to defined I²C states. When each of these states is entered, a status interrupt is requested (SI = 1). A valid status code is present in I2C_STATUS0 one cycle after SI is set by hardware and is still present one cycle after SI has been reset by software. In addition, states 00H stands for a Bus Error. A Bus Error occurs when a START or STOP condition is present at an illegal position in the formation frame. Example of illegal position are during the serial transfer of an address byte, a data byte or an acknowledge bit.

I²C Clock Divided Register (I2C_CLKDIV)

Register	Offset	R/W	Description	Reset Value
I2C_CLKDIV	I2Cn_BA+0x10	R/W	I²C Clock Divided Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:12]	**NFCNT**	**Noise Filter Count** The register bits control the input filter width. 0 : filter width 3*PCLK 1 : filter width 4*PCLK *N* : filter width (3+*N*)*PCKL **Note:** Filter width Min :3*PCLK, Max : 18*PCLK
[11:10]	**Reserved**	Reserved.
[9:0]	**DIVIDER**	**I²C Clock Divided** Indicates the I²C clock rate: Data Baud Rate of I²C = (system clock) / (4x (I2C_CLKDIV+1)). **Note:** The minimum value of I2C_CLKDIV is 4.

I²C Time-out Control Register (I2C_TOCTL)

Register	Offset	R/W	Description	Reset Value
I2C_TOCTL	I2Cn_BA+0x14	R/W	I²C Time-out Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					TOCEN	TOCDIV4	TOIF

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**TOCEN**	**Time-out Counter Enable Bit** When enabled, the 14-bit time-out counter will start counting when SI is cleared. Setting flag SI to '1' will reset counter and re-start up counting after SI is cleared. 0 = Time-out counter Disabled. 1 = Time-out counter Enabled.
[1]	**TOCDIV4**	**Time-out Counter Input Clock Divided by 4** When enabled, the time-out period is extended 4 times. 0 = Time-out period is extend 4 times Disabled. 1 = Time-out period is extend 4 times Enabled.
[0]	**TOIF**	**Time-out Flag** This bit is set by hardware when I²C time-out happened and it can interrupt CPU if I²C interrupt enable bit (INTEN) is set to 1. **Note:** Software can write 1 to clear this bit.

I²C Slave Address Register (ADDRx)

Register	Offset	R/W	Description	Reset Value
I2C_ADDR0	I2Cn_BA+0x04	R/W	I²C Slave Address Register0	0x0000_0000
I2C_ADDR1	I2Cn_BA+0x18	R/W	I²C Slave Address Register1	0x0000_0000
I2C_ADDR2	I2Cn_BA+0x1C	R/W	I²C Slave Address Register2	0x0000_0000
I2C_ADDR3	I2Cn_BA+0x20	R/W	I²C Slave Address Register3	0x0000_0000

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

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10:1]	**ADDR**	**I²C Address** The content of this register is irrelevant when I²C is in Master mode. In the slave mode, the seven most significant bits must be loaded with the chip's own address. The I²C hardware will react if either of the address is matched. **Note:** When software set 10'h000, the address can not be used.
[0]	**GC**	**General Call Function** 0 = General Call Function Disabled. 1 = General Call Function Enabled.

I²C Slave Address Mask Register (ADDRMSKx)

Register	Offset	R/W	Description	Reset Value
I2C_ADDRMSK0	I2Cn_BA+0x24	R/W	I²C Slave Address Mask Register0	0x0000_0000
I2C_ADDRMSK1	I2Cn_BA+0x28	R/W	I²C Slave Address Mask Register1	0x0000_0000
I2C_ADDRMSK2	I2Cn_BA+0x2C	R/W	I²C Slave Address Mask Register2	0x0000_0000
I2C_ADDRMSK3	I2Cn_BA+0x30	R/W	I²C Slave Address Mask Register3	0x0000_0000

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

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10:1]	**ADDRMSK**	**I²C Address Mask** 0 = Mask Disabled (the received corresponding register bit should be exact the same as address register.). 1 = Mask Enabled (the received corresponding address bit is don't care.). I²C bus controllers support multiple address recognition with four address mask register. When the bit in the address mask register is set to one, it means the received corresponding address bit is don't-care. If the bit is set to zero, that means the received corresponding register bit should be exact the same as address register. **Note:** The wake-up function can not use address mask.
[0]	**Reserved**	Reserved.

I²C Wake-up Control Register (I2C_WKCTL)

Register	Offset	R/W	Description	Reset Value
I2C_WKCTL	I2Cn_BA+0x3C	R/W	I²C Wake-up Control Register	0x0000_0000

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**NHDBUSEN**	**I²C No Hold BUS Enable Bit** 0 = I²C hold bus after wake-up. 1= I²C don't hold bus after wake-up. **Note:** The I²C controller could respond when WKIF event is not clear, it may cause error data transmitted or received. If data transmitted or received when WKIF event is not clear, user must reset I²C controller and execute the original operation again.
[6:1]	**Reserved**	Reserved.
[0]	**WKEN**	**I²C Wake-up Enable Bit** 0 = I²C wake-up function Disabled. 1= I²C wake-up function Enabled.

I²C Wake-up Status Register (I2C_WKSTS)

Register	Offset	R/W	Description	Reset Value
I2C_WKSTS	I2Cn_BA+0x40	R/W	I²C Wake-up Status Register	0x0000_0000

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

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**WRSTSWK**	**Read/Write Status Bit in Address Wakeup Frame** 0 = Write command be record on the address match wakeup frame. 1 = Read command be record on the address match wakeup frame. **Note:** This bit will be cleared when software can write 1 to WKAKDONE bit.
[1]	**WKAKDONE**	**Wakeup Address Frame Acknowledge Bit Done** 0 = The ACK bit cycle of address match frame isn't done. 1 = The ACK bit cycle of address match frame is done in power-down. **Note:** This bit can't release WKIF. Software can write 1 to clear this bit.
[0]	**WKIF**	**I²C Wake-up Flag** When chip is woken up from Power-down mode by I²C, this bit is set to 1. Software can write 1 to clear this bit.

I²C Control Register 1 (I2C_CTL1)

Register	Offset	R/W	Description	Reset Value
I2C_CTL1	I2Cn_BA+0x44	R/W	I²C Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						ADDR10EN	PDMASTR
7	6	5	4	3	2	1	0
Reserved					PDMARST	RXPDMAEN	TXPDMAEN

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**ADDR10EN**	**Address 10-bit Function Enable Bit** 0 = Address match 10-bit function Disabled. 1 = Address match 10-bit function Enabled.
[8]	**PDMASTR**	**PDMA Stretch Bit** 0 = I²C send STOP automatically after PDMA transfer done. (only master TX) 1 = I²C SCL bus is stretched by hardware after PDMA transfer done if the SI is not cleared. (only master TX)
[7:3]	**Reserved**	Reserved.
[2]	**PDMARST**	**PDMA Reset** 0 = No effect. 1 = Reset the I²C request to PDMA.
[1]	**RXPDMAEN**	**PDMA Receive Channel Available** 0 = Receive PDMA function Disabled. 1 = Receive PDMA function Enabled.
[0]	**TXPDMAEN**	**PDMA Transmit Channel Available** 0 = Transmit PDMA function Disabled. 1 = Transmit PDMA function Enabled.

I²C Status Register 1 (I2C_STATUS1)

Register	Offset	R/W	Description	Reset Value
I2C_STATUS1	I2Cn_BA+0x48	R/W	I²C Status Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							ONBUSY
7	6	5	4	3	2	1	0
Reserved				ADMAT3	ADMAT2	ADMAT1	ADMAT0

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**ONBUSY**	**On Bus Busy (Read Only)** Indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. It is cleared by hardware when a STOP condition is detected. 0 = The bus is IDLE (both SCLK and SDA High). 1 = The bus is busy.
[7:4]	**Reserved**	Reserved.
[3]	**ADMAT3**	**I²C Address 3 Match Status** When address 3 is matched, hardware will inform which address used. This bit will set to 1, and software can write 1 to clear this bit.
[2]	**ADMAT2**	**I²C Address 2 Match Status** When address 2 is matched, hardware will inform which address used. This bit will set to 1, and software can write 1 to clear this bit.
[1]	**ADMAT1**	**I²C Address 1 Match Status** When address 1 is matched, hardware will inform which address used. This bit will set to 1, and software can write 1 to clear this bit.
[0]	**ADMAT0**	**I²C Address 0 Match Status** When address 0 is matched, hardware will inform which address used. This bit will set to 1, and software can write 1 to clear this bit.

I²C Timing Configure Control Register (I2C_TMCTL)

Register	Offset	R/W	Description	Reset Value
I2C_TMCTL	I2Cn_BA+0x4C	R/W	I²C Timing Configure Control Register	0x0000_0000

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

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24:16]	**HTCTL**	**Hold Time Configure Control** This field is used to generate the delay timing between SCL falling edge and SDA rising edge in transmission mode. The delay hold time is numbers of peripheral clock = HTCTL x PCLK.
[15:9]	**Reserved**	Reserved.
[8:0]	**STCTL**	**Setup Time Configure Control** This field is used to generate a delay timing between SDA falling edge and SCL rising edge in transmission mode. The delay setup time is numbers of peripheral clock = STCTL x PCLK. **Note:** Setup time setting should not make SCL output less than three PCLKs.

I²C Bus Manage Control Register (I2C_BUSCTL)

Register	Offset	R/W	Description	Reset Value
I2C_BUSCTL	I2Cn_BA+0x50	R/W	I²C Bus Management Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		PECDIEN	BCDIEN	ACKM9SI	PECCLR	TIDLE	PECTXEN
7	6	5	4	3	2	1	0
BUSEN	SCTLOEN	SCTLOSTS	ALERTEN	BMHEN	BMDEN	PECEN	ACKMEN

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**PECDIEN**	**Packet Error Checking Byte Transfer Done Interrupt Enable Bit** 0 = PEC transfer done interrupt Disabled. 1 = PEC transfer done interrupt Enabled. **Note:** This bit is used in PECEN =1.
[12]	**BCDIEN**	**Packet Error Checking Byte Count Done Interrupt Enable Bit** 0 = Byte count done interrupt Disabled. 1 = Byte count done interrupt Enabled. **Note:** This bit is used in PECEN =1.
[11]	**ACKM9SI**	**Acknowledge Manual Enable Extra SI Interrupt** 0 = There is no SI interrupt in the 9th clock cycle when the BUSEN =1 and ACKMEN =1. 1 = There is SI interrupt in the 9th clock cycle when the BUSEN =1 and ACKMEN =1.
[10]	**PECCLR**	**PEC Clear at Repeat START** The calculation of PEC starts when PECEN is set to 1 and it is cleared when the STA or STO bit is detected. This PECCLR bit is used to enable the condition of Repeat START can clear the PEC calculation. 0 = PEC calculation is cleared by “Repeat START” function Disabled. 1 = PEC calculation is cleared by “Repeat START” function Enabled.
[9]	**TIDLE**	**Timer Check in Idle State** The BUSTOUT is used to calculate the time-out of clock low in bus active and the idle period in bus Idle. This bit is used to define which condition is enabled. 0 = BUSTOUT is used to calculate the clock low period in bus active. 1 = BUSTOUT is used to calculate the IDLE period in bus Idle. **Note:** The BUSY (I2C_BUSSTS[0]) indicate the current bus state.
[8]	**PECTXEN**	**Packet Error Checking Byte Transmission/Reception**

		0 = No PEC transfer. 1 = PEC transmission is requested. **Note:** 1.This bit has no effect in slave mode when ACKMEN =0.
[7]	**BUSEN**	**BUS Enable Bit** 0 = The system management function Disabled. 1 = The system management function Enabled. **Note:** When the bit is enabled, the internal 14-bit counter is used to calculate the time out event of clock low condition.
[6]	**SCTLOEN**	**Suspend or Control Pin Output Enable Bit** 0 = The SUSCON pin in input. 1 = The output enable is active on the SUSCON pin.
[5]	**SCTLOSTS**	**Suspend/Control Data Output Status** 0 = The output of SUSCON pin is low. 1 = The output of SUSCON pin is high.
[4]	**ALERTEN**	**Bus Management Alert Enable Bit** Device Mode (BMHEN =0). 0 = Release the BM_ALERT pin high and Alert Response Header disabled: 0001100x followed by NACK if both of BMDEN and ACKMEN are enabled. 1 = Drive BM_ALERT pin low and Alert Response Address Header enables: 0001100x followed by ACK if both of BMDEN and ACKMEN are enabled. Host Mode (BMHEN =1). 0 = BM_ALERT pin not supported. 1 = BM_ALERT pin supported.
[3]	**BMHEN**	**Bus Management Host Enable Bit** 0 = Host function Disabled. 1 = Host function Enabled.
[2]	**BMDEN**	**Bus Management Device Default Address Enable Bit** 0 = Device default address Disable. When the address 0'b1100001x coming and the both of BMDEN and ACKMEN are enabled, the device responses NACK 1 = Device default address Enabled. When the address 0'b1100001x coming and the both of BMDEN and ACKMEN are enabled, the device responses ACK.
[1]	**PECEN**	**Packet Error Checking Calculation Enable Bit** 0 = Packet Error Checking Calculation Disabled. 1 = Packet Error Checking Calculation Enabled. **Note:** When I^2C enters Power-down mode, the bit should be enabled after wake-up if needed PEC calculation.
[0]	**ACKMEN**	**Acknowledge Control by Manual** In order to allow ACK control in slave reception including the command and data, slave byte control mode must be enabled by setting the ACKMEN bit. 0 = Slave byte control Disabled. 1 = Slave byte control Enabled. The 9th bit can response the ACK or NACK according the received data by user. When the byte is received, stretching the SCLK signal low between the 8th and 9th SCLK pulse. **Note:** If the BMDEN =1 and this bit is enabled, the information of I2C_STATUS0 will be fixed as 0xF0 in slave receive condition.

I²C Bus Management Timer Control Register (I2C_BUSTCTL)

Register	Offset	R/W	Description	Reset Value
I2C_BUSTCTL	I2Cn_BA+0x54	R/W	I²C Bus Management Timer Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			TORSTEN	CLKTOIEN	BUSTOIEN	CLKTOEN	BUSTOEN

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**TORSTEN**	**Time Out Reset Enable Bit** 0 = I²C state machine reset Disabled. 1 = I²C state machine reset Enabled. (The clock and data bus will be released to high)
[3]	**CLKTOIEN**	**Extended Clock Time Out Interrupt Enable Bit** 0 = Clock time out interrupt Disabled. 1 = Clock time out interrupt Enabled.
[2]	**BUSTOIEN**	**Time-out Interrupt Enable Bit** BUSY =1. 0 = SCLK low time-out interrupt Disabled. 1 = SCLK low time-out interrupt Enabled. BUSY =0. 0 = Bus IDLE time-out interrupt Disabled. 1 = Bus IDLE time-out interrupt Enabled.
[1]	**CLKTOEN**	**Cumulative Clock Low Time Out Enable Bit** 0 = Cumulative clock low time-out detection Disabled. 1 = Cumulative clock low time-out detection Enabled. For Master, it calculates the period from START to ACK For Slave, it calculates the period from START to STOP
[0]	**BUSTOEN**	**Bus Time Out Enable Bit** 0 = Bus clock low time-out detection Disabled. 1 = Bus clock low time-out detection Enabled (bus clock is low for more than TTime-out (in BIDLE=0) or high more than TTime-out (in BIDLE =1).

I²C Bus Management Status Register (I2C_BUSSTS)

Register	Offset	R/W	Description	Reset Value
I2C_BUSSTS	I2Cn_BA+0x58	R/W	I²C Bus Management Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
PECDONE	CLKTO	BUSTO	SCTLDIN	ALERT	PECERR	BCDONE	BUSY

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**PECDONE**	**PEC Byte Transmission/Receive Done** 0 = PEC transmission/ receive is not finished when the PECEN is set. 1 = PEC transmission/ receive is finished when the PECEN is set. **Note:** Software can write 1 to clear this bit.
[6]	**CLKTO**	**Clock Low Cumulate Time-out Status** 0 = Cumulative clock low is no any time-out. 1 = Cumulative clock low time-out occurred. **Note:** Software can write 1 to clear this bit.
[5]	**BUSTO**	**Bus Time-out Status** 0 = There is no any time-out or external clock time-out. 1 = A time-out or external clock time-out occurred. In bus busy, the bit indicates the total clock low time-out event occurred; otherwise, it indicates the bus idle time-out event occurred. **Note:** Software can write 1 to clear this bit.
[4]	**SCTLDIN**	**Bus Suspend or Control Signal Input Status** 0 = The input status of SUSCON pin is 0. 1 = The input status of SUSCON pin is 1.
[3]	**ALERT**	**SMBus Alert Status** Device Mode (BMHEN =0). 0 = SMBALERT pin state is low. 1 = SMBALERT pin state is high. Host Mode (BMHEN =1). 0 = No SMBALERT event. 1 = There is SMBALERT event (falling edge) is detected in SMALERT pin when the BMHEN = 1 (SMBus host configuration) and the ALERTEN = 1.

		Note: 1. The SMBALERT pin is an open-drain pin, the pull-high resistor is must in the system. 2. Software can write 1 to clear this bit.
[2]	**PECERR**	**PEC Error in Reception** 0 = PEC value equal the received PEC data packet. 1 = PEC value doesn't match the receive PEC data packet. **Note:** Software can write 1 to clear this bit.
[1]	**BCDONE**	**Byte Count Transmission/Receive Done** 0 = Byte count transmission/ receive is not finished when the PECEN is set. 1 = Byte count transmission/ receive is finished when the PECEN is set. **Note:** Software can write 1 to clear this bit.
[0]	**BUSY**	**Bus Busy** Indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. It is cleared by hardware when a STOP condition is detected 0 = Bus is IDLE (both SCLK and SDA High). 1 = Bus is busy.

I²C Byte Number Register (I2C_PKTSIZE)

Register	Offset	R/W	Description	Reset Value
I2C_PKTSIZE	I2Cn_BA+0x5C	R/W	I²C Packet Error Checking Byte Number Register	0x0000_0000

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

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8:0]	**PLDSIZE**	**Transfer Byte Number** The transmission or receive byte number in one transaction when the PECEN is set. The maximum transaction or receive byte is 256 Bytes. **Note:** The byte number counting includes address, command code, and data frame.

I²C PEC Value Register (I2C_PKTCRC)

Register	Offset	R/W	Description	Reset Value
I2C_PKTCRC	I2Cn_BA+0x60	R	I²C Packet Error Checking Byte Value Register	0x0000_0000

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

Bits	Description	
[31:8]	Reserved	Reserved.
[7:0]	PECCRC	**Packet Error Checking Byte Value** This byte indicates the packet error checking content after transmission or receive byte count by using the C(x) = X8 + X2 + X + 1. It is read only.

I²C Bus Management Timer Register (I2C_BUSTOUT)

Register	Offset	R/W	Description	Reset Value
I2C_BUSTOUT	I2Cn_BA+0x64	R/W	I²C Bus Management Timer Register	0x0000_0005

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**BUSTO**	**Bus Management Time-out Value** Indicates the bus time-out value in bus is IDLE or SCLK low. **Note:** If the user wants to revise the value of BUSTOUT, the TORSTEN (I2C_BUSTCTL[4]) bit shall be set to 1 and clear to 0 first in the BUSEN(I2C_BUSCTL[7]) is set.

I²C Clock Low Timer Register (I2C_CLKTOUT)

Register	Offset	R/W	Description	Reset Value
I2C_CLKTOUT	I2Cn_BA+0x68	R/W	I²C Bus Management Clock Low Timer Register	0x0000_0005

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**CLKTO**	**Bus Clock Low Timer** The field is used to configure the cumulative clock extension time-out. **Note:** If the user wants to revise the value of CLKLTOUT, the TORSTEN bit shall be set to 1 and clear to 0 first in the BUSEN is set.

6.26 USCI – UART Mode

6.26.1 Overview

The asynchronous serial channel UART covers the reception and the transmission of asynchronous data frames. It performs a serial-to-parallel conversion on data received from the peripheral, and a parallel-to-serial conversion on data transmitted from the controller. The receiver and transmitter being independent, frames can start at different points in time for transmission and reception.

The UART controller also provides auto flow control. There are two conditions to wake-up the system.

6.26.2 Features

- Supports one transmit buffer and two receive buffer for data payload
- Supports hardware auto flow control function
- Supports programmable baud-rate generator
- Supports 9-bit Data Transfer (Support 9-bit RS-485)
- Baud rate detection possible by built-in capture event of baud rate generator
- Supports PDMA capability
- Supports Wake-up function (Data and nCTS Wakeup Only)

6.26.3 Block Diagram

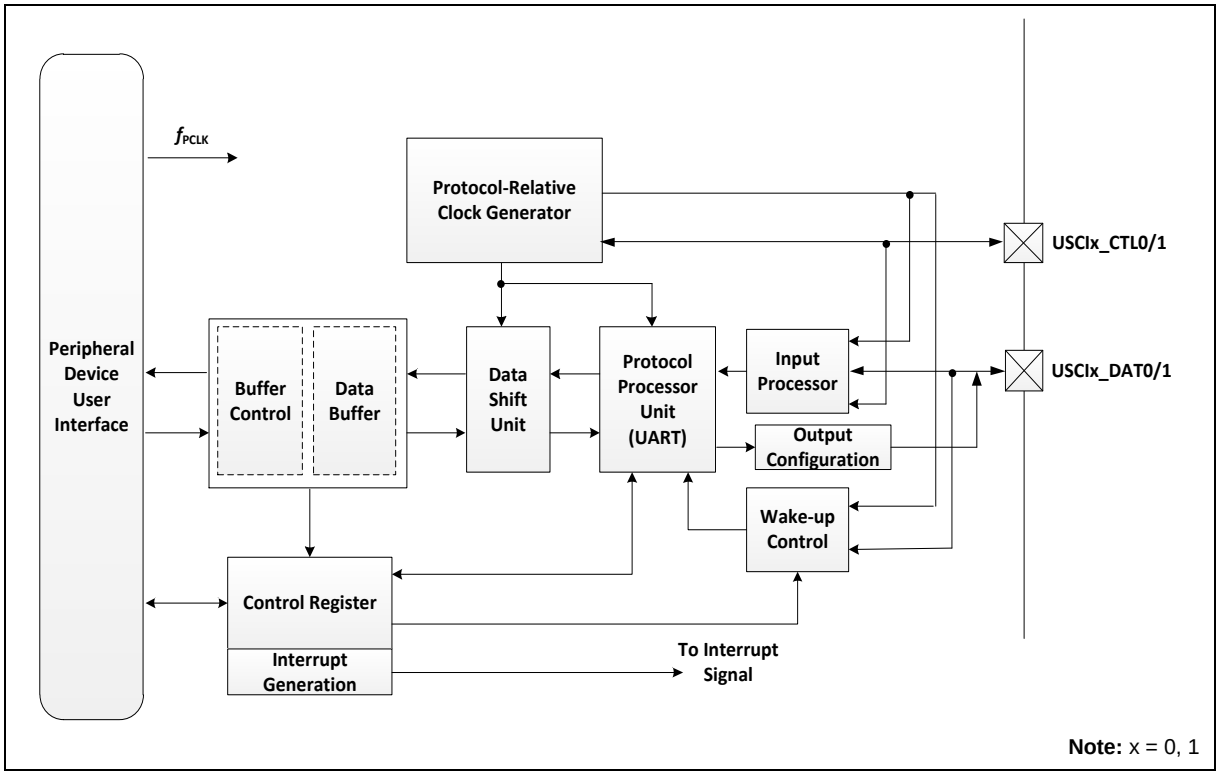


Figure 6.26-1 USCI-UART Mode Block Diagram

6.26.4 Basic Configuration

The basic configurations of USCI0_UART are as follows:

- Clock Source Configuration
  - Enable USCI0 peripheral clock in USCI0CKEN (CLK_APBCLK1[8]).
  - Enable USCI0_UART function in FUNMODE (UUART_CTL[2:0]=0x2).
- Reset Configuration
  - Reset USCI0 controller in USCI0RST (SYS_IPRST2[8]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
USCI0	USCI0_CTL0	PD.4	MFP3
		PD.14	MFP5
		PC.13	MFP6
		PE.6	MFP7
	USCI0_CTL1	PD.3	MFP3
		PB.15	MFP5
		PA.8	MFP6
		PE.5	MFP7
	USCI0_DAT0	PD.1	MFP3
		PB.13	MFP5
		PA.10	MFP6
		PE.3	MFP7
	USCI0_DAT1	PD.2	MFP3
		PB.14	MFP5
		PA.9	MFP6
		PE.4	MFP7

The basic configurations of USCI1_UART are as follows:

- Clock Source Configuration
  - Enable USCI1 peripheral clock in USCI1CKEN (CLK_APBCLK1[9]).
  - Enable USCI1_UART function in FUNMODE (UUART_CTL[2:0]=0x2).
- Reset Configuration
  - Reset USCI1 controller in USCI1RST (SYS_IPRST2[9]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
USCI0	USCI1_CTL0	PB.10	MFP4
		PD.3, PE.9	MFP6

		PB.5	MFP8
	USCI1_CTL1	PB.9	MFP4
		PD.4, PE.8	MFP6
		PB.4	MFP8
	USCI1_DAT0	PB.7	MFP4
		PD.5, PE.10	MFP6
		PB.2	MFP8
	USCI1_DAT1	PB.6	MFP4
		PD.6, PE.11	MFP6
		PB.3	MFP8

6.26.5 Functional Description

6.26.5.1 USCI Common Functional Description

Please refer to section USCI for detailed information.

6.26.5.2 Signal Description

An UART connection is characterized by the use of a single connection line between a transmitter and a receiver. The receiver input signal (RXD) is handled by the input stage USCIx_DAT0 and the transmit output (TXD) signal is handled by the output stage of USCIx_DAT1.

For full-duplex communication, an independent communication line is needed for each transfer direction. Figure 6.26-2 shows an example with a point-to-point full-duplex connection between two communication partners UART module A and UART module B.

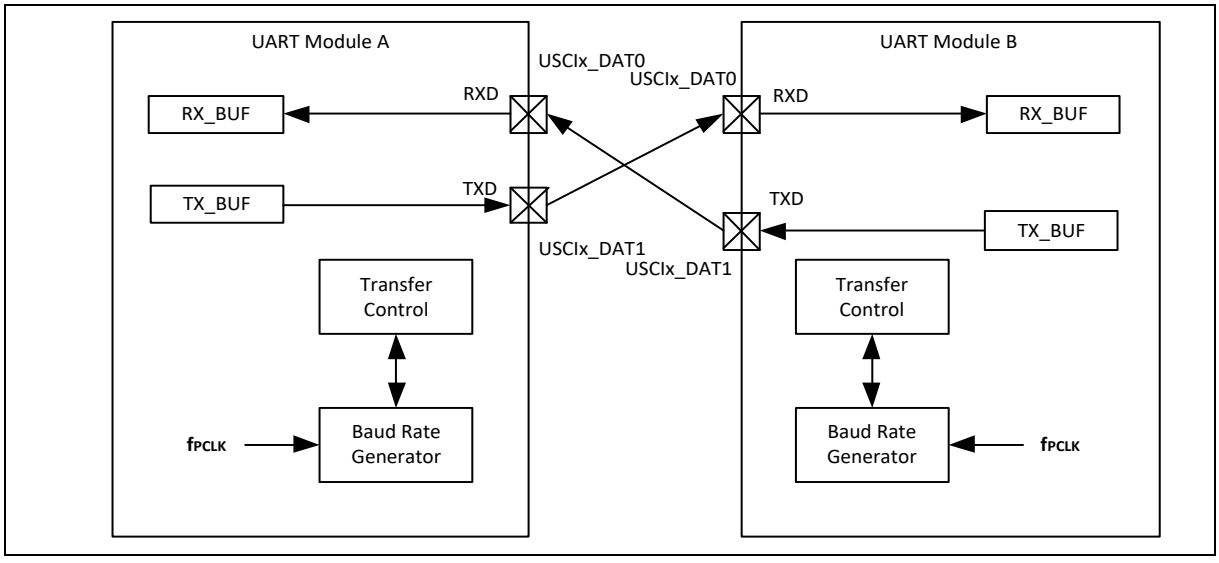


Figure 6.26-2 UART Signal Connection for Full-Duplex Communication

Input Signal

For UART protocol, the number of input signals is shown in Table 6.26-1. Each input signal is handled by an input processor for signal conditioning, such as signal inverse selection control, or a digital input filter. They can be classified according to their meaning for the protocols (see Table 6.26-1).

Selected Protocol		UART
Control Input	USCIx_CTL0	nCTS
	USCIx_CTL1	X
Data Input	USCIx_DAT0	RX
	USCIx_DAT1	X

Table 6.26-1 Input Signals for UART Protocol

Output Signals

For UART protocol, up to each protocol-related output signals are available. The number of actually used outputs depends on the selected protocol. They can be classified according to their meaning for the protocols.

Selected Protocol		UART
Control Output	USCIx_CTL0	X
	USCIx_CTL1	nRTS
Data Output	USCIx_DAT0	X
	USCIx_DAT1	TX

Table 6.26-2 Output Signals for UART Protocol

6.26.5.3 Frame Format

A standard UART frame is shown in Figure 6.26-3. It consists of:

- An idle time with the signal level 1.
- One start of frame bit (SOF) with the signal level 0.
- 6~13 bit data
- A parity bit (P), programmable for either even or odd parity. It is optionally possible to handle frames without parity bit.
- One or two stop bits with the signal level 1.

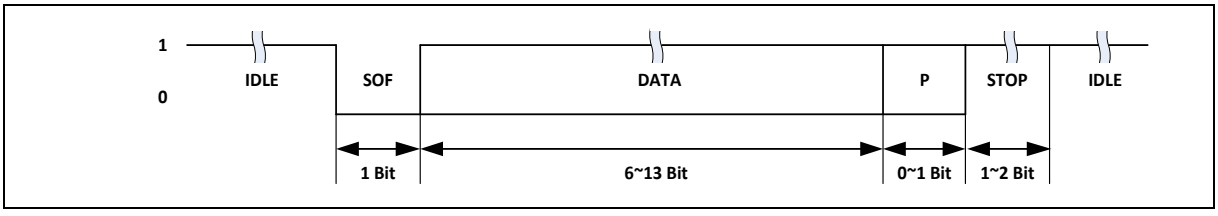


Figure 6.26-3 UART Standard Frame Format

The protocol specific bits (SOF, P, STOP) are automatically handled by the UART protocol state machine and do not appear in the data flow via the receive and transmit buffers.

Start Bit

The receiver input signal USCIx_DAT0 is checked for a falling edge. An SOF bit is detected when a falling edge occurs while the receiver is idle or after the sampling point of the last stop bit. To increase noise immunity, the SOF bit timing starts with the first falling edge that is detected. If the sampled bit value of the SOF is 1, the previous falling edge is considered to be due to noise and the receiver is

considered to be idle again.

Data Field

The length of the data field (number of data bits) can be programmed by the bit field of DWIDTH (UUART_LINECTL[11:8]). It can vary between 6 to 13 data bits.

Note: In UART protocol, the data transmission order is LSB first by setting LSB (UUART_LINECTL[0]) to 1.

Parity Bit

The UART allows parity generation for transmission and parity check for reception on frame base. The type of parity can be selected by bit field PARITYEN (UUART_PROTCTL[1]) and EVENPARITY (UUART_PROTCTL[2]), common for transmission and reception (no parity, even or odd parity). If the parity handling is disabled, the UART frame does not contain any parity bit. For consistency reasons, all communication partners have to be programmed to the same parity mode.

After the last data bit of the data field, the transmitter automatically sends out its calculated parity bit if parity generation has been enabled. The receiver interprets this bit as received parity and compares it to its internally calculated one. The result of the parity check and frame check (STOP bit) are monitored in the protocol status registers (UUART_PROTSTS). The register contains bits to monitor a protocol-related status and protocol-related error indication (FRMERR, PARITYERR).

Stop Bit

Each UART frame is completed by 1 or 2 of stop bits with the signal level 1 (same level as the idle level). The number of stop bits is programmable by bit STOPB (UUART_PROTCTL[0]). A new start bit can be transferred directly after the last stop bit.

Transfer Status Indication

RXBUSY (UUART_PROTSTS[10]) indicates the receiver status.

The receiver status can be monitored by RXBUSY bit. In this case, bit RXBUSY is set during a complete frame reception from the beginning of the start of frame bit to the end of the last stop bit.

6.26.5.4 Operating Mode

To operate the UART protocol, the following issues have to be considered:

Select UART Mode

The UART protocol can be selected by setting FUNMODE (UUART_CTL[2:0]) to 0x2 and the UART protocol can be enabled by setting PROTEN (UUART_PROTCTL [31]) to 1. Note that the FUNMODE must be set 0 before protocol changing and it is recommended to configure all parameters of the UART before UART protocol is enabled.

Pin Connections

The USCIx_DAT0 pin is used for UART receive data input signal (RX) in UART protocol. The property of input data signal can be configured in UUART_DATIN0. It is suggested to set EDGEDET (UUART_DATIN0[4:3]) as 10B for start bit detection.

The USCIx_DAT1 pin is used for UART transmit data output signal (TX) in UART protocol. The property of output data signal can be configured in UUART_LINECTL.

The USCIx_CTL0 pin is used for UART clear to send signal (nCTS) in UART protocol. The property of input control signal can be configured in UUART_CTLIN0.

The USCIx_CTL1 pin is used for UART request to send signal (nRTS) in UART protocol. The property of output control signal can be configured in UUART_LINECTL.

Bit Timing Configuration

The desired baud rate setting has to be selected, comprising the baud rate generator and the bit timing.

Frame Format Configuration

The word length, the stop bit number, and the parity mode has to be set up according to the application requirements by programming UUART_LINECTL and the UUART_PROTCTL register. If required by the application, the data input and output signals can be inverted. The data transmission order is LSB first by setting LSB (UUART_LINECTL[0]) to 1.

6.26.5.5 Bit Timing

In UART mode, each frame bit is divided into data sample time in order to provide granularity in the sub-bit range to adjust the sample point to the application requirements. The number of data sample time per bit is defined by bit fields DSCNT (UUART_BRGEN[14:10]) and the length of a data sample time is given by PDSCNT (UUART_BRGEN[9:8]).

In the example given in Figure 6.26-4, one bit time is composed of 16 data sample time DSCNT(UUART_BRGEN[14:10]) = 15. It is not recommended to program less and equal than 4 data sample time per bit time.

The position of the sampling point for the bit value is fixed in 1/2 samples time. It is possible to sample the bit value to take the average of samples.

The bit timing setup (number of data sample time) is common for the transmitter and the receiver because they use the same hardware circuit.

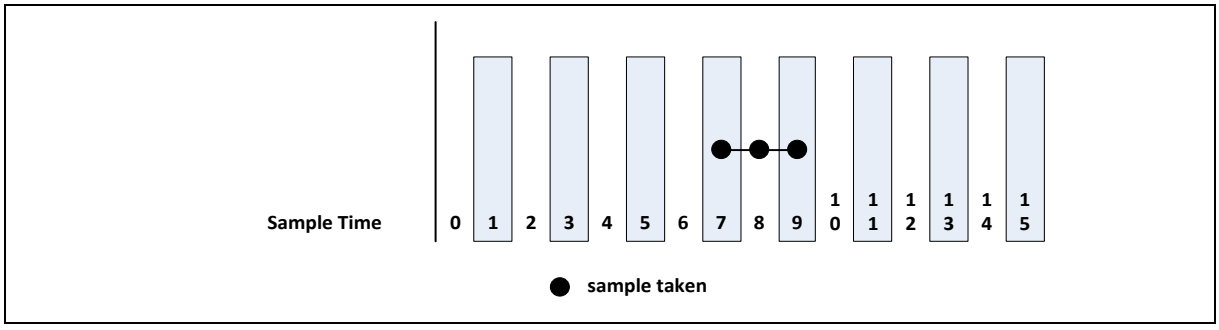


Figure 6.26-4 UART Bit Timing (Data Sample Time)

6.26.5.6 Baud Rate Generation

The baud rate f_{UART} in UART mode depends on the number of data sample time per bit time and their timing. The baud rate setting should only be changed while the transmitter and the receiver are idle. The bits RCLKSEL, SPCLKSEL, PDSCNT, and DSCNT define the baud rate setting:

RCLKSEL (UUART_BRGEN [0])

to define the input frequency f_{REF_CLK}

SPCLKSEL (UUART_BRGEN[3:2])

to define the multiple source of the sample clock f_{SAMP_CLK}

PDSCNT (UUART_BRGEN [9:8])

to define the length of a data sample time (division of f_{REF_CLK} by 1, 2, 3, or 4)

DSCNT (UUART_BRGEN [14:10])

to define the number of data sample time per bit time

The standard setting is given by RCLKSEL = 0 ($f_{REF\_CLK} = f_{PCLK}$), PTCLKSEL = 0 ($f_{PROT\_CLK} = f_{REF\_CLK}$) and SPCLKSEL = 0 ($f_{SAMP\_CLK} = f_{DIV\_CLK}$). Under these conditions, the baud rate is given by:

$$f_{\mathrm{UART}} = f_{REF\_CLK} \times \frac{1}{\mathrm{CLKDIV}+1} \times \frac{1}{\mathrm{PDSCNT}+1} \times \frac{1}{\mathrm{DSCNT}+1}$$

In order to generate slower frequencies, additional divide-by-2 stages can be selected by PTCLKSEL

= 1 (f_{PROT_CLK} = f_{REF_CLK2}), leading to:

$$f_{\mathrm{UART}} = \frac{f_{REF\_CLK}}{2} \times \frac{1}{\mathrm{CLKDIV}+1} \times \frac{1}{\mathrm{PDSCNT}+1} \times \frac{1}{\mathrm{DSCNT}+1}$$

If SPCLKSEL = 2 (f_{SAMP_CLK} = f_{SCLK}), and RCLKSEL = 0 (f_{REF_CLK} = f_{PCLK}), PTCLKSEL = 0 (f_{PROT_CLK} = f_{REF_CLK}). The baud rate is given by:

$$f_{\mathrm{UART}} = f_{REF\_CLK} \times \frac{1}{\mathrm{CLKDIV}+1} \times \frac{1}{2} \times \frac{1}{\mathrm{PDSCNT}+1} \times \frac{1}{\mathrm{DSCNT}+1}$$

There is error tolerance for the UART baud rate after setting the baud rate parameter. Table 6.26-3 lists the baud rate setting examples and the relative error percentage for user to calculate his relative baud rate setting. The clock source is standard setting (SPCLKSEL=0, PTCLKSEL=0 and RCLKSEL=0).

HCLK Source	PCLK Source	Expect Baud Rate	CLKDIV (UUART_BRGEN[25:16])	DSCNT (UUART_BRGEN[14:10])	PDSCNT	Active Baud Rate	Error Percentage
12 MHz	HCLK	115200	0xC	0x7	0x0	115384	0.16%
12 MHz	HCLK	9600	0x7C	0x9	0x0	9600	0%
12 MHz	HCLK	2400	0x1F3	0x9	0x0	2400	0%

Table 6.26-3 Baud Rate Relationship

6.26.5.7 Auto Baud Rate Detection

The UART controller supports auto baud rate detection function. It is used to identify the input baud rate from the receiver signal (USCIx_DAT0) and then revised the baud rate clock divider CLKDIV (UUART_BRGEN[25:16]) after the baud rate function done to meet the detected baud rate information. According the section of Timing Measurement Counter, the timing measurement counter is used for time interval measurement of the input signal (USCIx_DAT0) and the actual timer value is captured into bit field BRDETITV (UUART_PROTCTL [24:16]) in each falling edge of the detected signal.

When the ABREN (UUART_PROTCTL[6]) bit is enabled, the 0x55 data patterns is necessary for auto baud rate detection. The falling edge of input signal starts the baud rate counter and it loads the timing measurement counter value into the BRDETITV (UUART_PROTCTL [24:16]) in the next falling edge. It is suggested to use the f_{DIV_CLK} (TMCNTSRC (UUART_BRGENC[5]) =1) as the counter source.

The CLKDIV (UUART_BRGEN[25:16]) will be revised by BRDETITV (UUART_PROTCTL [25:16]) after the auto baud rate function done (the time of 4th falling edge of input signal). If the user want to receive the next successive frame correctly, it is better to set the value of CLKDIV (UUART_BRGEN[25:16]) and DSCNT (UUART_BRGEN[14:10]) as the same value (the value shall be among the rang of 0xF and 0x5 because the DSCNT is used to define the sample counter of each bit and the PDSCNT (UUART_BRGEN[9:8]) is 0x0.

During the auto baud rate detection, the ABRDETIF (UUART_PROTSTS[9]) and the BRDETITV (UUART_PROTCTL [24:16]) will be updated after each falling edge of input signal and the auto baud rate pattern, 0x55, won't be received into the receiver buffer after the frame done. The bit of ABREN will be cleared by hardware after the 4th falling edge of input signal is detected thus the user can read the status of ABREN to know the auto baud rate function is done or not.

If the CLKDIV and DSCNT are not set as the same value in calculation the auto baud rate function, the user shall calculate the proper average baud rate by the value of BRDETITV and CLKDIV after the auto baud rate function done.

If the baud rate of input signal is very slower and the bit time of timing measurement counter can't calculate the correct period of the input bit time, there is a ABERRSTS bit (UUART_PROTSTS[11]) to indicate the error information of the auto baud rate detection. At this time, the user shall revise the

value of CLKDIV and require the Host device to send the 0x55 pattern again.

According the limitation of timing measurement counter, the maximum auto baud rate detection is 0x1FE for BRDETITV. The UART Auto Baud Rate Control is shown in Figure 6.26-5.

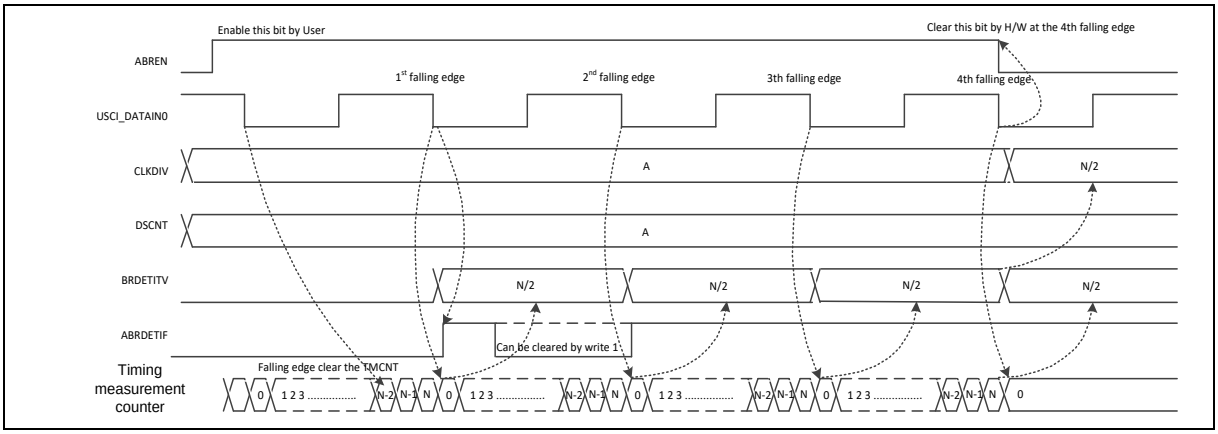


Figure 6.26-5 UART Auto Baud Rate Control

6.26.5.8 Auto Flow Control

The UART supports hardware auto-flow control that provides nRTS flow control by indicator RXFULL (UUART_BUFSTS[1]) on receiver buffer. When the buffer is full (RXFULL = 1), the nRTS is de-asserted.

The UART also provides nCTS flow control on transmitter. The nCTS is used to control the transmitted data is sent out when the nCTS is asserted.

6.26.5.9 RS-485 Support

The UART controller can play the role of the RS-485 master transmitter will identify an address character by setting the parity (9-th bit) to 1. For data characters, the parity is set to 0. Software can use the bit15 of each data to control the parity bit (PARITYEN (UUART_PROTCTL[1]) be set) when the STICKEN (UUART_PROTCTL[26]) is set. For example, if the STICKEN is set to 1 and data sequence are 0x8015, 0x8033, 0x0055, 0x0033 and 0x80AA the transmitted parity of data 0x15, 0x33, 0x55, 0x33 and 0xAA will be 1, 1, 0, 0 and 1.

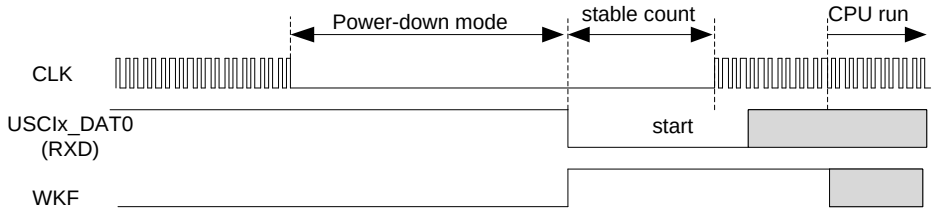
The UART controller can also play as an RS-485 addressable slave, the protocol-related error of PARITYERR (UUART_PROTSTS[5]) can be acted as the address bit detection when the PARITYEN (UUART_PROTCTL[1]), EVENPARITY (UUART_PROTCTL[2]) and STICKEN (UUART_PROTCTL[26]) were set. If the PARITYERR was set, it means that the address bit in the received bus is detected otherwise, the data is received into Buffer.

6.26.5.10 Wake-up Function

The USCI Controller in UART mode supports wake-up system function. The wake-up source includes incoming data and nCTS pin. Each wake-up source description is as follows:

- (a) Incoming data wake-up

  When system is in power-down and both of the WKEN (UUART_WKCTL [0]) and DATWKEN (UUART_PROTCTL[9]) are set, the toggle of incoming data pin can wake-up the system. In order to receive the incoming data after the system wake-up, the WAKECNT (UUART_PROTCTL[14:11]) shall be set. These bits field of WAKECNT (UUART_PROCTL[14:11]) indicate how many clock cycle selected by f_{PDS_CNT} do the controller can get the 1st bit (start bit) when the device is wakeup from Power-down mode. The incoming data wake-up is shown in Figure 6.26-6.



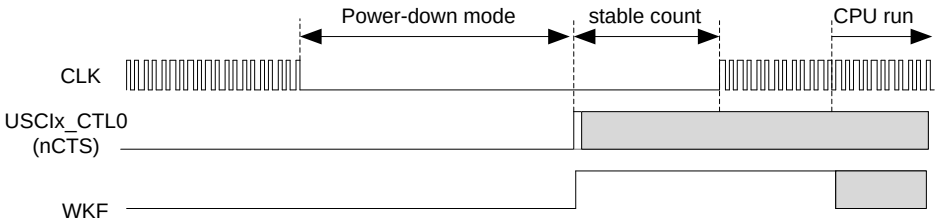
Note: Stable count means HCLK source recovery stable count.

Figure 6.26-6 Incoming Data Wake-Up

- (b) nCTS pin wake-up

  When system is in power-down and both of the WKEN (UUART_WKCTL [0]) and CTSWKEN (UUART_PROTCTL[10]) are set, the toggle of nCTS pin can wake-up the system. The nCTS wake-up is shown in Figure 6.26-7 and Figure 6.26-8.

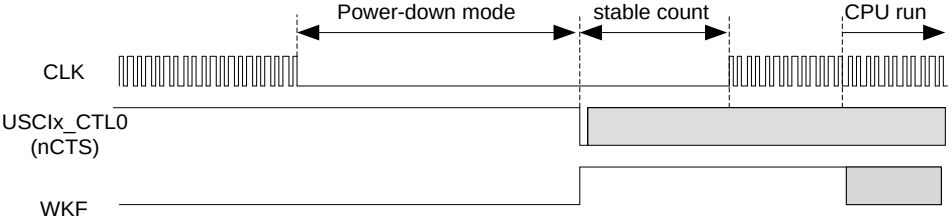
Case 1(nCTS transition from low to high):



Note: Stable count means HCLK source recovery stable count.

Figure 6.26-7 nCTS Wake-Up Case 1

Case 2 (nCTS transition from high to low):



Note: Stable count means HCLK source recovery stable count.

Figure 6.26-8 nCTS Wake-Up Case 2

6.26.5.11 Interrupt Events

The UART provides interrupt for protocol event and data transfer event. The description show below:

Protocol Interrupt Events

The following protocol-related events are generated in UART mode and can lead to a protocol interrupt.

Please note that the bits in register UUART_PROTSTS are not automatically cleared by hardware and

have to be cleared by software in order to monitor new incoming events.

Receiver Line Status

The protocol-related error FRMERR (UUART_PROTSTS[6]) or PARITYERR (UUART_PROTSTS[5]) are two flags that are assigned to each received data word in the corresponding receiver buffer status registers.

In UART mode, the result of the parity check by the protocol-related error indication PARITYERR (0 = received parity bit equal to calculated parity value), and the result of frame check by the protocol-related error indication FRMERR (0 = received stop bit equal to the format value '1'). This information is elaborated for each data frame.

The break error flag BREAK (UUART_PROTSTS[7]) is assigned when the receive data is 0, the received parity and the stop bit are also 0.

The interrupt indicates that there are parity error, frame error or the break data detection in the BREAK, FRMERR, PARITYERR (UUART_PROTSTS[7:5]) bits.

Auto Baud Rate Detection

The auto baud rate interrupt, ABRDETIF (UUART_PROTSTS [9]), indicates that the timing measurement counter has getting 2-bit duration for auto baud rate capture function.

The auto baud rate detection function will be enabled in the first falling edge of receiver signal. The auto baud rate detection function is measurement after the next following falling is detected and it is finished when the frame transfer done. After the transfer done, the timing measurement counter value divided by twice is equal to the number of sample time per bit. The user can read the value of BRDETITV (UUART_PROTCTL[24:16]) and write into the baud rate generator register CLKDIV (UUART_BRGEN[25:16]).

Data Transfer Interrupt Handling

The data transfer interrupts indicate events related to UART frame handling.

Transmit Start Interrupt

Bit TXSTIF (UUART_PROTSTS [1]) is set after the start bit of a data word. In buffer mode, this is the earliest point in time when a new data word can be written to UUART_TXDAT.

Transmitter Finished

This interrupt indicates that the transmitter has completely finished all data in the buffer. Bit TXENDIF (UUART_PROTSTS [2]) becomes set at the end of the last stop bit.

Receiver Starts Interrupt

Bit RXSTIF (UUART_PROTSTS [3]) is set after the sample point of the start bit.

Receiver Frame Finished

This interrupt indicates that the receiver has completely finished a frame. Bit RXENDIF (UUART_PROTSTS [4]) becomes set at the end of the last receive bit.

6.26.5.12 Programming Example

The following steps are used to configure the UART protocol setting and the data transmission.

1. Set FUNMODE (UUART_CTL[2:0]) to 0x2 to select UART protocol.
2. Write baud rate generator register UUART_BRGEN to select desired baud rate.
   - Set SPCLKSEL (UUART_BRGEN[3:2]), PTCLKSEL (UUART_BRGEN[1]) and RCLKSEL (UUART_BRGEN[0]) to select the clock source.
   - Configure CLKDIV (UUART_BRGEN[25:16]), DSCNT (UUART_BRGEN[14:10]) and
   - PDSCNT (UUART_BRGEN[9:8]) to determine the baud rate divider.
3. Write line control register UUART_LINECTL and protocol control register UUART_PROTCTL

to configure the transmission data format and UART protocol setting.

- Program data field length in DWIDTH (UUART_LINECTL[11:8]).
- Enable parity bit and determine the parity bit type by setting EVENPARITY (UUART_PROTCTL[2]) and PARITYEN (UUART_PROTCTL[1]).
- Configure stop bit length by setting STOPB (UUART_PROTCTL[0]).
- Enable LSB (UUART_LINECTL[0]) to select LSB first transmission for UART protocol.

- Set EDGEDET (UUART_DATIN0[4:3]) to 0x2 to select the detected edge as falling edge for receiver start bit detection.

4. Set PROTEN (UUART_PROTCTL[31]) to 1 to enable UART protocol.
5. Transmit and receive data.
   - Write transmit data register UUART_TXDAT to transmit data.
   - Wait until TXSTIF(UUART_PROTSTS[1]) is set and then user can write the next data in UUART_TXDAT.
   - When TXENDIF(UUART_PROTSTS[2]) is set, the transmit buffer is empty and the stop bit of the last data has been transmitted.
   - If RXENDIF(UUART_PROTSTS[4]) is set, the receiver has finished a data frame completely. User can get the data by reading receive data register UUART_RXDAT.

6.26.6 Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
UUART_UART Base Address: **UUARTn_BA = 0x400D_0000 + (0x1000 * n)** **n= 0, 1** **UUART_UART non-secure base address is UUARTn_BA + 0x1000_0000.**				
UUART_CTL	UUARTn_BA+0x00	R/W	USCI Control Register	0x0000_0000
UUART_INTEN	UUARTn_BA+0x04	R/W	USCI Interrupt Enable Register	0x0000_0000
UUART_BRGEN	UUARTn_BA+0x08	R/W	USCI Baud Rate Generator Register	0x0000_3C00
UUART_DATIN0	UUARTn_BA+0x10	R/W	USCI Input Data Signal Configuration Register 0	0x0000_0000
UUART_CTLIN0	UUARTn_BA+0x20	R/W	USCI Input Control Signal Configuration Register 0	0x0000_0000
UUART_CLKIN	UUARTn_BA+0x28	R/W	USCI Input Clock Signal Configuration Register	0x0000_0000
UUART_LINECTL	UUARTn_BA+0x2C	R/W	USCI Line Control Register	0x0000_0000
UUART_TXDAT	UUARTn_BA+0x30	W	USCI Transmit Data Register	0x0000_0000
UUART_RXDAT	UUARTn_BA+0x34	R	USCI Receive Data Register	0x0000_0000
UUART_BUFCTL	UUARTn_BA+0x38	R/W	USCI Transmit/Receive Buffer Control Register	0x0000_0000
UUART_BUFSTS	UUARTn_BA+0x3C	R/W	USCI Transmit/Receive Buffer Status Register	0x0000_0101
UUART_PDMACTL	UUARTn_BA+0x40	R/W	USCI PDMA Control Register	0x0000_0000
UUART_WKCTL	UUARTn_BA+0x54	R/W	USCI Wake-up Control Register	0x0000_0000
UUART_WKSTS	UUARTn_BA+0x58	R/W	USCI Wake-up Status Register	0x0000_0000
UUART_PROTCTL	UUARTn_BA+0x5C	R/W	USCI Protocol Control Register	0x0000_0000
UUART_PROTIEN	UUARTn_BA+0x60	R/W	USCI Protocol Interrupt Enable Register	0x0000_0000
UUART_PROTSTS	UUARTn_BA+0x64	R/W	USCI Protocol Status Register	0x0000_0000

6.26.7 Register Description

USCI Control Register (UUART_CTL)

Register	Offset	R/W	Description	Reset Value
UUART_CTL	UUARTn_BA+0x00	R/W	USCI Control Register	0x0000_0000

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

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2:0]	**FUNMODE**	**Function Mode** This bit field selects the protocol for this USCI controller. Selecting a protocol that is not available or a reserved combination disables the USCI. When switching between two protocols, the USCI has to be disabled before selecting a new protocol. Simultaneously, the USCI will be reset when user write 000 to FUNMODE. 000 = The USCI is disabled. All protocol related state machines are set to idle state. 001 = The SPI protocol is selected. 010 = The UART protocol is selected. 100 = The I^2C protocol is selected. Others = Reserved.

USCI Interrupt Enable Register (UUART_INTEN)

Register	Offset	R/W	Description	Reset Value
UUART_INTEN	UUARTn_BA+0x04	R/W	USCI Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			**RXENDIEN**	**RXSTIEN**	**TXENDIEN**	**TXSTIEN**	**Reserved**

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**RXENDIEN**	**Receive End Interrupt Enable Bit** This bit enables the interrupt generation in case of a receive finish event. 0 = The receive end interrupt Disabled. 1 = The receive end interrupt Enabled.
[3]	**RXSTIEN**	**Receive Start Interrupt Enable Bit** This bit enables the interrupt generation in case of a receive start event. 0 = The receive start interrupt Disabled. 1 = The receive start interrupt Enabled.
[2]	**TXENDIEN**	**Transmit End Interrupt Enable Bit** This bit enables the interrupt generation in case of a transmit finish event. 0 = The transmit finish interrupt Disabled. 1 = The transmit finish interrupt Enabled.
[1]	**TXSTIEN**	**Transmit Start Interrupt Enable Bit** This bit enables the interrupt generation in case of a transmit start event. 0 = The transmit start interrupt Disabled. 1 = The transmit start interrupt Enabled.
[0]	**Reserved**	Reserved.

USCI Baud Rate Generator Register (UUART_BRGEN)

Register	Offset	R/W	Description	Reset Value
UUART_BRGEN	UUARTn_BA+0x08	R/W	USCI Baud Rate Generator Register	0x0000_3C00

31	30	29	28	27	26	25	24
Reserved						CLKDIV	
23	22	21	20	19	18	17	16
CLKDIV							
15	14	13	12	11	10	9	8
Reserved	DSCNT					PDSCNT	
7	6	5	4	3	2	1	0
Reserved		TMCNTSRC	TMCNTEN	SPCLKSEL		PTCLKSEL	RCLKSEL

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25:16]	**CLKDIV**	**Clock Divider** This bit field defines the ratio between the protocol clock frequency f_{PROT_CLK} and the clock divider frequency f_{DIV_CLK} ($f_{DIV\_CLK} = f_{PROT\_CLK} / (CLKDIV+1)$). **Note:** In UART function, it can be updated by hardware in the 4th falling edge of the input data 0x55 when the auto baud rate function (ABREN(UUART_PROTCTL[6])) is enabled. The revised value is the average bit time between bit 5 and bit 6. The user can use revised CLKDIV and new BRDETITV (UUART_PROTCTL[24:16]) to calculate the precise baud rate.
[15]	**Reserved**	Reserved.
[14:10]	**DSCNT**	**Denominator for Sample Counter** This bit field defines the divide ratio of the sample clock f_{SAMP_CLK}. The divided frequency $f_{DS\_CNT} = f_{PDS\_CNT} / (DSCNT+1)$. **Note:** The maximum value of DSCNT is 0xF on UART mode and suggest to set over 4 to confirm the receiver data is sampled in right value.
[9:8]	**PDSCNT**	**Pre-divider for Sample Counter** This bit field defines the divide ratio of the clock division from sample clock f_{SAMP_CLK}. The divided frequency $f_{PDS\_CNT} = f_{SAMP\_CLK} / (PDSCNT+1)$.
[7:6]	**Reserved**	Reserved.
[5]	**TMCNTSRC**	**Timing Measurement Counter Clock Source Selection** 0 = Timing measurement counter with f_{PROT_CLK}. 1 = Timing measurement counter with f_{DIV_CLK}.
[4]	**TMCNTEN**	**Timing Measurement Counter Enable Bit** This bit enables the 10-bit timing measurement counter. 0 = Timing measurement counter is Disabled. 1 = Timing measurement counter is Enabled.
[3:2]	**SPCLKSEL**	**Sample Clock Source Selection**

		This bit field used for the clock source selection of a sample clock (f_{SAMP_CLK}) for the protocol processor. 00 = f_{SAMP_CLK} is selected to f_{DIV_CLK}. 01 = f_{SAMP_CLK} is selected to f_{PROT_CLK}. 10 = f_{SAMP_CLK} is selected to f_{SCLK}. 11 = f_{SAMP_CLK} is selected to f_{REF_CLK}.
[1]	**PTCLKSEL**	**Protocol Clock Source Selection** This bit selects the source signal of protocol clock (f_{PROT_CLK}). 0 = Reference clock f_{REF_CLK}. 1 = f_{REF_CLK2} (its frequency is half of f_{REF_CLK}).
[0]	**RCLKSEL**	**Reference Clock Source Selection** This bit selects the source signal of reference clock (f_{REF_CLK}). 0 = Peripheral device clock f_{PCLK}. 1 = Reserved.

USCI Input Data Signal Configuration (UUART_DATIN0)

Register	Offset	R/W	Description	Reset Value
UUART_DATIN0	UUARTn_BA+0x10	R/W	USCI Input Data Signal Configuration Register 0	0x0000_0000

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

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4:3]	**EDGEDET**	**Input Signal Edge Detection Mode** This bit field selects which edge actives the trigger event of input data signal. 00 = The trigger event activation is disabled. 01 = A rising edge activates the trigger event of input data signal. 10 = A falling edge activates the trigger event of input data signal. 11 = Both edges activate the trigger event of input data signal. **Note:** In UART function mode, it is suggested to set this bit field as 0x2.
[2]	**ININV**	**Input Signal Inverse Selection** This bit defines the inverter enable of the input asynchronous signal. 0 = The un-synchronized input signal will not be inverted. 1 = The un-synchronized input signal will be inverted.
[1]	**Reserved**	Reserved.
[0]	**SYNCSEL**	**Input Signal Synchronization Selection** This bit selects if the un-synchronized input signal (with optionally inverted) or the synchronized (and optionally filtered) signal can be used as input for the data shift unit. 0 = The un-synchronized signal can be taken as input for the data shift unit. 1 = The synchronized signal can be taken as input for the data shift unit.

USCI Input Control Signal Configuration (UUART_CTLIN0)

Register	Offset	R/W	Description	Reset Value
UUART_CTLIN0	UUARTn_BA+0x20	R/W	USCI Input Control Signal Configuration Register 0	0x0000_0000

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

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**ININV**	**Input Signal Inverse Selection** This bit defines the inverter enable of the input asynchronous signal. 0 = The un-synchronized input signal will not be inverted. 1 = The un-synchronized input signal will be inverted.
[1]	**Reserved**	Reserved.
[0]	**SYNCSEL**	**Input Synchronization Signal Selection** This bit selects if the un-synchronized input signal (with optionally inverted) or the synchronized (and optionally filtered) signal can be used as input for the data shift unit. 0 = The un-synchronized signal can be taken as input for the data shift unit. 1 = The synchronized signal can be taken as input for the data shift unit.

USCI Input Clock Signal Configuration Register (UUART_CLKIN)

Register	Offset	R/W	Description	Reset Value
UUART_CLKIN	UUARTn_BA+0x28	R/W	USCI Input Clock Signal Configuration Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**SYNCSEL**	**Input Synchronization Signal Selection** This bit selects if the un-synchronized input signal or the synchronized (and optionally filtered) signal can be used as input for the data shift unit. 0 = The un-synchronized signal can be taken as input for the data shift unit. 1 = The synchronized signal can be taken as input for the data shift unit.

USCI Line Control Register (UUART_LINECTL)

Register	Offset	R/W	Description	Reset Value
UUART_LINECTL	UUARTn_BA+0x2C	R/W	USCI Line Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				DWIDTH			
7	6	5	4	3	2	1	0
CTLOINV	Reserved	DATOINV	Reserved				LSB

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:8]	**DWIDTH**	**Word Length of Transmission** This bit field defines the data word length (amount of bits) for reception and transmission. The data word is always right-aligned in the data buffer. USCI support word length from 4 to 16 bits. 0000 = The data word contains 16 bits located at bit positions [15:0]. 0001 = Reserved. 0010 = Reserved. 0011 = Reserved. 0100 = The data word contains 4 bits located at bit positions [3:0]. 0101 = The data word contains 5 bits located at bit positions [4:0]. 0110 = The data word contains 6 bits located at bit positions [5:0]. 0111 = The data word contains 7 bits located at bit positions [6:0]. 1000 = The data word contains 8 bits located at bit positions [7:0]. 1001 = The data word contains 9 bits located at bit positions [8:0]. 1010 = The data word contains 10 bits located at bit positions [9:0]. 1011 = The data word contains 11 bits located at bit positions [10:0]. 1100 = The data word contains 12 bits located at bit positions [11:0]. 1101 = The data word contains 13 bits located at bit positions [12:0]. 1110 = The data word contains 14 bits located at bit positions [13:0]. 1111 = The data word contains 15 bits located at bit positions [14:0]. **Note:** In UART protocol, the length can be configured as 6~13 bits.
[7]	**CTLOINV**	**Control Signal Output Inverse Selection** This bit defines the relation between the internal control signal and the output control signal. 0 = No effect. 1 = The control signal will be inverted before its output. **Note:** In UART protocol, the control signal means nRTS signal.

[6]	**Reserved**	Reserved.
[5]	**DATOINV**	**Data Output Inverse Selection** This bit defines the relation between the internal shift data value and the output data signal of USCIx_DAT1 pin. 0 = The value of USCIx_DAT1 is equal to the data shift register. 1 = The value of USCIx_DAT1 is the inversion of data shift register.
[4:1]	**Reserved**	Reserved.
[0]	**LSB**	**LSB First Transmission Selection** 0 = The MSB, which bit of transmit/receive data buffer depends on the setting of DWIDTH, is transmitted/received first. 1 = The LSB, the bit 0 of data buffer, will be transmitted/received first.

USCI Transmit Data Register (UUART_TXDAT)

Register	Offset	R/W	Description	Reset Value
UUART_TXDAT	UUARTn_BA+0x30	W	USCI Transmit Data Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**TXDAT**	**Transmit Data** Software can use this bit field to write 16-bit transmit data for transmission.

USCI Receive Data Register (UUART_RXDAT)

Register	Offset	R/W	Description	Reset Value
UUART_RXDAT	UUARTn_BA+0x34	R	USCI Receive Data Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**RXDAT**	**Received Data** This bit field monitors the received data which stored in receive data buffer. **Note:** RXDAT[15:13] indicate the same frame status of BREAK, FRMERR and PARITYERR (UUART_PROTSTS[7:5]).

USCI Transmitter/Receive Buffer Control Register (UUART_BUFCTL)

Register	Offset	R/W	Description	Reset Value
UUART_BUFCTL	UUARTn_BA+0x38	R/W	USCI Transmit/Receive Buffer Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						RXRST	TXRST
15	14	13	12	11	10	9	8
RXCLR	RXOVIEN	Reserved					
7	6	5	4	3	2	1	0
TXCLR	Reserved						

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**RXRST**	**Receive Reset** 0 = No effect. 1 = Reset the receive-related counters, state machine, and the content of receive shift register and data buffer. **Note 1:** It is cleared automatically after one PCLK cycle. **Note 2:** It is suggested to check the RXBUSY (UUART_PROTSTS[10]) before this bit will be set to 1.
[16]	**TXRST**	**Transmit Reset** 0 = No effect. 1 = Reset the transmit-related counters, state machine, and the content of transmit shift register and data buffer**.** **Note:** It is cleared automatically after one PCLK cycle.
[15]	**RXCLR**	**Clear Receive Buffer** 0 = No effect. 1 = The receive buffer is cleared (filling level is cleared and output pointer is set to input pointer value). Should only be used while the buffer is not taking part in data traffic. **Note:** It is cleared automatically after one PCLK cycle.
[14]	**RXOVIEN**	**Receive Buffer Overrun Error Interrupt Enable Bit** 0 = Receive overrun interrupt Disabled. 1 = Receive overrun interrupt Enabled.
[13:8]	**Reserved**	Reserved.
[7]	**TXCLR**	**Clear Transmit Buffer** 0 = No effect. 1 = The transmit buffer is cleared (filling level is cleared and output pointer is set to input pointer value). It should only be used while the buffer is not taking part in data traffic. **Note:** It is cleared automatically after one PCLK cycle.

[6:0]	**Reserved**	Reserved.

USCI Transmit/Receive Buffer Status Register (UUART_BUFSTS)

Register	Offset	R/W	Description	Reset Value
UUART_BUFSTS	UUARTn_BA+0x3C	R/W	USCI Transmit/Receive Buffer Status Register	0x0000_0101

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						TXFULL	TXEMPTY
7	6	5	4	3	2	1	0
Reserved				RXOVIF	Reserved	RXFULL	RXEMPTY

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**TXFULL**	**Transmit Buffer Full Indicator (Read Only)** 0 = Transmit buffer is not full. 1 = Transmit buffer is full.
[8]	**TXEMPTY**	**Transmit Buffer Empty Indicator (Read Only)** 0 = Transmit buffer is not empty. 1 = Transmit buffer is empty.
[7:4]	**Reserved**	Reserved.
[3]	**RXOVIF**	**Receive Buffer Over-run Error Interrupt Status** This bit indicates that a receive buffer overrun error event has been detected. If RXOVIEN (UUART_BUFCTL[14]) is enabled, the corresponding interrupt request is activated. 0 = A receive buffer overrun error event has not been detected. 1 = A receive buffer overrun error event has been detected. **Note:** It is cleared by software writing 1 into this bit.
[2]	**Reserved**	Reserved.
[1]	**RXFULL**	**Receive Buffer Full Indicator (Read Only)** 0 = Receive buffer is not full. 1 = Receive buffer is full.
[0]	**RXEMPTY**	**Receive Buffer Empty Indicator (Read Only)** 0 = Receive buffer is not empty. 1 = Receive buffer is empty.

USCI PDMA Control Register (UUART_PDMACTL)

Register	Offset	R/W	Description	Reset Value
UUART_PDMACTL	UUARTn_BA+0x40	R/W	USCI PDMA Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				PDMAEN	RXPDMAEN	TXPDMAEN	PDMARST

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**PDMAEN**	**PDMA Mode Enable Bit** 0 = PDMA function Disabled. 1 = PDMA function Enabled.
[2]	**RXPDMAEN**	**PDMA Receive Channel Available** 0 = Receive PDMA function Disabled. 1 = Receive PDMA function Enabled.
[1]	**TXPDMAEN**	**PDMA Transmit Channel Available** 0 = Transmit PDMA function Disabled. 1 = Transmit PDMA function Enabled.
[0]	**PDMARST**	**PDMA Reset** 0 = No effect. 1 = Reset the USCI's PDMA control logic. This bit will be cleared to 0 automatically.

USCI Wake-up Control Register (UUART_WKCTL)

Register	Offset	R/W	Description	Reset Value
UUART_WKCTL	UUARTn_BA+0x54	R/W	USCI Wake-up Control Register	0x0000_0000

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

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**PDBOPT**	**Power Down Blocking Option** 0 = If user attempts to enter Power-down mode by executing WFI while the protocol is in transferring, MCU will stop the transfer and enter Power-down mode immediately. 1 = If user attempts to enter Power-down mode by executing WFI while the protocol is in transferring, the on-going transfer will not be stopped and MCU will enter idle mode immediately.
[1]	**Reserved**	Reserved.
[0]	**WKEN**	**Wake-up Enable Bit** 0 = Wake-up function Disabled. 1 = Wake-up function Enabled.

USCI Wake-up Status Register (UUART_WKSTS)

Register	Offset	R/W	Description	Reset Value
UUART_WKSTS	UUARTn_BA+0x58	R/W	USCI Wake-up Status Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**WKF**	**Wake-up Flag** When chip is woken up from Power-down mode, this bit is set to 1. Software can write 1 to clear this bit.

USCI Protocol Control Register – UART (UUART_PROTCTL)

Register	Offset	R/W	Description	Reset Value
UUART_PROTCTL	UUARTn_BA+0x5C	R/W	USCI Protocol Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PROTEN	DGE	BCEN	Reserved		STICKEN	Reserved	BRDETITV
23	22	21	20	19	18	17	16
BRDETITV							
15	14	13	12	11	10	9	8
Reserved	WAKECNT				CTSWKEN	DATWKEN	Reserved
7	6	5	4	3	2	1	0
Reserved	ABREN	RTSAUDIREN	CTSAUTOEN	RTSAUTOEN	EVENPARITY	PARITYEN	STOPB

Bits	Description	
[31]	**PROTEN**	**UART Protocol Enable Bit** 0 = UART Protocol Disabled. 1 = UART Protocol Enabled.
[30]	**DGE**	**Deglitch Enable Bit** 0 = Deglitch Disabled. 1 = Deglitch Enabled. **Note:** When this bit is set to logic 1, any pulse width less than about 300 ns will be considered a glitch and will be removed in the serial data input (RX). This bit acts only on RX line and has no effect on the transmitter logic.
[29]	**BCEN**	**Transmit Break Control Enable Bit** 0 = Transmit Break Control Disabled. 1 = Transmit Break Control Enabled. **Note:** When this bit is set to logic 1, the serial data output (TX) is forced to the Spacing State (logic 0). This bit acts only on TX line and has no effect on the transmitter logic.
[28:27]	**Reserved**	Reserved.
[26]	**STICKEN**	**Stick Parity Enable Bit** 0 = Stick parity Disabled. 1 = Stick parity Enabled. **Note:** Refer to RS-485 Support section for detailed information.
[25]	**Reserved**	Reserved.
[24:16]	**BRDETITV**	**Baud Rate Detection Interval** This bit fields indicate how many clock cycle selected by TMCNTSRC (UUART_BRGEN [5]) does the slave calculates the baud rate in one bits. The order of the bus shall be 1 and 0 step by step (e.g. the input data pattern shall be 0x55). The user can read the value to know the current input baud rate of the bus whenever the ABRDETIF (UUART_PROTCTL[9]) is set. **Note:** This bit can be cleared to 0 by software writing '0' to the **BRDETITV**.

[15]	**Reserved**	Reserved.
[14:11]	**WAKECNT**	**Wake-up Counter** These bits field indicate how many clock cycle selected by f_{PDS_CNT} do the slave can get the 1st bit (start bit) when the device is woken up from Power-down mode.
[10]	**CTSWKEN**	**nCTS Wake-up Mode Enable Bit** 0 = nCTS wake-up mode Disabled. 1 = nCTS wake-up mode Enabled.
[9]	**DATWKEN**	**Data Wake-up Mode Enable Bit** 0 = Data wake-up mode Disabled. 1 = Data wake-up mode Enabled.
[8:7]	**Reserved**	Reserved.
[6]	**ABREN**	**Auto-baud Rate Detect Enable Bit** 0 = Auto-baud rate detect function Disabled. 1 = Auto-baud rate detect function Enabled. **Note:** When the auto - baud rate detect operation finishes, hardware will clear this bit. The associated interrupt ABRDETIF (UUART_PROTSTS[9]) will be generated (If ABRIEN (UUART_PROTIEN [1]) is enabled).
[5]	**RTSAUDIREN**	**nRTS Auto Direction Enable Bit** When nRTS auto direction is enabled, if the transmitted bytes in the TX buffer is empty, the nRTS signal is inactive automatically. 0 = nRTS auto direction control Disabled. 1 = nRTS auto direction control Enabled. **Note 1:** This bit is used for nRTS auto direction control for RS485. **Note 2:** This bit has effect only when the RTSAUTOEN is not set.
[4]	**CTSAUTOEN**	**nCTS Auto-flow Control Enable Bit** When nCTS auto-flow is enabled, the UART will send data to external device when nCTS input assert (UART will not send data to device if nCTS input is dis-asserted). 0 = nCTS auto-flow control Disabled. 1 = nCTS auto-flow control Enabled.
[3]	**RTSAUTOEN**	**nRTS Auto-flow Control Enable Bit** When nRTS auto-flow is enabled, if the receiver buffer is full (RXFULL (UUART_BUFSTS[1] =1), the UART will de-assert nRTS signal. 0 = nRTS auto-flow control Disabled. 1 = nRTS auto-flow control Enabled. **Note:** This bit has effect only when the RTSAUDIREN is not set.
[2]	**EVENPARITY**	**Even Parity Enable Bit** 0 = Odd number of logic 1's is transmitted and checked in each word. 1 = Even number of logic 1's is transmitted and checked in each word. **Note:** This bit has effect only when PARITYEN is set.
[1]	**PARITYEN**	**Parity Enable Bit** This bit defines the parity bit is enabled in an UART frame. 0 = The parity bit Disabled. 1 = The parity bit Enabled.
[0]	**STOPB**	**Stop Bits** This bit defines the number of stop bits in an UART frame.

		0 = The number of stop bits is 1. 1 = The number of stop bits is 2.

USCI Protocol Interrupt Enable Register – UART (UUART_PROTIEN)

Register	Offset	R/W	Description	Reset Value
UUART_PROTIEN	UUARTn_BA+0x60	R/W	USCI Protocol Interrupt Enable Register	0x0000_0000

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

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**RLSIEN**	**Receive Line Status Interrupt Enable Bit** 0 = Receive line status interrupt Disabled. 1 = Receive line status interrupt Enabled. **Note:** UUART_PROTSTS[7:5] indicates the current interrupt event for receive line status interrupt.
[1]	**ABRIEN**	**Auto-baud Rate Interrupt Enable Bit** 0 = Auto-baud rate interrupt Disabled. 1 = Auto-baud rate interrupt Enabled.
[0]	**Reserved**	Reserved.

USCI Protocol Status Register – UART (UUART_PROTSTS)

Register	Offset	R/W	Description	Reset Value
UUART_PROTSTS	UUARTn_BA+0x64	R/W	USCI Protocol Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						CTSLV	CTSSYNCLV
15	14	13	12	11	10	9	8
Reserved				ABERRSTS	RXBUSY	ABRDETIF	Reserved
7	6	5	4	3	2	1	0
BREAK	FRMERR	PARITYERR	RXENDIF	RXSTIF	TXENDIF	TXSTIF	Reserved

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**CTSLV**	**nCTS Pin Status (Read Only)** This bit used to monitor the current status of nCTS pin input. 0 = nCTS pin input is low level voltage logic state. 1 = nCTS pin input is high level voltage logic state.
[16]	**CTSSYNCLV**	**nCTS Synchronized Level Status (Read Only)** This bit used to indicate the current status of the internal synchronized nCTS signal. 0 = The internal synchronized nCTS is low. 1 = The internal synchronized nCTS is high.
[15:12]	**Reserved**	Reserved.
[11]	**ABERRSTS**	**Auto-baud Rate Error Status** This bit is set when auto-baud rate detection counter overrun. When the auto-baud rate counter overrun, the user shall revise the CLKDIV (UUART_BRGEN[25:16]) value and enable ABREN (UUART_PROTCTL[6]) to detect the correct baud rate again. 0 = Auto-baud rate detect counter is not overrun. 1 = Auto-baud rate detect counter is overrun. **Note 1:** This bit is set at the same time of ABRDETIF. **Note 2:** This bit can be cleared by writing “1” to ABRDETIF or ABERRSTS.
[10]	**RXBUSY**	**RX Bus Status Flag (Read Only)** This bit indicates the busy status of the receiver. 0 = The receiver is Idle. 1 = The receiver is BUSY.
[9]	**ABRDETIF**	**Auto-baud Rate Interrupt Flag** This bit is set when auto-baud rate detection is done among the falling edge of the input data. If the

		ABRIEN (UUART_PROTIEN[1]) is set, the auto-baud rate interrupt will be generated. This bit can be set 4 times when the input data pattern is 0x55 and it is cleared before the next falling edge of the input bus. 0 = Auto-baud rate detect function is not done. 1 = One Bit auto-baud rate detect function is done. **Note:** This bit can be cleared by writing “1” to it.
[8]	**Reserved**	Reserved.
[7]	**BREAK**	**Break Flag** This bit is set to logic 1 whenever the received data input (RX) is held in the “spacing state” (logic 0) for longer than a full word transmission time (that is, the total time of “start bit” + data bits + parity + stop bits). 0 = No Break is generated. 1 = Break is generated in the receiver bus. **Note:** This bit can be cleared by writing “1” among the BREAK, FRMERR and PARITYERR bits.
[6]	**FRMERR**	**Framing Error Flag** This bit is set to logic 1 whenever the received character does not have a valid “stop bit” (that is, the stop bit following the last data bit or parity bit is detected as logic 0). 0 = No framing error is generated. 1 = Framing error is generated. **Note:** This bit can be cleared by writing “1” among the BREAK, FRMERR and PARITYERR bits.
[5]	**PARITYERR**	**Parity Error Flag** This bit is set to logic 1 whenever the received character does not have a valid “parity bit”. 0 = No parity error is generated. 1 = Parity error is generated. **Note:** This bit can be cleared by writing “1” among the BREAK, FRMERR and PARITYERR bits.
[4]	**RXENDIF**	**Receive End Interrupt Flag** 0 = A receive finish interrupt status has not occurred. 1 = A receive finish interrupt status has occurred. **Note:** It is cleared by software writing 1 into this bit.
[3]	**RXSTIF**	**Receive Start Interrupt Flag** 0 = A receive start interrupt status has not occurred. 1 = A receive start interrupt status has occurred. **Note:** It is cleared by software writing 1 into this bit.
[2]	**TXENDIF**	**Transmit End Interrupt Flag** 0 = A transmit end interrupt status has not occurred. 1 = A transmit end interrupt status has occurred. **Note:** It is cleared by software writing 1 into this bit.
[1]	**TXSTIF**	**Transmit Start Interrupt Flag** 0 = A transmit start interrupt status has not occurred. 1 = A transmit start interrupt status has occurred. **Note 1:** It is cleared by software writing one into this bit. **Note 2:** Used for user to load next transmit data when there is no data in transmit buffer.
[0]	**Reserved**	Reserved.

6.27 USCI - SPI Mode

6.27.1 Overview

The SPI protocol of USCI controller applies to synchronous serial data communication and allows full duplex transfer. It supports both Master and Slave operation mode with the 4-wire bi-direction interface. SPI mode of USCI controller performs a serial-to-parallel conversion on data received from a peripheral device, and a parallel-to-serial conversion on data transmitted to a peripheral device. The SPI mode is selected by FUNMODE (USPI_CTL[2:0]) = 0x1

This SPI protocol can operate as Master or Slave mode by setting the SLAVE (USPI_PROTCTL[0]) to communicate with the off-chip SPI Slave or Master device. The application block diagrams in Master and Slave mode are shown below.

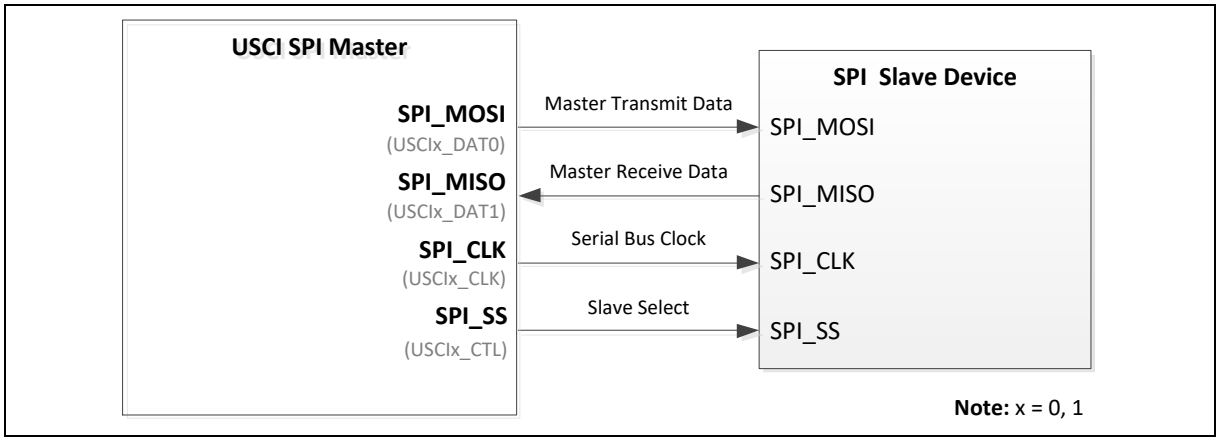


Figure 6.27-1 SPI Master Mode Application Block Diagram

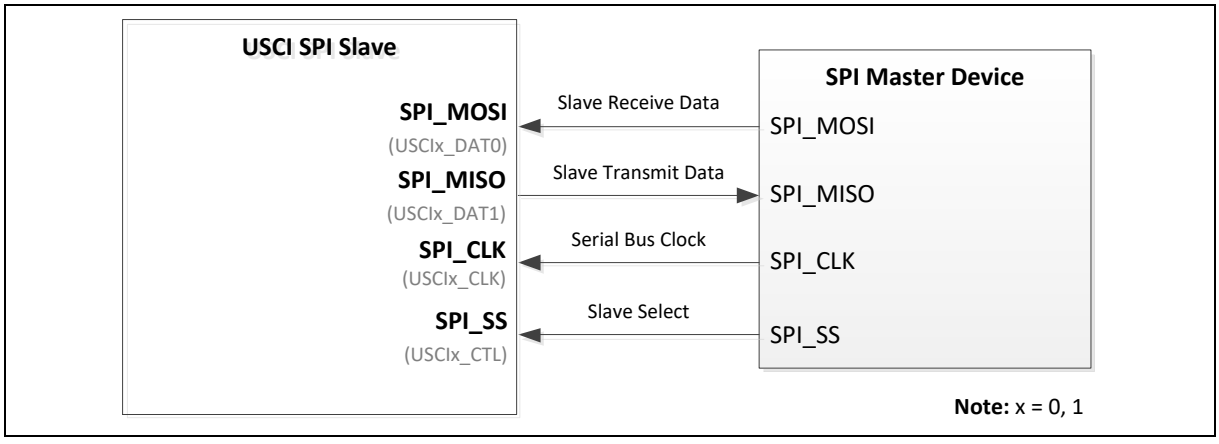


Figure 6.27-2 SPI Slave Mode Application Block Diagram

6.27.2 Features

- Supports Master or Slave mode operation (the maximum frequency -- Master < f_{PCLK} / 2, Slave < f_{PCLK} / 5)
- Configurable bit length of a transfer word from 4 to 16-bit
- Supports one transmit buffer and two receive buffers for data payload

- Supports MSB first or LSB first transfer sequence
- Supports Word Suspend function
- Supports PDMA transfer
- Supports 3-wire, no slave select signal, bi-direction interface
- Supports wake-up function by slave select signal in Slave mode
- Supports one data channel half-duplex transfer

6.27.3 Block Diagram

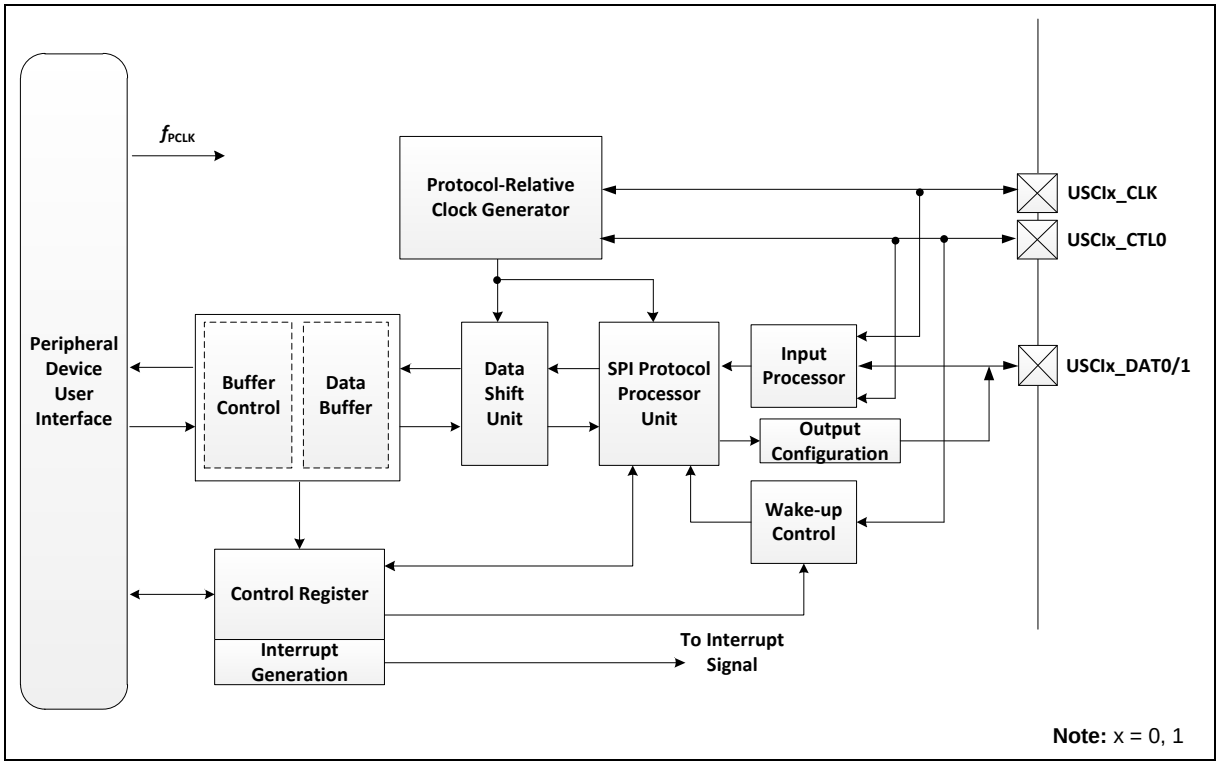


Figure 6.27-3 USCI SPI Mode Block Diagram

6.27.4 Basic Configuration

6.27.4.1 USCI0 SPI Basic Configurations

- Clock Source Configuration
  - Enable USCI0 peripheral clock in USCI0CKEN (CLK_APBCLK1[8]).
  - Enable USCI0_SPI function register in FUNMODE (USPI_CTL[2:0]=0x1).
- Reset Configuration
  - Reset USCI0 controller in USCI0RST (SYS_IPRST2[8]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
USCI0	USCI0_CLK	PD.0	MFP3

		PB.12	MFP5
		PA.11	MFP6
		PE.2	MFP7
	USCI0_CTL0	PD.4	MFP3
		PD.14	MFP5
		PC.13	MFP6
		PE.6	MFP7
	USCI0_DAT0	PD.1	MFP3
		PB.13	MFP5
		PA.10	MFP6
		PE.3	MFP7
	USCI0_DAT1	PD.2	MFP3
		PB.14	MFP5
		PA.9	MFP6
		PE.4	MFP7

6.27.4.2 USCI1 SPI Basic Configurations

- Clock source Configuration
  - Enable USCI1 peripheral clock in USCI1CKEN (CLK_APBCLK1[9]).
  - Enable USCI1_SPI function register in FUNMODE (USPI_CTL[2:0]=0x1).
- Reset Configuration
  - Reset USCI1 controller in USCI1RST (SYS_IPRST2[9]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
USCI1	USCI1_CLK	PB.8	MFP4
		PD.7, PE.12	MFP6
		PB.1	MFP8
	USCI1_CTL0	PB.10	MFP4
		PD.3, PE.9	MFP6
		PB.5	MFP8
	USCI1_DAT0	PB.7	MFP4
		PD.5, PE.10	MFP6
		PB.2	MFP8
	USCI1_DAT1	PB.6	MFP4
		PD.6, PE.11	MFP6

		PB.3	MFP8

6.27.5 Functional Description

6.27.5.1 USCI Common Function Description

Please refer to section USCI for detailed information.

6.27.5.2 Signal Description

A device operating in Master mode controls the start and end of a data transfer, as well as the generation of the SPI bus clock and slave select signal. The slave select signal indicates the start and the end of a data transfer, and the Master device can use it to enable the transmitting or receiving operations of Slave device. Slave device receives the SPI bus clock and optionally a slave select signal for data transaction. The signals for SPI communication are shown in Table 6.27-1.

SPI Mode	Receive Data	Transmit Data	Serial Bus Clock	Slave Select
Full-duplex SPI Master	SPI_MISO (USCIx_DAT1)	SPI_MOSI (USCIx_DAT0)	SPI_CLK (USCIx_CLK)	SPI_SS (USCIx_CTL0)
Full-duplex SPI Slave	SPI_MOSI (USCIx_DAT0)	SPI_MISO (USCIx_DAT1)	SPI_CLK (USCIx_CLK)	SPI_SS (USCIx_CTL0)
Half-duplex SPI Master/Slave	SPI_MOSI (USCIx_DAT0)	SPI_MOSI (USCIx_DAT0)	SPI_CLK (USCIx_CLK)	SPI_SS (USCIx_CTL0)

Table 6.27-1 Signals for SPI communication

- ***SPI Communication Signals***

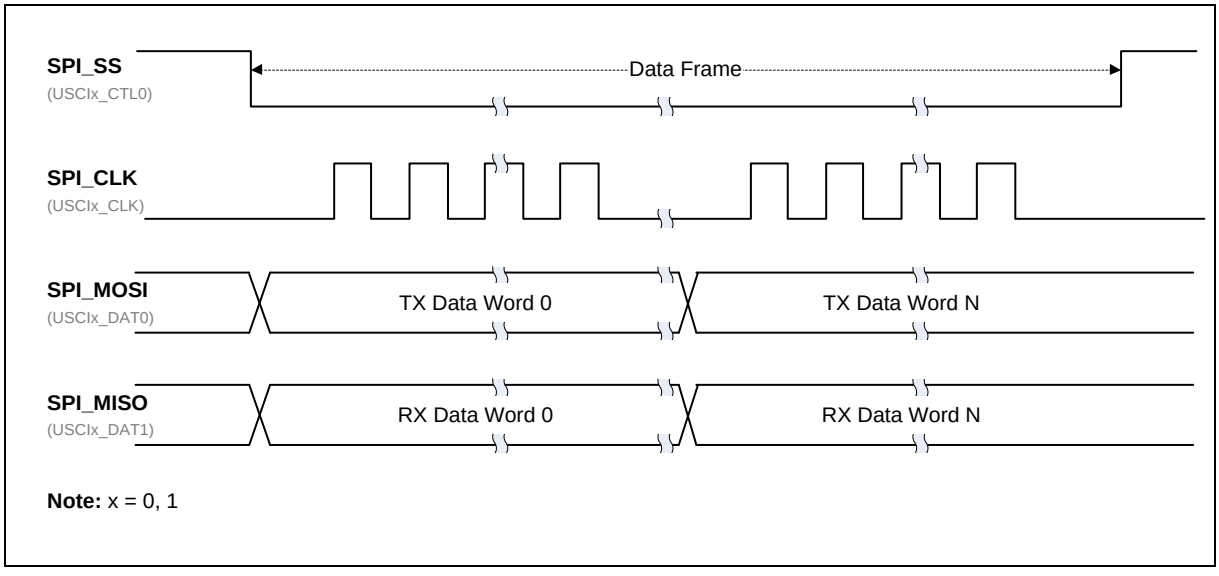


Figure 6.27-4 4-Wire Full-Duplex SPI Communication Signals (Master Mode)

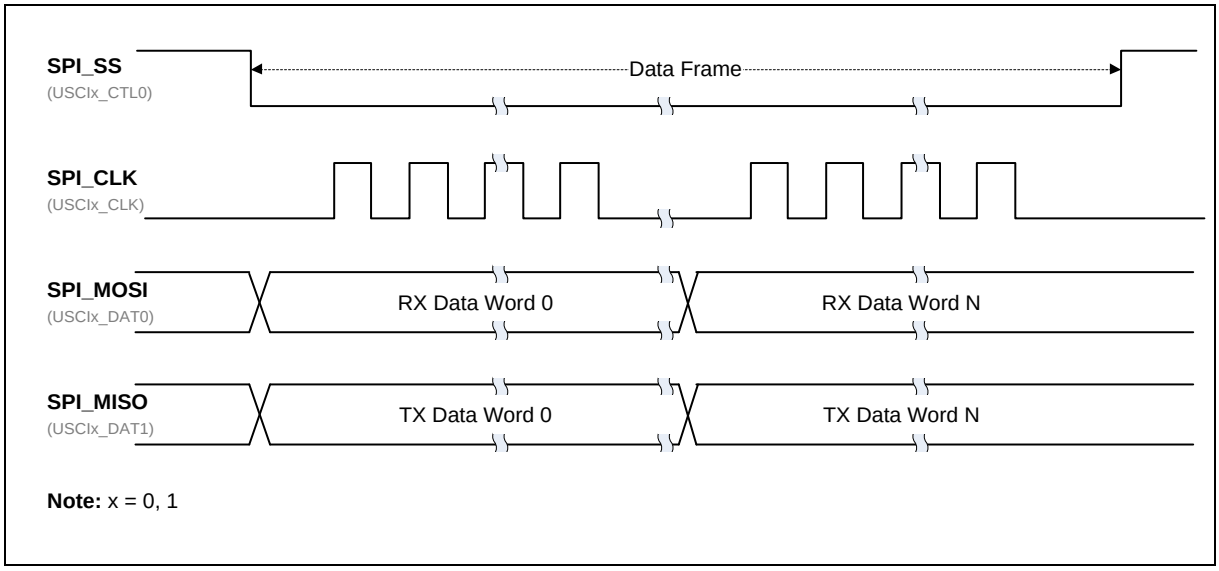


Figure 6.27-5 4-Wire Full-Duplex SPI Communication Signals (Slave Mode)

6.27.5.3 Serial Bus Clock Configuration

The USCI controller needs the peripheral clock to drive the USCI logic unit to perform the data transfer. The peripheral clock frequency is equal to PCLK frequency.

In Master mode, the frequency of the SPI bus clock is determined by protocol-relative clock generator. In general, the SPI bus clock is denoted as SPI clock. The frequency of SPI clock is half of f_{SAMP_CLK}, which can be selected by SPCLKSEL (USPI_BRGEN[3:2]). Refer to USCI for details of protocol-relative clock generator.

In Slave mode, the SPI bus clock is provided by an off-chip Master device. The peripheral clock frequency, f_{PCLK}, of SPI Slave device must be 5-times faster than the serial bus clock rate of the SPI Master device connected together (i.e. the clock rate of serial bus clock < 1/5 peripheral clock f_{PCLK} in Slave mode).

In SPI protocol, SCLKMODE (USPI_PROTCTL[7:6]) defines not only the idle state of serial bus clock but also the serial clock edge used for transmit and receive data. Both Master and Slave devices on the same communication bus should have the same SCLKMODE configuration. The four kinds of serial bus clock configuration are shown below.

SCLKMODE [1:0]	SPI Clock Idle State	Transmit Timing	Receive Timing
0x0	Low	Falling edge	Rising edge
0x1	Low	Rising edge	Falling edge
0x2	High	Rising edge	Falling edge
0x3	High	Falling edge	Rising edge

Table 6.27-2 Serial Bus Clock Configuration

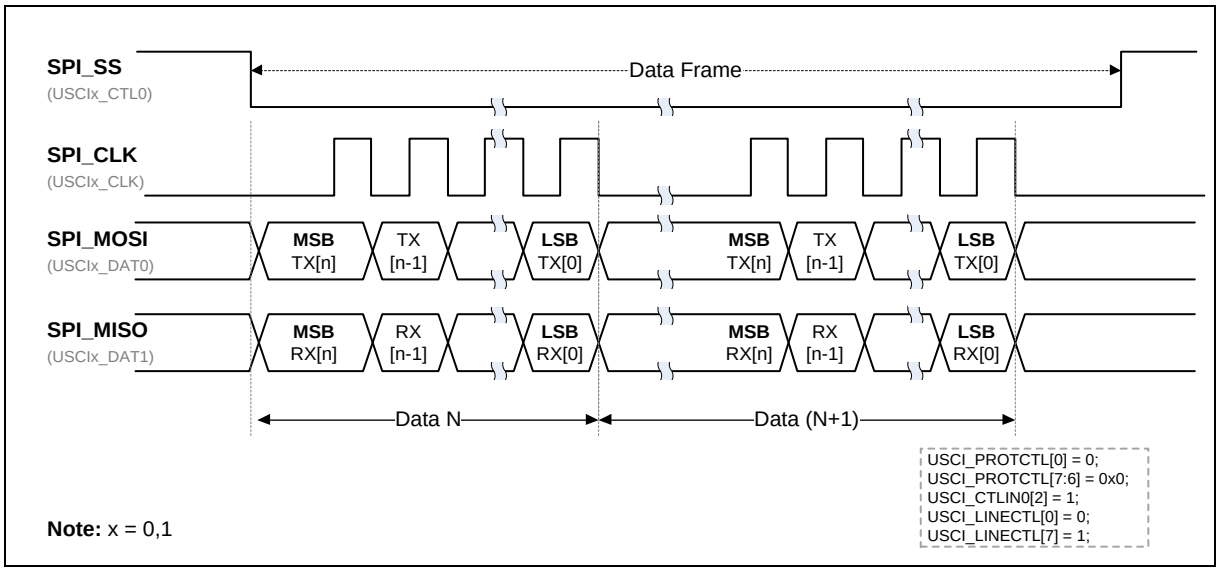


Figure 6.27-6 SPI Communication with Different SPI Clock Configuration (SCLKMODE=0x0)

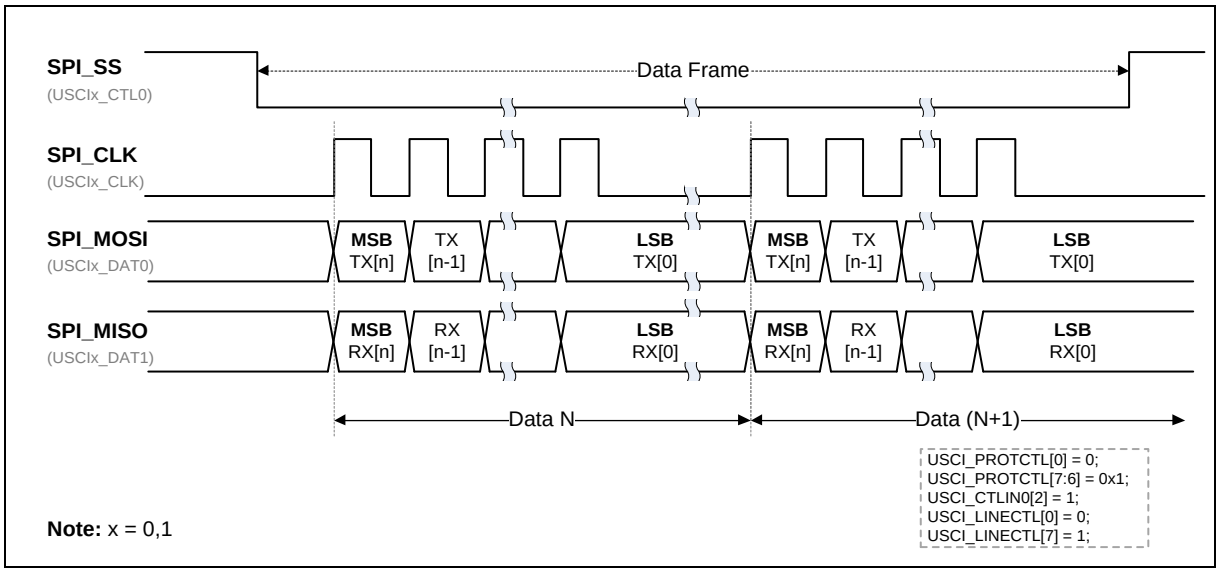


Figure 6.27-7 SPI Communication with Different SPI Clock Configuration (SCLKMODE=0x1)

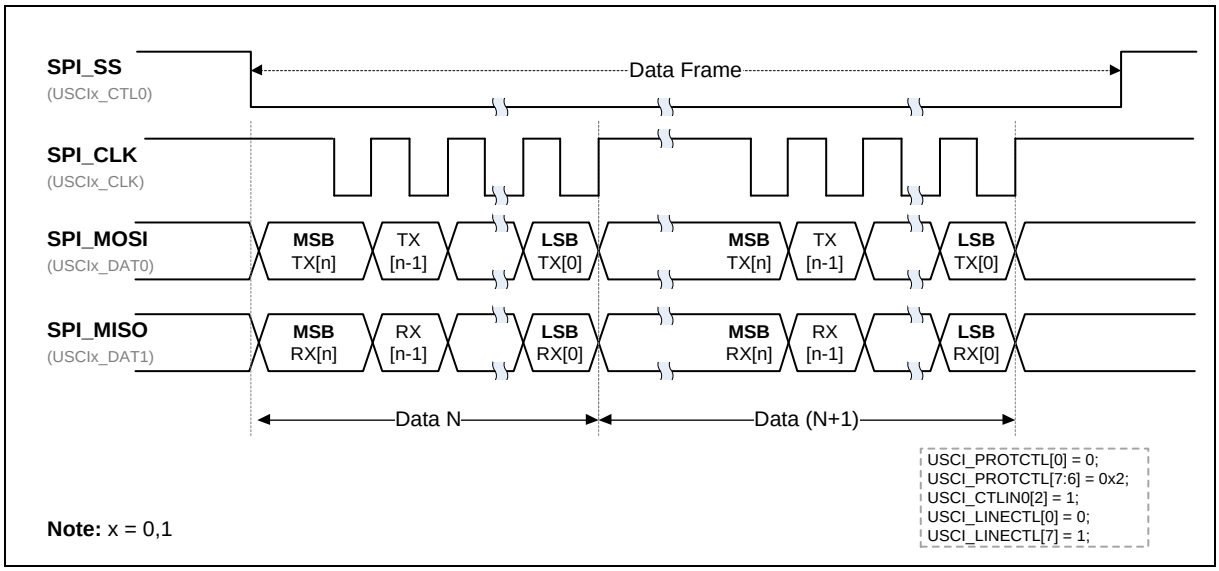


Figure 6.27-8 SPI Communication with Different SPI Clock Configuration (SCLKMODE=0x2)

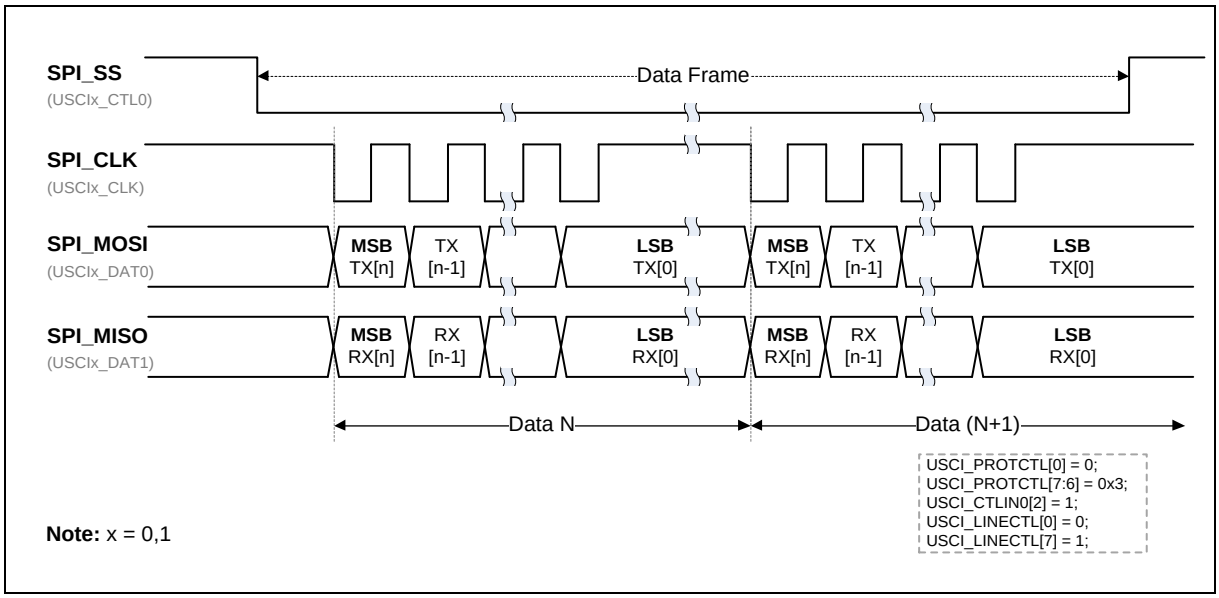


Figure 6.27-9 SPI Communication with Different SPI Clock Configuration (SCLKMODE=0x3)

6.27.5.4 Slave Select Signal

The slave selection signal of SPI protocol is active high by default. In SPI Master mode, the USCI controller can drive the control signal to off-chip SPI Slave device through slave select pin SPI_SS (USCIx_CTL0). In SPI Slave mode, the received slave select signal can be inverted by ININV (USPI_CTLIN0[2]).

If the slave select signal of external SPI Master device is low active, the ININV (USPI_CTLIN0[2]) setting of Slave device should be set to 1 for the inversion of input control signal. If USCI operates as SPI Master mode, the output slave select inversion CTLOINV (USPI_LINECTL[7]) is also needed to set as 1 for the external SPI Slave device whose slave select signal is active low.

The duration between the slave select active edge and the first SPI clock input edge shall over 2 USCI peripheral clock cycles.

The input slave select signal of SPI Slave has to be kept inactive for at least 2 USCI peripheral clock cycles between two consecutive frames in order to correctly detect the end of a frame. In Slave mode, the SPI_MISO (USCIx_DAT1) signal type is controlled by the SPI_SS (USCIx_CTL0) pin. If the SPI_SS (USCIx_CTL0) pin is inactive state, the SPI_MISO (USCIx_DAT1) is set to tri-state to avoid interference with other Slave devices. If the SPI_SS (USCIx_CTL0) pin is active state, the SPI_MISO (USCIx_DAT1) is set to output mode for data transfer.

6.27.5.5 Transmit and Receive Data

The bit length of a transmit/receive data word in SPI protocol of USCI controller is defined in DWIDTH (USPI_LINECTL[11:8]), and it can be configured up to 16-bit length for transmitting and receiving data in SPI communication.

The LSB bit (USPI_LINECTL[0]) defines the order of transfer data bit. If the LSB bit is set to 1, the transmission data sequence is LSB first. If the LSB bit is cleared to 0, the transmission data sequence is MSB first.

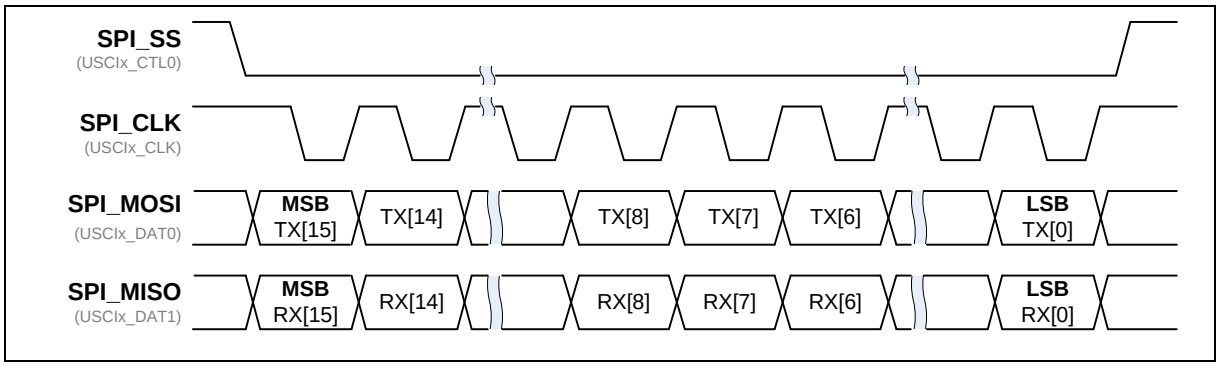
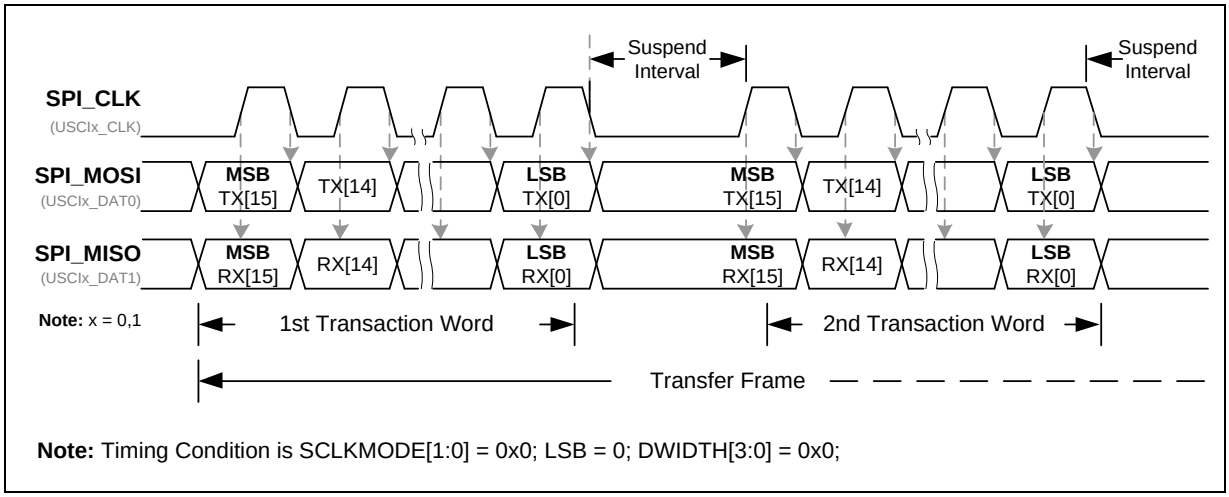


Figure 6.27-10 16-bit Data Length in One Word Transaction with MSB First Format

6.27.5.6 Word Suspend

SUSPITV (USPI_PROTCTL[11:8]) provides a configurable suspend interval, 0.5 ~ 15.5 SPI clock periods, between two successive transaction words in Master mode. The definition of the suspend interval is the interval between the last clock edge of the preceding transaction word and the first clock edge of the following transaction word. The default value of SUSPITV (USPI_PROTCTL[11:8]) is 0x3 (3.5 SPI clock cycles).



Note: Timing Condition is SCLKMODE[1:0] = 0x0; LSB = 0; DWIDTH[3:0] = 0x0;

Figure 6.27-11 Word Suspend Interval between Two Transaction Words

6.27.5.7 Automatic Slave Select Function

AUTOSS (USPI_PROTCTL[3]) is used for SPI Master mode to enable the automatic slave select function. If the bit AUTOSS (USPI_PROTCTL[3]) is set, the slave select signal will be generated automatically and the setting value of SS (USPI_PROTCTL[2]) will not affect the output slave select (through USCIx_CTL0 line). This means that the slave select signal will be asserted by the USCI controller when the SPI data transfer is started by writing to the transmit buffer. And, it will be de-asserted after either all transaction is finished or one word transaction done if the value of SUSPITV (USPI_PROTCTL[11:8]) is equal to or greater than 3.

If the AUTOSS bit (USPI_PROTCTL[3]) is cleared, the slave select on USCIx_CTL0 pin will be asserted/de-asserted by setting/clearing the SS (USPI_PROTCTL[2]). The internal slave select signal is active high and the CTLOINV (USPI_LINECTL[7]) can be used for the inversion of the slave select signal.

In SPI Master mode, if the value of SUSPITV (USPI_PROTCTL[11:8]) is less than 3 and the AUTOSS (USPI_PROTCTL[3]) is set as 1, the slave select signal will be kept at active state between two successive word transactions.

In SPI Slave mode, to recognize the inactive state of the slave select signal, the inactive period of the received slave select signal must be larger than 2 peripheral clock cycles between two successive transactions.

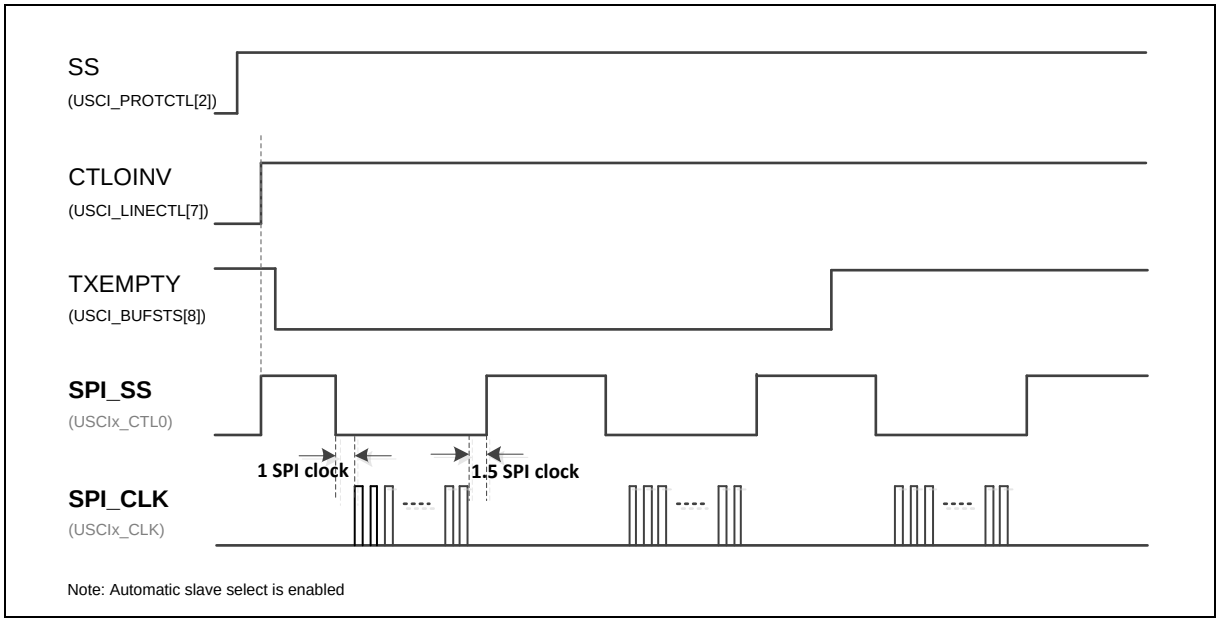


Figure 6.27-12 Auto Slave Select (SUSPITV ≧ 0x3)

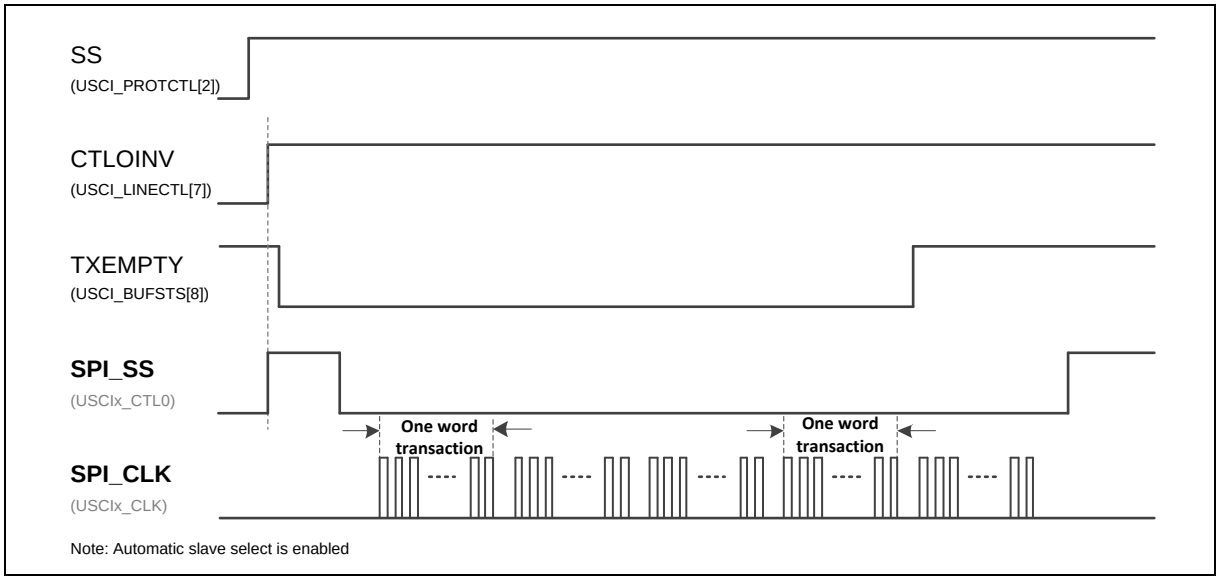


Figure 6.27-13 Auto Slave Select (SUSPITV < 0x3)

6.27.5.8 Slave 3-wire Mode

When the SLV3WIRE (USPI_PROTCTL[1]) is set by software to enable the Slave 3-wire mode, the USCI SPI communication can work with no slave select signal in Slave mode. The SLV3WIRE (USPI_PROTCTL[1]) only takes effect in SPI Slave mode. Only three pins, SPI_CLK (through USCIx_CLK line), SPI_MOSI (through USCIx_DAT0 line), and SPI_MISO (through USCIx_DAT1 line), are required to communicate with a SPI Master. When the SLV3WIRE (USPI_PROTCTL[1]) is set to 1, the SPI Slave will be ready to transmit/receive data after the SPI protocol is enabled by setting FUNMODE (USPI_CTL[2:0]) to 0x1.

6.27.5.9 Data Transfer Mode

The USCI controller supports full-duplex SPI transfer and one data channel half-duplex SPI transfer.

- Full-duplex SPI transfer

  In full-duplex SPI transfer, there are two data pins. One is used for transmitting data and the other is used for receiving data. Thus, data transmission and data reception can be performed simultaneously.

  SCLKMODE (USPI_PROTCTL[7:6]) defines the transition timing of the data shift output signal on USCIx_DAT0 pin. The transition may happen at the corresponding edge of SPI bus clock or active edge of slave select signal. The level of the last data bit of a data word is held on USCIx_DAT0 pin until the next data word begins with the next corresponding edge of the serial bus clock.

- One data channel half-duplex SPI transfer

  In one data channel half-duplex SPI transfer, there is only one data pin for data transfer. Thus, the data transmission and data reception are at different time interval. The data shift direction is determined by PORTDIR (USPI_TXDAT[16]). Refer to the register description for more detail information.

  The function of one data channel half-duplex SPI transfer is similar to the full-duplex SPI protocol. All the transfer data timing is the same as the full-duplex SPI transfer.

  Figure 6.27-14 and Figure 6.27-15 show the one output data channel and one input data channel half-duplex transfer diagrams with the external device.

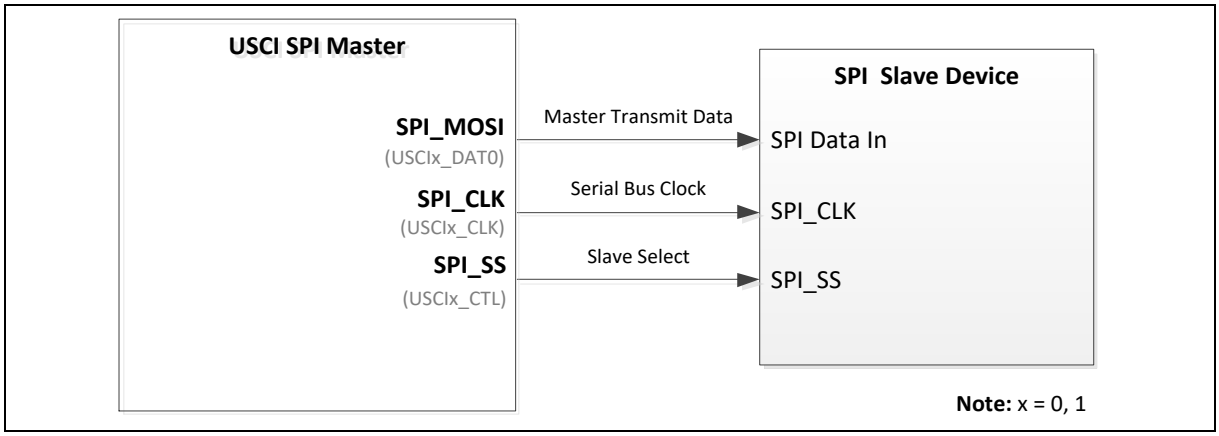


Figure 6.27-14 One Output Data Channel Half-duplex (SPI Master Mode)

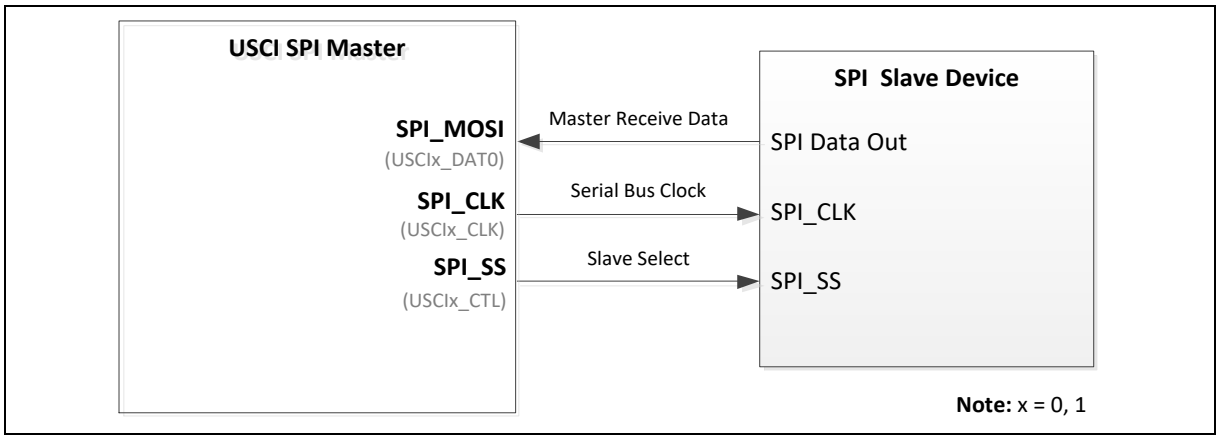


Figure 6.27-15 One Input Data Channel Half-duplex (SPI Master Mode)

The one data channel half-duplex transfer mode can be configured by TSMSEL[2:0] (USPI_PROTCTL[14:12]) and PORTDIR (USPI_TXDAT[16]) settings. When TSMSEL (USPI_PROTCTL[14:12]) is set to 0x4, one data channel half-duplex transfer mode is selected. The PORTDIR (USPI_TXDAT[16]) is used to define the direction of the corresponding transmit data. When the PORTDIR bit is set to 0, the USCI controller will send the corresponding data to external SPI device. When the PORTDIR bit is set to 1, the controller will read the corresponding data from the external SPI device.

For example, in one data channel half-duplex transfer mode with PORTDIR=0, USCI SPI transmits data through USCIx_DAT0 pin; if PORTDIR=1, USCI SPI receives data through USCIx_DAT0 pin.

6.27.5.10 Interrupt

Data Transfer Interrupts

- Transmit start interrupt

  The interrupt event TXSTIF (USPI_PROTSTS[1]) is set after the start of the first data bit of a transmit data word. It can be cleared only by writing 1 to it.

- Transmit end interrupt

  The interrupt event TXENDIF (USPI_PROTSTS[2]) is set after the start of the last data bit of the last transmit data which has been stored in transmit buffer. It can be cleared only by writing 1 to it.

- Receive start interrupt

  The interrupt event RXSTIF (USPI_PROTSTS[3]) is set after the start of the first data bit of a receive data word. It can be cleared only by writing 1 to it.

- Receive end interrupt

  The interrupt event RXENDIF (USPI_PROTSTS[4]) is set after the start of the last data bit of a receive data word. It can be cleared only by writing 1 to it.

Protocol-Related Interrupts

- SPI slave select interrupt

  In SPI Slave mode, there are slave select active and in-active interrupt flags, SSACTIF (USPI_PROTSTS[9]) and SSINAIF (USPI_PROTSTS[8]), will be set to 1 when SLAVE (USPI_PROTCTL[0]) is set to 1 and Slave senses the slave select signal active or inactive. The SPI controller will issue an interrupt if SSACTIEN (USPI_PROTIEN[1]) or SSINAIEN (USPI_PROTIEN[0]), is set to 1. Because the internal slave select signal in SPI function is active high, the ININV (USPI_CTLIN0[2]) can be used for inverting the slave select signal comes from an active low device.

- Slave time-out interrupt

  In SPI Slave mode, there is Slave time-out function for user to know that there is no serial clock input during the period of one word transaction. The Slave time-out function uses the timing measurement counter for the calculation of Slave time-out period which is defined by SLVTOCNT (USPI_PROTCTL[25:16]). TMCNTSRC (USPI_BRGEN[5]) can be used for clock frequency selection of timing measurement counter to calculate the Slave time-out period.

  When the timing measurement counter is enabled by TMCNTEN (USPI_BRGEN[4]) and the setting value of SLVTOCNT (USPI_PROTCTL[25:16]) is not 0 in SPI Slave mode, the timing measurement counter will start counting after the first input serial clock of each received word data. This counter will be reset while receiving the following input serial clock and then keep counting. Finally, the timing measurement counter will be cleared and stopped after the finish of the current word transaction. If the value of the time-out counter is equal to or greater than the value of SLVTOCNT (USPI_PROTCTL[25:16]) before one word transaction is done, the Slave time-out interrupt event occurs and the SLVTOIF (USPI_PROTSTS[5]) will be set to 1.

Buffer-Related Interrupts

The buffer-related interrupts are available if there is transmit/receive buffer in USCI controller.

- Receive buffer overrun interrupt

  If there is receive buffer overrun event, RXOVIF (USPI_BUFSTS[3]) will be set as 1. It can be cleared by write 1 into it.

- Transmit buffer under-run interrupt

  If there is transmit buffer under-run event, TXUDRIF (USPI_BUFSTS[11]) will be set as 1. It can be cleared by write 1 into it.

6.27.5.11 Timing Diagram

The slave select signal of USCI SPI protocol is active high by default, and it can be inverted by CTLOINV (USPI_LINECTL[7]) setting.

The idle state of serial bus clock and the serial bus clock edge used for transmit/receive data can be configured by setting SCLKMODE (USPI_PROTCTL[7:6]). The bit length of a transaction word data is determined by DWIDTH (USPI_LINECTL[11:8]), and data bit transfer sequence is determined by LSB (USPI_LINECTL[0]). Four SPI timing diagrams for Master/Slave operations and the related settings

are shown below.

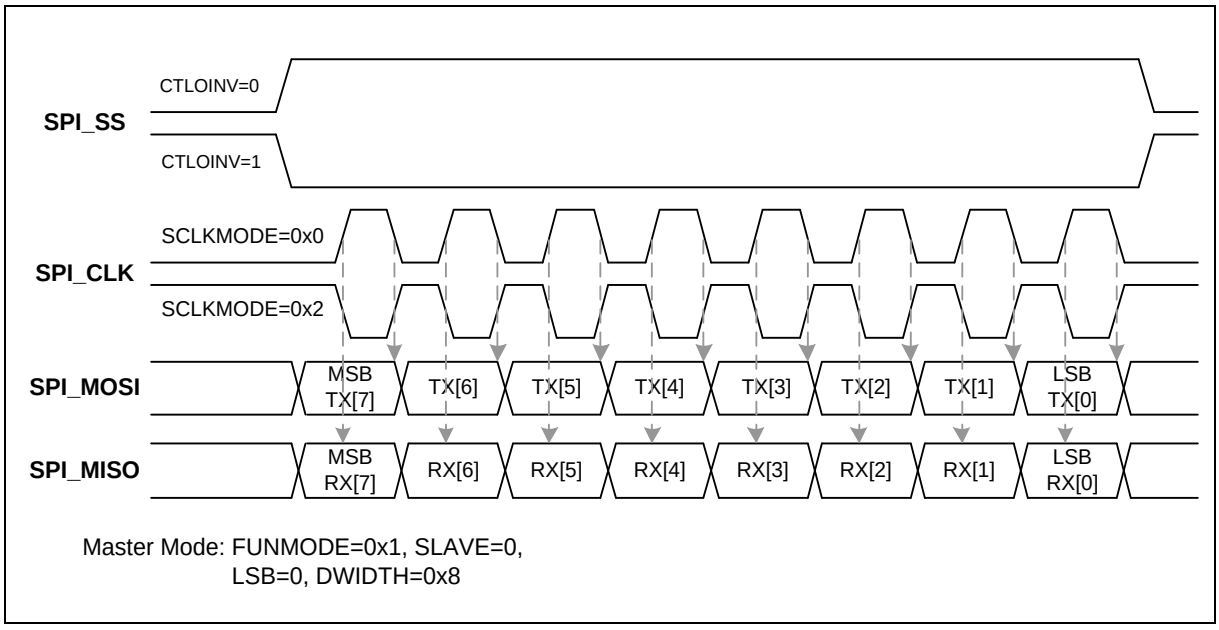


Figure 6.27-16 SPI Timing in Master Mode

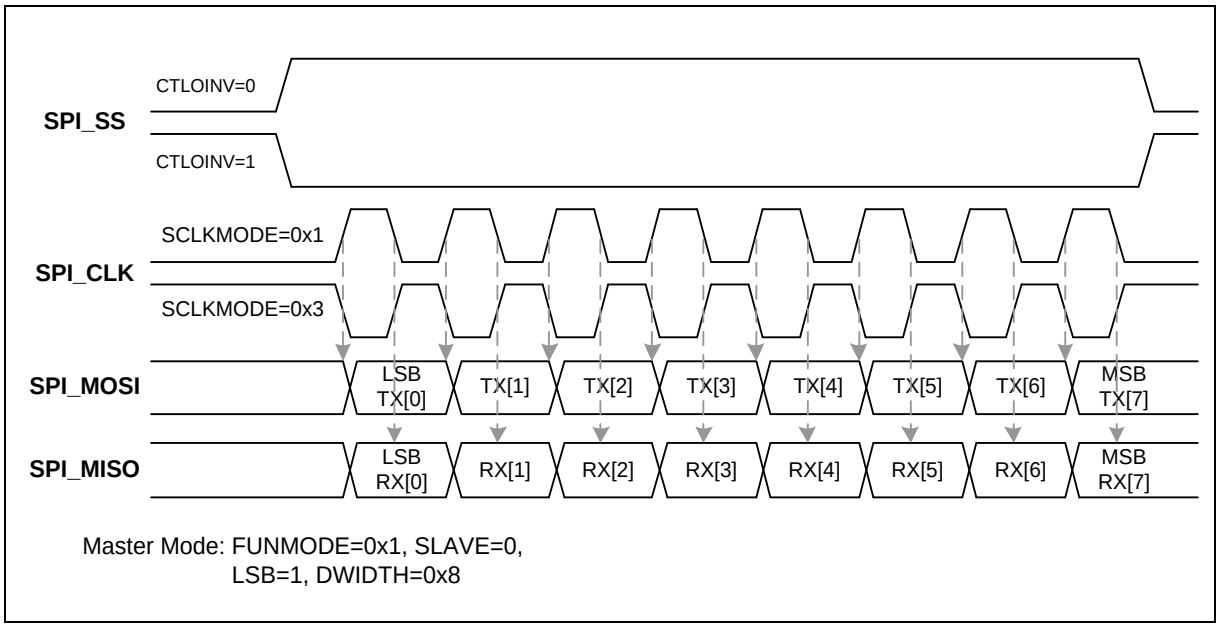


Figure 6.27-17 SPI Timing in Master Mode (Alternate Phase of Serial Bus Clock)

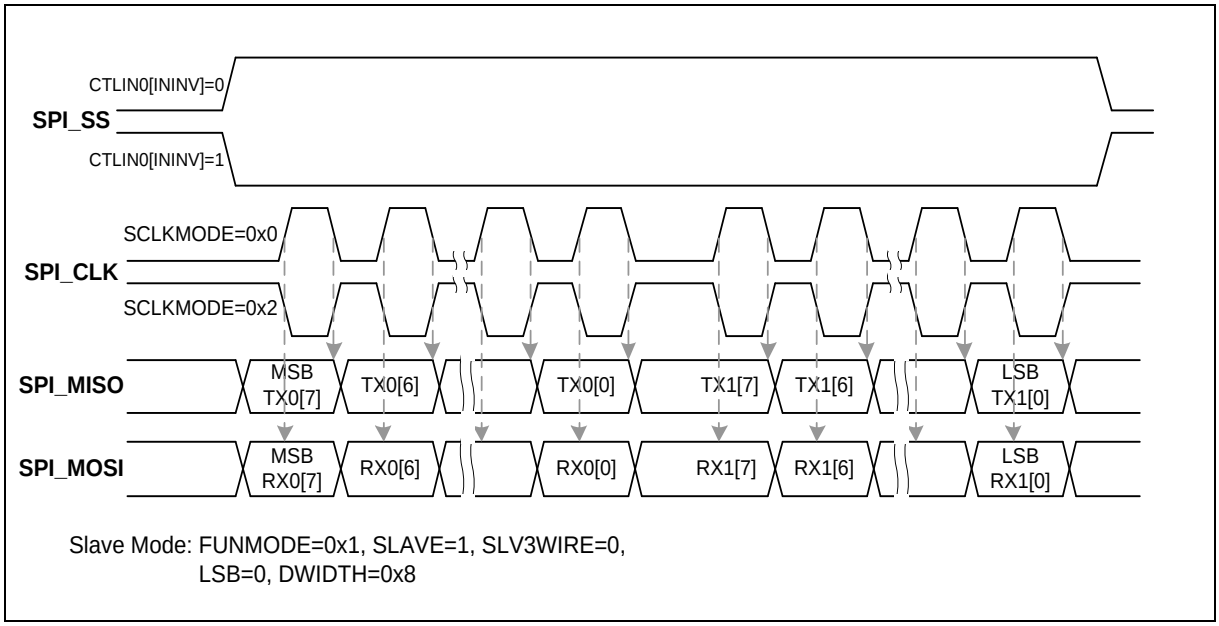


Figure 6.27-18 SPI Timing in Slave Mode

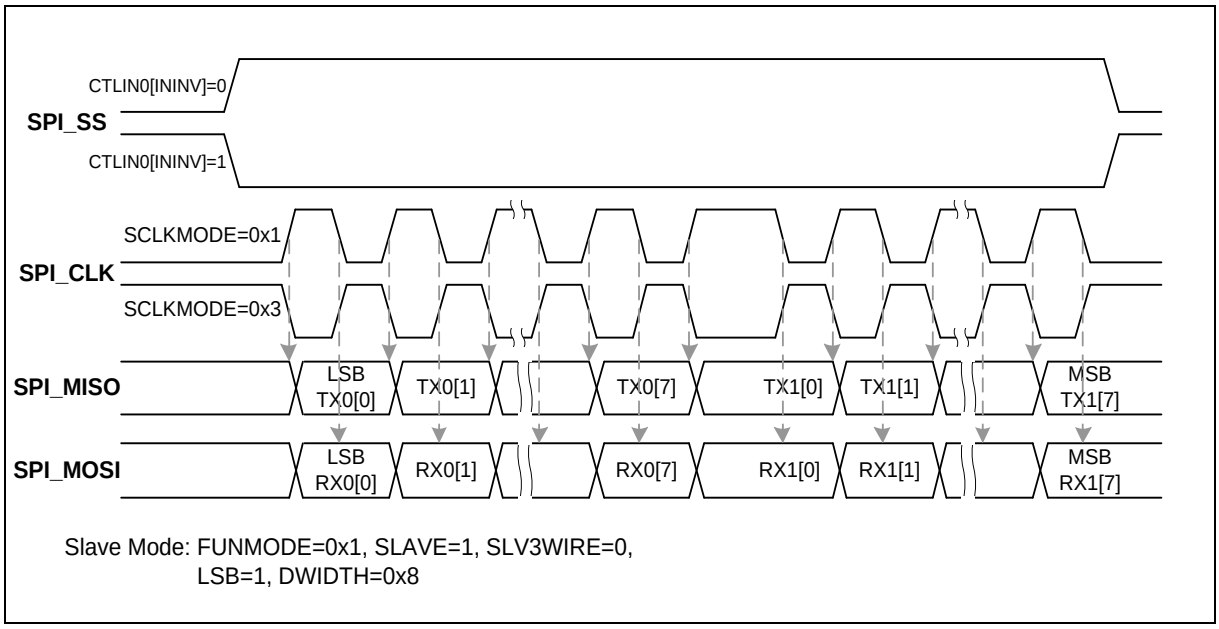


Figure 6.27-19 SPI Timing in Slave Mode (Alternate Phase of Serial Bus Clock)

6.27.5.12 Programming Flow

This section describes the programming flow for USCI SPI data transfer.

For Master mode:

1. Enable USCI peripheral clock by setting CLK_APBCLK1 register.
2. Configure user-specified pins as USCI function pins by setting corresponding multiple function control registers.
3. Set FUNMODE (USPI_CTL[2:0]) to 1 to select SPI mode.

4. Set USPI_BRGEN register to determine the SPI bus clock frequency.
5. According to the requirements of user's application, configure the settings as follows.
   - CTLOINV (USPI_LINECTL[7]): If the slave selection signal is active low, set this bit to 1; otherwise, set it to 0.
   - DWIDTH (USPI_LINECTL[11:8]): Data width setting.
   - LSB (USPI_LINECTL[0]): LSB first or MSB first.
   - TSMSEL (USPI_PROTCTL[14:12]): Full-duplex SPI transfer or one channel half-duplex SPI transfer.
   - SCLKMODE (USPI_PROTCTL[7:6]): Determine the clock timing.
   - AUTOSS (USPI_PROTCTL[3]): Enable automatic slave select function or not.
   - SLAVE (USPI_PROTCTL[0]): Set to 0 for Master mode.
   - Set PROTEN (USPI_PROTCTL[31]) to 1 to enable SPI protocol.
6. If automatic slave select function is disabled (AUTOSS=0), set SS (USPI_PROTCTL[2]) to 1 before data transfer; set SS to 0 to inactivate the slave selection signal by user's application.
7. Write USPI_TXDAT register to trigger SPI transfer. In half-duplex SPI transfer, the data pin direction is determined by PORTDIR (USPI_TXDAT[16]) setting.
8. User can get the received data by reading USPI_RXDAT register as long as RXEMPTY (USPI_BUFSTS[0]) is 0. The SPI data transfer can be triggered by writing USPI_TXDAT register as long as TXFULL (USPI_BUFSTS[9]) is 0.

For Slave mode:

1. Enable USCI peripheral clock by setting CLK_APBCLK1 register.
2. Configure user-specified pins as USCI function pins by setting corresponding multiple function control registers.
3. Set FUNMODE (USPI_CTL[2:0]) to 1 to select SPI mode.
4. According to the requirements of user's application, configure the settings as follows.
   - ININV (USPI_CTLIN0[2]): If the slave selection signal is active low, set this bit to 1; otherwise, set it to 0.
   - DWIDTH (USPI_LINECTL[11:8]): Data width setting.
   - LSB (USPI_LINECTL[0]): LSB first or MSB first.
   - TSMSEL (USPI_PROTCTL[14:12]): Full-duplex SPI transfer or one channel half-duplex SPI transfer.
   - SCLKMODE (USPI_PROTCTL[7:6]): Determine the clock timing.
   - SLAVE (USPI_PROTCTL[0]): Set to 1 for Slave mode.
5. Set PROTEN (USPI_PROTCTL[31]) to 1 to enable SPI protocol.
6. Write USPI_TXDAT register for transmission. In half-duplex SPI transfer, the data pin direction is determined by PORTDIR (USPI_TXDAT[16]) setting.
7. User can get the received data by reading USPI_RXDAT register as long as RXEMPTY (USPI_BUFSTS[0]) is 0. The next datum for transmission can be written to USPI_TXDAT register as long as TXFULL (USPI_BUFSTS[9]) is 0.

6.27.5.13 Wake-up Function

The USCI Controller in SPI mode supports wake-up system function. The wake-up source in SPI

protocol is the transition of input slave select signal.

6.27.6 Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
USCI_SPI Base Address: **USPIn_BA = 0x400D_0000 + (0x1000 * n)** **n= 0, 1** **USCI_SPI non-secure base address is USPIn_BA + 0x1000_0000.**				
USPI_CTL	USPIn_BA+0x00	R/W	USCI Control Register	0x0000_0000
USPI_INTEN	USPIn_BA+0x04	R/W	USCI Interrupt Enable Register	0x0000_0000
USPI_BRGEN	USPIn_BA+0x08	R/W	USCI Baud Rate Generator Register	0x0000_3C00
USPI_DATIN0	USPIn_BA+0x10	R/W	USCI Input Data Signal Configuration Register 0	0x0000_0000
USPI_CTLIN0	USPIn_BA+0x20	R/W	USCI Input Control Signal Configuration Register 0	0x0000_0000
USPI_CLKIN	USPIn_BA+0x28	R/W	USCI Input Clock Signal Configuration Register	0x0000_0000
USPI_LINECTL	USPIn_BA+0x2C	R/W	USCI Line Control Register	0x0000_0000
USPI_TXDAT	USPIn_BA+0x30	W	USCI Transmit Data Register	0x0000_0000
USPI_RXDAT	USPIn_BA+0x34	R	USCI Receive Data Register	0x0000_0000
USPI_BUFCTL	USPIn_BA+0x38	R/W	USCI Transmit/Receive Buffer Control Register	0x0000_0000
USPI_BUFSTS	USPIn_BA+0x3C	R/W	USCI Transmit/Receive Buffer Status Register	0x0000_0101
USPI_PDMACTL	USPIn_BA+0x40	R/W	USCI PDMA Control Register	0x0000_0000
USPI_WKCTL	USPIn_BA+0x54	R/W	USCI Wake-up Control Register	0x0000_0000
USPI_WKSTS	USPIn_BA+0x58	R/W	USCI Wake-up Status Register	0x0000_0000
USPI_PROTCTL	USPIn_BA+0x5C	R/W	USCI Protocol Control Register	0x0000_0300
USPI_PROTIEN	USPIn_BA+0x60	R/W	USCI Protocol Interrupt Enable Register	0x0000_0000
USPI_PROTSTS	USPIn_BA+0x64	R/W	USCI Protocol Status Register	0x0000_0000

6.27.7 Register Description

USCI Control Register (USPI_CTL)

Register	Offset	R/W	Description	Reset Value
USPI_CTL	USPIn_BA+0x00	R/W	USCI Control Register	0x0000_0000

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

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2:0]	**FUNMODE**	**Function Mode** This bit field selects the protocol for this USCI controller. Selecting a protocol that is not available or a reserved combination disables the USCI. When switching between two protocols, the USCI has to be disabled before selecting a new protocol. Simultaneously, the USCI will be reset when user write 000 to FUNMODE. 000 = The USCI is disabled. All protocol related state machines are set to idle state. 001 = The SPI protocol is selected. 010 = The UART protocol is selected. 100 = The I^2C protocol is selected. **Note:** Other bit combinations are reserved.

USCI Interrupt Enable Register (USPI_INTEN)

Register	Offset	R/W	Description	Reset Value
USPI_INTEN	USPIn_BA+0x04	R/W	USCI Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			**RXENDIEN**	**RXSTIEN**	**TXENDIEN**	**TXSTIEN**	**Reserved**

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**RXENDIEN**	**Receive End Interrupt Enable Bit** This bit enables the interrupt generation in case of a receive finish event. 0 = The receive end interrupt Disabled. 1 = The receive end interrupt Enabled.
[3]	**RXSTIEN**	**Receive Start Interrupt Enable Bit** This bit enables the interrupt generation in case of a receive start event. 0 = The receive start interrupt Disabled. 1 = The receive start interrupt Enabled.
[2]	**TXENDIEN**	**Transmit End Interrupt Enable Bit** This bit enables the interrupt generation in case of a transmit finish event. 0 = The transmit finish interrupt Disabled. 1 = The transmit finish interrupt Enabled.
[1]	**TXSTIEN**	**Transmit Start Interrupt Enable Bit** This bit enables the interrupt generation in case of a transmit start event. 0 = The transmit start interrupt Disabled. 1 = The transmit start interrupt Enabled.
[0]	**Reserved**	Reserved.

USCI Baud Rate Generator Register (USPI_BRGEN)

Register	Offset	R/W	Description	Reset Value
USPI_BRGEN	USPIn_BA+0x08	R/W	USCI Baud Rate Generator Register	0x0000_3C00

31	30	29	28	27	26	25	24
Reserved						**CLKDIV**	
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CLKDIV							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**TMCNTSRC**	**TMCNTEN**	**SPCLKSEL**		**PTCLKSEL**	**RCLKSEL**

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25:16]	**CLKDIV**	**Clock Divider** This bit field defines the ratio between the protocol clock frequency f_{PROT_CLK} and the clock divider frequency f_{DIV_CLK} ($f_{DIV\_CLK} = f_{PROT\_CLK} / (CLKDIV+1)$).
[15:6]	**Reserved**	Reserved.
[5]	**TMCNTSRC**	**Time Measurement Counter Clock Source Selection** 0 = Time measurement counter with f_{PROT_CLK}. 1 = Time measurement counter with f_{DIV_CLK}.
[4]	**TMCNTEN**	**Time Measurement Counter Enable Bit** This bit enables the 10-bit timing measurement counter. 0 = Time measurement counter Disabled. 1 = Time measurement counter Enabled.
[3:2]	**SPCLKSEL**	**Sample Clock Source Selection** This bit field used for the clock source selection of sample clock (f_{SAMP_CLK}) for the protocol processor. 00 = f_{DIV_CLK}. 01 = f_{PROT_CLK}. 10 = f_{SCLK}. 11 = f_{REF_CLK}.
[1]	**PTCLKSEL**	**Protocol Clock Source Selection** This bit selects the source of protocol clock (f_{PROT_CLK}). 0 = Reference clock f_{REF_CLK}. 1 = f_{REF_CLK2} (its frequency is half of f_{REF_CLK}).
[0]	**RCLKSEL**	**Reference Clock Source Selection** This bit selects the source of reference clock (f_{REF_CLK}).

		0 = Peripheral device clock $f_{PCLK.}$ 1 = Reserved.

USCI Input Data Signal Configuration (USPI_DATIN0)

Register	Offset	R/W	Description	Reset Value
USPI_DATIN0	USPIn_BA+0x10	R/W	USCI Input Data Signal Configuration Register 0	0x0000_0000

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

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**ININV**	**Input Signal Inverse Selection** This bit defines the inverter enable of the input asynchronous signal. 0 = The un-synchronized input signal will not be inverted. 1 = The un-synchronized input signal will be inverted. **Note:** In SPI protocol, it is suggested this bit should be set as 0.
[1]	**Reserved**	Reserved.
[0]	**SYNCSEL**	**Input Signal Synchronization Selection** This bit selects if the un-synchronized input signal (with optionally inverted) or the synchronized (and optionally filtered) signal can be used as input for the data shift unit. 0 = The un-synchronized signal can be taken as input for the data shift unit. 1 = The synchronized signal can be taken as input for the data shift unit. **Note:** In SPI protocol, it is suggested this bit should be set as 0.

USCI Input Control Signal Configuration (USPI_CTLIN0)

Register	Offset	R/W	Description	Reset Value
USPI_CTLIN0	USPIn_BA+0x20	R/W	USCI Input Control Signal Configuration Register 0	0x0000_0000

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

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**ININV**	**Input Signal Inverse Selection** This bit defines the inverter enable of the input asynchronous signal. 0 = The un-synchronized input signal will not be inverted. 1 = The un-synchronized input signal will be inverted.
[1]	**Reserved**	Reserved.
[0]	**SYNCSEL**	**Input Synchronization Signal Selection** This bit selects if the un-synchronized input signal (with optionally inverted) or the synchronized (and optionally filtered) signal can be used as input for the data shift unit. 0 = The un-synchronized signal can be taken as input for the data shift unit. 1 = The synchronized signal can be taken as input for the data shift unit. **Note:** In SPI protocol, it is suggested this bit should be set as 0.

USCI Input Clock Signal Configuration (USPI_CLKIN)

Register	Offset	R/W	Description	Reset Value
USPI_CLKIN	USPIn_BA+0x28	R/W	USCI Input Clock Signal Configuration Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**SYNCSEL**	**Input Synchronization Signal Selection** This bit selects if the un-synchronized input signal or the synchronized (and optionally filtered) signal can be used as input for the data shift unit. 0 = The un-synchronized signal can be taken as input for the data shift unit. 1 = The synchronized signal can be taken as input for the data shift unit. **Note:** In SPI protocol, it is suggested this bit should be set as 0.

USCI Line Control Register (USPI_LINECTL)

Register	Offset	R/W	Description	Reset Value
USPI_LINECTL	USPIn_BA+0x2C	R/W	USCI Line Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**DWIDTH**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CTLOINV	**Reserved**	**DATOINV**	**Reserved**				**LSB**

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:8]	**DWIDTH**	**Word Length of Transmission** This bit field defines the data word length (amount of bits) for reception and transmission. The data word is always right-aligned in the data buffer. USCI support word length from 4 to 16 bits. 0x0: The data word contains 16 bits located at bit positions [15:0]. 0x1: Reserved. 0x2: Reserved. 0x3: Reserved. 0x4: The data word contains 4 bits located at bit positions [3:0]. 0x5: The data word contains 5 bits located at bit positions [4:0]. ... 0xF: The data word contains 15 bits located at bit positions [14:0].
[7]	**CTLOINV**	**Control Signal Output Inverse Selection** This bit defines the relation between the internal control signal and the output control signal. 0 = No effect. 1 = The control signal will be inverted before its output. **Note:** The control signal has different definitions in different protocol. In SPI protocol, the control signal means slave select signal.
[6]	**Reserved**	Reserved.
[5]	**DATOINV**	**Data Output Inverse Selection** This bit defines the relation between the internal shift data value and the output data signal of USCIx_DAT0/1 pin. 0 = Data output level is not inverted. 1 = Data output level is inverted.
[4:1]	**Reserved**	Reserved.

[0]	**LSB**	**LSB First Transmission Selection** 0 = The MSB, which bit of transmit/receive data buffer depends on the setting of DWIDTH, is transmitted/received first. 1 = The LSB, the bit 0 of data buffer, will be transmitted/received first.

USCI Transmit Data Register (USPI_TXDAT)

Register	Offset	R/W	Description	Reset Value
USPI_TXDAT	USPIn_BA+0x30	W	USCI Transmit Data Register	0x0000_0000

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

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**PORTDIR**	**Port Direction Control** This bit field is only available while USCI operates in SPI protocol (FUNMODE = 0x1) with half-duplex transfer. It is used to define the direction of the data port pin. When software writes USPI_TXDAT register, the transmit data and its port direction are settled simultaneously. 0 = The data pin is configured as output mode. 1 = The data pin is configured as input mode.
[15:0]	**TXDAT**	**Transmit Data** Software can use this bit field to write 16-bit transmit data for transmission. In order to avoid overwriting the transmit data, user have to check TXEMPTY (USPI_BUFSTS[8]) status before writing transmit data into this bit field.

USCI Receive Data Register (USPI_RXDAT)

Register	Offset	R/W	Description	Reset Value
USPI_RXDAT	USPIn_BA+0x34	R	USCI Receive Data Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**RXDAT**	**Received Data** This bit field monitors the received data which stored in receive data buffer.

USCI Transmit/Receive Buffer Control Register (USPI_BUFCTL)

Register	Offset	R/W	Description	Reset Value
USPI_BUFCTL	USPIn_BA+0x38	R/W	USCI Transmit/Receive Buffer Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						RXRST	TXRST
15	14	13	12	11	10	9	8
RXCLR	RXOVIEN	Reserved					
7	6	5	4	3	2	1	0
TXCLR	TXUDRIEN	Reserved					

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**RXRST**	**Receive Reset** 0 = No effect. 1 = Reset the receive-related counters, state machine, and the content of receive shift register and data buffer. **Note:** It is cleared automatically after one PCLK cycle.
[16]	**TXRST**	**Transmit Reset** 0 = No effect. 1 = Reset the transmit-related counters, state machine, and the content of transmit shift register and data buffer**.** **Note1:** It is cleared automatically after one PCLK cycle. **Note2:** Write 1 to this bit will set the output data pin to zero if USPI_PROTCTL[28]=0.
[15]	**RXCLR**	**Clear Receive Buffer** 0 = No effect. 1 = The receive buffer is cleared. Should only be used while the buffer is not taking part in data traffic. **Note:** It is cleared automatically after one PCLK cycle.
[14]	**RXOVIEN**	**Receive Buffer Overrun Interrupt Enable Bit** 0 = Receive overrun interrupt Disabled. 1 = Receive overrun interrupt Enabled.
[13:8]	**Reserved**	Reserved.
[7]	**TXCLR**	**Clear Transmit Buffer** 0 = No effect. 1 = The transmit buffer is cleared. Should only be used while the buffer is not taking part in data traffic. **Note:** It is cleared automatically after one PCLK cycle.

[6]	**TXUDRIEN**	**Slave Transmit Under-run Interrupt Enable Bit** 0 = Transmit under-run interrupt Disabled. 1 = Transmit under-run interrupt Enabled.
[5:0]	**Reserved**	Reserved.

USCI Transmit/Receive Buffer Status Register (USPI_BUFSTS)

Register	Offset	R/W	Description	Reset Value
USPI_BUFSTS	USPIn_BA+0x3C	R/W	USCI Transmit/Receive Buffer Status Register	0x0000_0101

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**TXUDRIF**	**Reserved**	**TXFULL**	**TXEMPTY**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**RXOVIF**	**Reserved**	**RXFULL**	**RXEMPTY**

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11]	**TXUDRIF**	**Transmit Buffer Under-run Interrupt Status** This bit indicates that a transmit buffer under-run event has been detected. If enabled by TXUDRIEN (USPI_BUFCTL[6]), the corresponding interrupt request is activated. It is cleared by software writes 1 to this bit 0 = A transmit buffer under-run event has not been detected. 1 = A transmit buffer under-run event has been detected.
[10]	**Reserved**	Reserved.
[9]	**TXFULL**	**Transmit Buffer Full Indicator (Read Only)** 0 = Transmit buffer is not full. 1 = Transmit buffer is full.
[8]	**TXEMPTY**	**Transmit Buffer Empty Indicator (Read Only)** 0 = Transmit buffer is not empty. 1 = Transmit buffer is empty and available for the next transmission datum.
[7:4]	**Reserved**	Reserved.
[3]	**RXOVIF**	**Receive Buffer Over-run Interrupt Status** This bit indicates that a receive buffer overrun event has been detected. If RXOVIEN (USPI_BUFCTL[14]) is enabled, the corresponding interrupt request is activated. It is cleared by software writes 1 to this bit. 0 = A receive buffer overrun event has not been detected. 1 = A receive buffer overrun event has been detected.
[2]	**Reserved**	Reserved.
[1]	**RXFULL**	**Receive Buffer Full Indicator (Read Only)** 0 = Receive buffer is not full. 1 = Receive buffer is full.

[0]	**RXEMPTY**	**Receive Buffer Empty Indicator (Read Only)** 0 = Receive buffer is not empty. 1 = Receive buffer is empty.

USCI PDMA Control Register (USPI_PDMACTL)

Register	Offset	R/W	Description	Reset Value
USPI_PDMACTL	USPIn_BA+0x40	R/W	USCI PDMA Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**PDMAEN**	**RXPDMAEN**	**TXPDMAEN**	**PDMARST**

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**PDMAEN**	**PDMA Mode Enable Bit** 0 = PDMA function Disabled. 1 = PDMA function Enabled.
[2]	**RXPDMAEN**	**PDMA Receive Channel Available** 0 = Receive PDMA function Disabled. 1 = Receive PDMA function Enabled.
[1]	**TXPDMAEN**	**PDMA Transmit Channel Available** 0 = Transmit PDMA function Disabled. 1 = Transmit PDMA function Enabled. **Note1:** In SPI Master mode with full duplex transfer, if both TX and RX PDMA functions are enabled, RX PDMA function cannot be enabled prior to TX PDMA function. User can enable TX PDMA function firstly or enable both functions simultaneously. **Note2:** In SPI Master mode with full duplex transfer, if both TX and RX PDMA functions are enabled, TX PDMA function cannot be disabled prior to RX PDMA function. User can disable RX PDMA function firstly or disable both functions simultaneously.
[0]	**PDMARST**	**PDMA Reset** 0 = No effect. 1 = Reset the USCI's PDMA control logic. This bit will be cleared to 0 automatically.

USCI Wake-up Control Register (USPI_WKCTL)

Register	Offset	R/W	Description	Reset Value
USPI_WKCTL	USPIn_BA+0x54	R/W	USCI Wake-up Control Register	0x0000_0000

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

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**PDBOPT**	**Power Down Blocking Option** 0 = If user attempts to enter Power-down mode by executing WFI while the protocol is in transferring, MCU will stop the transfer and enter Power-down mode immediately. 1 = If user attempts to enter Power-down mode by executing WFI while the protocol is in transferring, the on-going transfer will not be stopped and MCU will enter idle mode immediately.
[1]	**Reserved**	Reserved.
[0]	**WKEN**	**Wake-up Enable Bit** 0 = Wake-up function Disabled. 1 = Wake-up function Enabled.

USCI Wake-up Status Register (USPI_WKSTS)

Register	Offset	R/W	Description	Reset Value
USPI_WKSTS	USPIn_BA+0x58	R/W	USCI Wake-up Status Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**WKF**	**Wake-up Flag** When chip is woken up from Power-down mode, this bit is set to 1. Software can write 1 to clear this bit.

USCI Protocol Control Register – SPI (USPI_PROTCTL)

Register	Offset	R/W	Description	Reset Value
USPI_PROTCTL	USPIn_BA+0x5C	R/W	USCI Protocol Control Register	0x0000_0300

31	30	29	28	27	26	25	24
PROTEN	Reserved		TXUDRPOL	Reserved		SLVTOCNT	
23	22	21	20	19	18	17	16
SLVTOCNT							
15	14	13	12	11	10	9	8
Reserved	TSMSEL			SUSPITV			
7	6	5	4	3	2	1	0
SCLKMODE		Reserved		AUTOSS	SS	SLV3WIRE	SLAVE

Bits	Description	
[31]	**PROTEN**	**SPI Protocol Enable Bit** 0 = SPI Protocol Disabled. 1 = SPI Protocol Enabled.
[30:29]	**Reserved**	Reserved.
[28]	**TXUDRPOL**	**Transmit Under-run Data Polarity (Slave Only)** This bit defines the transmitting data level when no data is available for transferring. 0 = The output data level is 0 if TX under-run event occurs. 1 = The output data level is 1 if TX under-run event occurs.
[27:26]	**Reserved**	Reserved.
[25:16]	**SLVTOCNT**	**Slave Mode Time-out Period (Slave Only)** In Slave mode, this bit field is used for Slave time-out period. This bit field indicates how many clock periods (selected by TMCNTSRC, USPI_BRGEN[5]) between the two edges of input SCLK will assert the Slave time-out event. Writing 0x0 into this bit field will disable the Slave time-out function. Example: Assume SLVTOCNT is 0x0A and TMCNTSRC (USPI_BRGEN[5]) is 1, it means the time-out event will occur if the state of SPI bus clock pin is not changed more than (10+1) periods of f_{DIV_CLK}.
[15]	**Reserved**	Reserved.
[14:12]	**TSMSEL**	**Transmit Data Mode Selection** This bit field describes how receive and transmit data is shifted in and out. TSMSEL = 000b: Full-duplex SPI. TSMSEL = 100b: Half-duplex SPI. Others = Reserved. **Note:** Changing the value of this bit field will produce the TXRST and RXRST to clear the TX/RX data buffer automatically.
[11:8]	**SUSPITV**	**Suspend Interval (Master Only)**

		This bit field provides the configurable suspend interval between two successive transmit/receive transaction in a transfer. The definition of the suspend interval is the interval between the last clock edge of the preceding transaction word and the first clock edge of the following transaction word. The default value is 0x3. The period of the suspend interval is obtained according to the following equation. (SUSPITV[3:0] + 0.5) * period of SPI_CLK clock cycle Example: SUSPITV = 0x0 … 0.5 SPI_CLK clock cycle. SUSPITV = 0x1 … 1.5 SPI_CLK clock cycle. …… SUSPITV = 0xE … 14.5 SPI_CLK clock cycle. SUSPITV = 0xF … 15.5 SPI_CLK clock cycle.
[7:6]	**SCLKMODE**	**Serial Bus Clock Mode** This bit field defines the SCLK idle status, data transmit, and data receive edge. 00 = MODE0. The idle state of SPI clock is low level. Data is transmitted with falling edge and received with rising edge. 01 = MODE1. The idle state of SPI clock is low level. Data is transmitted with rising edge and received with falling edge. 10 = MODE2. The idle state of SPI clock is high level. Data is transmitted with rising edge and received with falling edge. 11 = MODE3. The idle state of SPI clock is high level. Data is transmitted with falling edge and received with rising edge.
[5:4]	**Reserved**	Reserved.
[3]	**AUTOSS**	**Automatic Slave Select Function Enable (Master Only)** 0 = Slave select signal will be controlled by the setting value of SS (USPI_PROTCTL[2]) bit. 1 = Slave select signal will be generated automatically. The slave select signal will be asserted by the SPI controller when transmit/receive is started, and will be de-asserted after each transmit/receive is finished.
[2]	**SS**	**Slave Select Control (Master Only)** If AUTOSS bit is cleared, setting this bit to 1 will set the slave select signal to active state, and setting this bit to 0 will set the slave select signal back to inactive state. If the AUTOSS function is enabled (AUTOSS = 1), the setting value of this bit will not affect the current state of slave select signal. **Note:** In SPI protocol, the internal slave select signal is active high.
[1]	**SLV3WIRE**	**Slave 3-wire Mode Selection (Slave Only)** The SPI protocol can work with 3-wire interface (without slave select signal) in Slave mode. 0 = 4-wire bi-direction interface. 1 = 3-wire bi-direction interface.
[0]	**SLAVE**	**Slave Mode Selection** 0 = Master mode. 1 = Slave mode.

USCI Protocol Interrupt Enable Register – SPI (USPI_PROTIEN)

Register	Offset	R/W	Description	Reset Value
USPI_PROTIEN	USPIn_BA+0x60	R/W	USCI Protocol Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**SLVBEIEN**	**SLVTOIEN**	**SSACTIEN**	**SSINAIEN**

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**SLVBEIEN**	**Slave Mode Bit Count Error Interrupt Enable Bit** If data transfer is terminated by slave time-out or slave select inactive event in Slave mode, so that the transmit/receive data bit count does not match the setting of DWIDTH (USPI_LINECTL[11:8]), bit count error event occurs. 0 = The Slave mode bit count error interrupt Disabled. 1 = The Slave mode bit count error interrupt Enabled.
[2]	**SLVTOIEN**	**Slave Time-out Interrupt Enable Bit** In SPI protocol, this bit enables the interrupt generation in case of a Slave time-out event. 0 = The Slave time-out interrupt Disabled. 1 = The Slave time-out interrupt Enabled.
[1]	**SSACTIEN**	**Slave Select Active Interrupt Enable Bit** This bit enables/disables the generation of a slave select interrupt if the slave select changes to active. 0 = Slave select active interrupt generation Disabled. 1 = Slave select active interrupt generation Enabled.
[0]	**SSINAIEN**	**Slave Select Inactive Interrupt Enable Bit** This bit enables/disables the generation of a slave select interrupt if the slave select changes to inactive. 0 = Slave select inactive interrupt generation Disabled. 1 = Slave select inactive interrupt generation Enabled.

USCI Protocol Status Register – SPI (USPI_PROTSTS)

Register	Offset	R/W	Description	Reset Value
USPI_PROTSTS	USPIn_BA+0x64	R/W	USCI Protocol Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					SLVUDR	BUSY	SSLINE
15	14	13	12	11	10	9	8
Reserved						SSACTIF	SSINAIF
7	6	5	4	3	2	1	0
Reserved	SLVBEIF	SLVTOIF	RXENDIF	RXSTIF	TXENDIF	TXSTIF	Reserved

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18]	**SLVUDR**	**Slave Mode Transmit Under-run Status (Read Only)** In Slave mode, if there is no available transmit data in buffer while transmit data shift out caused by input serial bus clock, this status flag will be set to 1. This bit indicates whether the current shift-out data of word transmission is switched to TXUDRPOL (USPI_PROTCTL[28]) or not. 0 = Slave transmit under-run event does not occur. 1 = Slave transmit under-run event occurs.
[17]	**BUSY**	**Busy Status (Read Only)** 0 = SPI is in idle state. 1 = SPI is in busy state. The following listing are the bus busy conditions: k. USPI_PROTCTL[31] = 1 and the TXEMPTY = 0. l. For SPI Master mode, the TXEMPTY = 1 but the current transaction is not finished yet. m. For SPI Slave mode, the USPI_PROTCTL[31] = 1 and there is serial clock input into the SPI core logic when slave select is active. n. For SPI Slave mode, the USPI_PROTCTL[31] = 1 and the transmit buffer or transmit shift register is not empty even if the slave select is inactive.
[16]	**SSLINE**	**Slave Select Line Bus Status (Read Only)** This bit is only available in Slave mode. It used to monitor the current status of the input slave select signal on the bus. 0 = The slave select line status is 0. 1 = The slave select line status is 1.
[15:10]	**Reserved**	Reserved.
[9]	**SSACTIF**	**Slave Select Active Interrupt Flag (Slave Only)** This bit indicates that the internal slave select signal has changed to active. It is cleared by

		software writes one to this bit 0 = The slave select signal has not changed to active. 1 = The slave select signal has changed to active. **Note:** The internal slave select signal is active high.
[8]	**SSINAIF**	**Slave Select Inactive Interrupt Flag (Slave Only)** This bit indicates that the internal slave select signal has changed to inactive. It is cleared by software writes 1 to this bit 0 = The slave select signal has not changed to inactive. 1 = The slave select signal has changed to inactive. **Note:** The internal slave select signal is active high.
[7]	**Reserved**	Reserved.
[6]	**SLVBEIF**	**Slave Bit Count Error Interrupt Flag (Slave Only)** 0 = Slave bit count error event did not occur. 1 = Slave bit count error event occurred. **Note:** It is cleared by software write 1 to this bit.
[5]	**SLVTOIF**	**Slave Time-out Interrupt Flag (Slave Only)** 0 = Slave time-out event did not occur. 1 = Slave time-out event occurred. **Note:** It is cleared by software write 1 to this bit
[4]	**RXENDIF**	**Receive End Interrupt Flag** 0 = Receive end event did not occur. 1 = Receive end event occurred. **Note:** It is cleared by software write 1 to this bit.
[3]	**RXSTIF**	**Receive Start Interrupt Flag** 0 = Receive start event did not occur. 1 = Receive start event occurred. **Note:** It is cleared by software write 1 to this bit.
[2]	**TXENDIF**	**Transmit End Interrupt Flag** 0 = Transmit end event did not occur. 1 = Transmit end event occurred. **Note:** It is cleared by software write 1 to this bit.
[1]	**TXSTIF**	**Transmit Start Interrupt Flag** 0 = Transmit start event did not occur. 1 = Transmit start event occurred. **Note:** It is cleared by software write 1 to this bit.
[0]	**Reserved**	Reserved.

6.28 USCI - I²C Mode

6.28.1 Overview

On I²C bus, data is transferred between a Master and a Slave. Data bits transfer on the SCL and SDA lines are synchronously on a byte-by-byte basis. Each data byte is 8-bit. There is one SCL clock pulse for each data bit with the MSB being transmitted first, and an acknowledge bit follows each transferred byte. Each bit is sampled during the high period of SCL; therefore, the SDA line may be changed only during the low period of SCL and must be held stable during the high period of SCL. A transition on the SDA line while SCL is high is interpreted as a command (START or STOP). Please refer to Figure 6.28-1 for more detailed I²C BUS Timing.

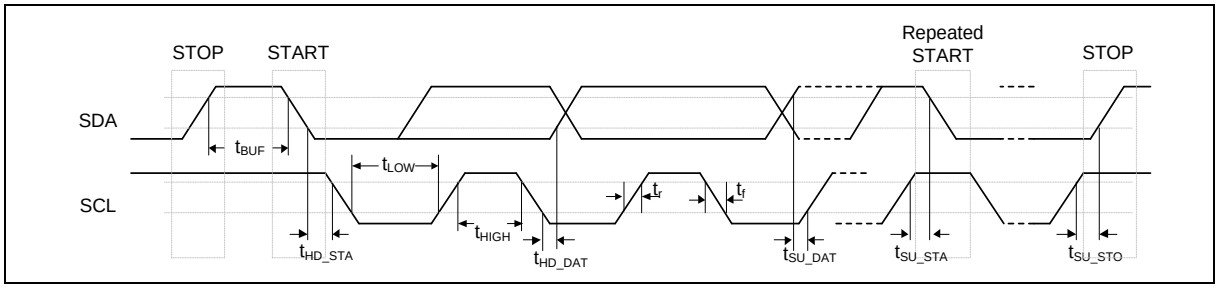


Figure 6.28-1 I²C Bus Timing

The device's on-chip I²C provides the serial interface that meets the I²C bus standard mode specification. The I²C port handles byte transfers autonomously. The I²C mode is selected by FUNMODE (UI2C_CTL [2:0]) = 100B. When enable this port, the USCI interfaces to the I²C bus via two pins: SDA and SCL. When I/O pins are used as I²C ports, user must set the pins function to I²C in advance.

Note: Pull-up resistor is needed for I²C operation because the SDA and SCL are set to open-drain pins when USCI is selected to I²C operation mode.

6.28.2 Features

- Full master and slave device capability
- Supports of 7-bit addressing, as well as 10-bit addressing
- Communication in standard mode (100 kBit/s) or in fast mode (up to 400 kBit/s)
- Supports multi-master bus
- Supports one transmit buffer and two receive buffer for data payload
- Supports 10-bit bus time-out capability
- Supports bus monitor mode.
- Supports Power down wake-up by received 'START' symbol or address match
- Supports setup/hold time programmable
- Supports multiple address recognition (two slave address with mask option)

6.28.3 Block Diagram

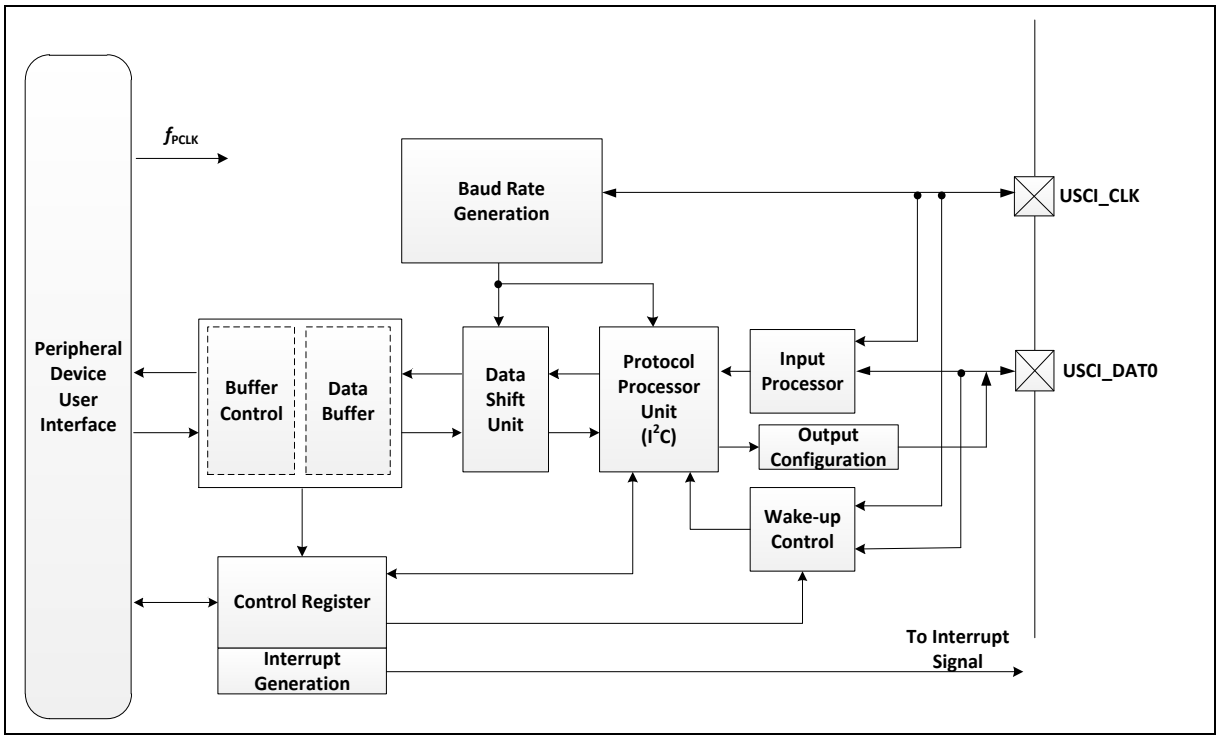


Figure 6.28-2 USCI I²C Mode Block Diagram

6.28.4 Basic Configuration

6.28.4.1 USCI0 I²C Basic Configurations

- Clock Source Configuration
  - Enable USCI0 peripheral clock in USCI0CKEN (CLK_APBCLK1[8]).
  - Enable USCI0_I2C function UI2C_CTL[2:0]) register, UI2C_CTL[2:0]=3'b100
- Reset Configuration
  - Reset USCI0 controller in USCI0RST (SYS_IPRST2[8]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
USCI0	USCI0_CLK	PD.0	MFP3
		PB.12	MFP5
		PA.11	MFP6
		PE.2	MFP7
	USCI0_CTL0	PD.4	MFP3
		PD.14	MFP5
		PC.13	MFP6
		PE.PE.6	MFP7

	USCI0_CTL1	PD.3	MFP3
		PB.15	MFP5
		PA.8	MFP6
		PE.5	MFP7
	USCI0_DAT0	PD.1	MFP3
		PB.13	MFP5
		PA.10	MFP6
		PE.3	MFP7
	USCI0_DAT1	PD.2	MFP3
		PB.14	MFP5
		PA.9	MFP6
		PE.4	MFP7

USCI1I^{2}C Basic Configurations Clock Source Configuration

- Enable USCI1 peripheral clock in USCI1CKEN (CLK_APBCLK1[9]).
- Enable USCI1_I2C function UI2C_CTL[2:0]) register, UI2C_CTL[2:0]=3'b100

- Reset Configuration
  - Reset USCI1 controller in USCI1RST (SYS_IPRST2[9]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
USCI1	USCI1_CLK	PB.8	MFP4
		PD.7, PE.12	MFP6
		PB.1	MFP8
	USCI1_CTL0	PB.PB.10	MFP4
		PD.PD.3, PE.9	MFP6
		PB.PB.5	MFP8
	USCI1_CTL1	PB.9	MFP4
		PD.4, PE.8	MFP6
		PB.4	MFP8
	USCI1_DAT0	PB.7	MFP4
		PD.5, PE.10	MFP6
		PB.2	MFP8
	USCI1_DAT1	PB.6	MFP4
		PD.6, PE.11	MFP6
		PB.3	MFP8

6.28.5 Functional Description

6.28.5.1 START or Repeated START Signal

Figure 6.28-3 shows the typical I²C protocol. Normally, a standard communication consists of four parts:

- START or Repeated START signal generation
- Slave address and R/W bit transfer
- Data transfer
- STOP signal generation

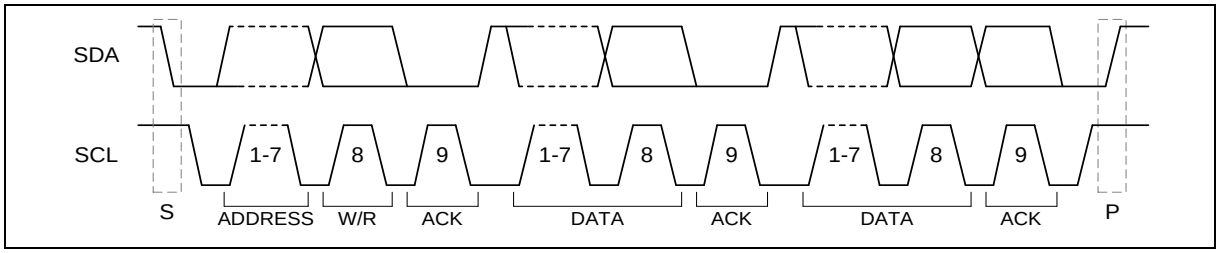


Figure 6.28-3 I²C Protocol

When the bus is free/idle, meaning no master device is engaging the bus (both SCL and SDA lines are high), a master can initiate a transfer by sending a START signal. A START signal, usually referred to as the “S” bit, is defined as a HIGH to LOW transition on the SDA line while SCL is HIGH. The START signal denotes the beginning of a new data transmission.

A Repeated START is not a STOP signal between two START signals and usually referred to as the “Sr” bit. The master uses this method to communicate with another slave or the same slave in a different transfer direction (e.g. from writing to a device to reading from a device) without releasing the bus idle flag.

6.28.5.2 STOP Signal

The master can terminate the communication by generating a STOP signal. A STOP signal, usually referred to as the “P” bit, is defined as a LOW to HIGH transition on the SDA line while SCL is HIGH. The section between STOP and START is called bus free.

Figure 6.28-4 shows the waveform of START, Repeat START and STOP.

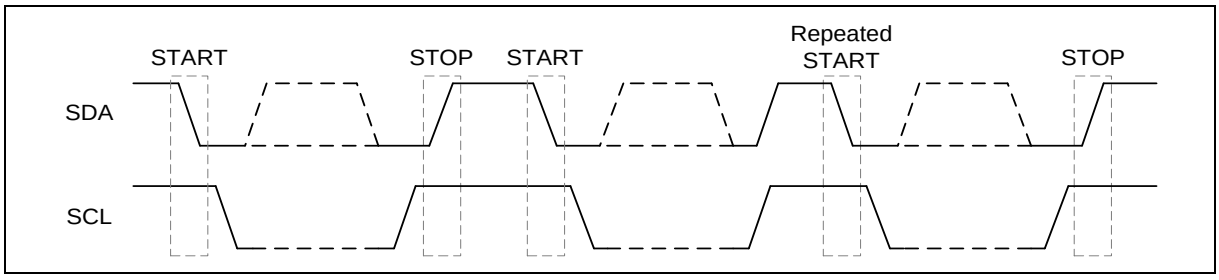


Figure 6.28-4 START and STOP Conditions

6.28.5.3 Slave Address Transfer

After a (Repeated) START condition, the master sends a slave address to identify the target device of the communication. The start address can comprise one or two address bytes (for 7-bit or for 10-bit addressing schemes). After an address byte, a slave sensitive to the transmitted address has to acknowledge the reception.

Therefore, the slave's address can be programmed in the device, where it is compared to the received address. In case of a match, the slave answers with an acknowledge (SDA = 0). Slaves that are not targeted answer with a non-acknowledge (SDA = 1). In addition to the match of the programmed address, another address byte value has to be answered with an acknowledge if the slave is capable to handle the corresponding requests. The address byte 00H indicates a general call address that can be acknowledged.

In order to allow selective acknowledges for the different values of the address byte(s), the following control mechanism is implemented:

- If the GCFUNC bit (UI2C_PROTCTL [0]) is set the I^2C port hardware will respond to General Call address (00H). Clear GC bit to disable general call function.
- The I^2C port is equipped with one device address registers, UI2C_DEVADDRn (n = 0~1). In 7-bit address mode, the first 7 bits of a received first address byte are compared to the programmed slave address (UI2C_DEVADDRn [6:0]). If these bits match, the slave sends an acknowledge.
- For 10 bit addressing mode, if the slave address is programmed to 1111 0XXB, the XX bits are compared to the bits UI2C_DEVADDR [9:8] to check for address match and also sends an acknowledge when ADDR10EN (UI2C_PROTCTL [4]) is set. The slave waits for a second address byte compares it with UI2C_DEVADDR [7:0] and sends an acknowledge accordingly to cover the 10 bit addressing mode. The user has to take care about reserved addresses (refer to I^2C specification for more detailed description). Only the address 1111 0XXB is supported. Under each of these conditions, bit SLASEL (UI2C_PROTSTS [14]) will be set when the addressing delivered a match. This SLASEL (UI2C_PROTSTS [14]) bit is cleared automatically by a (Repeated) START or STOP condition.
- The I^2C port is equipped with multiple address recognition with one address mask registers I2C_ADDRMSKn (n = 0~1). When the bit in the address mask register is set to 1, it means the received corresponding address bit is "Don't care". If the bit is set to 0, it means the received corresponding register bit should be exactly the same as address register.

6.28.5.4 Data Transfer

When a slave receives a correct address with an R/W bit, the data will follow R/W bit specified to transfer. Each transferred byte is followed by an acknowledge bit on the 9th SCL clock cycle. If the slave signals a Not Acknowledge (NACK), the master can generate a STOP signal to abort the data transfer or generate a Repeated START signal and start a new transfer cycle.

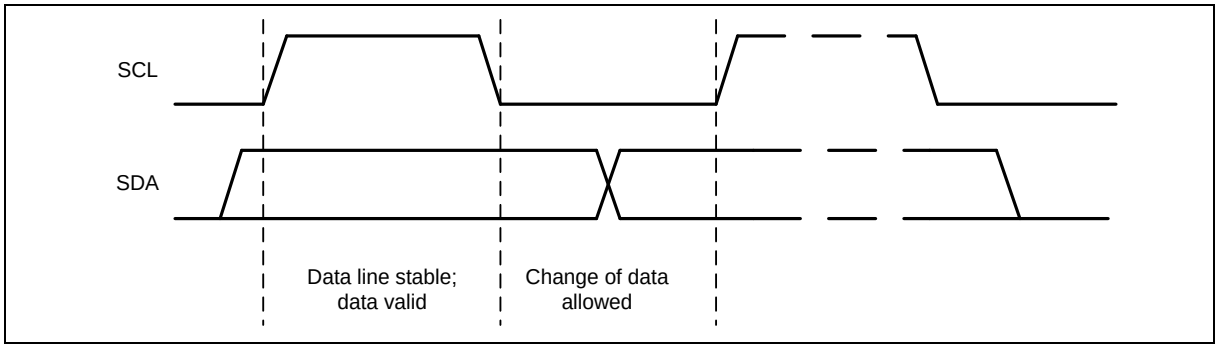


Figure 6.28-5 Bit Transfer on the I^2C Bus

If the master receives data, does Not Acknowledge (NACK) the slave, the slave releases the SDA line for the master to generate a STOP or Repeated START signal.

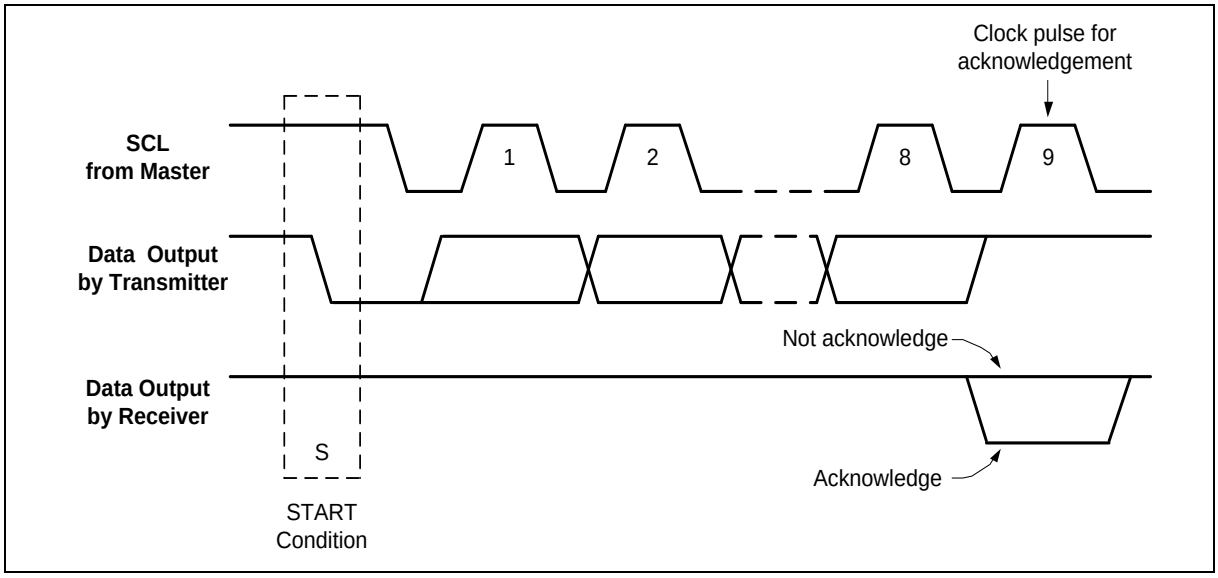


Figure 6.28-6 Acknowledge on the I²C Bus

6.28.5.5 Clock Baud Rate Bits

The data baud rate of I²C is determined by UI2C_BRGEN register when I²C is in Master Mode, and it is not necessary in a Slave mode. In the Slave mode, I²C will automatically synchronize it with any clock frequency from master I²C device. The bits RCLKSEL, SPCLKSEL, PDSCNT, and DSCNT define the baud rate setting:

- RCLKSEL (UI2C_BRGEN [0])

  to define the input frequency f_{REF_CLK}
- SPCLKSEL (UI2C_BRGEN[3:2])

  to define the multiple source of the sample clock f_{SAMP_CLK}
- PDSCNT (UI2C_BRGEN [9:8])

  to define the length of a data sample time (division of f_{REF_CLK} by 1, 2, 3, or 4)
- DSCNT (UI2C_BRGEN [14:10])

  to define the number of data sample time per bit time

  The standard setting is given by RCLKSEL = 0 ($f_{REF\_CLK} = f_{PCLK}$), PTCLKSEL = 0 ($f_{PROT\_CLK} = f_{REF\_CLK}$) and SPCLKSEL = 2'b00 ($f_{SAMP\_CLK} = f_{DIV\_CLK}$). Under these conditions, the baud rate is given by:

$$f_{I2C} = f_{REF\_CLK} \times \frac{1}{\text{CLKDIV} + 1} \times \frac{1}{\text{PDSCNT} + 1} \times \frac{1}{\text{DSCNT} + 1}$$

  In order to generate slower frequencies, additional divide-by-2 stages can be selected by PTCLKSEL = 1 (f_{PROT_CLK} = fREF_CLK2), leading to:

$$f_{I2C} = \frac{f_{REF\_CLK}}{2} \times \frac{1}{\text{CLKDIV} + 1} \times \frac{1}{\text{PDSCNT} + 1} \times \frac{1}{\text{DSCNT} + 1}$$

  If SPCLKSEL = 2'b10 ($f_{SAMP\_CLK} = f_{SCLK}$), and RCLKSEL = 0 (fREF_CLK = f_{PCLK}), PTCLKSEL = 0 ($f_{PROT\_CLK} = f_{REF\_CLK}$). The baud rate is given by:

$$f_{I2C} = f_{REF\_CLK} \times \frac{1}{\text{CLKDIV} + 1} \times \frac{1}{2} \times \frac{1}{\text{PDSCNT} + 1} \times \frac{1}{\text{DSCNT} + 1}$$

6.28.5.6 Byte Stretching

If a device is selected as master/slave transmit mode and should transmit a data byte but the transmit buffer TXDAT does not contain valid data to be transmitted, the device ties down SCL = 0 at the end of the previous acknowledge bit. The waiting period is finished if software writes 1 to PTRG (UI2C_PROTCTL [5]).

6.28.5.7 Multi-master Arbitration

In some applications, there are two or more masters on the same I^{2}C bus to access slaves, and the masters may transmit data simultaneously. The I^{2}C supports multi-master by including collision detection and arbitration to prevent data corruption.

If two masters sometimes initiate I^{2}C command at the same time, the arbitration procedure determines which master wins and can continue with the command. Arbitration is performed on the SDA signal while the SCL signal is high. Each master checks if the SDA signal on the bus corresponds to the generated SDA signal. If the SDA signal on the bus is low but it should be high, then this master has lost arbitration. Master I^{2}C device that has lost arbitration can generate SCL pulses until the byte ends and must then release the bus and go into slave mode. The arbitration procedure can continue until all the data is transferred. This means that in multi-master system each I^{2}C master must monitor the I^{2}C bus for collisions and act accordingly. Figure 6.28-7 describes master1 data and master2 data are compete arbitration.

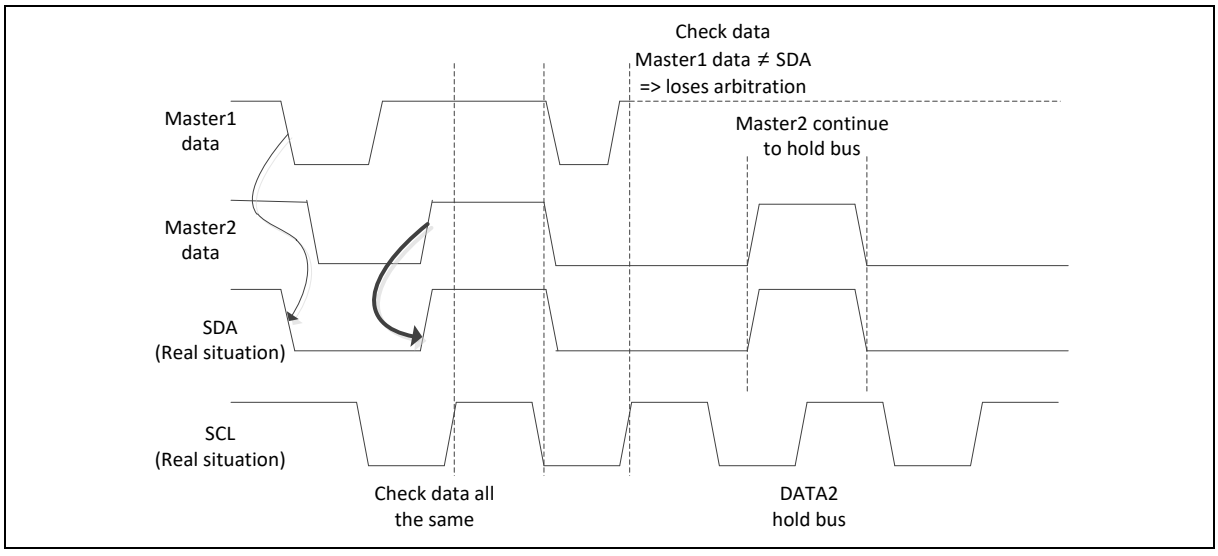


Figure 6.28-7 Arbitration Lost

In this case, during the address and data transmission, the master transmitter checks at the rising edge of SCL for each data bit if the value it is sending is equal to the value read on the SDA line. If yes, master can hold bus continuously. If this is not the case (transmitted value = 1, value read = 0), the master has lost the transmit arbitration. This is indicated by interrupt flag ARBLOIF (UI2C_PROTSTS [11]) and can generate a protocol interrupt if enabled by ARBLOIEN (UI2C_PROTIEN [4]).

When the transmit arbitration has been lost, the software has to initialize the complete frame again, starting with the first address byte together with the START condition for a new master transmit attempt. Arbitration also takes place for the ACK bit. If master arbitration lost and match the device address, then master will turn to slave.

6.28.5.8 Transmission Chain

The I²C bus protocol requiring a kind of in-bit-response during the arbitration phase and while a slave is transmitting, the resulting loop delay of the transmission chain can limit the reachable maximal baud rate, strongly depending on the bus characteristics (bus load, module frequency, etc.).

The shift clock SCL is generated by the master device, output on the wire, then it passes through the input stage and the input filter. Now, the edges can be detected and the SDA data signal can be generated accordingly. The SDA signal passes through the output stage and the wire to the master receiver part. There, it passes through the input stage and the input filter before it is sampled.

This complete loop has to be finished (including all settling times to obtain stable signal levels) before the SCL signal changes again. The delays in this path have to be taken into account for the calculation of the baud rate as a function of f_{PCLK} and f_{PROT_CLK}. We suggest user adopt f_{PCLK}.

6.28.5.9 Non-Acknowledge and Error Conditions

In case of a non-acknowledge (NACKIF (UI2C_PROTSTS [10])) or an error (ERRIF(UI2C_PROTSTS [12])), no further transmission will take place. User software doesn't invalidate the transmit buffer and disable transmissions, before configuring the transmission (by writing TXDAT) again with appropriate values to react on the previous event.

6.28.5.10 I²C Protocol Interrupt Events

The following protocol-related events are generated in I²C mode and can lead to a protocol interrupt.

Please note that the bits in register UI2C_PROTSTS are not all automatically cleared by hardware and have to be cleared by software in order to monitor new incoming events.

- START condition received at a correct position in a frame (STARIF (UI2C_PROTSTS [8]))
- STOP condition transferred at a correct position in a frame (STORIF (UI2C_PROTSTS [9]))
- Master arbitration lost (ARBLOIF (UI2C_PROTSTS [11]))
- Slave read requested (SLAREAD (UI2C_PROTSTS [15]))
- Acknowledge received (ACKIF (UI2C_PROTSTS [13]))
- Non-acknowledge received (NACKIF (UI2C_PROTSTS [10]))
- START condition not at the expected position in a frame (ERRIF (UI2C_PROTSTS [12]))
- STOP condition not at the expected position in a frame (ERRIF (UI2C_PROTSTS [12]))

6.28.5.11 Operating the I2C

To operate the I²C protocol, the following issues have to be considered:

Select I²C Mode

It is recommended to configure all parameters of the I²C that do not change during run time while FUNMODE (UI2C_CTL [2:0]) = 000B. The I²C control flow has to be done while FUNMODE (UI2C_CTL [2:0]) = 000B to avoid unintended edges of the input signals and the I²C mode can be enabled by FUNMODE (UI2C_CTL [2:0]) = 100B afterwards.

Step 1. Set FUNMODE (UI2C_CTL [2:0]) = 000B

Step 2. Set FUNMODE (UI2C_CTL [2:0]) = 100B

Pin Connections

The pins used for SDA and SCL have to be set to open-drain mode by USCI controller to support the wired-AND structure of the I²C bus lines.

Note: The step to enable the alternate output port functions should only be done after the I²C mode is

enabled, to avoided unintended spikes on the output.

Bit Timing Configuration

In standard mode (100 kBit/s) a minimum module frequency of 2 MHz is necessary, whereas in fast mode (400 kBit/s) a minimum of 10 MHz is required. Additionally, if the digital filter stage should be used to eliminate spikes up to 50 ns, a filter frequency of 20 MHz is necessary. There could be an uncertainty in the SCL high phase timing of maximum $1/f_{PROT\_CLK}$ if another I^2C participant lengthens the SCL low phase on the bus. Note that the SCL maximum frequency is $f_{SAMP\_CLK}/2$ and the SPCLKSEL (UI2C_BRGEN [3:2]) must be set to 0 for selecting $f_{SAMP\_CLK} = f_{DIV\_CLK}$.

Data Format Configuration

The data format has to be configured for 8 data bits (DWIDTH (UI2C_LINECTL [11:8]) = 8), and MSB shifted first (LSB (UI2C_LINECTL [0]) = 0). As a result, UI2C_LINECTL has to be set to 0x800.

Control Flow

The on-chip I^2C ports support three operation modes, Master, Slave, and General Call Mode.

In a given application, I^2C port may operate as a master or as a slave. In Slave mode, the I^2C port hardware looks for its own slave address and the general call address. If one of these addresses is detected, and if the slave is willing to receive or transmit data from/to master(by setting the AA bit), acknowledge pulse will be transmitted out on the 9th clock, hence an interrupt is requested on both master and slave devices if interrupt is enabled. When the microcontroller wishes to become the bus master, hardware waits until the bus is free before entering Master mode so that a possible slave action is not be interrupted. If address arbitration is lost in Master mode, I^2C port switches to Slave mode immediately and can detect its own slave address in the same serial transfer.

To control the I^2C bus transfer in each mode, user needs to set UI2C_PROTCTL, UI2C_PROTIEN, TXDAT registers according to current status of UI2C_PROTSTS register. In other words, for each I^2C bus action, user needs to check current status by UI2C_PROTSTS register, and then set UI2C_PROTCTL, UI2C_PROTIEN, TXDAT registers to take bus action. Finally, check the response status by UI2C_PROTSTS.

The bits, STA, STO and AA in UI2C_PROTCTL register are used to control the next state of the I^2C hardware after interrupt signal is cleared. Upon completion of the new action, a new status will be updated in UI2C_PROTSTS register will be set. If the I^2C interrupt control bit of UI2C_PROTIEN is set, appropriate action or software branch of the new status can be performed in the Interrupt service routine.

Figure 6.28-8 shows the current I^2C STARIF (UI2C_PROTSTS [8]) is set to 1 by hardware, and then set TXDAT = SLA+W (Slave address + Write bit), (PTRG, STA, STO, AA) = (1, 0, 0, x) to send the address to I^2C bus, and write 1 to STARIF (UI2C_PROTSTS [8]) to clear flag. If a slave on the bus matches the address and response ACK, the UI2C_PROTSTS will be updated by ACKIF (UI2C_PROTSTS [13]) setting.

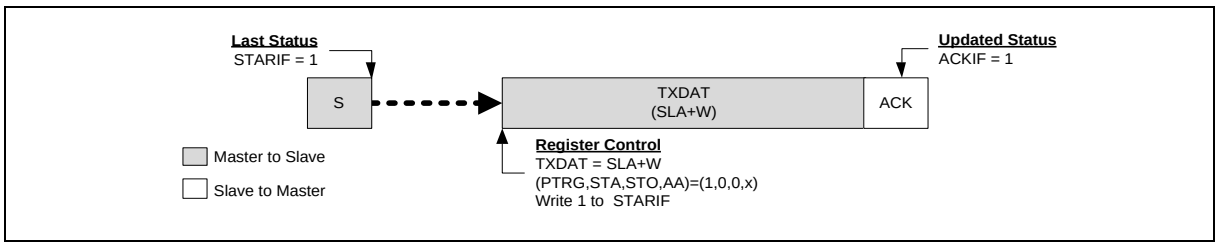


Figure 6.28-8 Control I^2C Bus according to Current I^2C Status

Data Transfer on the I^2C Bus

Figure 6.28-9 shows a master transmits data to slave. A master addresses a slave with a 7-bit address

and 1-bit write index to denote that the master wants to transmit data to the slave. The master keeps transmitting data after the slave returns acknowledge to the master.

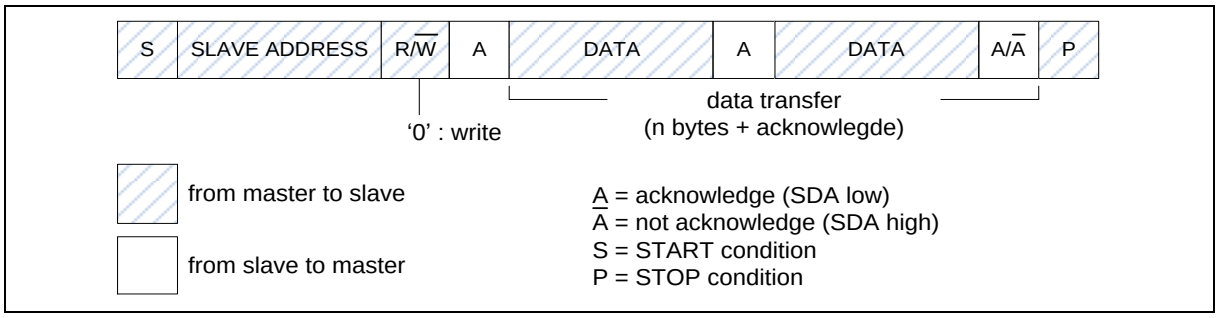


Figure 6.28-9 Master Transmits Data to Slave with a 7-bit Address

Figure 6.28-10 shows a master read data from slave. A master addresses a slave with a 7-bit address and 1-bit read index to denote that the master wants to read data from the slave. The slave will start transmitting data after the slave returns acknowledge to the master.

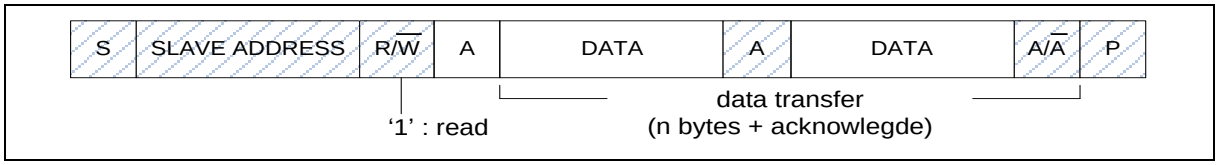


Figure 6.28-10 Master Reads Data from Slave with a 7-bit Address

Figure 6.28-11 shows a master transmits data to slave by 10-bit address. A master addresses a slave with a 10-bit address. First byte contains 10-bit address indicator (5'b11110) and 2-bit address with write index, second byte contains 8-bit address. The master keeps transmitting data after the second byte end. Note that 7-bit and 10-bit address device can work on the same bus.

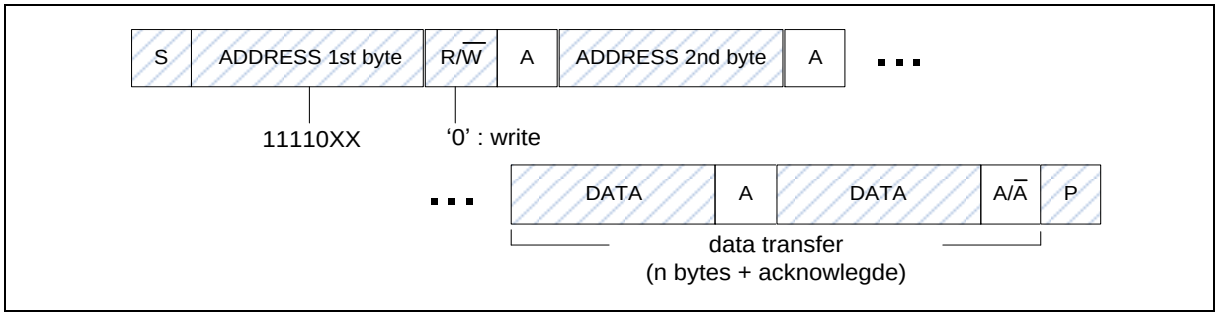


Figure 6.28-11 Master Transmits Data to Slave by 10-bit Address

Figure 6.28-12 shows a master read data from slave by 10-bit address. A master addresses a slave with a 10-bit address. First mater transmits 10-bit address to slave, after that master transmits first byte with read index. The slave will start transmitting data after the first byte with read index.

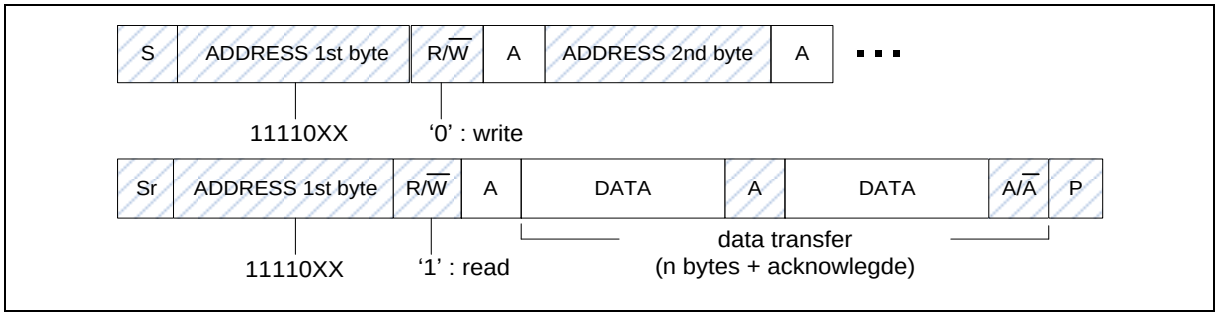


Figure 6.28-12 Master Reads Data from Slave by 10-bit Address

Master Mode

In Figure 6.28-13 and Figure 6.28-14, all possible protocols for I^{2}C master are shown. User needs to follow proper path of the flow to implement required I^{2}C protocol.

In other words, user can send a START signal to bus and I^{2}C will be in Master Transmitter mode (Figure 6.28-13) or Master receiver mode (Figure 6.28-14) after START signal has been sent successfully and new status register would be set STARIF (UI2C_PROTSTS [8]). Followed by START signal, user can send slave address, read/write bit, data and Repeat START, STOP to perform I^{2}C protocol.

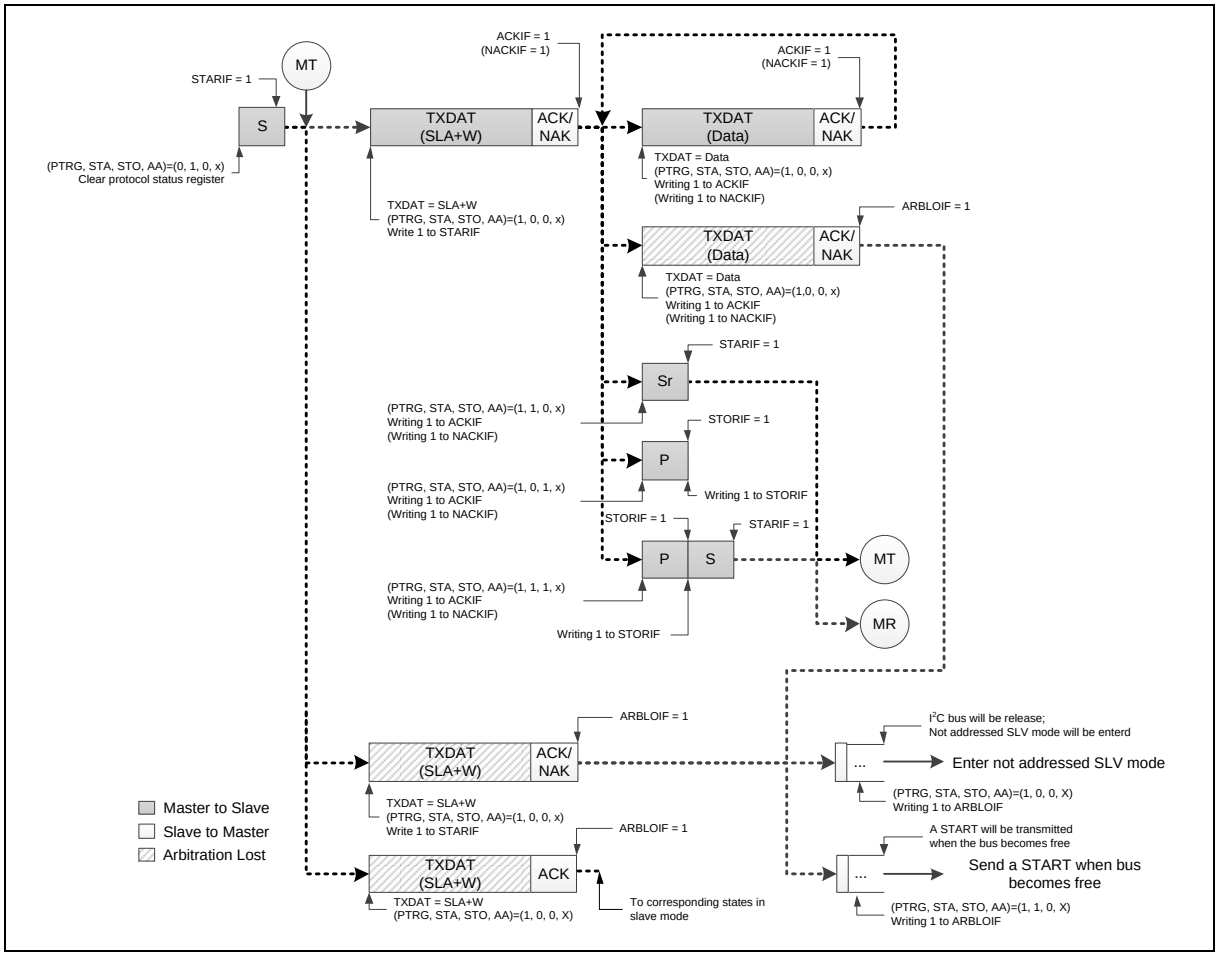


Figure 6.28-13 Master Transmitter Mode Control Flow with 7-bit Address

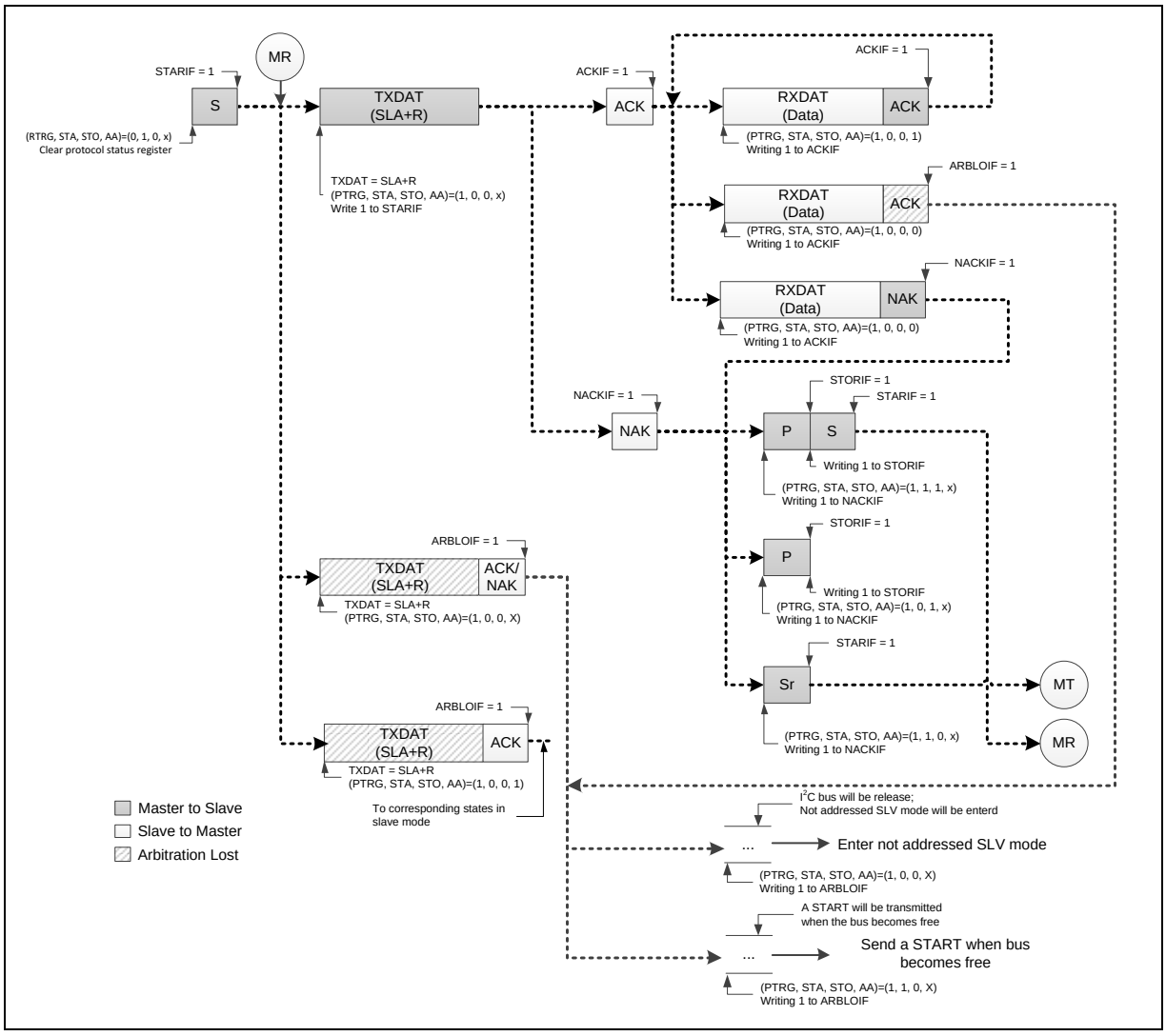


Figure 6.28-14 Master Receiver Mode Control Flow with 7-bit Address

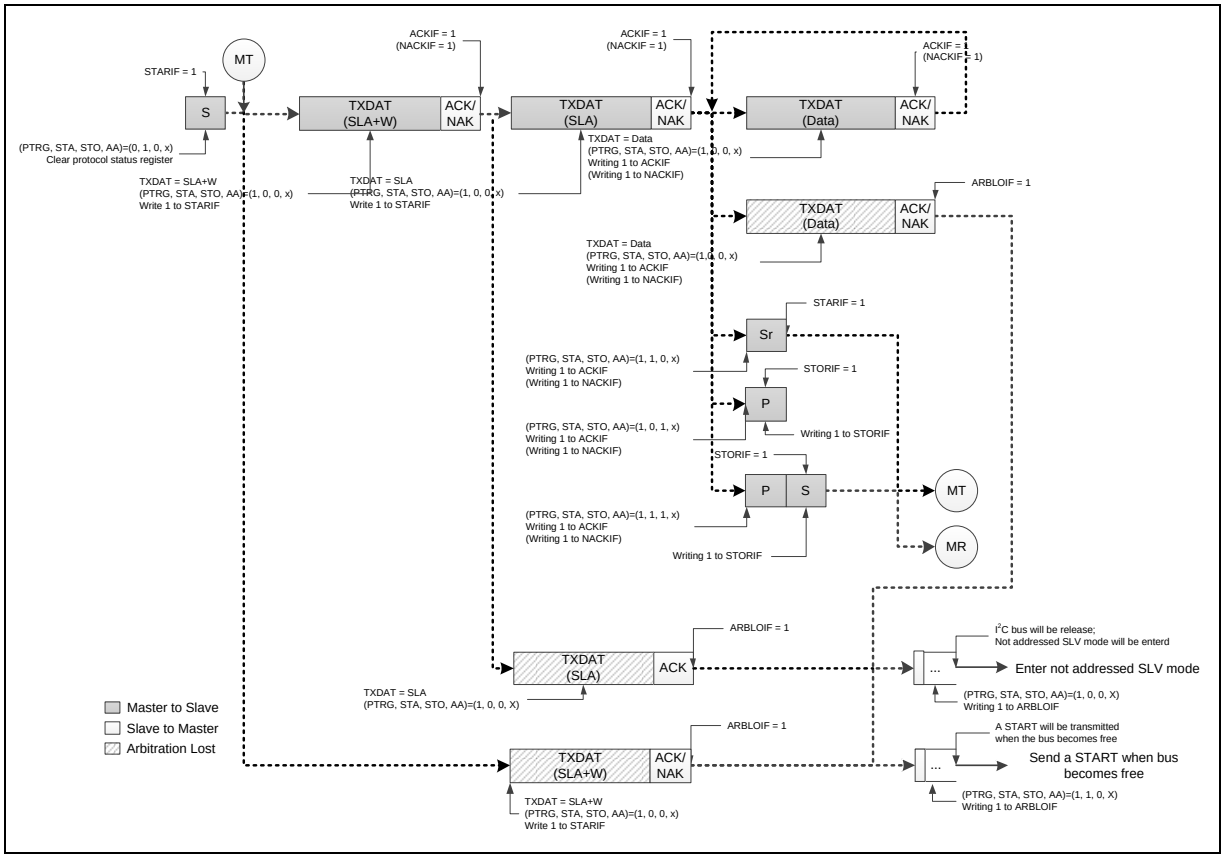


Figure 6.28-15 Master Transmitter Mode Control Flow with 10-bit Address

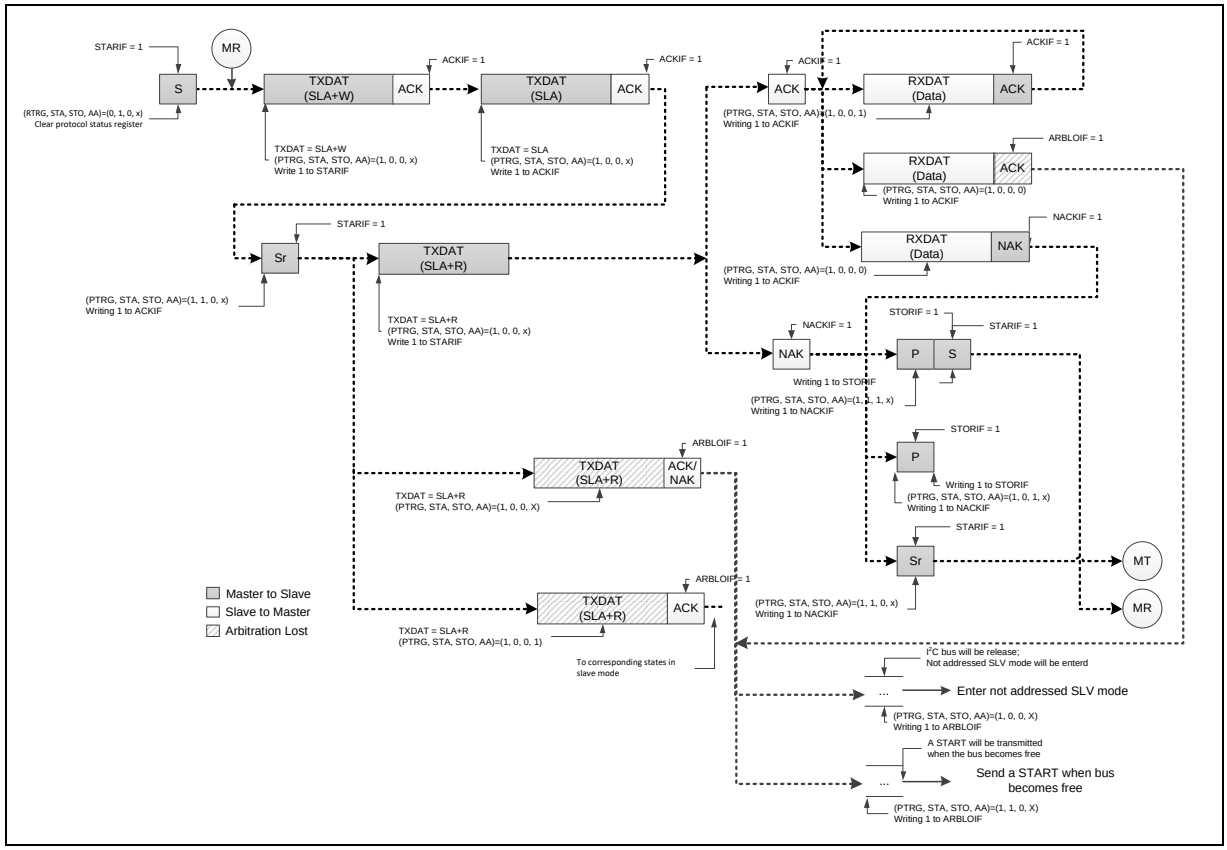


Figure 6.28-16 Master Receiver Mode Control Flow with 10-bit Address

If the I²C is in Master mode and gets arbitration lost, the bit of ARBLOIF (UI2C_PROTSTS [11]) will be set. User may writing 1 to ARBLOIF (UI2C_PROTSTS [11]) and set (PTRG, STA, STO, AA) = (1, 1, 0, X) to send START to re-start Master operation when bus becomes free. Otherwise, user may write 1 to ARBLOIF (UI2C_PROTSTS [11]) and set (PTRG, STA, STO, AA) = (1, 0, 0, X) to release I²C bus and enter not addressed Slave mode.

Slave Mode

When reset, I²C is not addressed and will not recognize the address on I²C bus. User can set device address by UI2C_DEVADDRn and set (PTRG, STA, STO, AA) = (1, 0, 0, 1) to let I²C recognize the address sent by master. Figure 6.28-17 shows all the possible flow for I²C in Slave mode. Users need to follow a proper flow (as shown in Figure 6.28-17 to implement their own I²C protocol).

If bus arbitration is lost in Master mode, I²C port switches to Slave mode immediately and can detect its own slave address in the same serial transfer. If the detected address is SLA+W (Master want to write data to Slave) or SLA+R (Master wants to read data from Slave) after arbitration lost, the ARBLOIF will be set to 1.

The I²C controller supports two slave address match flags, ADMAT0 and ADMAT1 on UI2C_ADMAT[1:0] register. Every control register represents which address is used and set 1 to inform software.

Note: During I²C communication, the SCL clock will be released when writing '1' to PTRG (UI2C_PROTCTL [5]) in Slave mode.

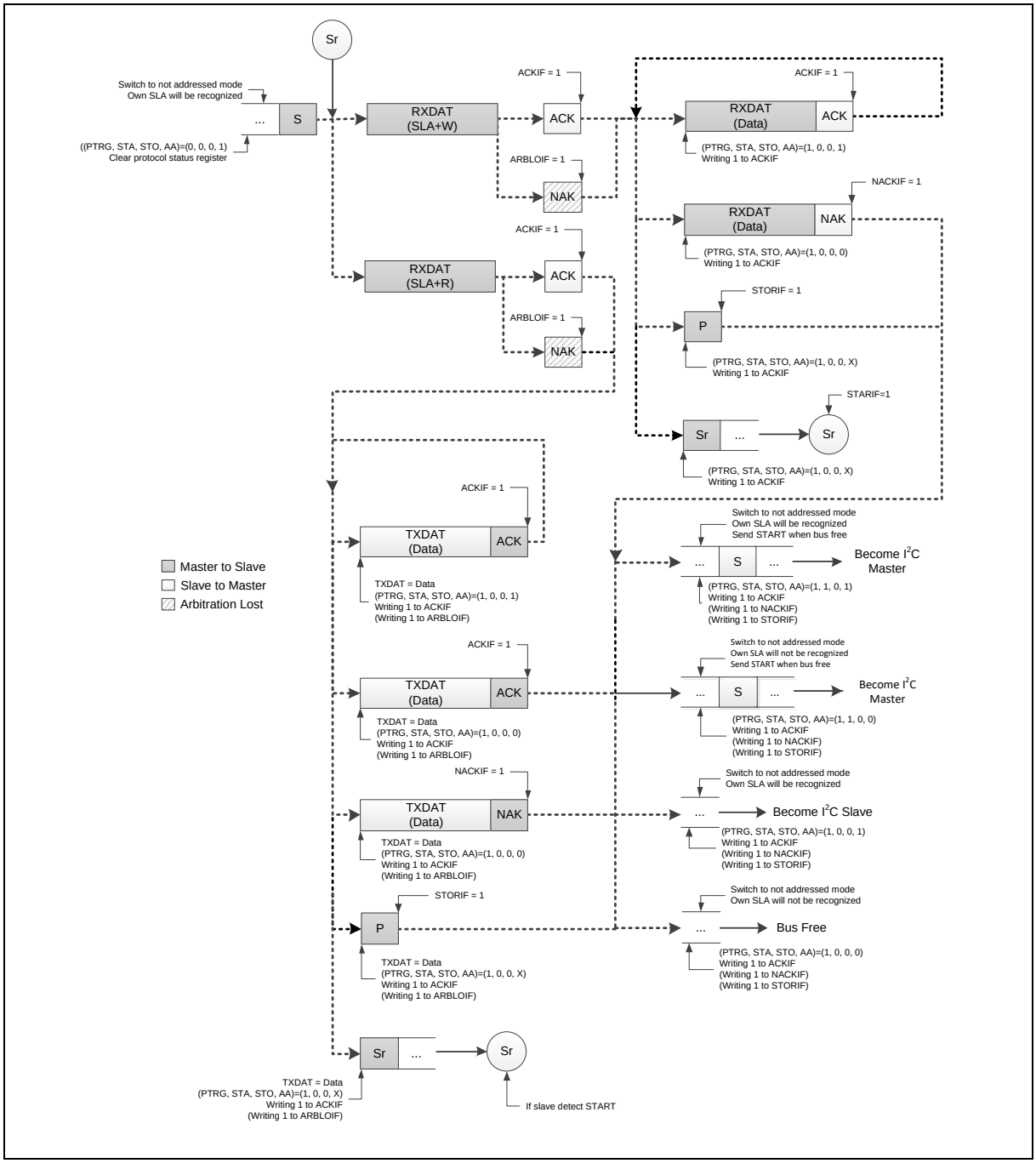


Figure 6.28-17 Save Mode Control Flow with 7-bit Address

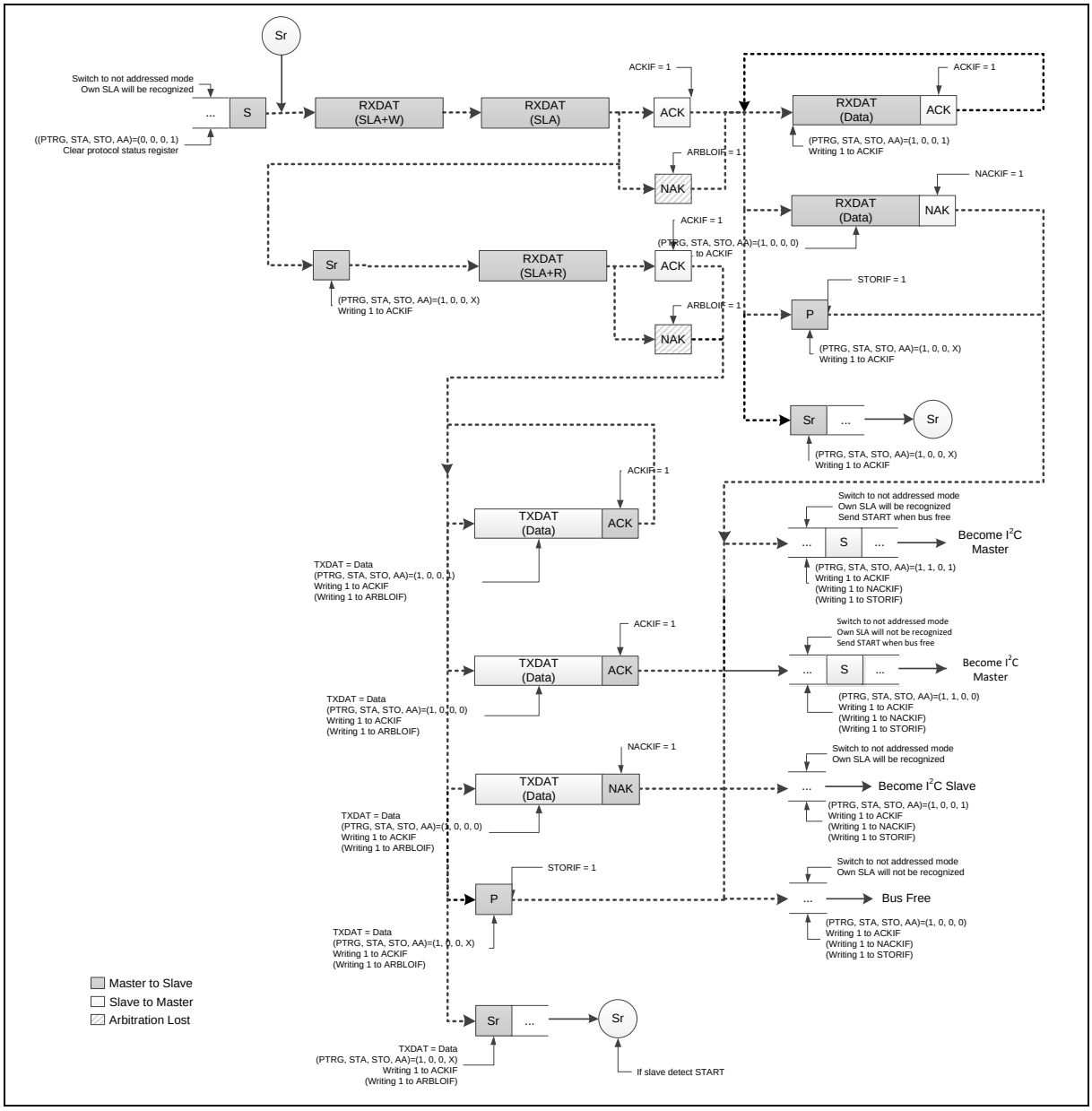


Figure 6.28-18 Save Mode Control Flow with 10-bit Address

If I²C is still transmitting and receiving data in addressed Slave mode but got a STOP or Repeat START, the register STORIF (UI2C_PROTSTS [9]) or STARIF (UI2C_PROTSTS [8]) will be set. User could follow the action for NACKIF (UI2C_PROTSTS [10]) as shown in the above figure when got STARIF (UI2C_PROTSTS [8]) is set.

Note: After slave gets interrupt flag of NACKIF (UI2C_PROTSTS [10]) and start/stop symbol including STARIF (UI2C_PROTSTS [8]) and STORIF (UI2C_PROTSTS [9]), slave can switch to not address mode and own SLA will not be recognized. If setting this interrupt flag, slave will not receive any I²C signal or address from master. At this status, I²C should be reset by setting FUNMODE (UI2C_CTL [2:0]) = 000B to leave this status.

General Call (GC) Mode

If the GCFUNC bit (UI2C_PROTCTL [0]) is set, the I²C port hardware will respond to General Call

address (00H). User can clear GC bit to disable general call function. When the GC bit is set and the I^{2}C in slave mode, it can receive the general call address by 0x00 after master send general call address to I^{2}C bus, and then it also will follow protocol status register.

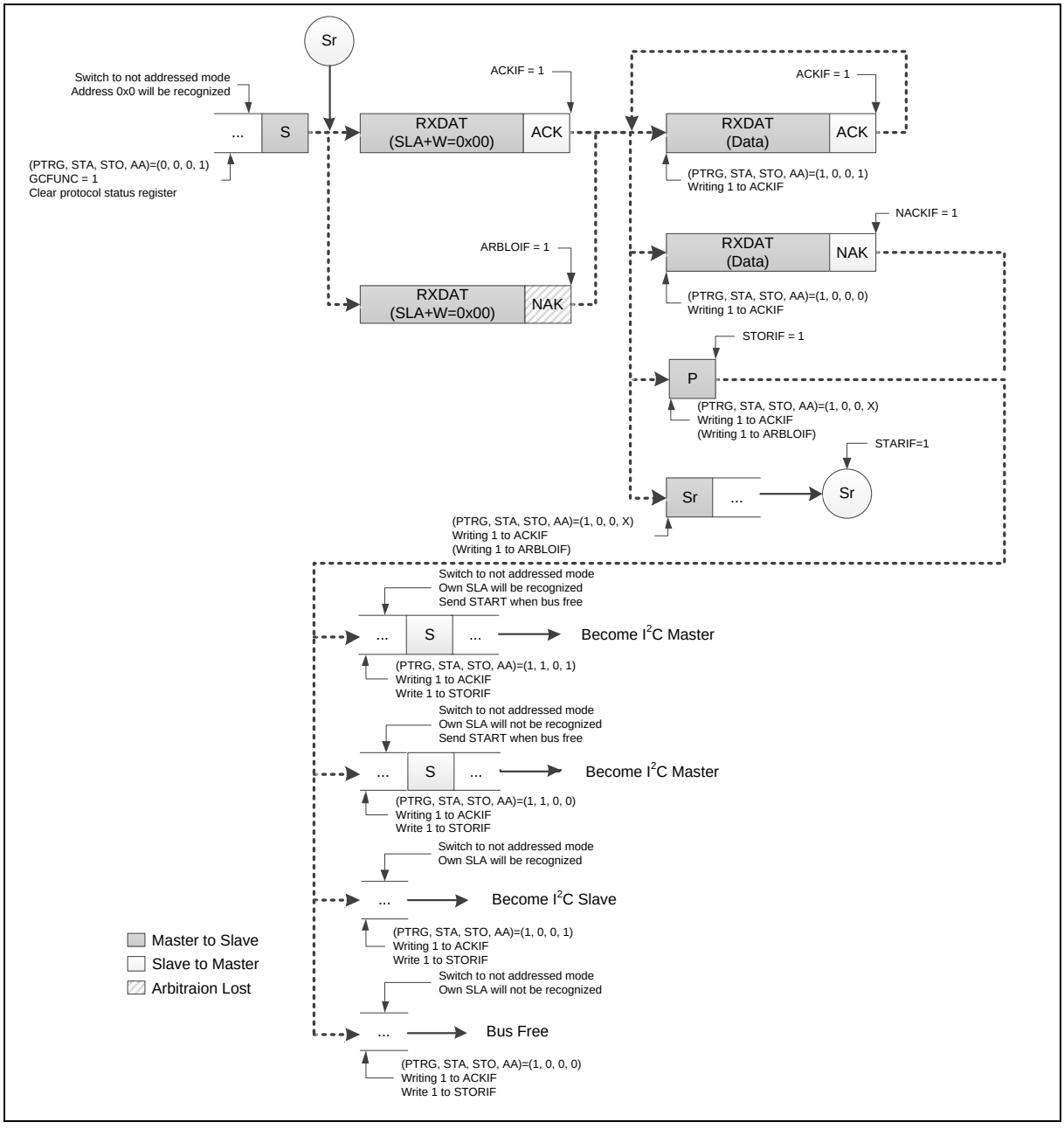


Figure 6.28-19 GC Mode with 7-bit Address

If I^{2}C is still receiving data in GC mode but got a STOP or Repeat START, the STORIF (UI2C_PROTSTS [9]) or STARIF (UI2C_PROTSTS [8]) will be set. User could follow the action for NACKIF (UI2C_PROTSTS [10]) in above figure when got STORIF (UI2C_PROTSTS [9]) or STARIF (UI2C_PROTSTS [8]) is set.

Note: After slave gets interrupt flag of NACKIF (UI2C_PROTSTS [10]) and start/stop symbol including STARIF (UI2C_PROTSTS [8]) and STORIF (UI2C_PROTSTS [9]), slave can switch to not address mode and own SLA will not be recognized. If setting this interrupt flag, slave will not receive any I^{2}C

signal or address from master. At this time, I²C controller should be reset by setting FUNMODE (UI2C_CTL [2:0]) = 000B to leave this status.

Protocol Functional Description

Monitor Mode

When I²C enters monitor mode, this device always returns NACK to master after each frame reception even address matching. Moreover, this device will store any receive data including address, command code, and data.

Interrupt in Monitor Mode

All interrupts will occur as normal process when the MONEN (UI2C_PROTCTL [9]) is set. Note that the first interrupt will occur when initial START, it's not the same as I²C slave, but the other interrupts are the same.

Subsequent to the address-match detection, interrupts will be generated after each data byte is received as slave mode control flow, or after each byte that the module believes it has transmitted for a slave-read transfer. In this second case, the data register will actually contain data transmitted by some other slave on the bus which was actually addressed by the master. If user wants to watch other device, user can set address mask and monitor.

If the monitor has not had time to respond to interrupt, the SCL signal will be pulled to low when SCLOUTEN (UI2C_PROTCTL [8]) is set to 1. User must set PTRG (UI2C_PROTCTL [5]) to release bus when SCLOUTEN (UI2C_PROTCTL [8]) is set to 1. If SCLOUTEN (UI2C_PROTCTL [8]) is not set to 1, user doesn't need to set PTRG (UI2C_PROTCTL [5]) to 1.

When device address match, but the device response NACK, this address will be received into buffer and NACK interrupt will be generated.

Following all of these interrupts, the processor may read the data register to see what was actually transmitted on the bus.

Loss of Arbitration in Monitor Mode

In monitor mode, the I²C module will not be able to respond to a request for information by the bus master or issue an ACK. Some other slave on the bus will respond instead. Software should be aware of the fact that the module is in monitor mode and should not respond to any loss of arbitration state that is detected.

Programmable Setup and Hold Time

In order to guarantee a correct data setup and hold time, the timing must be configured. By programming HTCTL (UI2C_TMCTL[24:16]) to configure hold time and STCTL (UI2C_TMCTL[8:0]) to configure setup time.

The delay timing refer peripheral clock (PCLK). When device stretch master clock, the setup and hold time configuration value will not affected by stretched.

User should focus the limitation of setup and hold time configuration, the timing setting must follow I²C protocol. Once setup time configuration greater than design limitation, that means if setup time setting make SCL output less than three PCLKs, I²C controller can't work normally due to SCL must sample three times. And once hold time configuration greater than I²C clock limitation, I²C will occur bus error. Suggest that user calculate suitable timing with baud rate and protocol before setting timing. Table 6.28-1 shows the relationship between I²C baud rate and PCLK, the number of table represent one clock duty contain how many PCLKs. Setup and hold time configuration even can program some extreme values in the design, but user should follow I²C protocol standard.

I²C Baud Rate	100k	200k	400k	800k	1200k

PCLK					
12 MHz	120	60	30	15	10
24 MHz	240	120	60	30	20
48 MHz	480	240	120	60	40
72 MHz	720	360	180	90	60

Table 6.28-1 Relationship between I^{2}C Baud Rate and PCLK

For setup time wrong adjustment example, assuming one SCL cycle contains ten PCLKs and set STCTL (UI2C_TMCTL[8:0]) to 3 that stretch three PCLKs for setup time setting. The setup time setting limitation: ST_{limit} = (UI2C_BRGEN[25:16]+1) - 6.

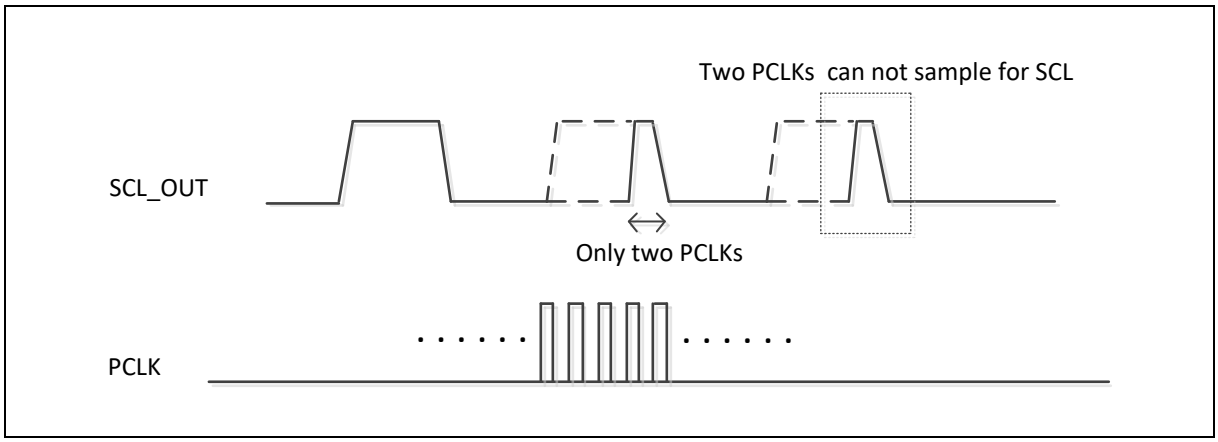


Figure 6.28-20 Setup Time Wrong Adjustment

For hold time wrong adjustment example, use I^{2}C Baud Rate = 1200k and PCLK = 72 MHz, the SCL high/low duty = 60 PCLK. When HTCTL (UI2C_TMCTL[24:16]) is set to 63 and STCTL (UI2C_TMCTL[8:0]) is set to 0, then SDA output delay will over SCL high duty and cause bus error. The hold time setting limitation: HT_{limit} = (UI2C_BRGEN[25:16]+1) - 9.

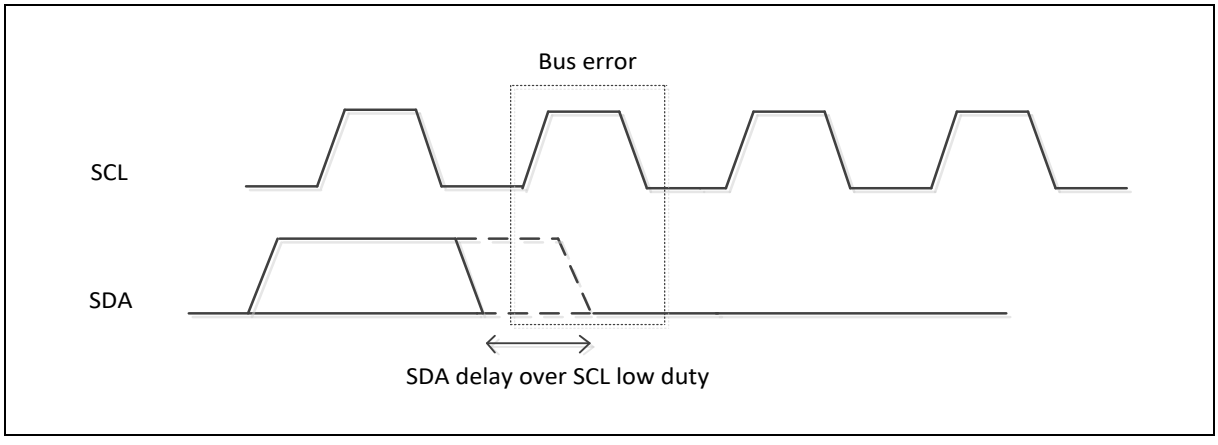


Figure 6.28-21 Hold Time Wrong Adjustment

I^{2}C Time-out Function

There is a 10 bits time-out counter TOCNT (UI2C_PROTCTL [25:16]) which can be used to deal with the I²C bus hang-up. If the time-out counter is enabled, the counter starts up counting until it equals TOCNT (UI2C_PROTCTL [25:16]) and generates I²C interrupt to CPU or stops counting by clearing TOIEN (UI2C_PROTIEN [0]) to 0 or setting all I²C interrupt signal (ACKIF, ERRIF, ARBLOIF, NACKIF, STORIF, STARIF). User may write 1 to clear TOIF(UI2C_PROTSTS[5]) to 0. When time-out counter is enabled, writing 1 to the TOIF will reset counter and re-start up counting after TOIF is cleared. Refer to Figure 6.28-22 for the time-out counter TOCNT (UI2C_PROTCTL [25:16]). T_{TOCNT} = (TOCNT (UI2C_PROTCTL [25:16]) +1) x32 (5-bit) x T_{PCLK}. Note that the time counter clock source TMCNTSRC (UI2C_ BRGEN [5]) must be set as 0.

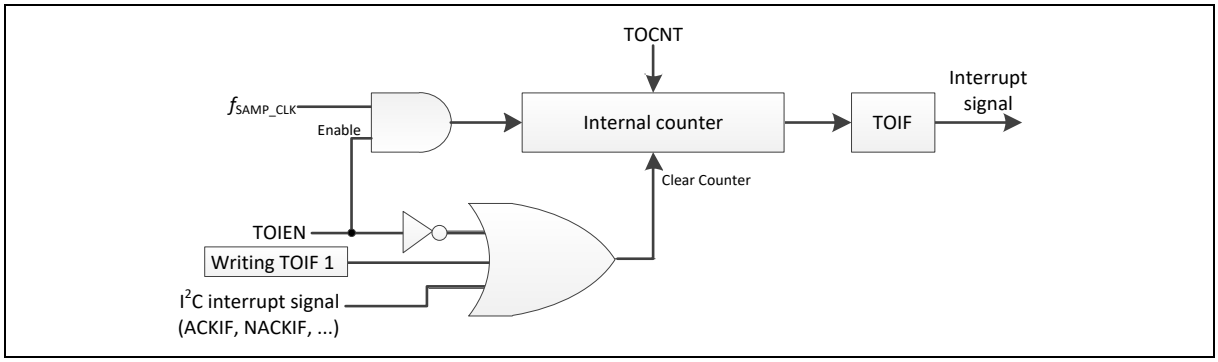


Figure 6.28-22 I²C Time-out Count Block Diagram

Wake-up Function

When chip enters Power-down mode and sets WKEN (WKCTL[0]) to 1, other I²C master can wake up the chip by addressing the I²C device. User must configure the related setting before entering sleep mode. The ACK bit cycle of address match frame is done in power-down. The controller will stretch the SCL to low when the address is matched the device's address and the ACK cycle done. The SCL is stretched until the bit is clear by user. If the frequency of SCL is low speed and the system has wakeup from address match frame, the user shall check this bit to confirm this frame has transaction done and then to do the wake-up procedure. Therefore, when the chip is woken up by address match with one of the device address register (UI2C_DEVADDRn), the user shall check the WKAKDONE (UI2C_PROTSTS [16]) bit is set to 1 to confirm if the address wakeup frame has been done. The WKAKDONE bit indicates that the ACK bit cycle of address match frame is done in power-down. The controller will stretch the SCL to low when the address is matched the device's slave address and the ACK cycle done. The SCL is stretched until the WKAKDONE bit is clear by user. If the frequency of SCL is low speed and the system has wakeup from address match frame, the user shall check this bit to confirm this frame has transaction done and then to do the wake-up procedure. Note that user must clear WKUPIF after clearing the WKAKDONE bit to 0.

The WRSTSWK (UI2C_PROTSTS [17]) bit records the Read/Write command on the address match wake-up frame. The user can use read this bit's status to prepare the next transmitted data (WRSTSWK = 0) or to wait the incoming data (WRSTSWK = 1) can be stored in time after the system is woken up by the address match frame.

When system is woken up by other I²C master device, WKF (UI2C_WKSTS [0]) is set to indicate this event. User needs write "1" to clear this bit.

Example for Random Read on EEPROM

The following steps are used to configure the USCI0_I2C related registers when using I²C protocol to read data from EEPROM.

1. Set USCI0_I2C the multi-function pin as SCL and SDA pins. The muti-function configuration reference Basic Configuration.
2. Enable USCI0 APB clock. The muti-function configuration reference Basic Configuration.

3. Set USCI0RST=1 to reset USCI controller then set USCI0RST=0 let USCI controller to normal operation. The reset controller configuration reference Basic Configuration.
4. Set FUNMODE =100 to enable USCI0_I2C controller in the “UI2C_CTL” register.
5. Give USCI0_I2C clock a divided register value for USCI0_I2C clock rate in the “UI2C_BRGEN”.
6. Enable system I2C0 IRQ in system “NVIC” control register.
7. Set ACKIEN, ERRIEN, ARBLOIEN, NACKIEN, STORIEN, STARIEN, and TOIEN to enable I^2C Interrupt in the “UI2C_PROTIEN” register.
8. Set USCI address registers “UI2C_ADDR0 ~ UI2C_ADDR1”.

Random read operation is one of the methods of access EEPROM. The method allows the master to access any address of EEPROM space. Figure 6.28-23 shows the EEPROM random read operation.

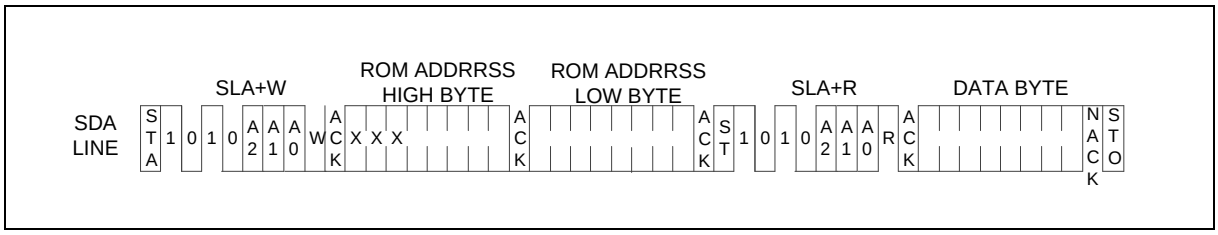


Figure 6.28-23 EEPROM Random Read

Figure 6.28-24 shows how to use I^2C controller to implement the protocol of EEPROM random read.

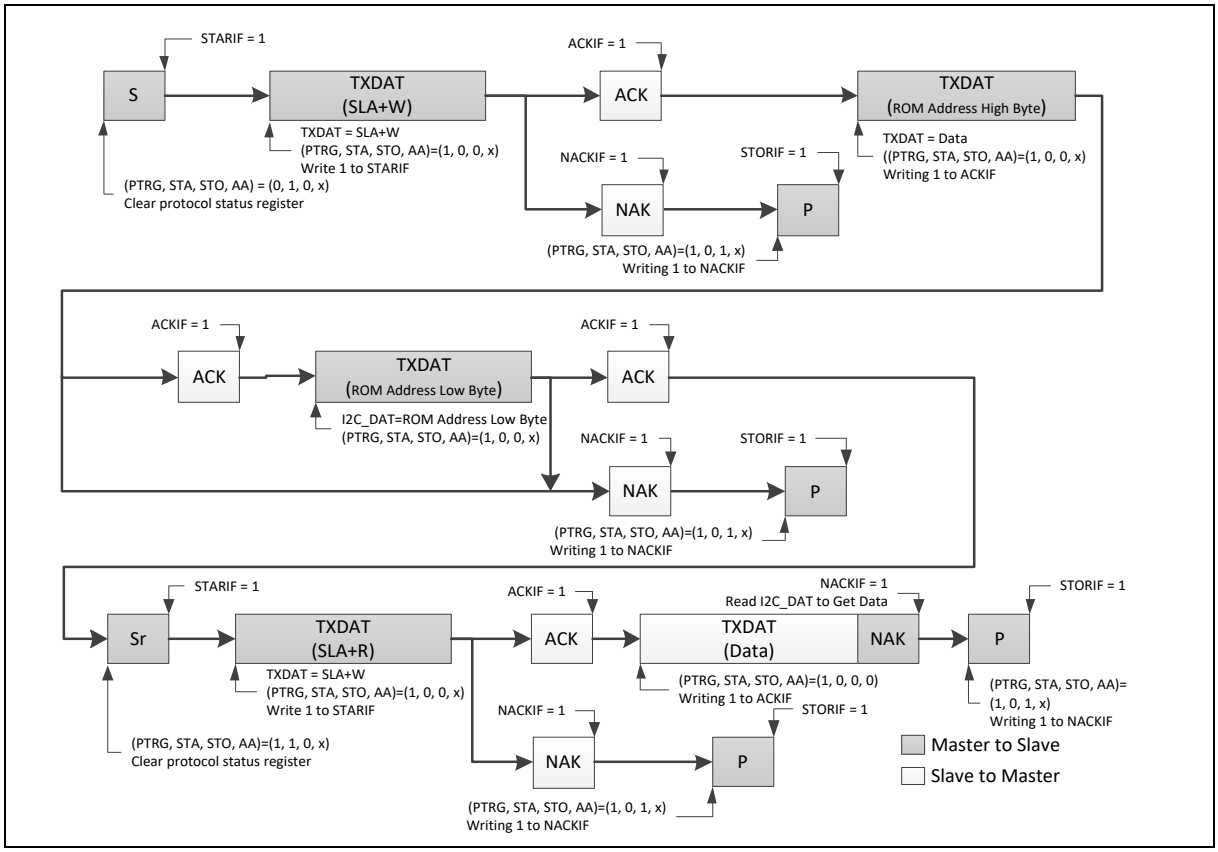


Figure 6.28-24 Protocol of EEPROM Random Read

The I^{2}C controller, which is a master, sends START to bus. Then, it sends a SLA+W (Slave address + Write bit) to EERPOM followed by two bytes data address to set the EEPROM address to read. Finally, a Repeat START followed by SLA+R is sent to read the data from EEPROM.

6.28.6 Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
UI2C_I2C Base Address: **UI2Cn_BA = 0x400D_0000 + (0x1000 * n)** **n= 0, 1** **UI2C_I2C non-secure base address is UI2Cn_BA + 0x1000_0000.,**				
UI2C_CTL	UI2Cn_BA+0x00	R/W	USCI Control Register	0x0000_0000
UI2C_BRGEN	UI2Cn_BA+0x08	R/W	USCI Baud Rate Generator Register	0x0000_3C00
UI2C_LINECTL	UI2Cn_BA+0x2C	R/W	USCI Line Control Register	0x0000_0000
UI2C_TXDAT	UI2Cn_BA+0x30	W	USCI Transmit Data Register	0x0000_0000
UI2C_RXDAT	UI2Cn_BA+0x34	R	USCI Receive Data Register	0x0000_0000
UI2C_DEVADDR0	UI2Cn_BA+0x44	R/W	USCI Device Address Register 0	0x0000_0000
UI2C_DEVADDR1	UI2Cn_BA+0x48	R/W	USCI Device Address Register 1	0x0000_0000
UI2C_ADDRMSK0	UI2Cn_BA+0x4C	R/W	USCI Device Address Mask Register 0	0x0000_0000
UI2C_ADDRMSK1	UI2Cn_BA+0x50	R/W	USCI Device Address Mask Register 1	0x0000_0000
UI2C_WKCTL	UI2Cn_BA+0x54	R/W	USCI Wake-up Control Register	0x0000_0000
UI2C_WKSTS	UI2Cn_BA+0x58	R/W	USCI Wake-up Status Register	0x0000_0000
UI2C_PROTCTL	UI2Cn_BA+0x5C	R/W	USCI Protocol Control Register	0x0000_0000
UI2C_PROTIEN	UI2Cn_BA+0x60	R/W	USCI Protocol Interrupt Enable Register	0x0000_0000
UI2C_PROTSTS	UI2Cn_BA+0x64	R/W	USCI Protocol Status Register	0x0000_0000
UI2C_ADMAT	UI2Cn_BA+0x88	R/W	I^2C Slave Match Address Register	0x0000_0000
UI2C_TMCTL	UI2Cn_BA+0x8C	R/W	I^2C Timing Configure Control Register	0x0000_0000

6.28.7 Register Description

USCI Control Register (UI2C_CTL)

Register	Offset	R/W	Description	Reset Value
UI2C_CTL	UI2Cn_BA+0x00	R/W	USCI Control Register	0x0000_0000

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

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2:0]	**FUNMODE**	**Function Mode** This bit field selects the protocol for this USCI controller. Selecting a protocol that is not available or a reserved combination disables the USCI. When switching between two protocols, the USCI has to be disabled before selecting a new protocol. Simultaneously, the USCI will be reset when user write 000 to FUNMODE. 000 = The USCI is disabled. All protocol related state machines are set to idle state. 001 = The SPI protocol is selected. 010 = The UART protocol is selected. 100 = The I^2C protocol is selected. **Note:** Other bit combinations are reserved.

USCI Baud Rate Generator Register (UI2C_BRGEN)

Register	Offset	R/W	Description	Reset Value
UI2C_BRGEN	UI2Cn_BA+0x08	R/W	USCI Baud Rate Generator Register	0x0000_3C00

31	30	29	28	27	26	25	24
NFCNT				Reserved		CLKDIV	
23	22	21	20	19	18	17	16
CLKDIV							
15	14	13	12	11	10	9	8
Reserved	DSCNT					PDSCNT	
7	6	5	4	3	2	1	0
Reserved		TMCNTSRC	TMCNTEN	SPCLKSEL		PTCLKSEL	RCLKSEL

Bits	Description	
[31:28]	**NFCNT**	**Noise Filter Count** The register bits control the input filter width. 0 = Filter width 3*PCLK. 1 = Filter width 4*PCLK. *N* = Filter width (3+*N*)*PCKL. **Note:** Filter width Min:3*PCLK, Max: 18*PCLK
[27:26]	**Reserved**	Reserved.
[25:16]	**CLKDIV**	**Clock Divider** This bit field defines the ratio between the protocol clock frequency f_{PROT_CLK} and the clock divider frequency f_{DIV_CLK} (fDIV_CLK = f_{PROT_CLK} / (CLKDIV+1)).
[15]	**Reserved**	Reserved.
[14:10]	**DSCNT**	**Denominator for Sample Counter** This bit field defines the divide ratio of the sample clock f_{SAMP_CLK}. The divided frequency f_{DS_CNT} = f_{PDS_CNT} / (DSCNT+1). **Note:** The maximum value of DSCNT is 0xF on UART mode and suggest to set over 4 to confirm the receiver data is sampled in right value.
[9:8]	**PDSCNT**	**Pre-divider for Sample Counter** This bit field defines the divide ratio of the clock division from sample clock f_{SAMP_CLK}. The divided frequency f_{PDS_CNT} = f_{SAMP_CLK} / (PDSCNT+1).
[7:6]	**Reserved**	Reserved.
[5]	**TMCNTSRC**	**Time Measurement Counter Clock Source Selection** 0 = Time measurement counter with f_{PROT_CLK}. 1 = Time measurement counter with f_{DIV_CLK}.
[4]	**TMCNTEN**	**Time Measurement Counter Enable Bit** This bit enables the 10-bit timing measurement counter.

		0 = Time measurement counter is Disabled. 1 = Time measurement counter is Enabled.
[3:2]	**SPCLKSEL**	**Sample Clock Source Selection** This bit field used for the clock source selection of a sample clock (f_{SAMP_CLK}) for the protocol processor. 00 = f_{SAMP_CLK} = f_{DIV_CLK}. 01 = f_{SAMP_CLK} = f_{PROT_CLK}. 10 = f_{SAMP_CLK} = f_{SCLK}. 11 = f_{SAMP_CLK} = f_{REF_CLK}.
[1]	**PTCLKSEL**	**Protocol Clock Source Selection** This bit selects the source signal of protocol clock (f_{PROT_CLK}). 0 = Reference clock f_{REF_CLK}. 1 = f_{REF_CLK2} (its frequency is half of f_{REF_CLK}).
[0]	**RCLKSEL**	**Reference Clock Source Selection** This bit selects the source signal of reference clock (f_{REF_CLK}). 0 = Peripheral device clock f_{PCLK}. 1 = Reserved.

USCI Line Control Register (UI2C_LINECTL)

Register	Offset	R/W	Description	Reset Value
UI2C_LINECTL	UI2Cn_BA+0x2C	R/W	USCI Line Control Register	0x0000_0000

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

Bits	Description	
[31:12]	Reserved	Reserved.
[11:8]	DWIDTH	**Word Length of Transmission** This bit field defines the data word length (amount of bits) for reception and transmission. The data word is always right-aligned in the data buffer. USCI support word length from 4 to 16 bits. 0x0: The data word contains 16 bits located at bit positions [15:0]. 0x1: Reserved. 0x2: Reserved. 0x3: Reserved. 0x4: The data word contains 4 bits located at bit positions [3:0]. 0x5: The data word contains 5 bits located at bit positions [4:0]. ... 0xF: The data word contains 15 bits located at bit positions [14:0].
[7:1]	Reserved	Reserved.
[0]	LSB	**LSB First Transmission Selection** 0 = The MSB, which bit of transmit/receive data buffer depends on the setting of DWIDTH, is transmitted/received first. 1 = The LSB, the bit 0 of data buffer, will be transmitted/received first.

USCI Transmit Data Register (UI2C_TXDAT)

Register	Offset	R/W	Description	Reset Value
UI2C_TXDAT	UI2Cn_BA+0x30	W	USCI Transmit Data Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**TXDAT**	**Transmit Data** Software can use this bit field to write 16-bit transmit data for transmission.

USCI Receive Data Register (UI2C_RXDAT)

Register	Offset	R/W	Description	Reset Value
UI2C_RXDAT	UI2Cn_BA+0x34	R	USCI Receive Data Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**RXDAT**	**Received Data** This bit field monitors the received data which stored in receive data buffer. **Note:** In I^2C protocol, RXDAT[12:8] indicate the different transmission conditions which defined in I^2C.

USCI Device Address Register (UI2C_DEVADDR)

Register	Offset	R/W	Description	Reset Value
UI2C_DEVADDR0	UI2Cn_BA+0x44	R/W	USCI Device Address Register 0	0x0000_0000
UI2C_DEVADDR1	UI2Cn_BA+0x48	R/W	USCI Device Address Register 1	0x0000_0000

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

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9:0]	**DEVADDR**	**Device Address** In I^2C protocol, this bit field contains the programmed slave address. If the first received address byte are 1111 $0AAX_B$, the AA bits are compared to the bits DEVADDR[9:8] to check for address match, where the X is R/W bit. Then the second address byte is also compared to DEVADDR[7:0]. **Note 1:** The DEVADDR [9:7] must be set 3'b000 when I^2C operating in 7-bit address mode. **Note 2:** When software sets 10'h000, the address can not be used.

USCI Device Address Mask Register (UI2C_ADDRMSK) – for I²C Only

Register	Offset	R/W	Description	Reset Value
UI2C_ADDRMSK0	UI2Cn_BA+0x4C	R/W	USCI Device Address Mask Register 0	0x0000_0000
UI2C_ADDRMSK1	UI2Cn_BA+0x50	R/W	USCI Device Address Mask Register 1	0x0000_0000

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

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9:0]	**ADDRMSK**	**USCI Device Address Mask** 0 = Mask Disabled (the received corresponding register bit should be exact the same as address register.). 1 = Mask Enabled (the received corresponding address bit is don't care.). USCI support multiple address recognition with two address mask register. When the bit in the address mask register is set to one, it means the received corresponding address bit is don't-care. If the bit is set to zero, that means the received corresponding register bit should be exact the same as address register. **Note:** The wake-up function can not use address mask.

USCI Wake-up Control Register (UI2C_WKCTL)

Register	Offset	R/W	Description	Reset Value
UI2C_WKCTL	UI2Cn_BA+0x54	R/W	USCI Wake-up Control Register	0x0000_0000

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

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**WKADDREN**	**Wake-up Address Match Enable Bit** 0 = The chip is woken up according to receive ‘START’ symbol. 1 = The chip is woken up according to address match.
[0]	**WKEN**	**Wake-up Enable Bit** 0 = Wake-up function Disabled. 1 = Wake-up function Enabled.

USCI Wake-up Status Register (UI2C_WKSTS)

Register	Offset	R/W	Description	Reset Value
UI2C_WKSTS	UI2Cn_BA+0x58	R/W	USCI Wake-up Status Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**WKF**	**Wake-up Flag** When chip is woken up from Power-down mode, this bit is set to 1. Software can write 1 to clear this bit.

USCI Protocol Control Register – I²C (UI2C_PROTCTL)

Register	Offset	R/W	Description	Reset Value
UI2C_PROTCTL	UI2Cn_BA+0x5C	R/W	USCI Protocol Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PROTEN	Reserved					TOCNT	
23	22	21	20	19	18	17	16
TOCNT							
15	14	13	12	11	10	9	8
Reserved						MONEN	SCLOUTEN
7	6	5	4	3	2	1	0
Reserved		PTRG	ADDR10EN	STA	STO	AA	GCFUNC

Bits	Description	
[31]	**PROTEN**	**I²C Protocol Enable Bit** 0 = I²C Protocol Disabled. 1 = I²C Protocol Enabled.
[30:26]	**Reserved**	Reserved.
[25:16]	**TOCNT**	**Time-out Clock Cycle** This bit field indicates how many clock cycle selected by TMCNTSRC (UI2C_BRGEN [5]) when each interrupt flags are clear. The time-out is enable when TOCNT bigger than 0. **Note:** The TMCNTSRC (UI2C_BRGEN [5]) must be set to 0 in I²C mode.
[15:10]	**Reserved**	Reserved.
[9]	**MONEN**	**Monitor Mode Enable Bit** This bit enables monitor mode. In monitor mode the SDA output will be put in high impedance mode. This prevents the I²C module from outputting data of any kind (including ACK) onto the I²C data bus. 0 = The monitor mode Disabled. 1 = The monitor mode Enabled. **Note:** Depending on the state of the SCLOUTEN bit, the SCL output may be also forced high, preventing the module from having control over the I²C clock line.
[8]	**SCLOUTEN**	**SCL Output Enable Bit** This bit enables monitor pulling SCL to low. This monitor will pull SCL to low until it has had time to respond to an I²C interrupt. 0 = SCL output will be forced high due to open drain mechanism. 1 = I²C module may act as a slave peripheral just like in normal operation, the I²C holds the clock line low until it has had time to clear I²C interrupt.
[7:6]	**Reserved**	Reserved.

[5]	**PTRG**	**I²C Protocol Trigger (Write Only)** When a new state is present in the UI2C_PROTSTS register, if the related interrupt enable bits are set, the I²C interrupt is requested. It must write one by software to this bit after the related interrupt flags are set to 1 and the I²C protocol function will go ahead until the STOP is active or the PROTEN is disabled. 0 = I2C's stretch disabled and the I²C protocol function will go ahead. 1 = I2C's stretch active.
[4]	**ADDR10EN**	**Address 10-bit Function Enable Bit** 0 = Address match 10 bit function Disabled. 1 = Address match 10 bit function Enabled.
[3]	**STA**	**I²C START Control** Setting STA to logic 1 to enter Master mode; the I²C hardware sends a START or repeats START condition to bus when the bus is free.
[2]	**STO**	**I²C STOP Control** In Master mode, setting STO to transmit a STOP condition to bus then I²C hardware will check the bus condition if a STOP condition is detected this bit will be cleared by hardware automatically. In a slave mode, setting STO resets I²C hardware to the defined "not addressed" slave mode when bus error (UI2C_PROTSTS.ERRIF = 1).
[1]	**AA**	**Assert Acknowledge Control** When AA =1 prior to address or data received, an acknowledged (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line when 1.) A slave is acknowledging the address sent from master, 2.) The receiver devices are acknowledging the data sent by transmitter. When AA=0 prior to address or data received, a Not acknowledged (high level to SDA) will be returned during the acknowledge clock pulse on the SCL line.
[0]	**GCFUNC**	**General Call Function** 0 = General Call Function Disabled. 1 = General Call Function Enabled.

USCI Protocol Interrupt Enable Register – I²C (UI2C_PROTIEN)

Register	Offset	R/W	Description	Reset Value
UI2C_PROTIEN	UI2Cn_BA+0x60	R/W	USCI Protocol Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved	ACKIEN	ERRIEN	ARBLOIEN	NACKIEN	STORIEN	STARIEN	TOIEN

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**ACKIEN**	**Acknowledge Interrupt Enable Bit** This bit enables the generation of a protocol interrupt if an acknowledge is detected by a master. 0 = The acknowledge interrupt Disabled. 1 = The acknowledge interrupt Enabled.
[5]	**ERRIEN**	**Error Interrupt Enable Bit** This bit enables the generation of a protocol interrupt if an I²C error condition is detected (indicated by ERR (UI2C_PROTSTS [16])). 0 = The error interrupt Disabled. 1 = The error interrupt Enabled.
[4]	**ARBLOIEN**	**Arbitration Lost Interrupt Enable Bit** This bit enables the generation of a protocol interrupt if an arbitration lost event is detected. 0 = The arbitration lost interrupt Disabled. 1 = The arbitration lost interrupt Enabled.
[3]	**NACKIEN**	**Non - Acknowledge Interrupt Enable Bit** This bit enables the generation of a protocol interrupt if a Non - acknowledge is detected by a master. 0 = The non - acknowledge interrupt Disabled. 1 = The non - acknowledge interrupt Enabled.
[2]	**STORIEN**	**STOP Condition Received Interrupt Enable Bit** This bit enables the generation of a protocol interrupt if a STOP condition is detected. 0 = The stop condition interrupt Disabled. 1 = The stop condition interrupt Enabled.
[1]	**STARIEN**	**START Condition Received Interrupt Enable Bit** This bit enables the generation of a protocol interrupt if a START condition is detected.

		0 = The start condition interrupt Disabled. 1 = The start condition interrupt Enabled.
[0]	**TOIEN**	**Time-out Interrupt Enable Bit** In I^2C protocol, this bit enables the interrupt generation in case of a time-out event. 0 = The time-out interrupt Disabled. 1 = The time-out interrupt Enabled.

USCI Protocol Status Register – I²C (UI2C_PROTSTS)

Register	Offset	R/W	Description	Reset Value
UI2C_PROTSTS	UI2Cn_BA+0x64	R/W	USCI Protocol Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved				ERRARBLO	BUSHANG	WRSTSWK	WKAKDONE
15	14	13	12	11	10	9	8
SLAREAD	SLASEL	ACKIF	ERRIF	ARBLOIF	NACKIF	STORIF	STARIF
7	6	5	4	3	2	1	0
Reserved	ONBUSY	TOIF	Reserved				

Bits	Description	
[31:20]	**Reserved**	Reserved.
[19]	**ERRARBLO**	**Error Arbitration Lost** This bit indicates bus arbitration lost due to bigger noise which is can't be filtered by input processor. The I²C can send start condition when ERRARBLO is set. Thus this bit doesn't be cared on slave mode. 0 = The bus is normal status for transmission. 1 = The bus is error arbitration lost status for transmission. **Note:** This bit has no interrupt signal, and it will be cleared automatically by hardware when a START condition is present.
[18]	**BUSHANG**	**Bus Hang-up** This bit indicates bus hang-up status. There is 4-bit counter count when SCL hold high and refer f_{SAMP_CLK}. The hang-up counter will count to overflow and set this bit when SDA is low. The counter will be reset by falling edge of SCL signal. 0 = The bus is normal status for transmission. 1 = The bus is hang-up status for transmission. **Note:** This bit has no interrupt signal, and it will be cleared automatically by hardware when a START condition is present.
[17]	**WRSTSWK**	**Read/Write Status Bit in Address Wake-up Frame** 0 = Write command be record on the address match wake-up frame. 1 = Read command be record on the address match wake-up frame.
[16]	**WKAKDONE**	**Wake-up Address Frame Acknowledge Bit Done** 0 = The ACK bit cycle of address match frame isn't done. 1 = The ACK bit cycle of address match frame is done in power-down. **Note:** This bit can't release when WKUPIF is set.
[15]	**SLAREAD**	**Slave Read Request Status** This bit indicates that a slave read request has been detected. 0 = A slave R/W bit is 1 has not been detected.

		1 = A slave R/W bit is 1 has been detected. **Note:** This bit has no interrupt signal, and it will be cleared automatically by hardware.
[14]	**SLASEL**	**Slave Select Status** This bit indicates that this device has been selected as slave. 0 = The device is not selected as slave. 1 = The device is selected as slave. **Note:** This bit has no interrupt signal, and it will be cleared automatically by hardware.
[13]	**ACKIF**	**Acknowledge Received Interrupt Flag** This bit indicates that an acknowledge has been received in master mode. A protocol interrupt can be generated if UI2C_PROTCTL.ACKIEN = 1. 0 = An acknowledge has not been received. 1 = An acknowledge has been received. **Note:** It is cleared by software writing 1 into this bit
[12]	**ERRIF**	**Error Interrupt Flag** This bit indicates that a Bus Error occurs when a START or STOP condition is present at an illegal position in the formation frame. Example of illegal position are during the serial transfer of an address byte, a data byte or an acknowledge bit. A protocol interrupt can be generated if UI2C_PROTCTL.ERRIEN = 1. 0 = An I^2C error has not been detected. 1 = An I^2C error has been detected. **Note 1:** It is cleared by software writing 1 into this bit **Note 2:** This bit is set for slave mode, and user must write 1 into STO register to the defined “not addressed” slave mode.
[11]	**ARBLOIF**	**Arbitration Lost Interrupt Flag** This bit indicates that an arbitration has been lost. A protocol interrupt can be generated if UI2C_PROTCTL.ARBLOIEN = 1. 0 = An arbitration has not been lost. 1 = An arbitration has been lost. **Note:** It is cleared by software writing 1 into this bit.
[10]	**NACKIF**	**Non - Acknowledge Received Interrupt Flag** This bit indicates that a non - acknowledge has been received in master mode. A protocol interrupt can be generated if UI2C_PROTCTL.NACKIEN = 1. 0 = A non - acknowledge has not been received. 1 = A non - acknowledge has been received. **Note:** It is cleared by software writing 1 into this bit.
[9]	**STORIF**	**Stop Condition Received Interrupt Flag** This bit indicates that a stop condition has been detected on the I^2C bus lines. A protocol interrupt can be generated if UI2C_PROTCTL.STORIEN = 1. 0 = A stop condition has not yet been detected. 1 = A stop condition has been detected. **Note 1:** It is cleared by software writing 1 into this bit.
[8]	**STARIF**	**Start Condition Received Interrupt Flag** This bit indicates that a start condition or repeated start condition has been detected on master mode. However, this bit also indicates that a repeated start condition has been detected on slave mode. A protocol interrupt can be generated if UI2C_PROTCTL.STARIEN = 1. 0 = A start condition has not yet been detected. 1 = A start condition has been detected. **Note:** It is cleared by software writing 1 into this bit.

[7]	**Reserved**	Reserved.
[6]	**ONBUSY**	**On Bus Busy** Indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. It is cleared by hardware when a STOP condition is detected 0 = The bus is IDLE (both SCLK and SDA High). 1 = The bus is busy.
[5]	**TOIF**	**Time-out Interrupt Flag** 0 = A time-out interrupt status has not occurred. 1 = A time-out interrupt status has occurred. **Note:** It is cleared by software writing 1 into this bit
[4:0]	**Reserved**	Reserved.

USCI Slave Match Address Register (UI2C_ADMAT)

Register	Offset	R/W	Description	Reset Value
UI2C_ADMAT	UI2Cn_BA+0x88	R/W	I^2C Slave Match Address Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						ADMAT1	ADMAT0

Bits	Description	
[31:2]	Reserved	Reserved.
[1]	ADMAT1	**USCI Address 1 Match Status Register** When address 1 is matched, hardware will inform which address used. This bit will set to 1, and software can write 1 to clear this bit.
[0]	ADMAT0	**USCI Address 0 Match Status Register** When address 0 is matched, hardware will inform which address used. This bit will set to 1, and software can write 1 to clear this bit.

USCI Timing Configure Control Register (UI2C_TMCTL)

Register	Offset	R/W	Description	Reset Value
UI2C_TMCTL	UI2Cn_BA+0x8C	R/W	I^2C Timing Configure Control Register	0x0000_0000

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

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24:16]	**HTCTL**	**Hold Time Configure Control** This field is used to generate the delay timing between SCL falling edge SDA edge in transmission mode. The delay hold time is numbers of peripheral clock = HTCTL x f_{PCLK}.
[15:9]	**Reserved**	Reserved.
[8:0]	**STCTL**	**Setup Time Configure Control** This field is used to generate a delay timing between SDA edge and SCL rising edge in transmission mode.. The delay setup time is numbers of peripheral clock = STCTL x f_{PCLK}.

6.29 Controller Area Network (CAN)

6.29.1 Overview

The C_CAN consists of the CAN Core, Message RAM, Message Handler, Control Registers and Module Interface. The CAN Core performs communication according to the CAN protocol version 2.0 part A and B. The bit rate can be programmed to values up to 1 Mbytesit/s. For the connection to the physical layer, additional transceiver hardware is required.

For communication on a CAN network, individual Message Objects are configured. The Message Objects and Identifier Masks for acceptance filtering of received messages are stored in the Message RAM. All functions concerning the handling of messages are implemented in the Message Handler. These functions include acceptance filtering, the transfer of messages between the CAN Core and the Message RAM, and the handling of transmission requests as well as the generation of the module interrupt.

The register set of the C_CAN can be accessed directly by the software through the module interface. These registers are used to control/configure the CAN Core and the Message Handler and to access the Message RAM.

6.29.2 Features

- Supports CAN protocol version 2.0 part A and B
- Bit rates up to 1 MBit/s
- 32 Message Objects
- Each Message Object has its own identifier mask
- Programmable FIFO mode (concatenation of Message Objects)
- Maskable interrupt
- Disabled Automatic Re-transmission mode for Time Triggered CAN applications
- Programmable loop-back mode for self-test operation
- 16-bit module interfaces to the AMBA APB bus
- Supports wake-up function

6.29.3 Block Diagram

The C_CAN interfaces with the AMBA APB bus. Figure 6.29-1 shows the block diagram of the C_CAN.

- CAN Core
  - CAN Protocol Controller and Rx/Tx Shift Register for serial/parallel conversion of messages.
- Message RAM
  - Stores Message Objects and Identifier Masks
- Registers
  - All registers used to control and to configure the C_CAN.
  - Message Handler
  - State Machine that controls the data transfer between the Rx/Tx Shift Register of the CAN Core and the Message RAM as well as the generation of interrupts as programmed in the Control and Configuration Registers.

- Module Interface
  - C_CAN interfaces to the AMBA APB 16-bit bus from CPU.

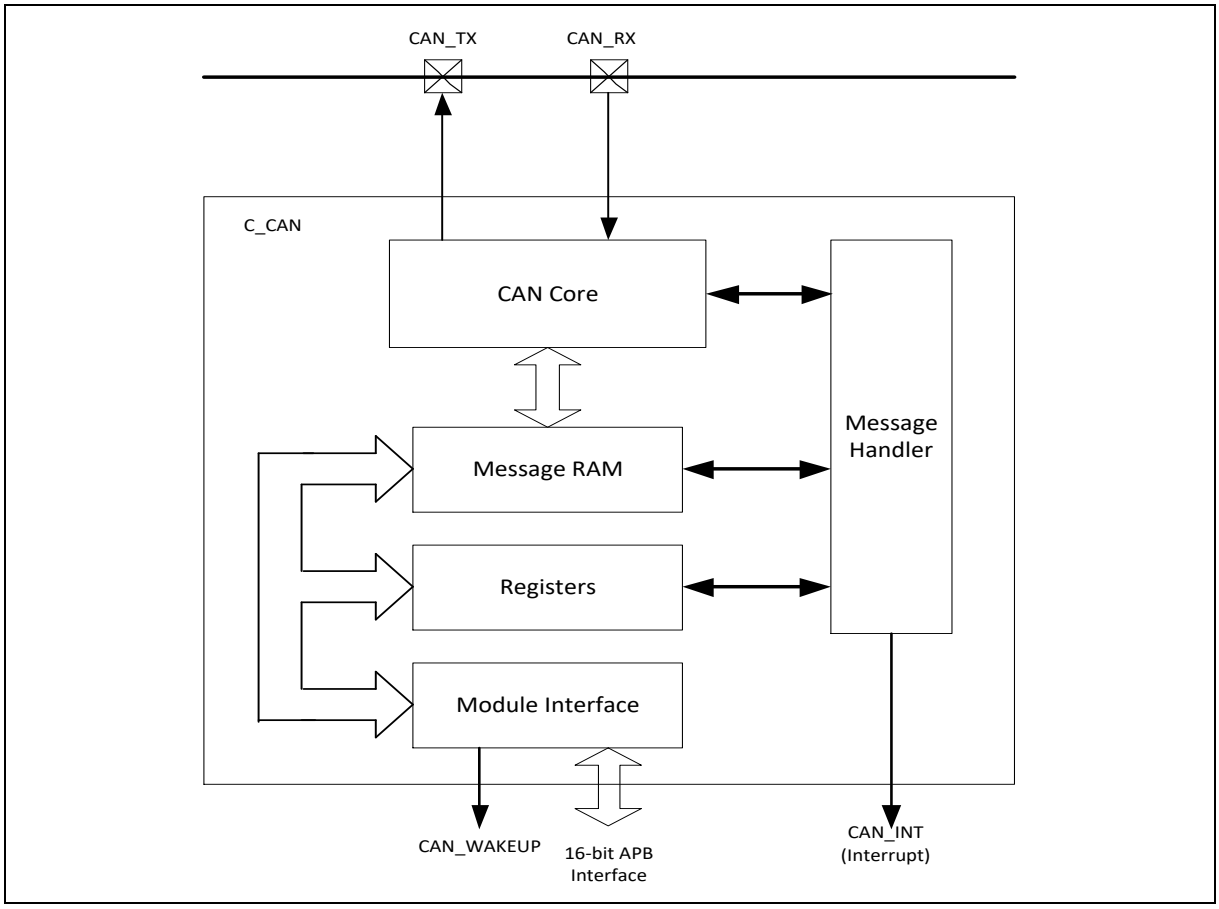


Figure 6.29-1 CAN Peripheral Block Diagram

6.29.4 Basic Configuration

6.29.4.1 CAN Basic Configuration

- Clock source Configuration
  - Enable CAN clock (CAN_EN (APBCLK0[24])).
- Reset Configuration
  - Reset CAN controller (CAN_RST (IPRSTC2[24])).
- Pin Configuration

Group	Pin Name	GPIO	MFP
CAN0	CAN0_RXD	PD.10, PE.15	MFP4
		PA.13	MFP6
		PB.10	MFP8
		PA.4, PC.4	MFP10
	CAN0_TXD	PD.11, PE.14	MFP4

		PA.12	MFP6
		PB.11	MFP8
		PC.5	MFP10

6.29.5 Functional Description

6.29.5.1 Software Initialization

The software initialization is started by setting the Init bit (CAN_CON[0]), either by a software or a hardware reset, or by going to bus-off state.

While the Init bit is set, all messages transfer to and from the CAN bus are stopped and the status of the CAN_TX output pin is recessive (HIGH). The Error Management Logic (EML) counters are unchanged. Setting the Init bit does not change any configuration register.

To initialize the CAN Controller, software has to set up the Bit Timing Register and each Message Object. If a Message Object is not required, the corresponding MsgVal bit (CAN_IFn_ARB2[15]) should be cleared. Otherwise, the entire Message Object has to be initialized.

Access to the Bit Timing Register and to the Baud Rate Prescaler Extension Register for configuring bit timing is enabled when both the Init and CCE (CAN_CON[6]) bits are set.

Resetting the Init bit (by software only) finishes the software initialization. Later, the Bit Stream Processor (BSP) (see Section 6.29.7.15: Configuring the Bit Timing) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (= Bus Idle) before it can take part in bus activities and start the message transfer.

The initialization of the Message Objects is independent of Init and can be done on the fly, but the Message Objects should all be configured to particular identifiers or set to not valid before the BSP starts the message transfer.

To change the configuration of a Message Object during normal operation, the software has to start by resetting the corresponding MsgVal bit. When the configuration is completed, MsgVal bit is set again.

6.29.5.2 CAN Message Transfer

Once the C_CAN is initialized and Init bit (CAN_CON[0]) is reset to zero, the C_CAN Core synchronizes itself to the CAN bus and starts the message transfer.

Received messages are stored in their appropriate Message Objects if they pass the Message Handler's acceptance filtering. The whole message including all arbitration bits, DLC (CAN_IFn_MCON[3:0]) and eight data bytes (CAN_IFn_DAT_A1/2; CAN_IFn_DAT_B1/2) are stored in the Message Object. If the Identifier Mask is used, the arbitration bits which are masked to "don't care" may be overwritten in the Message Object.

Software can read or write each message any time through the Interface Registers and the Message Handler guarantees data consistency in case of concurrent accesses.

Messages to be transmitted are updated by the application software. If a permanent Message Object (arbitration and control bits are set during configuration) exists for the message, only the data bytes are updated and the TxRqst bit (CAN_IFn_MCON[8]) with NewDat bit (CAN_IFn_MCON[15]) are set to start the transmission. If several transmit messages are assigned to the same Message Object (when the number of Message Objects is not sufficient), the whole Message Object has to be configured before the transmission of this message is requested.

The transmission of any number of Message Objects may be requested at the same time. Message objects are transmitted subsequently according to their internal priority. Messages may be updated or set to not valid any time, even when their requested transmission is still pending. The old data will be discarded when a message is updated before its pending transmission has started.

Depending on the configuration of the Message Object, the transmission of a message may be requested autonomously by the reception of a remote frame with a matching identifier.

Disabled Automatic Retransmission

In accordance with the CAN Specification (see ISO11898, 6.3.3 Recovery Management), the C_CAN provides means for automatic retransmission of frames that have lost arbitration or have been disturbed by errors during transmission. The frame transmission service will not be confirmed to the user before the transmission is successfully completed. This means that, by default, automatic retransmission is enabled. It can be disabled to enable the C_CAN to work within a Time Triggered CAN (TTCAN, see ISO11898-1) environment.

The Disabled Automatic Retransmission mode is enabled by setting the Disable Automatic Retransmission (DAR bit (CAN_CON[5])) to one. In this operation mode, the programmer has to consider the different behavior of bits TxRqst (CAN_IFn_MCON[8]) and NewDat (CAN_IFn_MCON[15]) of the Message Buffers:

- When a transmission starts, bit TxRqst of the respective Message Buffer is cleared, while bit NewDat remains set.
- When the transmission completed successfully, bit NewDat is cleared.
- When a transmission fails (lost arbitration or error), bit NewDat remains set.
- To restart the transmission, the software should set the bit TxRqst again.

6.29.6 Test Mode

Test Mode is entered by setting the Test bit (CAN_CON[7]). In Test Mode, bits Tx1 (CAN_TEST[6]), Tx0 (CAN_TEST[5]), LBack (CAN_TEST[4]), Silent (CAN_TEST[3]) and Basic (CAN_TEST[2]) are writeable. Bit Rx (CAN_TEST[7]) monitors the state of the CAN_RX pin and therefore is only readable. All Test Register functions are disabled when the Test bit is cleared.

6.29.6.1 Silent Mode

The CAN Core can be set in Silent Mode by programming the Silent bit (CAN_TEST[3]) to one. In Silent Mode, the C_CAN is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and it cannot start a transmission. If the CAN Core is required to send a dominant bit (ACK bit, Error Frames), the bit is rerouted internally so that the CAN Core monitors this dominant bit, although the CAN bus may remain in recessive state. The Silent Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. Figure 6.29-2 CAN Core in Silent Mode shows the connection of signals CAN_TX and CAN_RX to the CAN Core in Silent Mode.

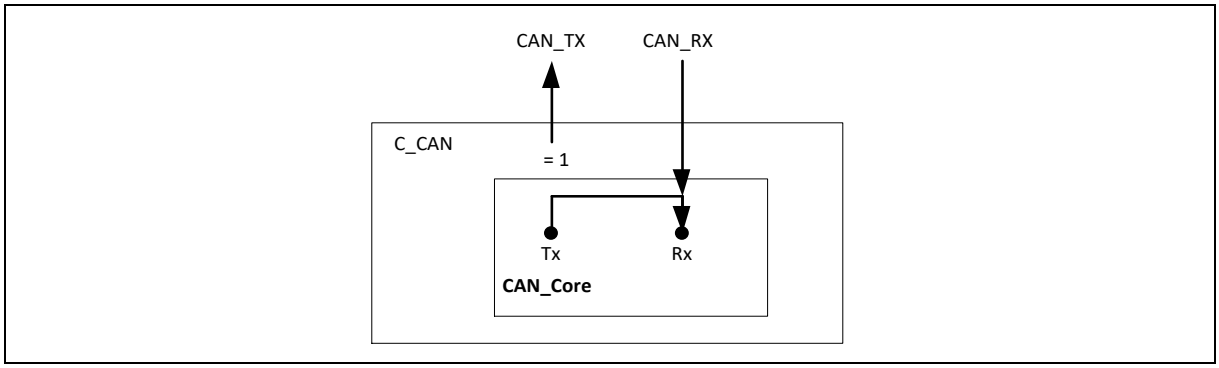


Figure 6.29-2 CAN Core in Silent Mode

6.29.6.2 Loop Back Mode

The CAN Core can be set in Loop Back Mode by programming the Test Register bit LBack (CAN_TEST[4]) to one. In Loop Back Mode, the CAN Core treats its own transmitted messages as received messages and stores them in a Receive Buffer (if they pass acceptance filtering). Figure 6.29-3 shows the connection of signals, CAN_TX and CAN_RX, to the CAN Core in Loop Back Mode.

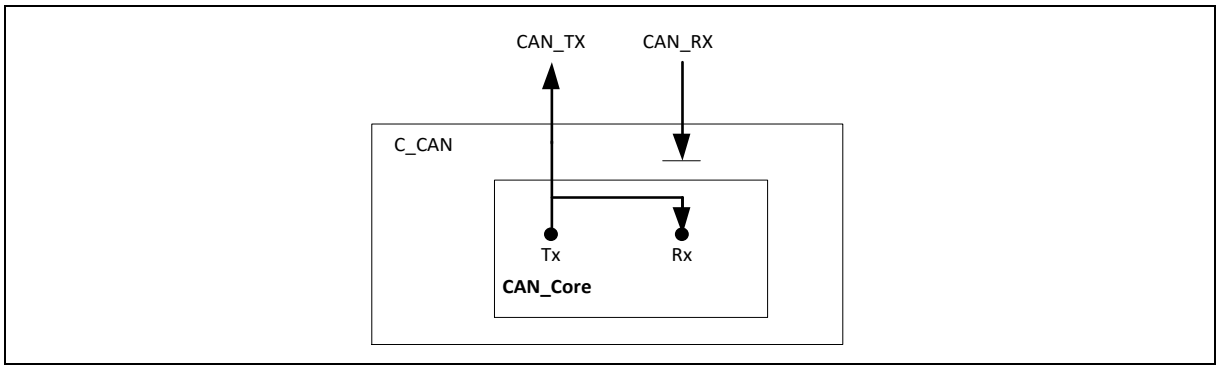


Figure 6.29-3 CAN Core in Loop Back Mode

This mode is provided for self-test functions. To be independent from external stimulation, the CAN Core ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/ remote frame) in Loop Back Mode. In this mode, the CAN Core performs an internal feedback from its Tx output to its Rx input. The actual value of the CAN_RX input pin is disregarded by the CAN Core. The transmitted messages can be monitored on the CAN_TX pin.

6.29.6.3 Loop Back Combined with Silent Mode

It is also possible to combine Loop Back Mode and Silent Mode by programming bits LBack (CAN_TEST[4]) and Silent (CAN_TEST[3]) to one at the same time. This mode can be used for a “Hot Selftest”, which means that C_CAN can be tested without affecting a running CAN system connected to the CAN_TX and CAN_RX pins. In this mode, the CAN_RX pin is disconnected from the CAN Core and the CAN_TX pin is held recessive. Figure 6.29-4 shows the connection of signals CAN_TX and CAN_RX to the CAN Core in case of the combination of Loop Back Mode with Silent Mode.

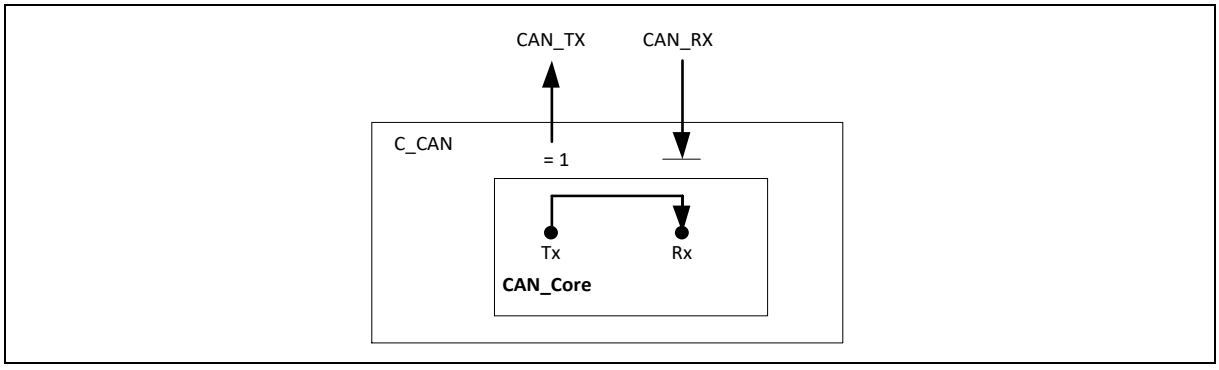


Figure 6.29-4 CAN Core in Loop Back Mode Combined with Silent Mode

6.29.6.4 Basic Mode

The CAN Core can be set in Basic Mode by programming the Basic bit (CAN_TEST[2]) to one. In this mode, the C_CAN runs without the Message RAM.

The IF1 Registers are used as Transmit Buffer. The transmission of the contents of the IF1 Registers is requested by writing the Busy bit (CAN_IFn_CREQ[15]) of the IF1 Command Request Register to one. The IF1 Registers are locked while the Busy bit is set. The Busy bit indicates that the

transmission is pending.

As soon the CAN bus is idle, the IF1 Registers are loaded into the shift register of the CAN Core and the transmission is started. When the transmission has been completed, the Busy bit is reset and the locked IF1 Registers are released.

A pending transmission can be aborted at any time by resetting the Busy bit in the IF1 Command Request Register while the IF1 Registers are locked. If the software has reset the Busy bit, a possible retransmission in case of lost arbitration or in case of an error is disabled.

The IF2 Registers are used as a Receive Buffer. After the reception of a message the contents of the shift register is stored into the IF2 Registers, without any acceptance filtering.

Additionally, the actual contents of the shift register can be monitored during the message transfer. Each time a read Message Object is initiated by writing the Busy bit of the IF2 Command Request Register to one, the contents of the shift register are stored into the IF2 Registers.

In Basic Mode, the evaluation of all Message Object related control and status bits and the control bits of the IFn Command Mask Registers are turned off. The message number of the Command request registers is not evaluated. The NewDat (CAN_IFn_MCON[15]) and MsgLst (CAN_IFn_MCON[14]) bits retain their function, DLC3-0 indicates the received DLC (CAN_IFn_MCON[3:0]), and the other control bits are read as '0'.

6.29.6.5 Software Control of CAN_TX Pin

Four output functions are available for the CAN transmit pin, CAN_TX. In addition to its default function (serial data output), the CAN transmit pin can drive the CAN Sample Point signal to monitor CAN_Core's bit timing and it can drive constant dominant or recessive values. The latter two functions, combined with the readable CAN receive pin CAN_RX, can be used to check the physical layer of the CAN bus.

The output mode for the CAN_TX pin is selected by programming the Tx1 (CAN_TEST[6]) and Tx0 (CAN_TEST[5]) bits.

The three test functions of the CAN_TX pin interfere with all CAN protocol functions. CAN_TX must be left in its default function when CAN message transfer or any of the test modes (Loop Back Mode, Silent Mode or Basic Mode) are selected.

6.29.7 CAN Communications

6.29.7.1 Managing Message Objects

The configuration of the Message Objects in the Message RAM (with the exception of the bits MsgVal, NewDat, IntPnd and TxRqst) will not be affected by resetting the chip. All the Message Objects must be initialized by the application software or they must be "not valid" (MsgVal bit = '0') and the bit timing must be configured before the application software clears the Init bit (CAN_CON[0]).

The configuration of a Message Object is done by programming Mask, Arbitration, Control and Data fields of one of the two interface registers to the desired values. By writing to the corresponding IFn Command Request Register, the IFn Message Buffer Registers are loaded into the addressed Message Object in the Message RAM.

When the Init bit is cleared, the CAN Protocol Controller state machine of the CAN_Core and the state machine of the Message Handler control the internal data flow of the C_CAN. Received messages that pass the acceptance filtering are stored into the Message RAM, messages with pending transmission request are loaded into the CAN_Core's Shift Register and are transmitted through the CAN bus.

The application software reads received messages and updates messages to be transmitted through the IFn Interface Registers. Depending on the configuration, the application software is interrupted on certain CAN message and CAN error events.

6.29.7.2 Message Handler State Machine

The Message Handler controls the data transfer between the Rx/Tx Shift Register of the CAN Core, the Message RAM and the IFn Registers.

The Message Handler FSM controls the following functions:

- Data Transfer from IFn Registers to the Message RAM
- Data Transfer from Message RAM to the IFn Registers
- Data Transfer from Shift Register to the Message RAM
- Data Transfer from Message RAM to Shift Register
- Data Transfer from Shift Register to the Acceptance Filtering unit
- Scanning of Message RAM for a matching Message Object
- Handling of TxRqst flags
- Handling of interrupts.

6.29.7.3 Data Transfer from/to Message RAM

When the application software initiates a data transfer between the IFn Registers and Message RAM, the Message Handler sets the Busy bit (CAN_IFn_CREQ[15]) to '1'. After the transfer has completed, the Busy bit is again cleared (see Figure 6.29-5).

The respective Command Mask Register specifies whether a complete Message Object or only parts of it will be transferred. Due to the structure of the Message RAM, it is not possible to write single bits/bytes of one Message Object. It is always necessary to write a complete Message Object into the Message RAM. Therefore, the data transfer from the IFn Registers to the Message RAM requires a read-modify-write cycle. First, those parts of the Message Object that are not to be changed are read from the Message RAM and then the complete contents of the Message Buffer Registers are written into the Message Object.

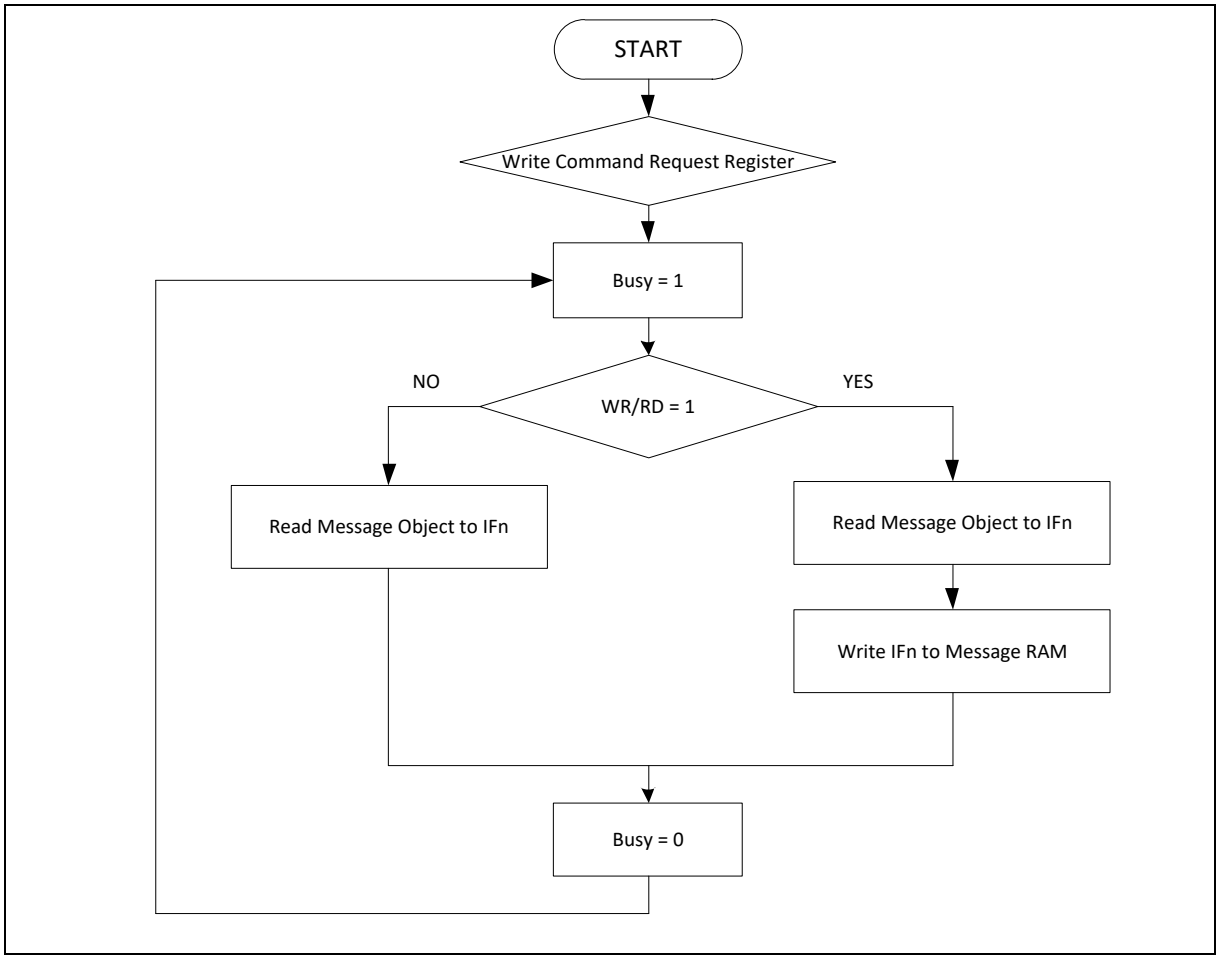


Figure 6.29-5 Data transfer between IFn Registers and Message

After a partial write of a Message Object, the Message Buffer Registers that are not selected in the Command Mask Register will set the actual contents of the selected Message Object.

After a partial read of a Message Object, the Message Buffer Registers that are not selected in the Command Mask Register will be left unchanged.

6.29.7.4 Message Transmission

If the shift register of the CAN Core cell is ready for loading and if there is no data transfer between the IFn Registers and Message RAM, the MsgVal bit (CAN_IFn_ARB2[15]) and TxRqst bits (CAN_TXREQ1/2) are evaluated. The valid Message Object with the highest priority pending transmission request is loaded into the shift register by the Message Handler and the transmission is started. The NewDat (CAN_IFn_MCON[15]) bit of the Message Object is reset.

After a successful transmission and also if no new data was written to the Message Object (NewDat = '0') since the start of the transmission, the TxRqst bit of the Message Control register (CAN_IFn_MCON[8]) will be reset. If TxIE bit (CAN_IFn_MCON[11]) is set, IntPnd bit (CAN_IFn_MCON[13]) of the Interrupt Identifier register will be set after a successful transmission. If the C_CAN has lost the arbitration or if an error occurred during the transmission, the message will be retransmitted as soon as the CAN bus is free again. Meanwhile, if the transmission of a message with higher priority has been requested, the messages will be transmitted in the order of their priority.

6.29.7.5 Acceptance Filtering of Received Messages

When the arbitration and control field (Identifier + IDE + RTR + DLC) of an incoming message is completely shifted into the Rx/Tx Shift Register of the CAN Core, the Message Handler FSM starts the scanning of the Message RAM for a matching valid Message Object.

To scan the Message RAM for a matching Message Object, the Acceptance Filtering unit is loaded with the arbitration bits from the CAN Core shift register. The arbitration and mask fields (including MsgVal (CAN_IFn_ARB2[15]), UMask (CAN_IFn_MCON[12]), NewDat (CAN_IFn_MCON[15]) and EoB (CAN_IFn_MCON[7])) of Message Object 1 are then loaded into the Acceptance Filtering unit and compared with the arbitration field from the shift register. This is repeated with each following Message Object until a matching Message Object is found or until the end of the Message RAM is reached.

If a match occurs, the scan is stopped and the Message Handler FSM proceeds depending on the type of frame (Data Frame or Remote Frame) received.

Reception of Data Frame

The Message Handler FSM stores the message from the CAN Core shift register into the respective Message Object in the Message RAM. Not only the data bytes, but all arbitration bits and the Data Length Code are stored into the corresponding Message Object. This is done to keep the data bytes connected with the identifier even if arbitration mask registers are used.

The NewDat bit (CAN_IFn_MCON[15]) is set to indicate that new data (not yet seen by the software) has been received. The application software should reset NewDat bit when the Message Object has been read. If at the time of reception, the NewDat bit was already set, MsgLst (CAN_IFn_MCON[14]) is set to indicate that the previous data (supposedly not seen by the software) is lost. If the RxIE bit (CAN_IFn_MCON[10]) is set, the IntPnd bit (CAN_IFn_MCON[13]) is set, causing the Interrupt Register to point to this Message Object.

The TxRqst bit (CAN_IFn_MCON[8]) of this Message Object is reset to prevent the transmission of a Remote Frame, while the requested Data Frame has just been received.

Reception of Remote Frame

When a Remote Frame is received, three different configurations of the matching Message Object have to be considered:

1. Dir (CAN_IFn_ARB2[13]) = ‘1’ (direction = transmit), RmtEn (CAN_IFn_MCON[9]) = ‘1’ and UMask (CAN_IFn_MCON[12]) = ‘1’ or ’0’
2. At the reception of a matching Remote Frame, the TxRqst bit of this Message Object is set. The rest of the Message Object remains unchanged.
3. Dir = ‘1’ (direction = transmit), RmtEn = ‘0’ and UMask = ’0’
4. At the reception of a matching Remote Frame, the TxRqst bit of this Message Object remains unchanged; the Remote Frame is ignored.
5. Dir = ‘1’ (direction = transmit), RmtEn = ‘0’ and UMask = ’1’
6. At the reception of a matching Remote Frame, the TxRqst bit of this Message Object is reset. The arbitration and control field (Identifier + IDE + RTR + DLC) from the shift register is stored in the Message Object of the Message RAM and the NewDat bit (CAN_IFn_MCON[15]) of this Message Object is set. The data field of the Message Object remains unchanged; the Remote Frame is treated similar to a received Data Frame.

6.29.7.6 Receive/Transmit Priority

The receive/transmit priority for the Message Objects is attached to the message number. Message Object 1 has the highest priority, while Message Object 32 has the lowest priority. If more than one transmission request is pending, they are serviced due to the priority of the corresponding Message

Object

6.29.7.7 Configuring a Transmit Object

Table 6.29-1 shows how a Transmit Object should be initialized.

Ms	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	1	0	0	0	appl.	0	appl.	0

Table 6.29-1 Initialization of a Transmit Object

Note: appl. = application software.

The Arbitration Register values (ID28-0 (CAN_IFn_ARB1/2) and Xtd bit (CAN_IFn_ARB2[14])) are provided by the application. They define the identifier and type of the outgoing message. If an 11-bit Identifier (“Standard Frame”) is used, it is programmed to ID28 - ID18. The ID17 - ID0 can then be disregarded.

If the TxIE bit (CAN_IFn_MCON[11]) is set, the IntPnd bit (CAN_IFn_MCON[13]) will be set after a successful transmission of the Message Object.

If the RmtEn bit (CAN_IFn_MCON[9]) is set, a matching received Remote Frame will cause the TxRqst bit (CAN_IFn_MCON[8]) to be set; the Remote Frame will autonomously be answered by a Data Frame.

The Data Register values (DLC3-0 (CAN_IFn_MCON[3:0]), Data(0)-(7)) are provided by the application, TxRqst and RmtEn may not be set before the data is valid.

The Mask Registers (Msk28-0, UMask, MXtd and MDir bits) may be used (UMask (CAN_IFn_MCON[12]) = ’1’) to allow groups of Remote Frames with similar identifiers to set the TxRqst bit. The Dir bit (CAN_IFn_ARB2[13]) should not be masked.

6.29.7.8 Updating a Transmit Object

The software may update the data bytes of a Transmit Object any time through the IFn Interface registers, neither MsgVal bit (CAN_IFn_ARB2[15]) nor TxRqst (CAN_IFn_MCON[8]) have to be reset before the update.

Even if only a part of the data bytes are to be updated, all four bytes of the corresponding IFn Data A Register or IFn Data B Register have to be valid before the contents of that register are transferred to the Message Object. Either the application software has to write all four bytes into the IFn Data Register or the Message Object is transferred to the IFn Data Register before the software writes the new data bytes.

When only the (eight) data bytes are updated, first 0x0087 is written to the Command Mask Register and then the number of the Message Object is written to the Command Request Register, concurrently updating the data bytes and setting TxRqst.

To prevent the reset of TxRqst at the end of a transmission that may already be in progress while the data is updated, NewDat (CAN_IFn_MCON[15]) has to be set together with TxRqst.

When NewDat is set together with TxRqst, NewDat will be reset as soon as the new transmission has started.

6.29.7.9 Configuring a Receive Object

Table 6.29-2 shows how a Receive Object should be initialized.

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	0	0	0	appl.	0	0	0	0

Table 6.29-2 Initialization of a Receive Object

The Arbitration Registers values (ID28-0 (CAN_IFn_ARB1/2) and Xtd bit (CAN_IFn_ARB2[14])) are provided by the application. They define the identifier and type of accepted received messages. If an 11-bit Identifier (“Standard Frame”) is used, it is programmed to ID28 - ID18. Then ID17 - ID0 can be disregarded. When a Data Frame with an 11-bit Identifier is received, ID17 - ID0 will be set to ‘0’.

If the RxIE bit (CAN_IFn_MCON[10]) is set, the IntPnd bit (CAN_IFn_MCON[13]) will be set when a received Data Frame is accepted and stored in the Message Object.

The Data Length Code (DLC3-0 (CAN_IFn_MCON[3:0])) is provided by the application. When the Message Handler stores a Data Frame in the Message Object, it will store the received Data Length Code and eight data bytes. If the Data Length Code is less than 8, the remaining bytes of the Message Object will be overwritten by unspecified values.

The Mask Registers (Msk28-0, UMask, MXtd and MDir bits) may be used (UMask (CAN_IFn_MCON[12]) = ’1’) to allow groups of Data Frames with similar identifiers to be accepted. The Dir bit (CAN_IFn_ARB2[13]) should not be masked in typical applications.

6.29.7.10 Handling Received Messages

The application software may read a received message any time through the IFn Interface registers. The data consistency is guaranteed by the Message Handler state machine.

Typically, the software will write first 0x007F to the Command Mask Register and then the number of the Message Object to the Command Request Register. This combination will transfer the whole received message from the Message RAM into the Message Buffer Register. Additionally, the bits NewDat (CAN_IFn_MCON[15]) and IntPnd (CAN_IFn_MCON[13]) are cleared in the Message RAM (not in the Message Buffer).

If the Message Object uses masks for acceptance filtering, the arbitration bits show which of the matching messages have been received.

The actual value of NewDat shows whether a new message has been received since the last time this Message Object was read. The actual value of MsgLst (CAN_IFn_MCON[14]) shows whether more than one message has been received since the last time this Message Object was read. MsgLst will not be automatically reset.

By means of a Remote Frame, the software may request another CAN node to provide new data for a receive object. Setting the TxRqst bit (CAN_IFn_MCON[8]) of a receive object will cause the transmission of a Remote Frame with the receive object’s identifier. This Remote Frame triggers the other CAN node to start the transmission of the matching Data Frame. If the matching Data Frame is received before the Remote Frame could be transmitted, the TxRqst bit is automatically reset.

6.29.7.11 Configuring a FIFO Buffer

With the exception of the EoB bit (CAN_IFn_MCON[7]), the configuration of Receive Objects belonging to a FIFO Buffer is the same as the configuration of a (single) Receive Object, see Section 6.5.7.9: Configuring a Receive Object.

To concatenate two or more Message Objects into a FIFO Buffer, the identifiers and masks (if used) of these Message Objects have to be programmed to matching values. Due to the implicit priority of the Message Objects, the Message Object with the lowest number will be the first Message Object of the FIFO Buffer. The EoB bit of all Message Objects of a FIFO Buffer except the last have to be

programmed to zero. The EoB bit of the last Message Object of a FIFO Buffer is set to one, configuring it as the End of the Block.

6.29.7.12 Receiving Messages with FIFO Buffers

Received messages with identifiers matching to a FIFO Buffer are stored into a Message Object of this FIFO Buffer starting with the Message Object with the lowest message number.

When a message is stored into a Message Object of a FIFO Buffer, the NewDat bit (CAN_IFn_MCON[15]) of this Message Object is set. By setting NewDat while EoB (CAN_IFn_MCON[7]) is zero, the Message Object is locked for further write access by the Message Handler until the application software has written the NewDat bit back to zero.

Messages are stored into a FIFO Buffer until the last Message Object of this FIFO Buffer is reached. If none of the preceding Message Objects is released by writing NewDat to zero, all further messages for this FIFO Buffer will be written into the last Message Object of the FIFO Buffer and therefore overwrite the previous messages.

6.29.7.13 Reading from a FIFO Buffer

When the application software transfers the contents of a Message Object to the IFn Message Buffer register by writing its number to the IFn Command Request Register, the corresponding Command Mask Register should be programmed in such a way that bits NewDat (CAN_IFn_MCON[15]) and IntPnd (CAN_IFn_MCON[13]) are reset to zero (TxRqst/NewDat (CAN_IFn_CMASK[2]) = ‘1’ and ClrIntPnd (CAN_IFn_CMASK[3]) = ‘1’). The values of these bits in the Message Control Register always reflect the status before resetting the bits.

To assure the correct function of a FIFO Buffer, the application software should read the Message Objects starting at the FIFO Object with the lowest message number.

Figure 6.29-6 shows how a set of Message Objects which are concatenated to a FIFO Buffer can be handled by the application software.

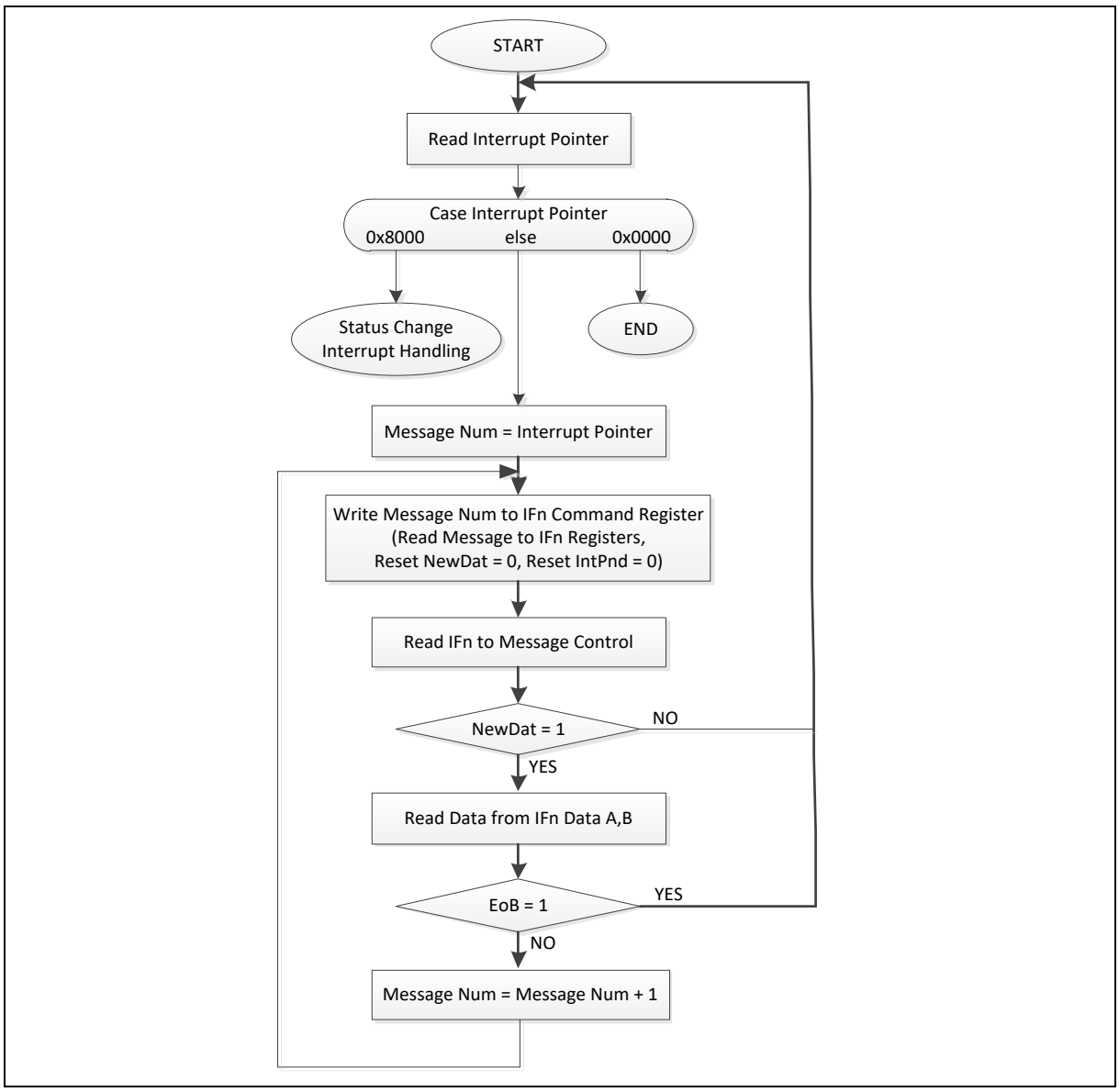


Figure 6.29-6 Application Software Handling of a FIFO Buffer

6.29.7.14 Handling Interrupts

If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority, disregarding their chronological order. An interrupt remains pending until the application software has cleared it.

The Status Interrupt has the highest priority. Among the message interrupts, interrupt priority of the Message Object decreases with increasing message number.

A message interrupt is cleared by clearing the IntPnd bit (CAN_IFn_MCON[13]) of the Message Object. The Status Interrupt is cleared by reading the Status Register.

The interrupt identifier, IntId, in the Interrupt Register, indicates the cause of the interrupt. When no interrupt is pending, the register will hold the value zero. If the value of the Interrupt Register is different from zero, then there is an interrupt pending and, if IE (CAN_CON[1]) is set, the CAN_INT interrupt signal is active. The interrupt remains active until the Interrupt Register is back to value zero

(the cause of the interrupt is reset) or until IE is reset.

The value 0x8000 indicates that an interrupt is pending because the CAN Core has updated (not necessarily changed) the Status Register (Error Interrupt or Status Interrupt). This interrupt has the highest priority. The application software can update (reset) the status bits RxOk (CAN_STATUS[4]), TxOk (CAN_STATUS[3]) and LEC (CAN_STATUS[2:0]), but a write access of the software to the Status Register can never generate or reset an interrupt.

All other values indicate that the source of the interrupt is one of the Message Objects. IntId points to the pending message interrupt with the highest interrupt priority.

The application software controls whether a change of the Status Register may cause an interrupt (bits EIE (CAN_CON[3]) and SIE (CAN_CON[2])) and whether the interrupt line becomes active when the Interrupt Register is different from zero (bit IE in the CAN Control Register). The Interrupt Register will be updated even when IE is reset.

The application software has two possibilities to follow the source of a message interrupt. First, it can follow the IntId in the Interrupt Register and second it can poll the Interrupt Pending Register.

An interrupt service routine that is reading the message that is the source of the interrupt may read the message and reset the Message Object's IntPnd at the same time (bit ClrIntPnd (CAN_IFn_CMASK[3])). When IntPnd is cleared, the Interrupt Register will point to the next Message Object with a pending interrupt.

6.29.7.15 Configuring the Bit Timing

Even if minor errors in the configuration of the CAN bit timing do not result in immediate failure, the performance of a CAN network can be reduced significantly.

In many cases, the CAN bit synchronization will amend a faulty configuration of the CAN bit timing to such a degree that only occasionally an error frame is generated. However, in the case of arbitration, when two or more CAN nodes simultaneously try to transmit a frame, a misplaced sample point may cause one of the transmitters to become error passive.

The analysis of such sporadic errors requires a detailed knowledge of the CAN bit synchronization inside a CAN node and interaction of the CAN nodes on the CAN bus.

6.29.7.16 Bit Time and Bit Rate

CAN supports bit rates in the range of lower than 1 Kbit/s up to 1000 Kbit/s. Each member of the CAN network has its own clock generator, usually a quartz oscillator. The timing parameter of the bit time (i.e. the reciprocal of the bit rate) can be configured individually for each CAN node, creating a common bit rate even though the oscillator periods of the CAN nodes (fosc) may be different.

The frequencies of these oscillators are not absolutely stable, small variations are caused by changes in temperature or voltage and by deteriorating components. As long as the variations remain inside a specific oscillator tolerance range (df), the CAN nodes are able to compensate for the different bit rates by re-synchronizing to the bit stream.

According to the CAN specification, the bit time is divided into four segments (see Figure 6.29-7). The Synchronization Segment, the Propagation Time Segment, the Phase Buffer Segment 1 and the Phase Buffer Segment 2. Each segment consists of a specific, programmable number of time quanta (Table 6.29-3). The length of the time quantum (tq), which is the basic time unit of the bit time, is defined by the CAN controller's APB clock fAPB and the BRP bit (CAN_BTIME[5:0]) : tq = BRP / fAPB.

The Synchronization Segment, Sync_Seg, is that part of the bit time where edges of the CAN bus level are expected to occur. The distance between an edge that occurs outside of Sync_Seg, and the Sync_Seg is called the phase error of that edge. The Propagation Time Segment, Prop_Seg, is intended to compensate for the physical delay time within the CAN network. The Phase Buffer Segments Phase_Seg1 and Phase_Seg2 surround the Sample Point. The (Re-)Synchronization Jump Width (SJW) defines how far a re-synchronization may move the Sample Point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.

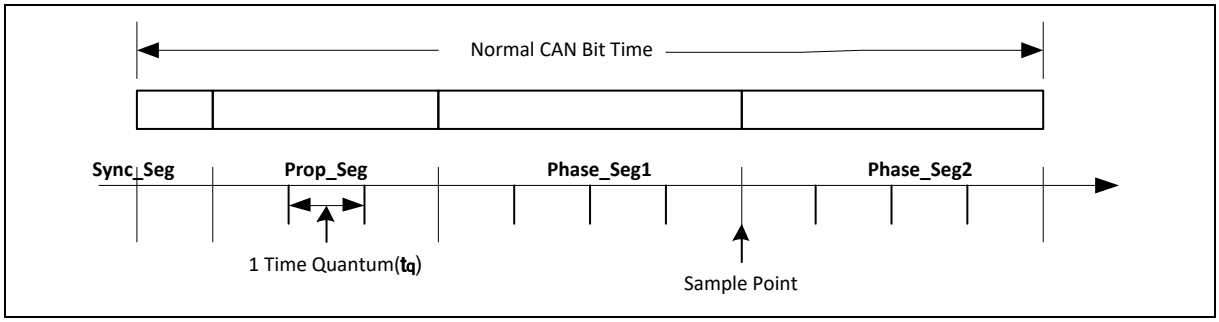


Figure 6.29-7 Bit Timing

Parameter	Range	Remark
BRP	[1..32]	Defines the length of the time quantum t_q
Sync_Seg	1 t_q	Fixed length, synchronization of bus input to APB clock
Prop_Seg	[1..8] t_q	Compensates for the physical delay time
Phase_Seg1	[1..8] t_q	Which may be lengthened temporarily by synchronization
Phase_Seg2	[1..8] t_q	Which may be shortened temporarily by synchronization
SJW	[1..4] t_q	Which may not be longer than either Phase Buffer Segment
This table describes the minimum programmable ranges required by the CAN protocol		

Table 6.29-3 CAN Bit Time Parameters

A given bit rate may be met by different bit time configurations, but for the proper function of the CAN network the physical delay time and the oscillator's tolerance range have to be considered.

6.29.7.17 Propagation Time Segment

This part of the bit time is used to compensate physical delay time within the network. These delay time consist of the signal propagation time on the bus and the internal delay time of the CAN nodes.

Any CAN node synchronized to the bit stream on the CAN bus will be out of phase with the transmitter of that bit stream, caused by the signal propagation time between the two nodes. The CAN protocol's non-destructive bitwise arbitration and the dominant acknowledge bit provided by receivers of CAN messages requires that a CAN node transmitting a bit stream must also be able to receive dominant bits transmitted by other CAN nodes that are synchronized to that bit stream. The example in Figure 6.29-8 shows the phase shift and propagation time between two CAN nodes.

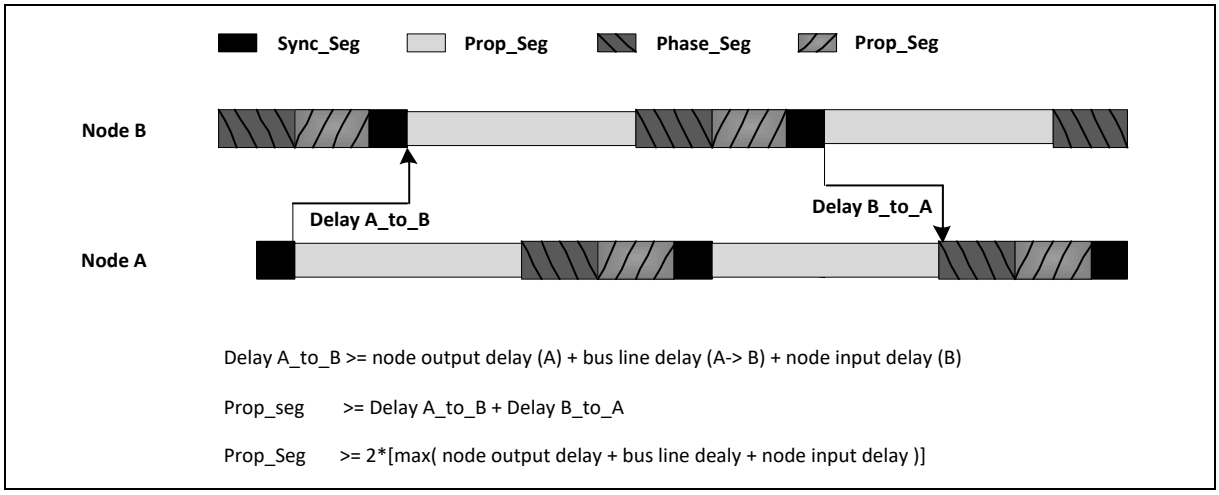


Figure 6.29-8 Propagation Time Segment

In this example, both nodes A and B are transmitters, performing an arbitration for the CAN bus. Node A has sent its Start of Frame bit less than one bit time earlier than node B, therefore node B has synchronized itself to the received edge from recessive to dominant. Since node B has received this edge delay (A_to_B) after it has been transmitted, B's bit timing segments are shifted with respect to A. Node B sends an identifier with higher priority and so it will win the arbitration at a specific identifier bit when it transmits a dominant bit while node A transmits a recessive bit. The dominant bit transmitted by node B will arrive at node A after the delay (B_to_A).

Due to oscillator tolerances, the actual position of node A's Sample Point can be anywhere inside the nominal range of node A's Phase Buffer Segments, so the bit transmitted by node B must arrive at node A before the start of Phase_Seg1. This condition defines the length of Prop_Seg.

If the edge from recessive to dominant transmitted by node B arrives at node A after the start of Phase_Seg1, it can happen that node A samples a recessive bit instead of a dominant bit, resulting in a bit error and the destruction of the current frame by an error flag.

The error occurs only when two nodes arbitrate for the CAN bus that have oscillators of opposite ends of the tolerance range and that are separated by a long bus line. This is an example of a minor error in the bit timing configuration (Prop_Seg is too short) that causes sporadic bus errors.

Some CAN implementations provide an optional 3 Sample Mode but the C_CAN does not. In this mode, the CAN bus input signal passes a digital low-pass filter, using three samples and a majority logic to determine the valid bit value. This results in an additional input delay of 1 tq, requiring a longer Prop_Seg.

6.29.7.18 Phase Buffer Segments and Synchronization

The Phase Buffer Segments (Phase_Seg1 and Phase_Seg2) and the Synchronization Jump Width (SJW) are used to compensate for the oscillator tolerance. The Phase Buffer Segments may be lengthened or shortened by synchronization.

Synchronizations occur on edges from recessive to dominant, their purpose is to control the distance between edges and Sample Points.

Edges are detected by sampling the actual bus level in each time quantum and comparing it with the bus level at the previous Sample Point. A synchronization may be done only if a recessive bit was sampled at the previous Sample Point and if the bus level at the actual time quantum is dominant.

An edge is synchronous if it occurs inside of Sync_Seg, otherwise the distance between edge and the end of Sync_Seg is the edge phase error, measured in time quanta. If the edge occurs before Sync_Seg, the phase error is negative, else it is positive.

Two types of synchronization exist, Hard Synchronization and Re-synchronization.

A Hard Synchronization is done once at the start of a frame and inside a frame only when Re-synchronizations occur.

Hard Synchronization

After a hard synchronization, the bit time is restarted with the end of Sync_Seg, regardless of the edge phase error. Thus hard synchronization forces the edge, which has caused the hard synchronization to lie within the synchronization segment of the restarted bit time.

Bit Re-synchronization

Re-synchronization leads to a shortening or lengthening of the bit time such that the position of the sample point is shifted with regard to the edge.

When the phase error of the edge which causes Re-synchronization is positive, Phase_Seg1 is lengthened. If the magnitude of the phase error is less than SJW, Phase_Seg1 is lengthened by the magnitude of the phase error, else it is lengthened by SJW.

When the phase error of the edge, which causes Re-synchronization is negative, Phase_Seg2 is shortened. If the magnitude of the phase error is less than SJW, Phase_Seg2 is shortened by the magnitude of the phase error, else it is shortened by SJW.

When the magnitude of the phase error of the edge is less than or equal to the programmed value of SJW, the results of Hard Synchronization and Re-synchronization are the same. If the magnitude of the phase error is larger than SJW, the Re-synchronization cannot compensate the phase error completely, an error (phase error - SJW) remains.

Only one synchronization may be done between two Sample Points. The Synchronizations maintain a minimum distance between edges and Sample Points, giving the bus level time to stabilize and filtering out spikes that are shorter than (Prop_Seg + Phase_Seg1).

Apart from noise spikes, most synchronizations are caused by arbitration. All nodes synchronize “hard” on the edge transmitted by the “leading” transceiver that started transmitting first, but due to propagation delay time, they cannot become ideally synchronized. The “leading” transmitter does not necessarily win the arbitration, therefore the receivers have to synchronize themselves to different transmitters that subsequently “take the lead” and that are differently synchronized to the previously “leading” transmitter. The same happens at the acknowledge field, where the transmitter and some of the receivers will have to synchronize to that receiver that “takes the lead” in the transmission of the dominant acknowledge bit.

Synchronizations after the end of the arbitration will be caused by oscillator tolerance, when the differences in the oscillator’s clock periods of transmitter and receivers sum up during the time between synchronizations (at most ten bits). These summarized differences may not be longer than the SJW, limiting the oscillator’s tolerance range.

The examples in Figure 6.29-9 show how the Phase Buffer Segments are used to compensate for phase errors. There are three drawings of each two consecutive bit timings. The upper drawing shows the synchronization on a “late” edge, the lower drawing shows the synchronization on an “early” edge, and the middle drawing is the reference without synchronization.

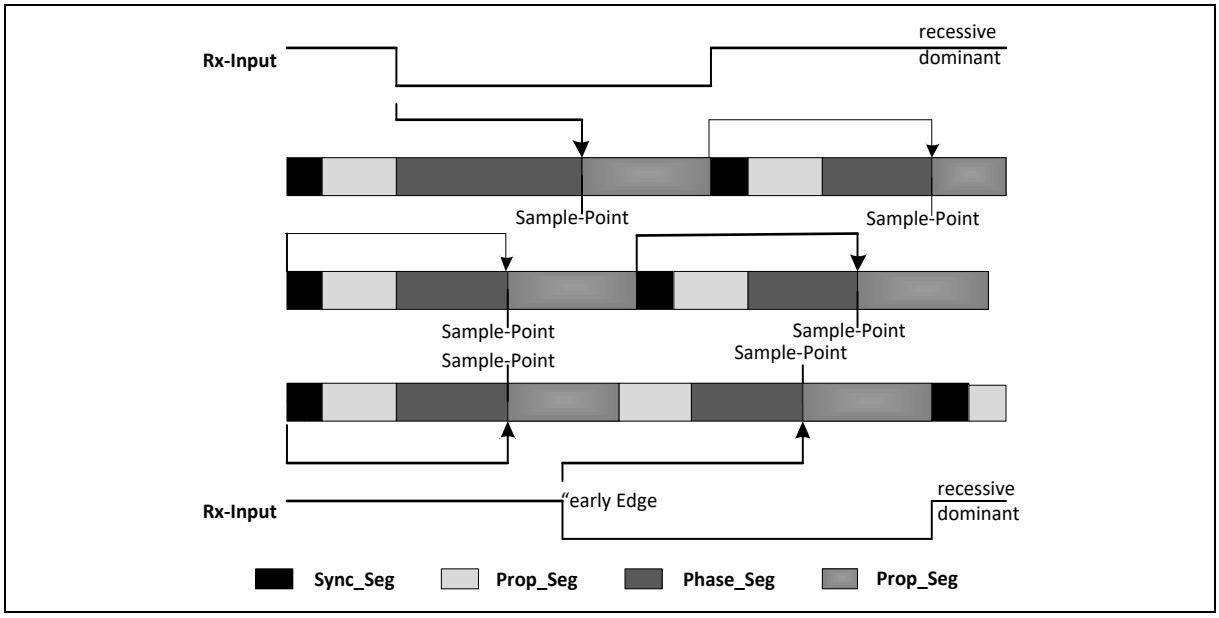


Figure 6.29-9 Synchronization on "late" and "early" Edges

In the first example an edge from recessive to dominant occurs at the end of Prop_Seg. The edge is "late" since it occurs after the Sync_Seg. Reacting to the "late" edge, Phase_Seg1 is lengthened so that the distance from the edge to the Sample Point is the same as it would have been from the Sync_Seg to the Sample Point if no edge had occurred. The phase error of this "late" edge is less than SJW, so it is fully compensated and the edge from dominant to recessive at the end of the bit, which is one nominal bit time long, occurs in the Sync_Seg.

In the second example an edge from recessive to dominant occurs during Phase_Seg2. The edge is "early" since it occurs before a Sync_Seg. Reacting to the "early" edge, Phase_Seg2 is shortened and Sync_Seg is omitted, so that the distance from the edge to the Sample Point is the same as it would have been from an Sync_Seg to the Sample Point if no edge had occurred. As in the previous example, the magnitude of this "early" edge's phase error is less than SJW, so it is fully compensated.

The Phase Buffer Segments are lengthened or shortened temporarily only; at the next bit time, the segments return to their nominal programmed values.

In these examples, the bit timing is seen from the point of view of the CAN implementation's state machine, where the bit time starts and ends at the Sample Points. The state machine omits Sync_Seg when synchronising on an "early" edge because it cannot subsequently redefine that time quantum of Phase_Seg2 where the edge occurs to be the Sync_Seg.

The examples in Figure 6.29-11 show how short dominant noise spikes are filtered by synchronisations. In both examples the spike starts at the end of Prop_Seg and has the length of (Prop_Seg + Phase_Seg1).

In the first example, the Synchronization Jump Width is greater than or equal to the phase error of the spike's edge from recessive to dominant. Therefore the Sample Point is shifted after the end of the spike; a recessive bus level is sampled.

In the second example, SJW is shorter than the phase error, so the Sample Point cannot be shifted far enough; the dominant spike is sampled as actual bus level.

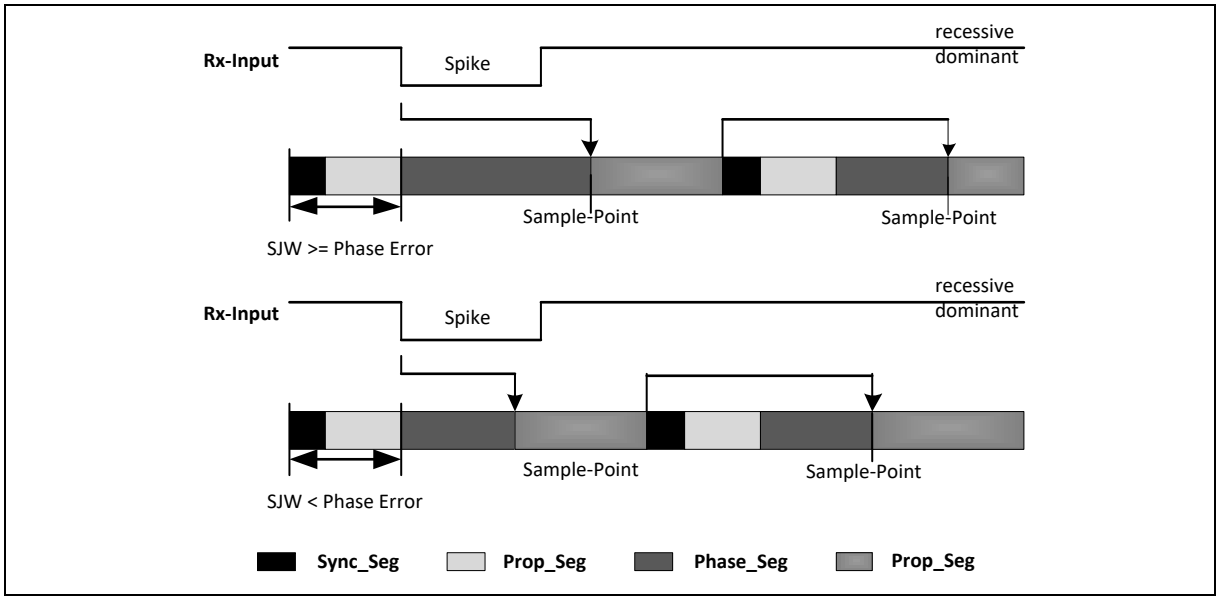


Figure 6.29-10 Filtering of Short Dominant Spikes

6.29.7.19 Oscillator Tolerance Range

The oscillator tolerance range was increased when the CAN protocol was developed from version 1.1 to version 1.2 (version 1.0 was never implemented in silicon). The option to synchronize on edges from dominant to recessive became obsolete, only edges from recessive to dominant are considered for synchronization. The protocol update to version 2.0 (A and B) had no influence on the oscillator tolerance.

The tolerance range d*f* for an oscillator frequency fosc around the nominal frequency fnom is:

(1 – d*f*) • fnom <= fosc <= (1 + d*f*) • fnom

It depends on the proportions of Phase_Seg1, Phase_Seg2, SJW and the bit time. The maximum tolerance df is the defined by two conditions (both shall be met):

$$I: df <= \frac{\min(\text{Phase\_Seg1}, \text{Phase\_Seg2})}{2 * (13 * \text{bit\_time} - Phase\_Seg2}$$

$$II: df <= \frac{\text{SJW}}{20 * \text{bit\_tim}}$$

Note: These conditions base on the APB cock = *f* osc.

It has to be considered that SJW may not be larger than the smaller of the Phase Buffer Segments and that the Propagation Time Segment limits that part of the bit time that may be used for the Phase Buffer Segments.

The combination Prop_Seg = 1 and Phase_Seg1 = Phase_Seg2 = SJW = 4 allows the largest possible oscillator tolerance of 1.58%. This combination with a Propagation Time Segment of only 10% of the bit time is not suitable for short bit times; it can be used for bit rates of up to 125 Kbit/s (bit time = 8us) with a bus length of 40 m.

6.29.7.20 Configuring the CAN Protocol Controller

In most CAN implementations and also in the C_CAN, the bit timing configuration is programmed in two register bytes. The sum of Prop_Seg and Phase_Seg1 (as TSEG1 (CAN_BTIME[11:8])) is combined with Phase_Seg2 (as TSEG2 (CAN_BTIME[14:12])) in one byte, SJW (CAN_BTIME[7:6]) and BRP (CAN_BTIME[5:0]) are combined in the other byte.

In these bit timing registers, the four components TSEG1, TSEG2, SJW and BRP have to be programmed to a numerical value that is one less than its functional value. Therefore, instead of values in the range of [1..n], values in the range of [0..n-1] are programmed. That way, e.g. SJW (functional range of [1..4]) is represented by only two bits.

Therefore the length of the bit time is (programmed values) [TSEG1 + TSEG2 + 3] tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

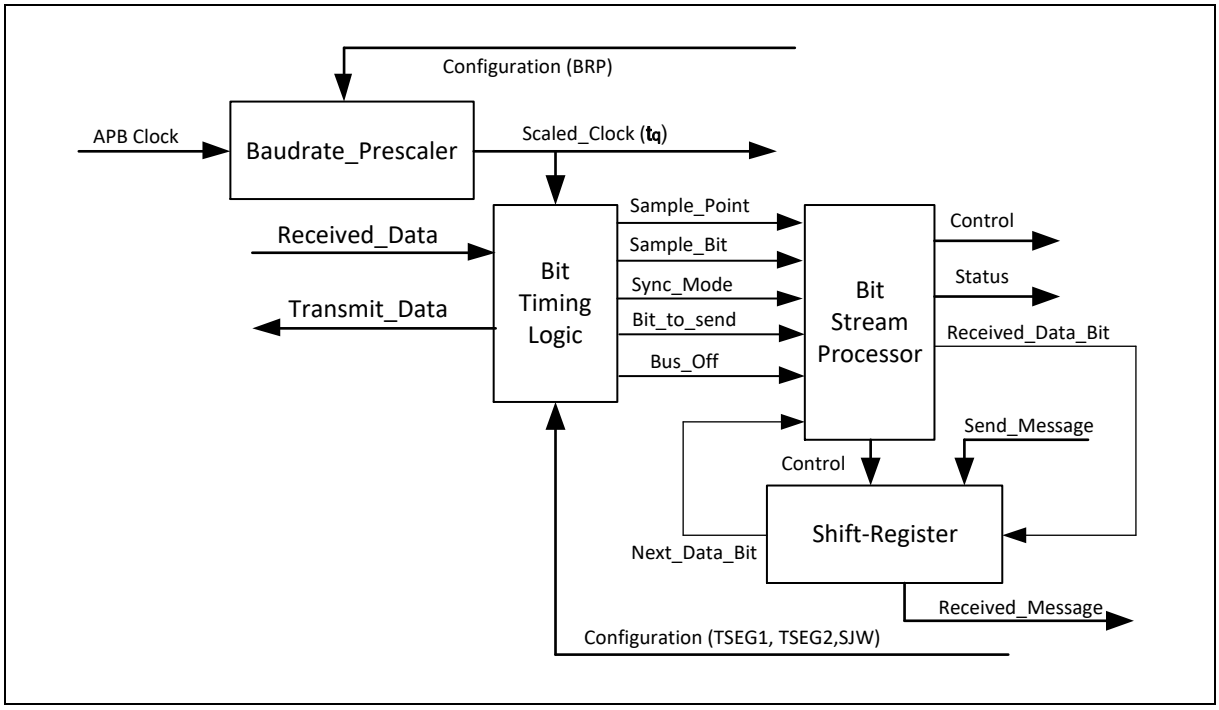


Figure 6.29-11 Structure of the CAN Core's CAN Protocol Controller

The data in the bit timing registers is the configuration input of the CAN protocol controller. The Baud Rate Prescaler (configured by BRP) defines the length of the time quantum, the basic time unit of the bit time; the Bit Timing Logic (configured by TSEG1, TSEG2 and SJW) defines the number of time quanta in the bit time.

The processing of the bit time, the calculation of the position of the Sample Point, and occasional synchronizations are controlled by the BTL (Bit Timing Logic) state machine, which is evaluated once each time quantum. The rest of the CAN protocol controller, the BSP (Bit Stream Processor) state machine is evaluated once each bit time, at the Sample Point.

The Shift Register sends the messages serially and parallelizes received messages. It's loading and shifting is controlled by the BSP.

The BSP translates messages into frames and vice versa. It generates and discards the enclosing fixed format bits, inserts and extracts stuff bits, calculates and checks the CRC code, performs the error management, and decides which type of synchronization is to be used. It is evaluated at the Sample Point and processes the sampled bus input bit. The time that is needed to calculate the next bit to be sent after the Sample point (e.g. data bit, CRC (Cyclic Redundancy Check) bit, stuff bit, error flag or idle) is called the Information Processing Time (IPT).

The IPT is application specific but may not be longer than 2 tq; the IPT for the C_CAN is 0 tq. Its length is the lower limit of the programmed length of Phase_Seg2. In case of a synchronization, Phase_Seg2 may be shortened to a value less than IPT, which does not affect bus timing.

6.29.7.21 Calculating Bit Timing Parameters

Usually, the calculation of the bit timing configuration starts with a desired bit rate or bit time. The resulting bit time (1/bit rate) must be an integer multiple of the APB clock period.

The bit time may consist of 4 to 25 time quanta, the length of the time quantum tq is defined by the Baud Rate Prescaler with tq = (Baud Rate Prescaler)/fapb_clk. Several combinations may lead to the desired bit time, allowing iterations of the following steps.

First part of the bit time to be defined is the Prop_Seg. Its length depends on the delay time measured in the APB clock. A maximum bus length as well as a maximum node delay has to be defined for expandible CAN bus systems. The resulting time for Prop_Seg is converted into time quanta (rounded up to the nearest integer multiple of tq).

The Sync_Seg is 1 tq long (fixed), leaving (bit time – Prop_Seg – 1) tq for the two Phase Buffer Segments. If the number of remaining tq is even, the Phase Buffer Segments have the same length, Phase_Seg2 = Phase_Seg1, else Phase_Seg2 = Phase_Seg1 + 1.

The minimum nominal length of Phase_Seg2 has to be regarded as well. Phase_Seg2 may not be shorter than the IPT of the CAN controller, which, depending on the actual implementation, is in the range of [0..2] tq.

The length of the Synchronization Jump Width is set to its maximum value, which is the minimum of 4 and Phase_Seg1.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formulas given in Section “Oscillator Tolerance Range”.

If more than one configuration is possible, that configuration allowing the highest oscillator tolerance range should be chosen.

CAN nodes with different system clocks require different configurations to come to the same bit rate. The calculation of the propagation time in the CAN network, based on the nodes with the longest delay time, is done once for the whole network.

The oscillator tolerance range of the CAN systems is limited by that node with the lowest tolerance range.

The calculation may shows that bus length or bit rate have to be decreased or that the stability of the oscillator frequency has to be increased in order to find a protocol compliant configuration of the CAN bit timing. The resulting configuration is written into the Bit Timing Register: (Phase_Seg2-1) & (Phase_Seg1+Prop_Seg-1) & (SynchronisationJumpWidth-1) & (Prescaler-1)

Example for Bit Timing at High Baud Rate

In this example, the frequency of APB_CLK is 10 MHz, BRP (CAN_BTIME[5:0]) is 0, and the bit rate is 1 MBit/s.

t_q 100 ns = t_{APB_CLK}

delay of bus driver 50ns

delay of receiver circuit 30ns

delay of bus line (40m) 220ns

t_{Prop} 600ns = 6 • t_q

t_{SJW} 100ns = 1 • t_q

t_{TSeg1} 700ns = t_{Prop} + t_{SJW}

t_{TSeg2} 200ns = Information Processing Time + 1 • t_q

$t_{Sync\text{-}Seg}$ 100ns = 1 • t_q

bit time 1000ns = $t_{Sync\text{-}Seg} + t_{TSeg1} + t_{TSeg2}$

tolerance for APB_CLK 0.39% = $\frac{Min(PB1, PB2)}{2 \times 13 \times (bit\ time - PB2))}$

$= \frac{0.1us}{2 \times 13 \times (1us - 0.2us))}$

In this example, the concatenated bit time parameters are $(2\text{-}1)_3$ & $(7\text{-}1)_4$ & $(1\text{-}1)_2$ & $(1\text{-}1)_6$, and the Bit Timing Register is programmed to 0x1600.

Note:

PB1/2: indicate the phase buffer segment 1/2

The subscript of $(2\text{-}1)_3$ indicates the number of bits in the corresponding bit of Bit Timing Register.

6.29.7.22 CAN Interface Reset State

After the hardware reset, the C_CAN registers hold the reset values which are given in the register description in 0.

Additionally the bus-off state is reset and the output CAN_TX is set to recessive (HIGH). The value 0x0001 (Init = '1') in the CAN Control Register enables the software initialization. The C_CAN does not influence the CAN bus until the application software resets the Init bit (CAN_CON[0]) to '0'.

The data stored in the Message RAM is not affected by a hardware reset. After powered on, the contents of the Message RAM are undefined.

Example for Bit Timing at Low Baud Rate

In this example, the frequency of APB_CLK is 2 MHz, BRP (CAN_BTIME[5:0]) is 1, and the bit rate is 100 Kbit/s.

t_q 1 us = 2 •t_{APB_CLK}

delay of bus driver 200ns

delay of receiver circuit 80ns

delay of bus line (40m) 220ns

t_{Prop} 1us = 1 • t_q

t_{SJW} 4us = 4 • t_q

t_{TSeg1} 5us = t_{Prop} + t_{SJW}

t_{TSeg2} 4us = Information Processing Time + 3 • t_q

$t_{Sync\text{-}Seg}$ 1us = 1 • t_q

bit time 10us = $t_{Sync\text{-}Seg} + t_{TSeg1} + t_{TSeg2}$

tolerance for APB_CLK 1.58 % = $\frac{Min(PB1, PB2)}{2 \times 13 \times (bit\ time - PB2))}$

$= \frac{4us}{2 \times (13 \times (10us - 4us))}$

In this example, the concatenated bit time parameters are $(4\text{-}1)_3$ & $(5\text{-}1)_4$ & $(4\text{-}1)_2$ & $(2\text{-}1)_6$, and the Bit Timing Register is programmed to 0x34C1.

6.29.8 Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
CAN Base Address: **CAN_BA = 0x400A_0000** **CAN non-secure base address is CAN_BA + 0x1000_0000.**				
CAN_CON	CAN_BA+0x00	R/W	CAN Control Register	0x0000_0001
CAN_STATUS	CAN_BA+0x04	R/W	CAN Status Register	0x0000_0000
CAN_ERR	CAN_BA+0x08	R	CAN Error Counter Register	0x0000_0000
CAN_BTIME	CAN_BA+0x0C	R/W	Bit Timing Register	0x0000_2301
CAN_IIDR	CAN_BA+0x10	R	Interrupt Identifier Register	0x0000_0000
CAN_TEST	CAN_BA+0x14	R/W	Test Register (Register Map Note 1)	0x0000_0080
CAN_BRPE	CAN_BA+0x18	R/W	Baud Rate Prescaler Extension Register	0x0000_0000
CAN_IFn_CREQ **n = 1,2**	CAN_BA+0x20 + (0x60 *(n-1))	R/W	IFn (Register Map Note 2) Command Request Registers	0x0000_0001
CAN_IFn_CMASK **n=1,2**	CAN_BA+0x24 + (0x60 *(n-1))	R/W	IFn Command Mask Registers	0x0000_0000
CAN_IFn_MASK1 **n=1,2**	CAN_BA+0x28 + (0x60 *(n-1))	R/W	IFn Mask 1 Registers	0x0000_FFFF
CAN_IFn_MASK2 **n=1,2**	CAN_BA+0x2C + (0x60 *(n-1))	R/W	IFn Mask 2 Registers	0x0000_FFFF
CAN_IFn_ARB1 **n=1,2**	CAN_BA+0x30 + (0x60 *(n-1))	R/W	IFn Arbitration 1 Registers	0x0000_0000
CAN_IFn_ARB2 **n=1,2**	CAN_BA+0x34 + (0x60 *(n-1))	R/W	IFn Arbitration 2 Registers	0x0000_0000
CAN_IFn_MCON **n=1,2**	CAN_BA+0x38 + (0x60 *(n-1))	R/W	IFn Message Control Registers	0x0000_0000
CAN_IFn_DAT_A1 **n=1,2**	CAN_BA+0x3C + (0x60 *(n-1))	R/W	IFn Data A1 Registers (Register Map Note 3)	0x0000_0000
CAN_IFn_DAT_A2 **n=1,2**	CAN_BA+0x40 + (0x60 *(n-1))	R/W	IFn Data A2 Registers (Register Map Note 3)	0x0000_0000
CAN_IFn_DAT_B1 **n=1,2**	CAN_BA+0x44 + (0x60 *(n-1))	R/W	IFn Data B1 Registers (Register Map Note 3)	0x0000_0000
CAN_IFn_DAT_B2 **n=1,2**	CAN_BA+0x48 + (0x60 *(n-1))	R/W	IFn Data B2 Registers (Register Map Note 3)	0x0000_0000
CAN_TXREQ1	CAN_BA+0x100	R	Transmission Request Register 1	0x0000_0000
CAN_TXREQ2	CAN_BA+0x104	R	Transmission Request Register 2	0x0000_0000

CAN_NDAT1	CAN_BA+0x120	R	New Data Register 1	0x0000_0000
CAN_NDAT2	CAN_BA+0x124	R	New Data Register 2	0x0000_0000
CAN_IPND1	CAN_BA+0x140	R	Interrupt Pending Register 1	0x0000_0000
CAN_IPND2	CAN_BA+0x144	R	Interrupt Pending Register 2	0x0000_0000
CAN_MVLD1	CAN_BA+0x160	R	Message Valid Register 1	0x0000_0000
CAN_MVLD2	CAN_BA+0x164	R	Message Valid Register 2	0x0000_0000
CAN_WU_EN	CAN_BA+0x168	R/W	Wake-up Enable Control Register	0x0000_0000
CAN_WU_STATUS	CAN_BA+0x16C	R/W	Wake-up Status Register	0x0000_0000

Note:

1. 0x00 & 0br0000000, where r signifies the actual value of the CAN_RX
2. IFn: The two sets of Message Interface Registers – IF1 and IF2, have identical function
3. An/Bn: The two sets of data registers – A1, A2 and B1, B2.
4. CAN_BA, where x = 0 or 1.

CAN Register Map for Each Bit Function

Addr Offset	Register Name	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00h	CAN_CON	Reserved								Test	CCE	DAR	Res	EIE	SIE	IE	Init
04h	CAN_STATUS	Reserved								BOff	EWarn	EPass	RxOk	TxOk	LEC		
08h	CAN_ERR	RP	REC6-0							TEC7-0							
0Ch	CAN_BTIME	Res	TSeg2			TSeg1				SJW		BRP					
10h	CAN_IIDR	IntId15-8								IntId7-0							
14h	CAN_TEST	Reserved								Rx	Tx1	Tx0	LBack	Silent	Basic	Reserved	
18h	CAN_BRPE	Reserved												BRPE			
20h	CAN_IF1_CREQ	Busy	Reserved									Message Number					
24h	CAN_IF1_CMASK	Reserved								WR/RD	Mask	Arb	Control	ClrIntPnd	TxRqst/	Data A	Data B
28h	CAN_IF1_MASK1	Msk15-0															
2Ch	CAN_IF1_MASK2	MXtd	MDir	Res	Msk28-16												
30h	CAN_IF1_ARB1	ID15-0															
34h	CAN_IF1_ARB2	MsgVal	Xtd	Dir	ID28-16												

Addr Offset	Register Name	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
38h	CAN_IF1_MCON	NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst	EoB	Reserved			DLC3-0			
3Ch	CAN_IF1_DAT_A1	Data(1)								Data(0)							
40h	CAN_IF1_DAT_A2	Data(3)								Data(2)							
44h	CAN_IF1_DAT_B1	Data(5)								Data(4)							
48h	CAN_IF1_DAT_B2	Data(7)								Data(6)							
80h	CAN_IF2_CREQ	Busy	Reserved									Message Number					
84h	CAN_IF2_CMASK	Reserved								WR/RD	Mask	Arb	Control	ClrIntPnd	TxRqst/	Data A	Data B
88h	CAN_IF2_MASK1	Msk15-0															
8Ch	CAN_IF2_MASK2	MXtd	MDir	Res.	Msk28-16												
90h	CAN_IF2_ARB1	ID15-0															
94h	CAN_IF2_ARB2	MsgVal	Xtd	Dir	ID28-16												
98h	CAN_IF2_MCON	NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst	EoB	Reserved			DLC3-0			
9Ch	CAN_IF2_DAT_A1	Data(1)								Data(0)							

Addr Offset	Register Name	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A0h	CAN_IF2_DAT_A2	Data(3)								Data(2)							
A4h	CAN_IF2_DAT_B1	Data(5)								Data(4)							
A8h	CAN_IF2_DAT_B2	Data(7)								Data(6)							
100h	CAN_TXREQ1	TxRqst16-1															
104h	CAN_TXREQ2	TxRqst32-17															
120h	CAN_NDAT1	NewDat16-1															
124h	CAN_NDAT2	NewDat32-17															
140h	CAN_IPND1	IntPnd16-1															
144h	CAN_IPND2	IntPnd32-17															
160h	CAN_MVLD1	MsgVal16-1															
164h	CAN_MVLD2	MsgVal32-17															
168h	CAN_WU_EN	Reserved															WAKUP_EN
16Ch	CAN_WU_STATUS	Reserved															WAKUP_STS
170h	CAN_RAM_CEN	Reserved															RAM_CEN
Others	Reserved	Reserved															

Table 6.29-4 CAN Register Map for Each Bit Function

Note: Reserved bits are read as 0’ except for IFn Mask 2 Register where they are read as ’1’.

Res. = Reserved

6.29.9 Register Description

The C_CAN allocates an address space of 256 bytes. The registers are organized as 16-bit registers.

The two sets of interface registers (IF1 and IF2) control the software access to the Message RAM. They buffer the data to be transferred to and from the RAM, avoiding conflicts between software accesses and message reception/transmission.

CAN Control Register (CAN_CON)

Register	Offset	R/W	Description	Reset Value
CAN_CON	CAN_BA+0x00	R/W	CAN Control Register	0x0000_0001

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Test	CCE	DAR	Reserved	EIE	SIE	IE	Init

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**Test**	**Test Mode Enable Bit** 0 = Normal Operation. 1 = Test Mode.
[6]	**CCE**	**Configuration Change Enable Bit** 0 = No write access to the Bit Timing Register. 1 = Write access to the Bit Timing Register (CAN_BTIME) allowed. (while Init bit (CAN_CON[0]) = 1).
[5]	**DAR**	**Automatic Re-transmission Disable Bit** 0 = Automatic Retransmission of disturbed messages Enabled. 1 = Automatic Retransmission Disabled.
[4]	**Reserved**	Reserved.
[3]	**EIE**	**Error Interrupt Enable Bit** 0 = Disabled - No Error Status Interrupt will be generated. 1 = Enabled - A change in the bits BOff (CAN_STATUS[7]) or EWarn (CAN_STATUS[6]) in the Status Register will generate an interrupt.
[2]	**SIE**	**Status Change Interrupt Enable Bit** 0 = Disabled - No Status Change Interrupt will be generated. 1 = Enabled - An interrupt will be generated when a message transfer is successfully completed or a CAN bus error is detected.
[1]	**IE**	**Module Interrupt Enable Bit** 0 = Funcrion interrupt Disabled. 1 = Funcrion interrupt Enabled.
[0]	**Init**	**Init Initialization** 0 = Normal Operation. 1 = Initialization is started.

Note: The bus-off recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or resetting the Init bit (CAN_CON[0]). If the device goes in the bus-off state, it will set Init of its own accord, stopping all bus activities. Once Init has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before resuming normal operations. At the end of the bus-off recovery sequence, the Error Management Counters will be reset.

During the waiting time after resetting Init, each time a sequence of 11 recessive bits has been monitored, a Bit0Error code is written to the Status Register, enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the proceeding of the bus-off recovery sequence.

CAN Status Register (CAN_STATUS)

Register	Offset	R/W	Description	Reset Value
CAN_STATUS	CAN_BA+0x04	R/W	CAN Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
BOff	EWarn	EPass	RxOK	TxOK	LEC		

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**BOff**	**Bus-off Status (Read Only)** 0 = The CAN module is not in bus-off state. 1 = The CAN module is in bus-off state.
[6]	**EWarn**	**Error Warning Status (Read Only)** 0 = Both error counters are below the error warning limit of 96. 1 = At least one of the error counters in the EML has reached the error warning limit of 96.
[5]	**EPass**	**Error Passive (Read Only)** 0 = The CAN Core is error active. 1 = The CAN Core is in the error passive state as defined in the CAN Specification.
[4]	**RxOK**	**Received a Message Successfully** 0 = No message has been successfully received since this bit was last reset by the CPU. This bit is never reset by the CAN Core. 1 = A message has been successfully received since this bit was last reset by the CPU (independent of the result of acceptance filtering).
[3]	**TxOK**	**Transmitted a Message Successfully** 0 = Since this bit was reset by the CPU, no message has been successfully transmitted. This bit is never reset by the CAN Core. 1 = Since this bit was last reset by the CPU, a message has been successfully (error free and acknowledged by at least one other node) transmitted.
[2:0]	**LEC**	**Last Error Code** The LEC field holds a code, which indicates the type of the last error to occur on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error. The unused code '7' may be written by the CPU to check for updates. Table 6.29-5 Last Error Codedescribes the error code. **Note:** Type of the Last Error to Occur on the CAN Bus**.**

Error Code	Meanings
0	No Error
1	Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
2	Form Error: A fixed format part of a received frame has the wrong format.
3	AckError: The message this CAN Core transmitted was not acknowledged by another node.
4	Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant.
5	Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), though the device wanted to send a dominant level (data or identifier bit logical value '0'), but the monitored Bus value was recessive. During bus-off recovery, this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceedings of the bus-off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).
6	CRCError: The CRC check sum was incorrect in the message received, the CRC received for an incoming message does not match with the calculated CRC for the received data.
7	Unused: When the LEC shows the value '7', no CAN bus event was detected since the CPU wrote this value to the LEC.

Table 6.29-5 Last Error Code

Status Interrupts

A Status Interrupt is generated by bits BOff (CAN_STATUS[7]) and EWarn (CAN_STATUS[6]) (Error Interrupt) or by RxOk (CAN_STATUS[4]), TxOk (CAN_STATUS[3]) and LEC (CAN_STATUS[2:0]) (Status Change Interrupt) assumed that the corresponding enable bits in the CAN Control Register are set. A change of bit EPass (CAN_STATUS[5]) or a write to RxOk, TxOk or LEC will never generate a Status Interrupt.

Reading the Status Register will clear the Status Interrupt value (8000h) in the Interrupt Register, if it is pending.

CAN Error Counter Register (CAN_ERR)

Register	Offset	R/W	Description	Reset Value
CAN_ERR	CAN_BA+0x08	R	CAN Error Counter Register	0x0000_0000

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

Bits	Description	
[31:16]	Reserved	Reserved.
[15]	RP	**Receive Error Passive** 0 = The Receive Error Counter is below the error passive level. 1 = The Receive Error Counter has reached the error passive level as defined in the CAN Specification.
[14:8]	REC	**Receive Error Counter** Actual state of the Receive Error Counter. Values between 0 and 127.
[7:0]	TEC	**Transmit Error Counter** Actual state of the Transmit Error Counter. Values between 0 and 255.

Bit Timing Register (CAN_BTIME)

Register	Offset	R/W	Description	Reset Value
CAN_BTIME	CAN_BA+0x0C	R/W	Bit Timing Register	0x0000_2301

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved	TSeg2			TSeg1			
7	6	5	4	3	2	1	0
SJW		BRP					

Bits	Description	
[31:15]	Reserved	Reserved.
[14:12]	TSeg2	**Time Segment After Sample Point** 0x0-0x7: Valid values for TSeg2 are [0…7]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
[11:8]	TSeg1	**Time Segment Before the Sample Point Minus Sync_Seg** 0x01-0x0F: valid values for TSeg1 are [1…15]. The actual interpretation by the hardware of this value is such that one more than the value programmed is used.
[7:6]	SJW	**Synchronization Jump Width** 0x0-0x3: Valid programmed values are [0…3]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. **Note:** It also means re-synchronization jump width.
[5:0]	BRP	**Baud Rate Prescaler** 0x01-0x3F: The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are [0…63]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Note: With a module clock APB_CLK of 8 MHz, the reset value of 0x2301 configures the C_CAN for a bit rate of 500 Kbit/s. The registers are only writable if bits CCE (CAN_CON[6]) and Init (CAN_CON[0]) are set.

Interrupt Identify Register (CAN_IIDR)

Register	Offset	R/W	Description	Reset Value
CAN_IIDR	CAN_BA+0x10	R	Interrupt Identifier Register	0x0000_0000

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

Bits	Description	
[15:0]	IntId	**Interrupt Identifier** These bits indicates the source of the interrupt. If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority, disregarding their chronological order. An interrupt remains pending until the application software has cleared it. If IntId is different from 0x0000 and IE (CAN_CON[1]) is set, the IRQ interrupt signal to the EIC is active. The interrupt remains active until IntId is back to value 0x0000 (the cause of the interrupt is reset) or until IE is reset. The Status Interrupt has the highest priority. Among the message interrupts, the Message Object' s interrupt priority decreases with increasing message number. A message interrupt is cleared by clearing the Message Object's IntPnd bit (CAN_IFn_MCON[13]). The Status Interrupt is cleared by reading the Status Register.

IntId Value	Meanings
0x0000	No Interrupt is Pending
0x0001-0x0020	Number of Message Object which caused the interrupt.
0x0021-0x7FFF	Unused
0x8000	Status Interrupt
0x8001-0xFFFF	Unused

Table 6.29-6 Source of Interrupts

Test Register (CAN_TEST)

Register	Offset	R/W	Description	Reset Value
CAN_TEST	CAN_BA+0x14	R/W	Test Register (Register Map Note 1)	0x0000_0080

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Rx	Tx		LBack	Silent	Basic	Reserved	

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**Rx**	**Monitors the Actual Value of CAN_RX Pin (Read Only)** 0 = The CAN bus is dominant (CAN_RX = '0'). 1 = The CAN bus is recessive (CAN_RX = '1'). **Note:** Reset value: 0000 0000 R000 0000 b (R:current value of RX pin)
[6:5]	**Tx**	**Tx[1:0]: Control of CAN_TX Pin** 00 = Reset value, CAN_TX pin is controlled by the CAN Core. 01 = Sample Point can be monitored at CAN_TX pin. 10 = CAN_TX pin drives a dominant ('0') value. 11 = CAN_TX pin drives a recessive ('1') value.
[4]	**LBack**	**Loop Back Mode Enable Bit** 0 = Loop Back Mode Disabled. 1 = Loop Back Mode Enabled.
[3]	**Silent**	**Silent Mode** 0 = Normal operation. 1 = The module is in Silent Mode.
[2]	**Basic**	**Basic Mode** 0 = Basic Mode Disabled. 1= IF1 Registers used as Tx Buffer, IF2 Registers used as Rx Buffer.
[1:0]	**Reserved**	Reserved.

Note: Write access to the Test Register is enabled by setting the Test bit (CAN_CON[7]). The different test functions may be combined, but Tx[1-0] "00" (CAN_TEST[6:5]) disturbs message transfer.

Baud Rate Prescaler Extension REGISTER (CAN_BRPE)

Register	Offset	R/W	Description	Reset Value
CAN_BRPE	CAN_BA+0x18	R/W	Baud Rate Prescaler Extension Register	0x0000_0000

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

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3:0]	**BRPE**	**Baud Rate Prescaler Extension** 0x00-0x0F: By programming BRPE, the Baud Rate Prescaler can be extended to values up to 1023. The actual interpretation by the hardware is that one more than the value programmed by BRPE (MSBs) and BTIME (LSBs) is used.

Message Interface Register Sets

There are two sets of Interface Registers, which are used to control the CPU access to the Message RAM. The Interface Registers avoid conflict between the CPU accesses to the Message RAM and CAN message reception and transmission by buffering the data to be transferred. A complete Message Object or parts of the Message Object may be transferred between the Message RAM and the IFn Message Buffer registers in one single transfer.

The function of the two interface register sets is identical except for the Basic test mode. They can be used the way one set of registers is used for data transfer to the Message RAM while the other set of registers is used for the data transfer from the Message RAM, allowing both processes to be interrupted by each other. Table 6.29-7 provides an overview of the two Interface Register sets.

Each set of Interface Registers consists of Message Buffer Registers controlled by their own Command Registers. The Command Mask Register specifies the direction of the data transfer and which parts of a Message Object will be transferred. The Command Request Register is used to select a Message Object in the Message RAM as target or source for the transfer and to start the action specified in the Command Mask Register.

Address	IF1 Register Set	Address	IF2 Register Set
CAN_BA+0x20	IF1 Command Request	CAN_BA+0x80	IF2 Command Request
CAN_BA+0x24	IF1 Command Mask	CAN_BA+0x84	IF2 Command Mask
CAN_BA+0x28	IF1 Mask 1	CAN_BA+0x88	IF2 Mask 1
CAN_BA+0x2C	IF1 Mask 2	CAN_BA+0x8C	IF2 Mask 2
CAN_BA+0x30	IF1 Arbitration 1	CAN_BA+0x90	IF2 Arbitration 1
CAN_BA+0x34	IF1 Arbitration 2	CAN_BA+0x94	IF2 Arbitration 2
CAN_BA+0x38	IF1 Message Control	CAN_BA+0x98	IF2 Message Control
CAN_BA+0x3C	IF1 Data A 1	CAN_BA+0x9C	IF2 Data A 1
CAN_BA+0x40	IF1 Data A 2	CAN_BA+0xA0	IF2 Data A 2
CAN_BA+0x44	IF1 Data B 1	CAN_BA+0xA4	IF2 Data B 1
CAN_BA+0x48	IF1 Data B 2	CAN_BA+0xA8	IF2 Data B 2

Table 6.29-7 IF1 and IF2 Message Interface Register

IFn Command Request Register (CAN_IFn_CREQ)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_CREQ	CAN_BA+0x20 + (0x60 *(n-1))	R/W	IFn (Register Map Note 2) Command Request Registers	0x0000_0001

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**Busy**	**Busy Flag** 0 = Read/write action has finished. 1 = Writing to the IFn Command Request Register is in progress. This bit can only be read by the software.
[14:6]	**Reserved**	Reserved.
[5:0]	**Message Number**	**Message Number** 0x01-0x20: Valid Message Number, the Message Object in the Message RAM is selected for data transfer. 0x00: Not a valid Message Number, interpreted as 0x20. 0x21-0x3F: Not a valid Message Number, interpreted as 0x01-0x1F.

A message transfer is started as soon as the application software has written the message number to the Command Request Register. With this write operation, the Busy bit (CAN_IFn_CREQ[15]) is automatically set to notify the CPU that a transfer is in progress. After a waiting time of 3 to 6 APB_CLK periods, the transfer between the Interface Register and the Message RAM is completed. The Busy bit is cleared.

Note: When a Message Number that is not valid is written into the Command Request Register, the Message Number will be transformed into a valid value and that Message Object will be transferred.

IFn Command Mask Register (CAN_IFn_CMASK)

The control bits of the IFn Command Mask Register specify the transfer direction and select which of the IFn Message Buffer Registers are source or target of the data transfer.

Register	Offset	R/W	Description	Reset Value
CAN_IFn_CMASK	CAN_BA+0x24 + (0x60 *(n-1))	R/W	IFn Command Mask Registers	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
WR_RD	Mask	Arb	Control	ClrIntPnd	TxRqst/NewDat	DAT_A	DAT_B

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**WR_RD**	**Write or Read Mode** 0 = Read: Transfer data from the Message Object addressed by the Command Request Register into the selected Message Buffer Registers. 1 = Write: Transfer data from the selected Message Buffer Registers to the Message Object addressed by the Command Request Register.
[6]	**Mask**	**Access Mask Bits** Write Operation: 0 = Mask bits unchanged. 1 = Transfer Identifier Mask + MDir + MXtd to Message Object. Read Operation: 0 = Mask bits unchanged. 1 = Transfer Identifier Mask + MDir + MXtd to IFn Message Buffer Register.
[5]	**Arb**	**Access Arbitration Bits** Write Operation: 0 = Arbitration bits unchanged. 1 = Transfer Identifier + Dir (CAN_IFn_ARB2[13]) + Xtd (CAN_IFn_ARB2[14]) + MsgVal (CAN_IFn_ARB2[15]) to Message Object. Read Operation: 0 = Arbitration bits unchanged. 1 = Transfer Identifier + Dir + Xtd + MsgVal to IFn Message Buffer Register.
[4]	**Control**	**Control Access Control Bits** Write Operation: 0 = Control Bits unchanged. 1 = Transfer Control Bits to Message Object.

		Read Operation: 0 = Control Bits unchanged. 1 = Transfer Control Bits to IFn Message Buffer Register.
[3]	**ClrIntPnd**	**Clear Interrupt Pending Bit** Write Operation: When writing to a Message Object, this bit is ignored. Read Operation: 0 = IntPnd bit (CAN_IFn_MCON[13]) remains unchanged. 1 = Clear IntPnd bit in the Message Object.
[2]	**TxRqst/NewDat**	**Access Transmission Request Bit When Write Operation** 0 = TxRqst bit unchanged. 1 = Set TxRqst bit. **Note:** If a transmission is requested by programming bit TxRqst/NewDat in the IFn Command Mask Register, bit TxRqst in the IFn Message Control Register will be ignored. Access New Data Bit when Read Operation. 0 = NewDat bit remains unchanged. 1 = Clear NewDat bit in the Message Object. **Note:** A read access to a Message Object can be combined with the reset of the control bits IntPnd and NewDat. The values of these bits transferred to the IFn Message Control Register always reflect the status before resetting these bits.
[1]	**DAT_A**	**Access Data Bytes [3:0]** Write Operation: 0 = Data Bytes [3:0] unchanged. 1 = Transfer Data Bytes [3:0] to Message Object. Read Operation: 0 = Data Bytes [3:0] unchanged. 1 = Transfer Data Bytes [3:0] to IFn Message Buffer Register.
[0]	**DAT_B**	**Access Data Bytes [7:4]** Write Operation: 0 = Data Bytes [7:4] unchanged. 1 = Transfer Data Bytes [7:4] to Message Object. Read Operation: 0 = Data Bytes [7:4] unchanged. 1 = Transfer Data Bytes [7:4] to IFn Message Buffer Register.

IFn Mask 1 Register (CAN_IFn_MASK1)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_MASK1	CAN_BA+0x28 + (0x60 *(n-1))	R/W	IFn Mask 1 Registers	0x0000_FFFF

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**Msk**	**Identifier Mask 15-0** 0 = The corresponding bit in the identifier of the message object cannot inhibit the match in the acceptance filtering. 1 = The corresponding identifier bit is used for acceptance filtering.

IFn Mask 2 Register (CAN_IFn_MASK2)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_MASK2	CAN_BA+0x2C + (0x60 *(n-1))	R/W	IFn Mask 2 Registers	0x0000_FFFF

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**MXtd**	**Mask Extended Identifier** 0 = The extended identifier bit (IDE) has no effect on the acceptance filtering. 1 = The extended identifier bit (IDE) is used for acceptance filtering. **Note:** When 11-bit ("standard") Identifiers are used for a Message Object, the identifiers of received Data Frames are written into bits ID28 to ID18 (CAN_IFn_ARB2[12:2]). For acceptance filtering, only these bits together with mask bits Msk28 to Msk18 (CAN_IFn_MASK2[12:2]) are considered.
[14]	**MDir**	**Mask Message Direction** 0 = The message direction bit (Dir (CAN_IFn_ARB2[13])) has no effect on the acceptance filtering. 1 = The message direction bit (Dir) is used for acceptance filtering.
[13]	**Reserved**	Reserved.
[12:0]	**Msk**	**Identifier Mask 28-16** 0 = The corresponding bit in the identifier of the message object cannot inhibit the match in the acceptance filtering. 1 = The corresponding identifier bit is used for acceptance filtering.

IFn Arbitration 1 Register (CAN_IFn_ARB1)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_ARB1	CAN_BA+0x30 + (0x60 *(n-1))	R/W	IFn Arbitration 1 Registers	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**ID**	**Message Identifier 15-0** ID28 - ID0, 29-bit Identifier (“Extended Frame”). ID28 - ID18, 11-bit Identifier (“Standard Frame”)

IFn Arbitration 2 Register (CAN_IFn_ARB2)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_ARB2	CAN_BA+0x34 + (0x60 *(n-1))	R/W	IFn Arbitration 2 Registers	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
MsgVal	Xtd	Dir	ID				
7	6	5	4	3	2	1	0
ID							

Bits	Description	
[31:16]	Reserved	Reserved.
[15]	MsgVal	**Message Valid** 0 = The Message Object is ignored by the Message Handler. 1 = The Message Object is configured and should be considered by the Message Handler. **Note:** The application software must reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init (CAN_CON[0]). This bit must also be reset before the identifier Id28-0 (CAN_IFn_ARB1/2), the control bits Xtd (CAN_IFn_ARB2[14]), Dir (CAN_IFn_ARB2[13]), or the Data Length Code DLC3-0 (CAN_IFn_MCON[3:0]) are modified, or if the Messages Object is no longer required.
[14]	Xtd	**Extended Identifier** 0 = The 11-bit (“standard”) Identifier will be used for this Message Object. 1 = The 29-bit (“extended”) Identifier will be used for this Message Object.
[13]	Dir	**Message Direction** 0 = Direction is receive. On TxRqst, a Remote Frame with the identifier of this Message Object is transmitted. On reception of a Data Frame with matching identifier, that message is stored in this Message Object. 1 = Direction is transmit. On TxRqst, the respective Message Object is transmitted as a Data Frame. On reception of a Remote Frame with matching identifier, the TxRqst bit (CAN_IFn_CMASK[2]) of this Message Object is set (if RmtEn (CAN_IFn_MCON[9]) = one).
[12:0]	ID	**Message Identifier 28-16** ID28 - ID0, 29-bit Identifier (“Extended Frame”). ID28 - ID18, 11-bit Identifier (“Standard Frame”)

IFn Message Control Register (CAN_IFn_MCON)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_MCON	CAN_BA+0x38 + (0x60 *(n-1))	R/W	IFn Message Control Registers	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst
7	6	5	4	3	2	1	0
EoB	Reserved			DLC			

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**NewDat**	**New Data** 0 = No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the application software. 1 = The Message Handler or the application software has written new data into the data portion of this Message Object.
[14]	**MsgLst**	**Message Lost** 0 = No message lost since last time this bit was reset by the CPU. 1 = The Message Handler stored a new message into this object when NewDat was still set, the CPU has lost a message. **Note:** Only valid for Message Objects with direction = receive.
[13]	**IntPnd**	**Interrupt Pending** 0 = This message object is not the source of an interrupt. 1 = This message object is the source of an interrupt. The Interrupt Identifier in the Interrupt Register will point to this message object if there is no other interrupt source with higher priority.
[12]	**UMask**	**Use Acceptance Mask** 0 = Mask ignored. 1 = Use Mask (Msk28-0, MXtd, and MDir) for acceptance filtering. **Note:** If the UMask bit is set to one, the Message Object's mask bits have to be programmed during initialization of the Message Object before MsgVal bit (CAN_IFn_ARB2[15]) is set to one.
[11]	**TxIE**	**Transmit Interrupt Enable Bit** 0 = IntPnd (CAN_IFn_MCON[13]) will be left unchanged after the successful transmission of a frame. 1 = IntPnd will be set after a successful transmission of a frame.
[10]	**RxIE**	**Receive Interrupt Enable Bit** 0 = IntPnd (CAN_IFn_MCON[13]) will be left unchanged after a successful reception of a frame. 1 = IntPnd will be set after a successful reception of a frame.

[9]	**RmtEn**	**Remote Enable Bit** 0 = At the reception of a Remote Frame, TxRqst (CAN_IFn_MCON[8]) is left unchanged. 1 = At the reception of a Remote Frame, TxRqst is set.
[8]	**TxRqst**	**Transmit Request** 0 = This Message Object is not waiting for transmission. 1 = The transmission of this Message Object is requested and is not yet done.
[7]	**EoB**	**End of Buffer** 0 = Message Object belongs to a FIFO Buffer and is not the last Message Object of that FIFO Buffer. 1 = Single Message Object or last Message Object of a FIFO Buffer. **Note:** This bit is used to concatenate two or more Message Objects (up to 32) to build a FIFO Buffer. For single Message Objects (not belonging to a FIFO Buffer), this bit must always be set to one.
[6:4]	**Reserved**	Reserved.
[3:0]	**DLC**	**Data Length Code** 0-8: Data Frame has 0-8 data bytes. 9-15: Data Frame has 8 data bytes **Note:** The Data Length Code of a Message Object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the Message Handler stores a data frame, it will write the DLC to the value given by the received message. Data(0): 1st data byte of a CAN Data Frame Data(1): 2nd data byte of a CAN Data Frame Data(2): 3rd data byte of a CAN Data Frame Data(3): 4th data byte of a CAN Data Frame Data(4): 5th data byte of a CAN Data Frame Data(5): 6th data byte of a CAN Data Frame Data(6): 7th data byte of a CAN Data Frame Data(7): 8th data byte of a CAN Data Frame **Note:** The Data(0) byte is the first data byte shifted into the shift register of the CAN Core during a reception while the Data(7) byte is the last. When the Message Handler stores a Data Frame, it will write all the eight data bytes into a Message Object. If the Data Length Code is less than 8, the remaining bytes of the Message Object will be overwritten by unspecified values.

IFn Data A1 Register (CAN_IFn_DAT_A1)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_DAT_A1	CAN_BA+0x3C + (0x60 *(n-1))	R/W	IFn Data A1 Registers (Register Map Note 3)	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**Data(1)**	**Data Byte 1** 2nd data byte of a CAN Data Frame
[7:0]	**Data(0)**	**Data Byte 0** 1st data byte of a CAN Data Frame

IFn Data A2 Register (CAN_IFn_DAT_A2)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_DAT_A2	CAN_BA+0x40 + (0x60 *(n-1))	R/W	IFn Data A2 Registers (Register Map Note 3)	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**Data(3)**	**Data Byte 3** 4th data byte of CAN Data Frame
[7:0]	**Data(2)**	**Data Byte 2** 3rd data byte of CAN Data Frame

IFn Data B1 Register (CAN_IFn_DAT_B1)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_DAT_B1	CAN_BA+0x44 + (0x60 *(n-1))	R/W	IFn Data B1 Registers (Register Map Note 3)	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**Data(5)**	**Data Byte 5** 6th data byte of CAN Data Frame
[7:0]	**Data(4)**	**Data Byte 4** 5th data byte of CAN Data Frame

IFn Data B2 Register (CAN_IFn_DAT_B2)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_DAT_B2	CAN_BA+0x48 + (0x60 *(n-1))	R/W	IFn Data B2 Registers (Register Map Note 3)	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**Data(7)**	**Data Byte 7** 8th data byte of CAN Data Frame.
[7:0]	**Data(6)**	**Data Byte 6** 7th data byte of CAN Data Frame.

In a CAN Data Frame, Data [0] is the first, Data [7] is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

Message Object in the Message Memory

There are 32 Message Objects in the Message RAM. To avoid conflicts between application software access to the Message RAM and CAN message reception and transmission, the CPU cannot directly access the Message Objects, these accesses are handled through the IFn Interface Registers. Table 6.29-8 provides an overview of the structures of a Message Object.

Message Object												
UMask	Msk [28:0]	MXtd	MDir	EoB	NewDat		MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
MsgVal	ID [28:0]	Xtd	Dir	DLC [3:0]	Data(0)	Data(1)	Data(2)	Data(3)	Data(4)	Data(5)	Data(6)	Data(7)

Table 6.29-8 Structure of a Message Object in the Message Memory

The Arbitration Registers ID28-0 (CAN_IFn_ARB1/2), Xtd (CAN_IFn_ARB2[14]) and Dir (CAN_IFn_ARB2[13]) are used to define the identifier and type of outgoing messages and are used (together with the mask registers Msk28-0 (CAN_IFn_MASK1/2), MXtd (CAN_IFn_MASK2[15]) and MDir (CAN_IFn_MASK2[14])) for acceptance filtering of incoming messages. A received message is stored in the valid Message Object with matching identifier and Direction = receive (Data Frame) or Direction = transmit (Remote Frame). Extended frames can be stored only in Message Objects with Xtd = one, standard frames in Message Objects with Xtd = zero. If a received message (Data Frame or Remote Frame) matches with more than one valid Message Object, it is stored into that with the lowest message number.

Message Handler Registers

All Message Handler registers are read only. Their contents (TxRqst (CAN_IFn_MCON[8]), NewDat (CAN_IFn_MCON[15]), IntPnd (CAN_IFn_MCON[13]) and MsgVal (CAN_IFn_ARB2[15]) bits of each Message Object and the Interrupt Identifier) are status information provided by the Message Handler FSM.

Transmission Request Register 1 (CAN_TXREQ1)

These registers hold the TxRqst bits of the 32 Message Objects. By reading the TxRqst bits, the software can check which Message Object in a Transmission Request is pending. The TxRqst bit of a specific Message Object can be set/reset by the application software through the IFn Message Interface Registers or by the Message Handler after reception of a Remote Frame or after a successful transmission.

Register	Offset	R/W	Description	Reset Value
CAN_TXREQ1	CAN_BA+0x100	R	Transmission Request Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TxRqst16-9							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TxRqst8-1							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**TxRqst16-1**	**Transmission Request Bits 16-1 of All Message Objects (Read Only)** 0 = This Message Object is not waiting for transmission. 1 = The transmission of this Message Object is requested and is not yet done.

Transmission Request Register 2 (CAN_TXREQ2)

Register	Offset	R/W	Description	Reset Value
CAN_TXREQ2	CAN_BA+0x104	R	Transmission Request Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TxRqst32-25							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TxRqst24-17							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**TxRqst32-17**	**Transmission Request Bits 32-17 of All Message Objects (Read Only)** 0 = This Message Object is not waiting for transmission. 1 = The transmission of this Message Object is requested and is not yet done.

New Data Register 1 (CAN_NDAT1)

These registers hold the NewDat bits of the 32 Message Objects. By reading out the NewDat bits, the software can check for which Message Object the data portion was updated. The NewDat bit of a specific Message Object can be set/reset by the software through the IFn Message Interface Registers or by the Message Handler after reception of a Data Frame or after a successful transmission.

Register	Offset	R/W	Description	Reset Value
CAN_NDAT1	CAN_BA+0x120	R	New Data Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NewData16-9							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NewData8-1							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**NewData16-1**	**New Data Bits 16-1 of All Message Objects** 0 = No new data has been written into the data portion of this Message Object by the Message Handler since the last time this flag was cleared by the application software. 1 = The Message Handler or the application software has written new data into the data portion of this Message Object.

New Data Register 2 (CAN_NDAT2)

Register	Offset	R/W	Description	Reset Value
CAN_NDAT2	CAN_BA+0x124	R	New Data Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
NewData32-25							
7	6	5	4	3	2	1	0
NewData24-17							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**NewData32-17**	**New Data Bits 32-17 of All Message Objects** 0 = No new data has been written into the data portion of this Message Object by the Message Handler since the last time this flag was cleared by the application software. 1 = The Message Handler or the application software has written new data into the data portion of this Message Object.

Interrupt Pending Register 1 (CAN_IPND1)

These registers contain the IntPnd bits of the 32 Message Objects. By reading the IntPnd bits, the software can check for which Message Object an interrupt is pending. The IntPnd bit of a specific Message Object can be set/reset by the application software through the IFn Message Interface Registers or by the Message Handler after reception or after a successful transmission of a frame. This will also affect the value of IntId in the Interrupt Register.

Register	Offset	R/W	Description	Reset Value
CAN_IPND1	CAN_BA+0x140	R	Interrupt Pending Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
IntPnd16-9							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IntPnd8-1							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**IntPnd16-1**	**Interrupt Pending Bits 16-1 of All Message Objects** 0 = This message object is not the source of an interrupt. 1 = This message object is the source of an interrupt.

Interrupt Pending Register 2 (CAN_IPND2)

Register	Offset	R/W	Description	Reset Value
CAN_IPND2	CAN_BA+0x144	R	Interrupt Pending Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
IntPnd32-25							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IntPnd24-17							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**IntPnd32-17**	**Interrupt Pending Bits 32-17 of All Message Objects** 0 = This message object is not the source of an interrupt. 1 = This message object is the source of an interrupt.

Message Valid Register 1 (CAN_MVLD1)

These registers hold the MsgVal bits of the 32 Message Objects. By reading the MsgVal bits, the application software can check which Message Object is valid. The MsgVal bit of a specific Message Object can be set/reset by the application software via the IFn Message Interface Registers.

Register	Offset	R/W	Description	Reset Value
CAN_MVLD1	CAN_BA+0x160	R	Message Valid Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
MsgVal16- 9							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MsgVal8-1							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**MsgVal16-1**	**Message Valid Bits 16-1 of All Message Objects (Read Only)** 0 = This Message Object is ignored by the Message Handler. 1 = This Message Object is configured and should be considered by the Message Handler. **Note:** CAN_MVLD1[0] means Message object No.1 is valid or not. If CAN_MVLD1[0] is set, message object No.1 is configured.

Message Valid Register 2 (CAN_MVLD2)

Register	Offset	R/W	Description	Reset Value
CAN_MVLD2	CAN_BA+0x164	R	Message Valid Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
MsgVal32-25							
7	6	5	4	3	2	1	0
MsgVal24-17							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**MsgVal32-17**	**Message Valid Bits 32-17 of All Message Objects (Read Only)** 0 = This Message Object is ignored by the Message Handler. 1 = This Message Object is configured and should be considered by the Message Handler. **Note:** CAN_MVLD2[15] means Message object No.32 is valid or not. If CAN_MVLD2[15] is set, message object No.32 is configured.

Wake-up Enable Control Register (CAN_WU_EN)

Register	Offset	R/W	Description	Reset Value
CAN_WU_EN	CAN_BA+0x168	R/W	Wake-up Enable Control Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**WAKUP_EN**	**Wake-up Enable Bit** 0 = The wake-up function Disabled. 1 = The wake-up function Enabled. **Note:** User can wake up system when there is a falling edge in the CAN_Rx pin.

Wake-up Status Register (CAN_WU_STATUS)

Register	Offset	R/W	Description	Reset Value
CAN_WU_STATUS	CAN_BA+0x16C	R/W	Wake-up Status Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**WAKUP_STS**	**Wake-up Status** 0 = No wake-up event occurred. 1 = Wake-up event occurred. **Note:** This bit can be cleared by writing ‘0’ to it.

6.30 Secure Digital Host Controller (SDH)

6.30.1 Overview

The Secure Digital Host Controller (SD Host) has DMAC unit and SD unit. The DMAC unit provides a DMA (Direct Memory Access) function for SD to exchange data between system memory and shared buffer (128 bytes), and the SD unit controls the interface of SD/SDHC. The SD host controller can support SD/SDHC and cooperated with DMAC to provide a fast data transfer between system memory and cards.

6.30.2 Features

- AMBA AHB master/slave interface compatible, for data transfer and register read/write.
- Supports single DMA channel.
- Supports hardware Scatter-Gather function.
- Using single 128 Bytes shared buffer for data exchange between system memory and cards.
- Synchronous design for DMA with single clock domain, AHB bus clock (HCLK).
- Interface with DMAC for register read/write and data transfer.
- Supports SD/SDHC card.
- Completely asynchronous design for Secure Digital with two clock domains, HCLK and Engine clock, note that frequency of HCLK should be higher than the frequency of peripheral clock.

6.30.3 Block Diagram

The block diagram and Card Pad Assignment of SD host Controller is shown as follows.

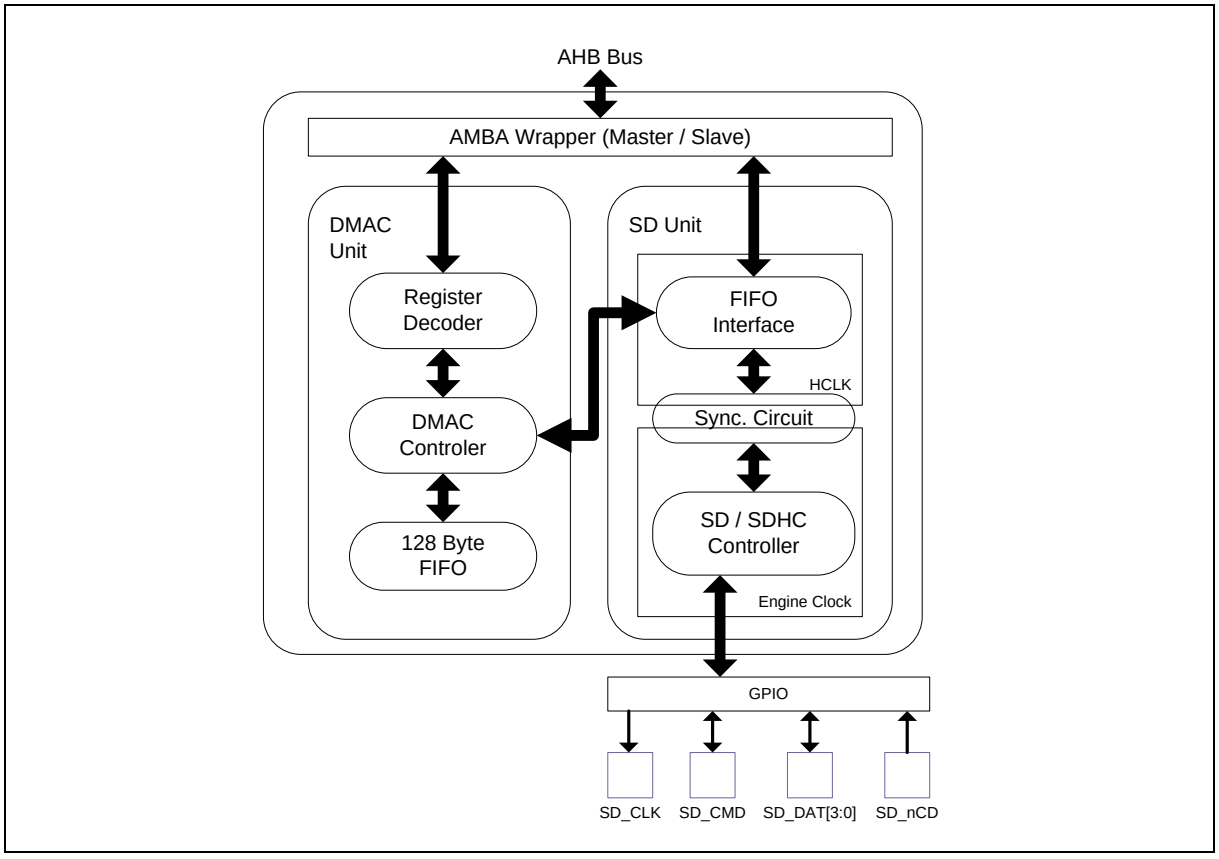


Figure 6.30-1 SD Host Controller Block Diagram

6.30.4 Basic Configuration

6.30.4.1 SD0 Basic Configuration

- Clock source Configuration
  - Select the source of SD0 engine clock on SDH0SEL (CLK_CLKSEL0[21:20]).
  - Select the clock divider number of SD0 engine clock on SDH0DIV (CLK_CLKDIV0[31:24]).
  - Enable SD0 engine clock in SDH0CKEN (CLK_AHBCLK[6]).
- Reset Configuration
  - Reset SD0 controller in SDH0RST (SYS_IPRST0[6]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
SD0	SD0_CLK	PB.1, PE.6	MFP3
	SD0_CMD	PB.0, PE.7	MFP3
	SD0_DAT0	PB.2, PE.2	MFP3

	SD0_DAT1	PB.3, PE.3	MFP3
	SD0_DAT2	PB.4, PE.4	MFP3
	SD0_DAT3	PB.5, PE.5	MFP3
	SD0_nCD	PB.12	MFP9

Table 6.30-1 SD0 Pin Configuration

6.30.5 Functional Description

The SD host provides an interface for SD/SDHC card access with 1-bit/4-bit data bus width.

The SD controller uses an independent clock source named SDCLK as engine clock. SDCLK can be completely asynchronous with system clock HCLK, software can change SD clock arbitrary. However, HCLK should be faster than SDCLK.

6.30.5.1 Basic Operation

This SD controller can generate all types of 48-bit command to the SD card and retrieve all types of response from SD card. After response in, the content of response will be stored at SDH_RESP0 register and SDH_RESP1 register. SD controller will calculate CRC7 and check its correctness for response. If CRC7 is error, CRCIF (SDH_INTSTS[1]) will be set and CRC7 (SDH_INTSTS[2]) will be '0'. For response R1b, software should notice that after response in, SD card will put busy signal on data line DAT0; software should check this status with clock polling until it became high. For response R3, CRC7 is invalid; but SD controller will still calculate CRC7 and get an error result, software should ignore this error and clear CRCIF flag (SDH_INTSTS[1]).

The SD controller is composed of two state machines – command/response part and data part. For command/response part, the trigger bits are COEN (SDH_CTL[0]), RIEN (SDH_CTL[1]), R2EN (SDH_CTL[4]), CLK74OEN (SDH_CTL[5]) and CLK8OEN (SDH_CTL[6]) in SDH_CTL register. If software enables all of these bits, the execution priority will be CLK74OEN (SDH_CTL[5])> COEN (SDH_CTL[0]) > RIEN (SDH_CTL[1])/R2EN (SDH_CTL[4]) > CLK8OEN (SDH_CTL[6]), note that RIEN (SDH_CTL[1]) and R2EN (SDH_CTL[4]) can't be triggered at the same time. For data part, there are DIEN and DOEN for selection. Software can only trigger one of them at one time. If DIEN is triggered, the SD controller waits start bit from data line DAT0 immediately, and then get the specified amount data from SD card. After data-in, the SD controller will check the correctness of CRC16; if it is incorrect, CRCIF (SDH_INTSTS[1]) will be set and CRC16 (SDH_INTSTS[3]) will be '0'. If DOEN is triggered, the SD controller will wait until response in is finished, and then send specified amount data to the SD card. After data-out, the SD controller will get CRC status from SD card and check its correctness; it should be '010', otherwise CRCIF (SDH_INTSTS[1]) will be set and CRCSTS (SDH_INTSTS[6:4]) will be the value it received.

If R2EN (SDH_CTL[4]) is triggered, the SD controller will receive response R2 (136 bits) from SD card, CRC7 and end bit will be dropped. The receiving data will be placed at DMA's buffer, starting from address offset 0x0.

6.30.5.2 Multiple Block Transfer

The SD controller also provides multiple block transfer function (change BLKLEN to change the block length). Software can use this function to accelerate data transfer throughput. If CRC7, CRC16 or CRC status is error, SD controller will stop transfer and set CRCIF (SDH_INTSTS[1]), software should do engine reset when this situation occurred.

There is a hardware time-out mechanism for response in and data in inside SD engine. Software can specify a 24-bit time-out value at TOUT, and then SD controller will decide when to time-out according to this value.

6.30.5.3 DMA Controller

The SD host DMA controller provides a DMA (Direct Memory Access) function for SD host controller to exchange data between system memory (SRAM) and shared buffer (128 bytes). Arbitration of DMA request between SD host is done by DMA's bus master. Software just simply fills in the starting address and enables DMA, and then let DMA to handle the data transfer automatically.

There is a 128 bytes shared buffer inside DMA, it can provide multi-block transfers for SD host. Software can access these shared buffers directly when SD host is not in busy.

6.30.5.4 Programming Flow

Here is a simple example programming flow without DMA Scatter-Gather enabled.

1. Set DMAEN (SDH_DMACTL[0]) to enable DMA.
2. Fill corresponding starting address in SDH_DMASA for SD host.
3. Trigger SD host to start DMA transfer.
4. Wait until transfer is finished.

Here is a simple example programming flow with DMA Scatter-Gather enabled.

1. Set DMAEN (SDH_DMACTL[0]) to enable DMA and SGEN (SDH_DMACTL[3]) to enable Scatter-Gather function.
2. Fill corresponding starting address of Physical Address Descriptor (PAD) table in SDH_DMASA for SD host.
3. When bit-0 of SDH_DMASA is 1, the PAD will fetch in out of order, otherwise, it is fetched in order from PAD. The first time of writing bit-0 with 1 or not is not available for this function. The bits will be available in PAD table.
4. Trigger SD host to start DMA transfer.
5. Wait until transfer is finished.

The format of PAD table is shown in Figure 6.30-2. Note that the total byte count of all Pads must be equal to the byte count filled in SD host. EOT should be set to 1 in the last descriptor.

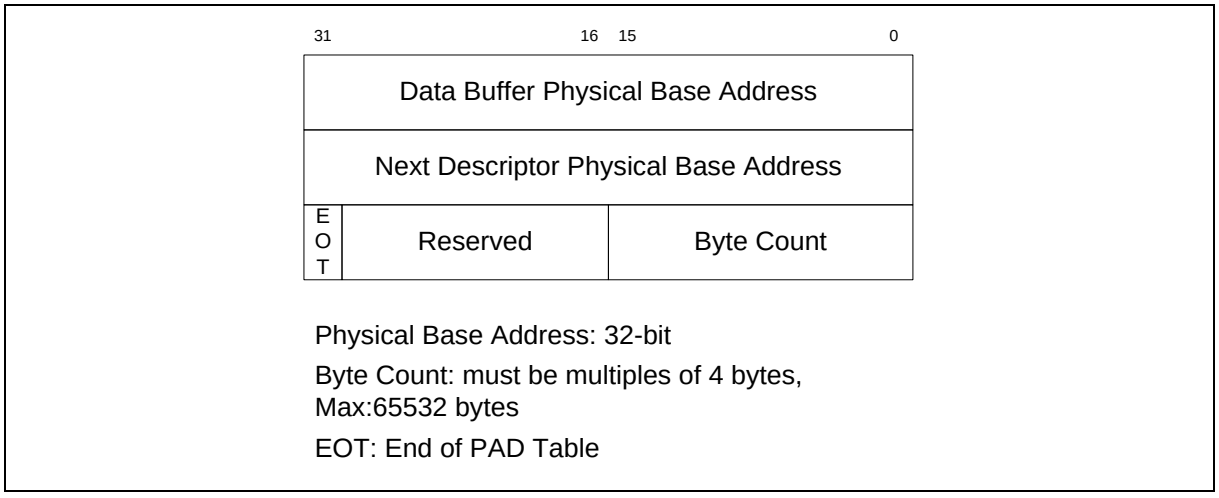


Figure 6.30-2 PAD (Physical Address Descriptor) Table Format

6.30.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
SDH Base address: **SDH0_BA = 0x4000_D000** **SDH non-secure base address is SDH0_BA + 0x1000_0000.**				
SDH_DMACTL	SDH0_BA+0x400	R/W	DMA Control and Status Register	0x0000_0000
SDH_DMASA	SDH0_BA+0x408	R/W	DMA Transfer Starting Address Register	0x0000_0000
SDH_DMABCNT	SDH0_BA+0x40C	R	DMA Transfer Byte Count Register	0x0000_0000
SDH_DMAINTEN	SDH0_BA+0x410	R/W	DMA Interrupt Enable Control Register	0x0000_0001
SDH_DMAINTSTS	SDH0_BA+0x414	R/W	DMA Interrupt Status Register	0x0000_0000
SDH_GCTL	SDH0_BA+0x800	R/W	Global Control and Status Register	0x0000_0000
SDH_GINTEN	SDH0_BA+0x804	R/W	Global Interrupt Control Register	0x0000_0001
SDH_GINTSTS	SDH0_BA+0x808	R/W	Global Interrupt Status Register	0x0000_0000
SDH_CTL	SDH0_BA+0x820	R/W	SD Control and Status Register	0x0101_0000
SDH_CMDARG	SDH0_BA+0x824	R/W	SD Command Argument Register	0x0000_0000
SDH_INTEN	SDH0_BA+0x828	R/W	SD Interrupt Control Register	0x0000_0A00
SDH_INTSTS	SDH0_BA+0x82C	R/W	SD Interrupt Status Register	0x000X_008C
SDH_RESP0	SDH0_BA+0x830	R	SD Receiving Response Token Register 0	0x0000_0000
SDH_RESP1	SDH0_BA+0x834	R	SD Receiving Response Token Register 1	0x0000_0000
SDH_BLEN	SDH0_BA+0x838	R/W	SD Block Length Register	0x0000_01FF
SDH_TOUT	SDH0_BA+0x83C	R/W	SD Response/Data-in Time-out Register	0x0000_0000

6.30.7 Register Description

DMA Control and Status Register (SDH_DMACTL)

Register	Offset	R/W	Description	Reset Value
SDH_DMACTL	SDH0_BA+0x400	R/W	DMA Control and Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						DMABUSY	Reserved
7	6	5	4	3	2	1	0
Reserved				SGEN	Reserved	DMARST	DMAEN

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**DMABUSY**	**DMA Transfer Is in Progress** This bit indicates if SD Host is granted and doing DMA transfer or not. 0 = DMA transfer is not in progress. 1 = DMA transfer is in progress.
[8:4]	**Reserved**	Reserved.
[3]	**SGEN**	**Scatter-gather Function Enable Bit** 0 = Scatter-gather function Disabled (DMA will treat the starting address in DMASA as starting pointer of a single block memory). 1 = Scatter-gather function Enabled (DMA will treat the starting address in DMASA as a starting address of Physical Address Descriptor (PAD) table. The format of these Pads' will be described later).
[2]	**Reserved**	Reserved.
[1]	**DMARST**	**Software Engine Reset** 0 = No effect. 1 = Reset internal state machine and pointers. The contents of control register will not be cleared. This bit will auto be cleared after few clock cycles. **Note:** The software reset DMA related registers.
[0]	**DMAEN**	**DMA Engine Enable Bit** 0 = DMA Disabled. 1 = DMA Enabled. **Note 1:** If this bit is cleared, DMA will ignore all requests from SD host and force bus master into IDLE state. **Note 2:** If target abort occurred, DMAEN will be cleared.

DMA Transfer Starting Address Register (SDH_DMASA)

Register	Offset	R/W	Description	Reset Value
SDH_DMASA	SDH0_BA+0x408	R/W	DMA Transfer Starting Address Register	0x0000_0000

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

Bits	Description	
[31:1]	DMASA	**DMA Transfer Starting Address** This field pads 0 as least significant bit indicates a 32-bit starting address of system memory (SRAM) for DMA to retrieve or fill in data. If DMA is not in normal mode, this field will be interpreted as a starting address of Physical Address Descriptor (PAD) table. **Note:** Starting address of the SRAM must be word aligned, for example, 0x0000_0000, 0x0000_0004.
[0]	ORDER	**Determined to the PAD Table Fetching Is in Order or Out of Order** 0 = PAD table is fetched in order. 1 = PAD table is fetched out of order. **Note:** The bit0 is valid in scatter-gather mode when SGEN = 1.

DMA Transfer Byte Count Register (SDH_DMABCNT)

Register	Offset	R/W	Description	Reset Value
SDH_DMABCNT	SDH0_BA+0x40C	R	DMA Transfer Byte Count Register	0x0000_0000

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

Bits	Description	
[31:26]	Reserved	Reserved.
[25:0]	BCNT	**DMA Transfer Byte Count (Read Only)** This field indicates the remained byte count of DMA transfer. The value of this field is valid only when DMA is busy; otherwise, it is 0.

DMA Interrupt Enable Control Register (SDH_DMAINTEN)

Register	Offset	R/W	Description	Reset Value
SDH_DMAINTEN	SDH0_BA+0x410	R/W	DMA Interrupt Enable Control Register	0x0000_0001

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

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**WEOTIEN**	**Wrong EOT Encountered Interrupt Enable Bit** 0 = Interrupt generation Disabled when wrong EOT is encountered. 1 = Interrupt generation Enabled when wrong EOT is encountered.
[0]	**ABORTIEN**	**DMA Read/Write Target Abort Interrupt Enable Bit** 0 = Target abort interrupt generation Disabled during DMA transfer. 1 = Target abort interrupt generation Enabled during DMA transfer.

DMA Interrupt Status Register (SDH_DMAINTSTS)

Register	Offset	R/W	Description	Reset Value
SDH_DMAINTSTS	SDH0_BA+0x414	R/W	DMA Interrupt Status Register	0x0000_0000

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

Bits	Description	
[31:2]	Reserved	Reserved.
[1]	WEOTIF	**Wrong EOT Encountered Interrupt Flag (Read Only)** When DMA Scatter-Gather function is enabled, and EOT of the descriptor is encountered before DMA transfer finished (that means the total sector count of all PAD is less than the sector count of SD host), this bit will be set. 0 = No EOT encountered before DMA transfer finished. 1 = EOT encountered before DMA transfer finished. **Note:** This bit is read only, but can be cleared by writing '1' to it.
[0]	ABORTIF	**DMA Read/Write Target Abort Interrupt Flag (Read Only)** 0 = No bus ERROR response received. 1 = Bus ERROR response received. **Note 1:** This bit is read only, but can be cleared by writing '1' to it. **Note 2:** When DMA's bus master received ERROR response, it means that target abort happened. DMA will stop transfer and respond this event and then go to IDLE state. When target abort occurred or WEOTIF is set, software must reset DMA and SD host, and then transfer those data again.

Global Control and Status Register (SDH_GCTL)

Register	Offset	R/W	Description	Reset Value
SDH_GCTL	SDH0_BA+0x800	R/W	Global Control and Status Register	0x0000_0000

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

Bits	Description	
[31:2]	Reserved	Reserved.
[1]	SDEN	**Secure Digital Functionality Enable Bit** 0 = SD functionality Disabled. 1 = SD functionality Enabled.
[0]	GCTLRST	**Software Engine Reset** 0 = No effect. 1 = Reset SD host. The contents of control register will not be cleared. This bit will auto cleared after reset complete.

Global Interrupt Control Register (SDH_GINTEN)

Register	Offset	R/W	Description	Reset Value
SDH_GINTEN	SDH0_BA+0x804	R/W	Global Interrupt Control Register	0x0000_0001

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

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	DTAIEN	**DMA READ/WRITE Target Abort Interrupt Enable Bit** 0 = DMA READ/WRITE target abort interrupt generation Disabled. 1 = DMA READ/WRITE target abort interrupt generation Enabled.

Global Interrupt Status Register (SDH_GINTSTS)

Register	Offset	R/W	Description	Reset Value
SDH_GINTSTS	SDH0_BA+0x808	R/W	Global Interrupt Status Register	0x0000_0000

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

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	DTAIF	**DMA READ/WRITE Target Abort Interrupt Flag (Read Only)** This bit indicates DMA received an ERROR response from internal AHB bus during DMA read/write operation. When Target Abort is occurred, please reset all engine. 0 = No bus ERROR response received. 1 = Bus ERROR response received. **Note:** This bit is read only, but can be cleared by writing '1' to it.

SD Control and Status Register (SDH_CTL)

Register	Offset	R/W	Description	Reset Value
SDH_CTL	SDH0_BA+0x820	R/W	SD Control and Status Register	0x0101_0000

31	30	29	28	27	26	25	24
Reserved				SDNWR			
23	22	21	20	19	18	17	16
BLKCNT							
15	14	13	12	11	10	9	8
DBW	CTLRST	CMDCODE					
7	6	5	4	3	2	1	0
CLKKEEP	CLK8OEN	CLK74OEN	R2EN	DOEN	DIEN	RIEN	COEN

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:24]	**SDNWR**	**NWR Parameter for Block Write Operation** This value indicates the NWR parameter for data block write operation in SD clock counts. The actual clock cycle will be SDNWR+1.
[23:16]	**BLKCNT**	**Block Counts to Be Transferred or Received** This field contains the block counts for data-in and data-out transfer. For READ_MULTIPLE_BLOCK and WRITE_MULTIPLE_BLOCK command, software can use this function to accelerate data transfer and improve performance. Do not fill 0x0 to this field. **Note:** For READ_MULTIPLE_BLOCK and WRITE_MULTIPLE_BLOCK command, the actual total length is BLKCNT * (BLKLEN +1).
[15]	**DBW**	**SD Data Bus Width (for 1-bit / 4-bit Selection)** 0 = Data bus width is 1-bit. 1 = Data bus width is 4-bit.
[14]	**CTLRST**	**Software Engine Reset** 0 = No effect. 1 = Reset the internal state machine and counters. The contents of control register will not be cleared (but RIEN (SDH_CTL[1]), DIEN (SDH_CTL[2]), DOEN (SDH_CTL[3]) and R2EN (SDH_CTL[4]) will be cleared). This bit will be auto cleared after few clock cycles.
[13:8]	**CMDCODE**	**SD Command Code** The bits contain the SD command code (0x00 – 0x3F).
[7]	**CLKKEEP**	**SD Clock Enable Control** 0 = SD host decides when to output clock and when to disable clock output automatically. 1 = SD clock always keeps free running.

[6]	**CLK8OEN**	**Generating 8 Clock Cycles Output Enable Bit** 0 = No effect. (Please use DMARST (SDH_DMACTL[1]) to clear this bit.) 1 = Enabled. The SD host will output 8 clock cycles. **Note:** When operation is finished, this bit will be cleared automatically, so don't write 0 to this bit (the controller will be abnormal).
[5]	**CLK74OEN**	**Initial 74 Clock Cycles Output Enable Bit** 0 = No effect. (Please use DMARST (SDH_DMACTL[1]) to clear this bit.) 1 = Enabled. The SD host will output 74 clock cycles to SD card. **Note:** When operation is finished, this bit will be cleared automatically, so don't write 0 to this bit (the controller will be abnormal).
[4]	**R2EN**	**Response R2 Input Enable Bit** 0 = No effect. (Please use DMARST (SDH_DMACTL[1]) to clear this bit.) 1 = Enabled. The SD host will wait to receive a response R2 from SD card and store the response data into DMC's Flash buffer (exclude CRC7). **Note:** When operation is finished, this bit will be cleared automatically, so do not write 0 to this bit (the controller will be abnormal).
[3]	**DOEN**	**Data Output Enable Bit** 0 = No effect. (Please use DMARST (SDH_DMACTL[1]) to clear this bit.) 1 = Enabled. The SD host will transfer block data and the CRC16 value to SD card. **Note:** When operation is finished, this bit will be cleared automatically, so do not write 0 to this bit (the controller will be abnormal).
[2]	**DIEN**	**Data Input Enable Bit** 0 = No effect. (Please use DMARST (SDH_DMACTL[1]) to clear this bit.) 1 = Enabled. The SD host will wait to receive block data and the CRC16 value from SD card. **Note:** When operation is finished, this bit will be cleared automatically, so do not write 0 to this bit (the controller will be abnormal).
[1]	**RIEN**	**Response Input Enable Bit** 0 = No effect. (Please use DMARST (SDH_DMACTL[1]) to clear this bit.) 1 = Enabled. The SD host will wait to receive a response from SD card. **Note:** When operation is finished, this bit will be cleared automatically, so do not write 0 to this bit (the controller will be abnormal).
[0]	**COEN**	**Command Output Enable Bit** 0 = No effect. (Please use DMARST (SDH_DMACTL[1]) to clear this bit.) 1 = Enabled. The SD host will output a command to SD card. **Note:** When operation is finished, this bit will be cleared automatically, so do not write 0 to this bit (the controller will be abnormal).

SD Command Argument Register (SDH_CMDARG)

Register	Offset	R/W	Description	Reset Value
SDH_CMDARG	SDH0_BA+0x824	R/W	SD Command Argument Register	0x0000_0000

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

Bits	Description	
[31:0]	ARGUMENT	**SD Command Argument** This register contains a 32-bit value specifies the argument of SD command from host controller to SD card. Before trigger COEN (SDH_CTL [0]), software should fill argument in this field.

SD Interrupt Control Register (SDH_INTEN)

Register	Offset	R/W	Description	Reset Value
SDH_INTEN	SDH0_BA+0x828	R/W	SD Interrupt Control Register	0x0000_0A00

31	30	29	28	27	26	25	24
Reserved	CDSRC	Reserved					
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved	WKIEN	DITOIEN	RTOIEN	Reserved			CDIEN
7	6	5	4	3	2	1	0
Reserved						CRCIEN	BLKDIEN

Bits	Description	
[31]	**Reserved**	Reserved.
[30]	**CDSRC**	**SD Card Detect Source Selection** 0 = From SD card's DAT3 pin. Host need clock to get data on pin DAT3. Please make sure CLKKEEP (SDH_CTL[7]) is 1 in order to generate free running clock for DAT3 pin. 1 = From GPIO pin.
[29:15]	**Reserved**	Reserved.
[14]	**WKIEN**	**Wake-up Signal Generating Enable Bit** Enable/Disable wake-up signal generating of SD controller when card is inserted or removed. 0 = SD Card interrupt to wake-up chip Disabled. 1 = SD Card interrupt to wake-up chip Enabled.
[13]	**DITOIEN**	**Data Input Time-out Interrupt Enable Bit** Enable/Disable interrupts generation of SD controller when data input time-out. Time-out value is specified at TOUT register. 0 = DITOIF (SDH_INTSTS [13]) trigger interrupt Disabled. 1 = DITOIF (SDH_INTSTS [13]) trigger interrupt Enabled.
[12]	**RTOIEN**	**Response Time-out Interrupt Enable Bit** Enable/Disable interrupts generation of SD controller when receiving response or R2 time-out. Time-out value is specified at TOUT register. 0 = RTOIF (SDH_INTSTS [12]) trigger interrupt Disabled. 1 = RTOIF (SDH_INTSTS [12]) trigger interrupt Enabled.
[11:9]	**Reserved**	Reserved.

[8]	**CDIEN**	**SD Card Detection Interrupt Enable Bit** Enable/Disable interrupts generation of SD controller when card is inserted or removed. 0 = CDIF (SDH_INTSTS [8]) trigger interrupt Disabled. 1 = CDIF (SDH_INTSTS [8]) trigger interrupt Enabled.
[7:2]	**Reserved**	Reserved.
[1]	**CRCIEN**	**CRC7, CRC16 and CRC Status Error Interrupt Enable Bit** 0 = CRCIF (SDH_INTSTS [1]) trigger interrupt Disabled. 1 = CRCIF (SDH_INTSTS [1]) trigger interrupt Enabled.
[0]	**BLKDIEN**	**Block Transfer Done Interrupt Enable Bit** 0 = BLKDIF (SDH_INTSTS [0]) trigger interrupt Disabled. 1 = BLKDIF (SDH_INTSTS [0]) trigger interrupt Enabled.

SD Interrupt Status Register (SDH_INTSTS)

Register	Offset	R/W	Description	Reset Value
SDH_INTSTS	SDH0_BA+0x82C	R/W	SD Interrupt Status Register	0x000X_008C

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					DAT1STS	Reserved	CDSTS
15	14	13	12	11	10	9	8
Reserved		DITOIF	RTOIF	Reserved		Reserved	CDIF
7	6	5	4	3	2	1	0
DAT0STS	CRCSTS			CRC16	CRC7	CRCIF	BLKDIF

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18]	**DAT1STS**	**DAT1 Pin Status of SD Card (Read Only)** This bit indicates the DAT1 pin status of SD card.
[17]	**Reserved**	Reserved.
[16]	**CDSTS**	**Card Detect Status of SD (Read Only)** This bit indicates the card detect pin status of SD, and is used for card detection. When there is a card inserted in or removed from SD, software should check this bit to confirm if there is really a card insertion or removal. If CDSRC (SDH_INTEN[30]) = 0, to select DAT3 for card detection:. 0 = Card removed. 1 = Card inserted. If CDSRC (SDH_INTEN[30]) = 1, to select GPIO for card detection:. 0 = Card inserted. 1 = Card removed.
[15:14]	**Reserved**	Reserved.
[13]	**DITOIF**	**Data Input Time-out Interrupt Flag (Read Only)** This bit indicates that SD host counts to time-out value when receiving data (waiting start bit). 0 = Not time-out. 1 = Data input time-out. **Note:** This bit is read only, but can be cleared by writing '1' to it.
[12]	**RTOIF**	**Response Time-out Interrupt Flag (Read Only)** This bit indicates that SD host counts to time-out value when receiving response or R2 (waiting start bit). 0 = Not time-out. 1 = Response time-out. **Note:** This bit is read only, but can be cleared by writing '1' to it.

[11:9]	**Reserved**	Reserved.
[8]	**CDIF**	**SD Card Detection Interrupt Flag (Read Only)** This bit indicates that SD card is inserted or removed. Only when CDIEN (SDH_INTEN[8]) is set to 1, this bit is active. 0 = No card is inserted or removed. 1 = There is a card inserted in or removed from SD. **Note:** This bit is read only, but can be cleared by writing '1' to it.
[7]	**DAT0STS**	**DAT0 Pin Status of Current Selected SD Port (Read Only)** This bit is the DAT0 pin status of current selected SD port.
[6:4]	**CRCSTS**	**CRC Status Value of Data-out Transfer (Read Only)** SD host will record CRC status of data-out transfer. Software could use this value to identify what type of error is during data-out transfer. 010 = Positive CRC status. 101 = Negative CRC status. 111 = SD card programming error occurs.
[3]	**CRC16**	**CRC16 Check Status of Data-in Transfer (Read Only)** SD host will check CRC16 correctness after data-in transfer. 0 = Fault. 1 = OK.
[2]	**CRC7**	**CRC7 Check Status (Read Only)** SD host will check CRC7 correctness during each response in. If that response does not contain CRC7 information (ex. R3), then software should turn off CRCIEN (SDH_INTEN[1]) and ignore this bit. 0 = Fault. 1 = OK.
[1]	**CRCIF**	**CRC7, CRC16 and CRC Status Error Interrupt Flag (Read Only)** This bit indicates that SD host has occurred CRC error during response in, data-in or data-out (CRC status error) transfer. When CRC error is occurred, software should reset SD engine. Some response (ex. R3) doesn't have CRC7 information with it; SD host will still calculate CRC7, get CRC error and set this flag. In this condition, software should ignore CRC error and clears this bit manually. 0 = No CRC error is occurred. 1 = CRC error is occurred. **Note:** This bit is read only, but can be cleared by writing '1' to it.
[0]	**BLKDIF**	**Block Transfer Done Interrupt Flag (Read Only)** This bit indicates that SD host has finished all data-in or data-out block transfer. If there is a CRC16 error or incorrect CRC status during multiple block data transfer, the transfer will be broken and this bit will also be set. 0 = Not finished yet. 1 = Done. **Note:** This bit is read only, but can be cleared by writing '1' to it.

SD Receiving Response Token Register 0 (SDH_RESP0)

Register	Offset	R/W	Description	Reset Value
SDH_RESP0	SDH0_BA+0x830	R	SD Receiving Response Token Register 0	0x0000_0000

31	30	29	28	27	26	25	24
RESPTK0							
23	22	21	20	19	18	17	16
RESPTK0							
15	14	13	12	11	10	9	8
RESPTK0							
7	6	5	4	3	2	1	0
RESPTK0							

Bits	Description	
[31:0]	RESPTK0	**SD Receiving Response Token 0 (Read Only)** SD host controller will receive a response token for getting a reply from SD card when RIEN (SDH_CTL[1]) is set. This field contains response bit 47-16 of the response token.

SD Receiving Response Token Register 1 (SDH_RESP1)

Register	Offset	R/W	Description	Reset Value
SDH_RESP1	SDH0_BA+0x834	R	SD Receiving Response Token Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
RESPTK1							

Bits	Description	
[7:0]	RESPTK1	**SD Receiving Response Token 1 (Read Only)** The SD host controller will receive a response token for getting a reply from SD card when RIEN (SDH_CTL[1]) is set. This register contains the bit 15-8 of the response token.

SD Block Length Register (SDH_BLEN)

Register	Offset	R/W	Description	Reset Value
SDH_BLEN	SDH0_BA+0x838	R/W	SD Block Length Register	0x0000_01FF

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

Bits	Description	
[10:0]	BLKLEN	**SD BLOCK LENGTH in Byte Unit** A 11-bit value specifies the SD transfer byte count of a block. The actual byte count is equal to BLKLEN+1. **Note:** The default SD block length is 512 bytes.

SD Response/Data-in Time-out Register (SDH_TOUT)

Register	Offset	R/W	Description	Reset Value
SDH_TOUT	SDH0_BA+0x83C	R/W	SD Response/Data-in Time-out Register	0x0000_0000

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

Bits	Description	
[23:0]	TOUT	**SD Response/Data-in Time-out Value** A 24-bit value specifies the time-out counts of response and data input. SD host controller will wait start bit of response or data-in until this value reached. The time period depends on SD engine clock frequency. Do not write a small number into this field, or you may never get response or data due to time-out. **Note:** Filling 0x0 into this field will disable hardware time-out function.

6.31 External Bus Interface (EBI)

6.31.1 Overview

This chip is equipped with an external bus interface (EBI) for external device use. To save the connections between an external device and a chip, EBI is operating at address bus and data bus multiplex mode. The EBI supports three chip selects that can connect three external devices with different timing setting requirements.

6.31.2 Features

- Supports up to three memory banks
- Supports dedicated external chip select pin with polarity control for each bank
- Supports accessible space up to 1 Mbytes for each bank, actually external addressable space is dependent on package pin out
- Supports 8-/16-bit data width
- Supports byte write in 16-bit data width mode
- Supports address bus and data bus multiplexe mode
- Supports address bus and data bus separate mode
- Supports Timing parameters individual adjustment for each memory block
- Supports LCD interface i80 mode
- Supports PDMA mode
- Supports variable external bus base clock (MCLK) which based on HCLK
- Supports configurable idle cycle for different access condition: Idle of Write command finish (W2X) and Idle of Read-to-Read (R2R)

6.31.3 Block Diagram

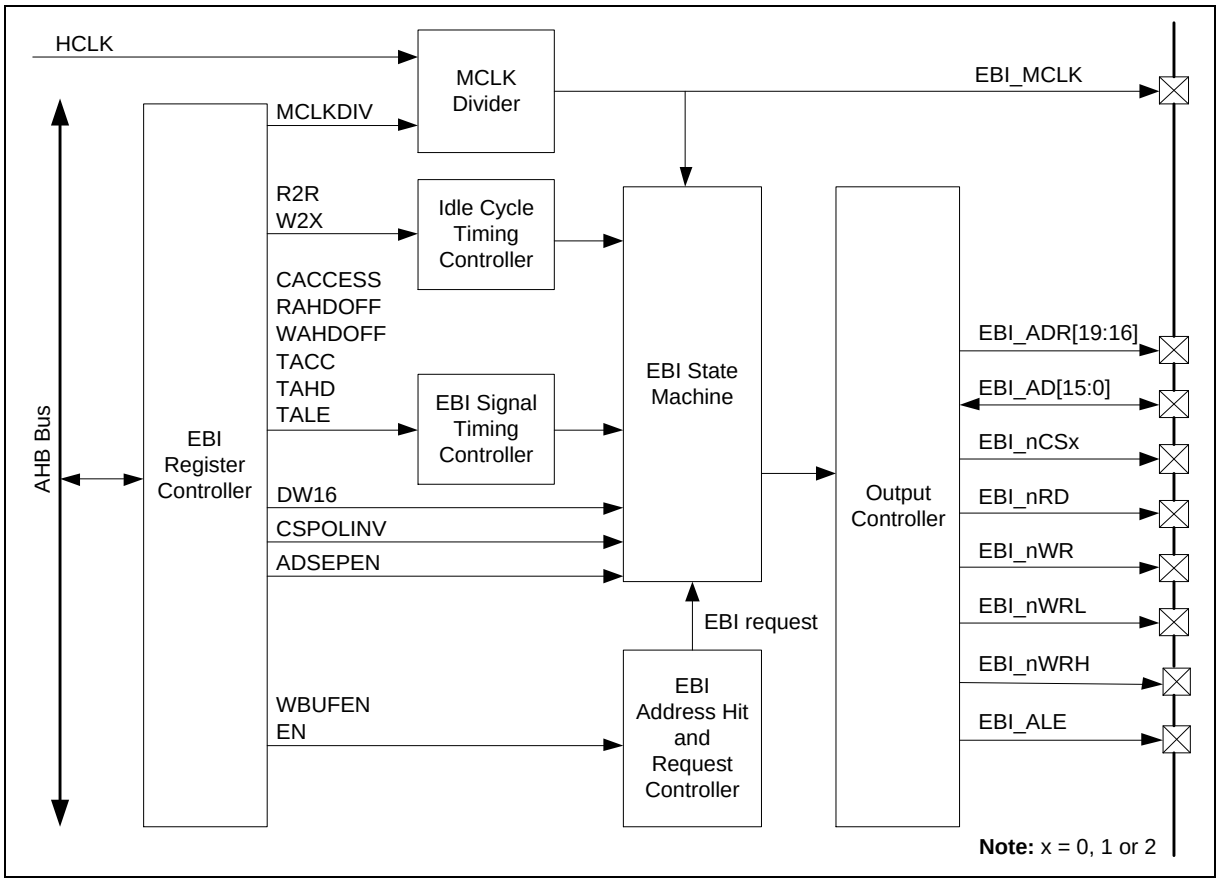


Figure 6.31-1 EBI Block Diagram

6.31.4 Basic Configuration

- Clock Source Configuration
  - Enable EBI controller clock in EBICKEN (CLK_AHBCLK[3]).
- Reset Configuration
  - Reset EBI controller in EBIRST (SYS_IPRST0[3]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
EBI	EBI_AD0	PC.0, PG.9	MFP2
	EBI_AD1	PC.1, PG.10	MFP2
	EBI_AD2	PC.2, PG.11	MFP2
	EBI_AD3	PC.3, PG.12	MFP2
	EBI_AD4	PC.4, PG.13	MFP2
	EBI_AD5	PC.5, PG.14	MFP2
	EBI_AD6	PA.6, PD.8	MFP2

	EBI_AD7	PA.7, PD.9	MFP2
	EBI_AD8	PC.6, PE.14	MFP2
	EBI_AD9	PC.7, PE.15	MFP2
	EBI_AD10	PD.3, PE.1	MFP2
	EBI_AD11	PD.2, PE.0	MFP2
	EBI_AD12	PB.15, PD.1, PH.8	MFP2
	EBI_AD13	PB.14, PD.0, PH.9	MFP2
	EBI_AD14	PB.13, PH.10	MFP2
	EBI_AD15	PB.12, PH.11	MFP2
	EBI_ADR0	PB.5, PH.7	MFP2
	EBI_ADR1	PB.4, PH.6	MFP2
	EBI_ADR2	PB.3, PH.5	MFP2
	EBI_ADR3	PB.2, PH.4	MFP2
	EBI_ADR4	PC.12	MFP2
	EBI_ADR5	PC.11	MFP2
	EBI_ADR6	PC.10	MFP2
	EBI_ADR7	PC.9	MFP2
	EBI_ADR8	PB.1	MFP2
	EBI_ADR9	PB.0	MFP2
	EBI_ADR10	PC.13, PE.8	MFP2
	EBI_ADR11	PE.9, PG.2	MFP2
	EBI_ADR12	PE.10, PG.3	MFP2
	EBI_ADR13	PE.11, PG.4	MFP2
	EBI_ADR14	PE.12, PF.11	MFP2
	EBI_ADR15	PE.13, PF.10	MFP2
	EBI_ADR16	PB.11, PC.8, PF.9	MFP2
	EBI_ADR17	PB.10, PF.8	MFP2
	EBI_ADR18	PB.9, PF.7	MFP2
	EBI_ADR19	PB.8, PF.6	MFP2
	EBI_ALE	PA.8, PE.2	MFP2
	EBI_MCLK	PA.9, PE.3	MFP2
	EBI_nCS0	PD.12, PD.14, PF.3	MFP2
		PF.6	MFP7
		PB.7	MFP8

	EBI_nCS1	PD.11, PF.2	MFP2
		PB.6	MFP8
	EBI_nCS2	PD.10	MFP2
	EBI_nRD	PA.11, PE.5	MFP2
	EBI_nWR	PA.10, PE.4	MFP2
	EBI_nWRH	PB.6	MFP2
	EBI_nWRL	PB.7	MFP2

6.31.5 Functional Description

6.31.5.1 EBI Area and Address Hit

The EBI mapping address is located at 0x6000_0000 ~ 0x602F_FFFF and the total memory space is 3 Mbytes. When system request address hits EBI's memory space, the corresponding EBI chip select signal is assert and EBI state machine operates.

Chip Select	Address Mapping
EBI_nCS0	0x6000_0000 ~ 0x600F_FFFF
EBI_nCS1	0x6010_0000 ~ 0x601F_FFFF
EBI_nCS2	0x6020_0000 ~ 0x602F_FFFF

Table 6.31-1 EBI Address Mapping

To map the whole EBI memory space, it requires 20-bit address for 8-bit data width device and 19-bit address for 16-bit data width device. For package that output less than 20-bit address, EBI will map device to mirror space. For example, the package with 18-bit EBI address, EBI will mapped external device (for Bank0/EBI_nCS0) to 0x6000_0000 ~ 0x6003_FFFF, 0x6004_0000 ~ 0x6007_FFFF, 0x6008_0000 ~ 0x600B_FFFF and 0x600C_0000 ~ 0x600F_FFFF simultaneously.

6.31.5.2 EBI Data Width Connection - Address Bus and Data Bus Multiplex Mode

The EBI supports the device whose address bus and data bus are multiplexed. For the external device with separated address and data bus, the connection to device needs additional latch device to latch the address. In this case, the pin EBI_ALE is connected to the latch device to latch the address value. Pin EBI_AD is the input of the latch device, and the output of the latch device is connected to the address of external device.

For 16-bit device, the EBI_AD[15:0] is shared by address and 16-bit data, and EBI_ADR[18:16] is dedicated for address and could be connected to 16-bit device directly. The EBI_ADR[19] will be ignored when EBI data width is set as 16-bit width. For 8-bit device, only EBI_AD[7:0] is shared by address and 8-bit data, EBI_AD[15:8] and EBI_ADR[19:16] are dedicated for address and could be connected to 8-bit device directly. Figure 6.31-2 shows the connection of 16-bit data width device and Figure 6.31-3 shows the connection of 8-bit data width device.

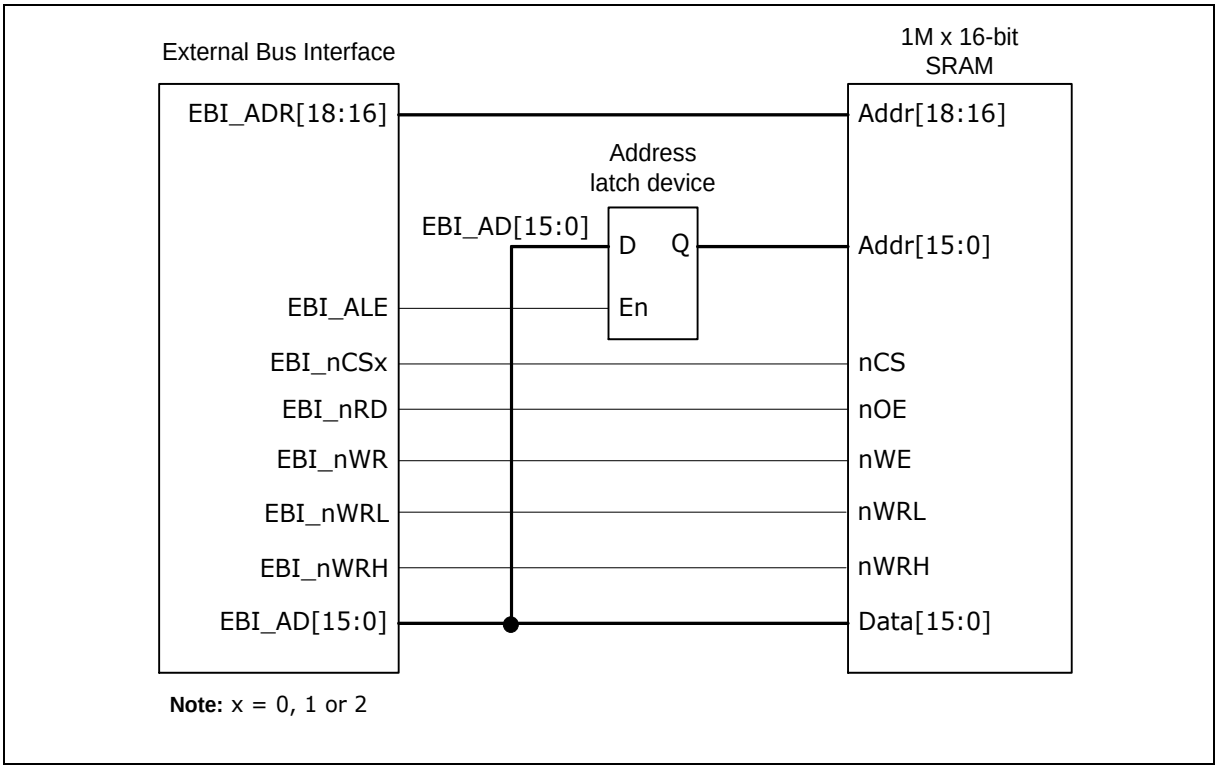


Figure 6.31-2 Connection of 16-bit EBI Data Width with 16-bit Device

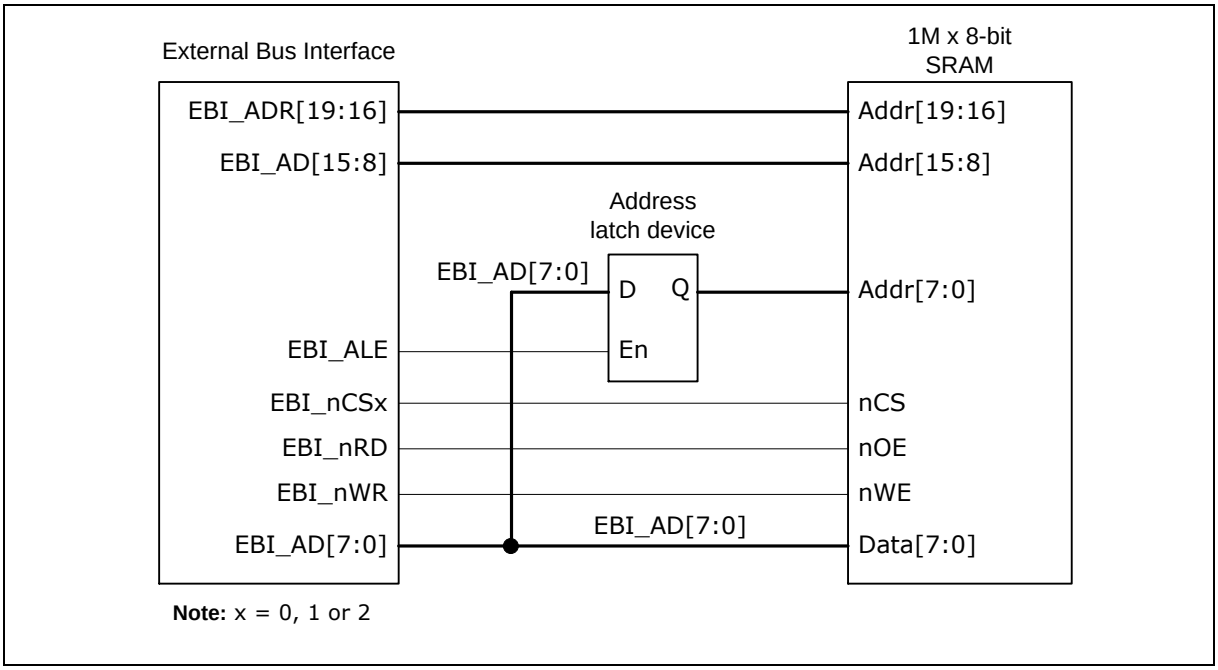


Figure 6.31-3 Connection of 8-bit EBI Data Width with 8-bit Device

When system access data width is larger than EBI data width, the EBI controller will finish a system access command by operating EBI access more than once. For example, if system requests a 32-bit data through EBI device, the EBI controller will operate accessing four times when setting EBI data width with 8-bit.

6.31.5.3 EBI Data Width Connection - Address Bus and Data Bus Separate Mode

The EBI supports address and data bus separate mode. User can enable this mode by setting ADSEPEN (EBI_CTLx[3]). When separate mode is enabled, EBI_AD is dedicated for data bus and connected directly to device data bus, EBI_ADR is dedicated for address bus.

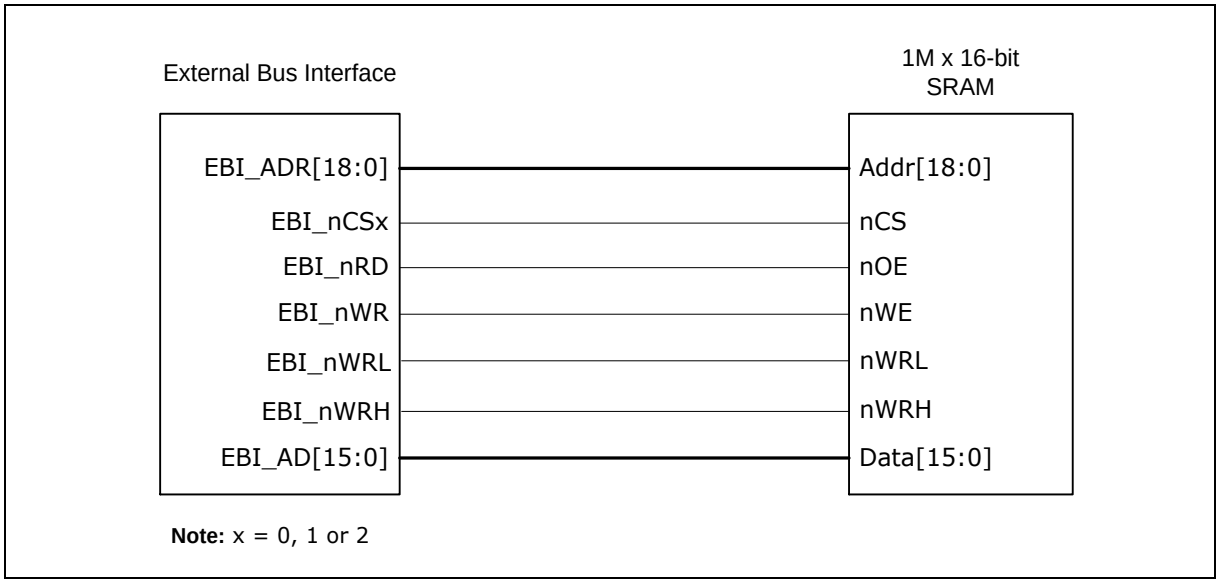


Figure 6.31-4 Connection of 16-bit EBI Data Width with 16-bit Device in Separate mode

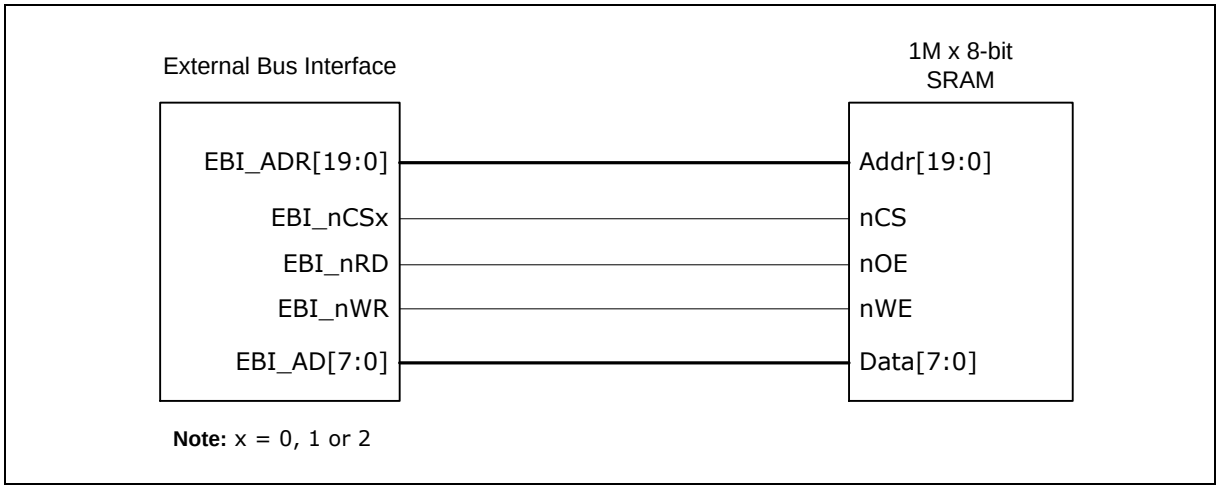


Figure 6.31-5 Connection of 8-bit EBI Data Width with 8-bit Device in Separate mode

6.31.5.4 EBI Operating Control

MCLK Control

In the chip, all EBI signals will be synchronized by EBI_MCLK when EBI is operating. When chip connects to the external device with slower operating frequency, the EBI_MCLK can divide most to HCLK/128 by setting MCLKDIV (EBI_CTLx[10:8]). Therefore, chip can be suitable for a wide frequency range of EBI device. If EBI_MCLK is set to HCLK/1, EBI signals are synchronized by positive edge of EBI_MCLK, else by negative edge of EBI_MCLK.

Operation and Access Timing Control

At the start of EBI access, chip select (EBI_nCS0, EBI_nCS1 and EBI_nCS2) asserts to low and wait one EBI_MCLK for address setup time (tASU) for address stable. Then EBI_ALE asserts to high after address is stable and keeps for a period of time (tALE) for address latch. After latch address, EBI_ALE asserts to low and wait one EBI_MCLK for latch hold time (tLHD) and another one EBI_MCLK cycle (tA2D) that is inserted behind address hold time to be the bus turn-around time for address change to data. Then EBI_nRD asserts to low when read access or EBI_nWR asserts to low when write access. Then EBI_nRD or EBI_nWR asserts to high after keeps access time (tACC) for reading output stable or writing finish. After that, EBI signals keep for data access hold time (tAHD) and chip select asserts to high, address is released by current access control.

The EBI controller provides a flexible timing control for different external device. In EBI timing control, tASU, tLHD and tA2D are fixed to 1 EBI_MCLK cycle, tAHD can modulate to 1~8 EBI_MCLK cycles by setting TAHD (EBI_TCTLx[10:8]), tACC can modulate to 1~32 EBI_MCLK cycles by setting TACC (EBI_TCTLx[7:3]), and tALE can modulate to 1~8 EBI_MCLK cycles by setting TALE (EBI_CTL0[18:16]). Some external device can support zero data access hold time accessing, the EBI controller can skipped tAHD to increase access speed by setting WAHDOFF (EBI_TCTLx[23]) and RAHDOFF (EBI_TCTLx[22]).

For each chip select, the EBI provides individual register with timing control except that tALE can only be controlled by EBI_CTL0.

Parameter	Value	Unit	Description
tASU	1	MCLK	Address Latch Setup Time.
tALE	1 ~ 8	MCLK	ALE High Period. Controlled by TALE (EBI_CTL0[18:16]).
tLHD	1	MCLK	Address Latch Hold Time.
tA2D	1	MCLK	Address To Data Delay (Bus Turn-Around Time).
tACC	1 ~ 32	MCLK	Data Access Time. Controlled by TACC (EBI_TCTLx[7:3]).
tAHD	1 ~ 8	MCLK	Data Access Hold Time. Controlled by TAHD (EBI_TCTLx[10:8]).
IDLE	0 ~ 15	MCLK	Idle Cycle. Controlled by R2R (EBI_TCTLx[27:24]) and W2X (EBI_TCTLx[15:12]).

Table 6.31-2 Timing Control Parameter

Figure 6.31-6 shows an example of setting 16-bit data width. In this example, EBI_AD bus is used for being address [15:0] and data [15:0]. When EBI_ALE assert to high, EBI_AD is address output. After address is latched, EBI_ALE asserts to low and the EBI_AD bus change to high impedance to wait device output data in read access operation, or it is used for being write data output.

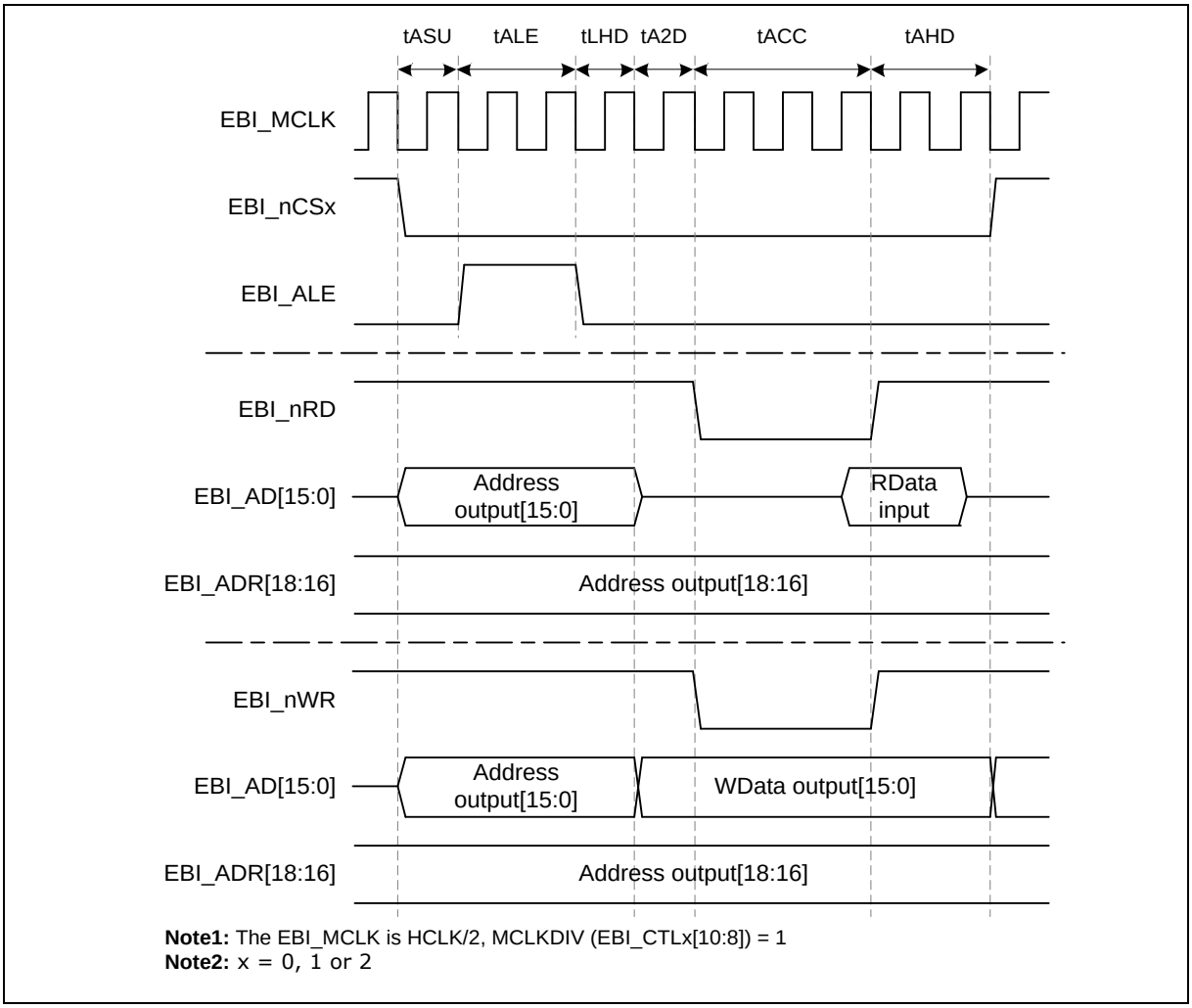


Figure 6.31-6 Timing Control Waveform for 16-bit Data Width

Figure 6.31-7 shows an example of setting 8-bit data width. The difference between 8-bit and 16-bit data width is EBI_AD[15:8]. In 8-bit data width setting, EBI_AD[15:8] is always Address [15:8] output so that external latch needs only 8-bit width.

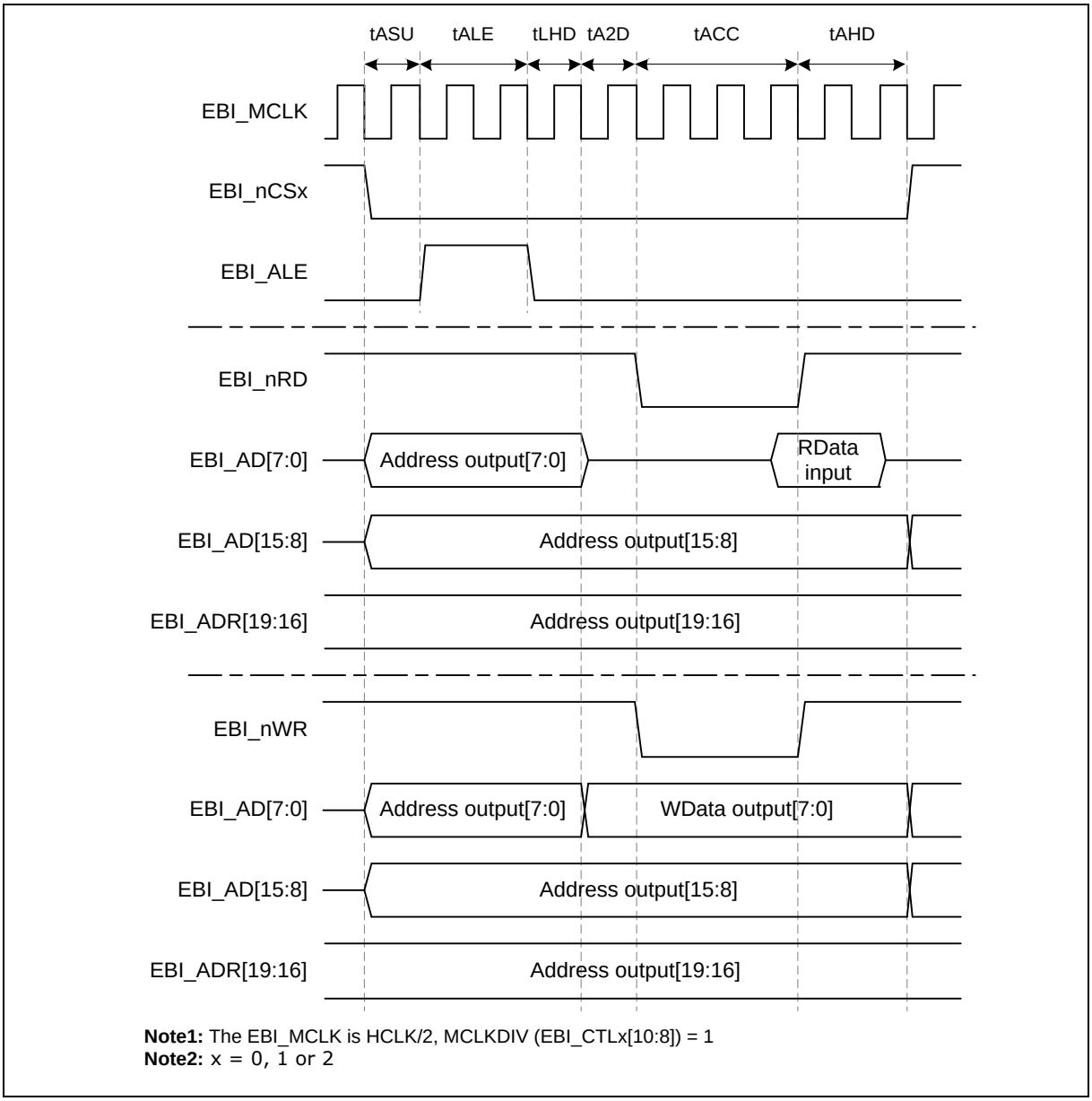
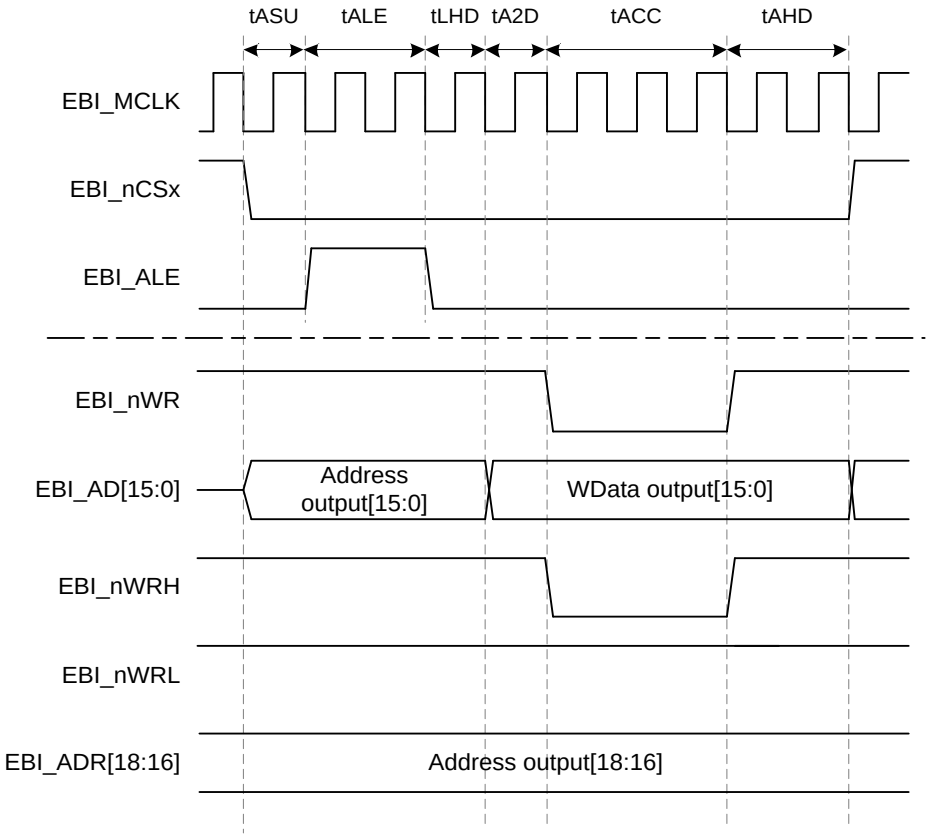


Figure 6.31-7 Timing Control Waveform for 8-bit Data Width

Byte Access

The EBI supports byte access when connected to 16-bit device. The pin EBI_nWRH and EBI_nWRL assertion indicate high byte enable and low byte enable in 16-bit data bus. Figure 6.31-8 shows the write operation of 8-bit width data in EBI_AD[15:8] with EBI_nWRH assertion.



Note1: The EBI_MCLK is HCLK/2, MCLKDIV (EBI_CTLx[10:8]) = 1
Note2: x = 0, 1 or 2
Note3: WData[15:8] is valid with EBI_nWRH assertion

Figure 6.31-8 Timing Control Waveform for Byte Write in 16-bit Data Mode

Insert Idle Cycle

When EBI accesses continuously, there may occur bus conflict if the device access time is much slow with system operating. The EBI controller supplies additional idle cycle to solve this problem. During idle cycle, all control signals of EBI are inactive. Figure 6.31-9 shows idle cycles.

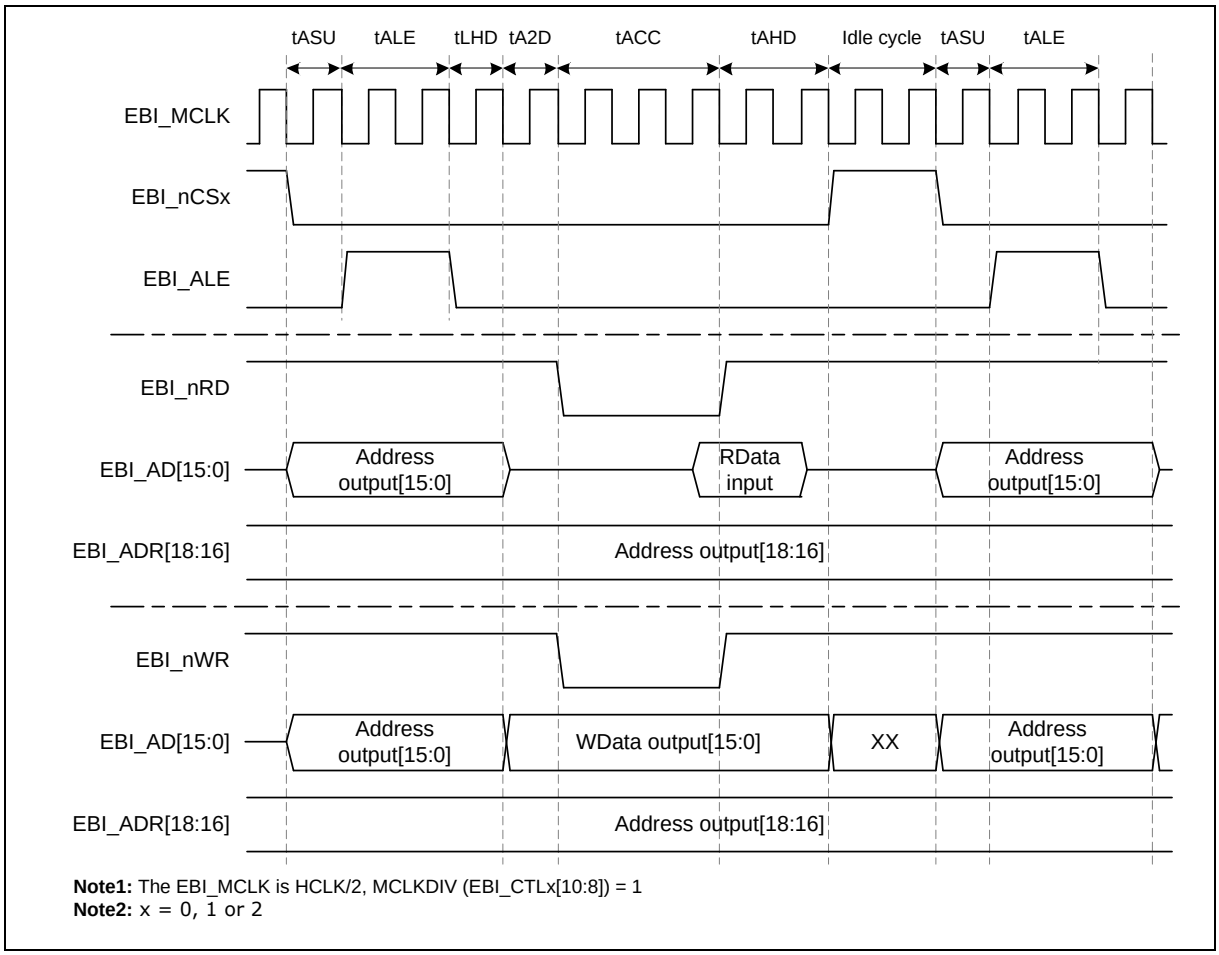


Figure 6.31-9 Timing Control Waveform for Insert Idle Cycle

There are two conditions that EBI can insert idle cycle by timing control:

1. After write access (W2X idle cycle)
2. After read access and before next read access (R2R idle cycle)

By setting W2X (EBI_TCTLx[15:12]), and R2R (EBI_TCTLx[27:24]), the time of idle cycle can be specified from 0~15 EBI_MCLK.

Chip Select Polarity Control

The EBI supports chip select polarity control for connecting to variable external device. When CSPOLINV (EBI_CTLx[2]) is set to 0, the chip select pins (EBI_nCSx) works as low active behavior. It means the external device can be access under EBI_nCSx at low state. When CSPOLINV (EBI_CTLx[2]) is set to 1, the chip select pin (EBI_nCS) works as high active behavior. It means the external device can be access under EBI_nCSx at high state.

Write Buffer

When user writes data to an external device through EBI bus, the EBI controller will start processing the write action immediately and the CPU is held until current EBI write action is finished. User can enable write buffer function to improve CPU and EBI access performance. When EBI write buffer function is enabled, the CPU can continuously execute other instruction during EBI controller process the write action to external device. There is one exception condition for this case. If CPU executes another data access through EBI bus when EBI process write action, the CPU will be held.

User can enable write buffer by setting WBUFEN (EBI_CTL0[24]).

Address Data Separate Mode

When EBI is set as separate mode, the tALE, tLHD, tA2D cycles are ignored. EBI_AD and EBI_ADR are dedicated for data and address bus separately.

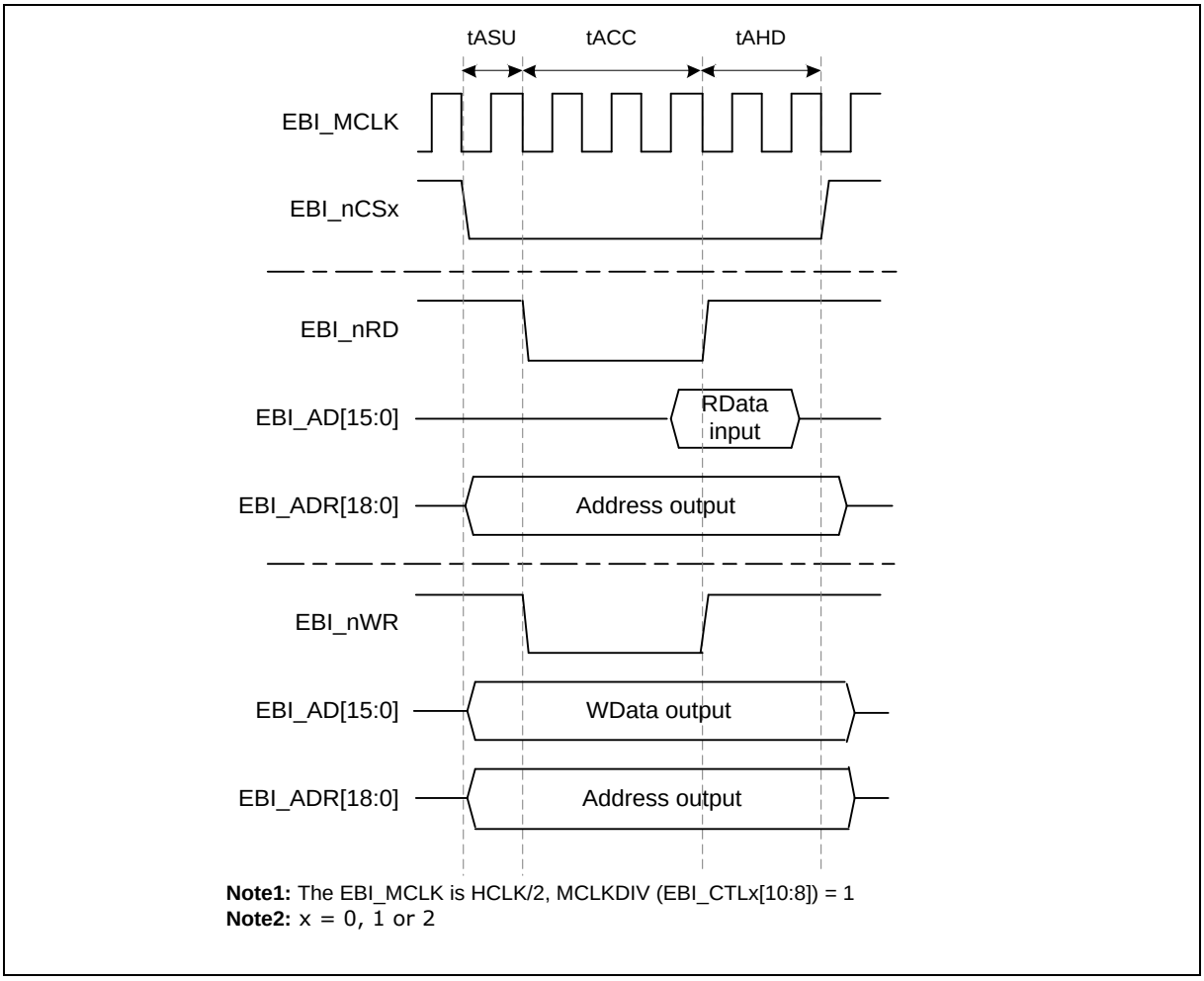


Figure 6.31-10 Timing Control Waveform for 16-bit Data Width for Separate Mode

Continuous Data Access Mode

The EBI supports continuous data access mode for the device which needs faster data access and do not need address control interface. User can enable this mode by setting CACCESS (EBI_CTLx[4]) for each bank. When EBI is set as continuous data access mode, the tASU, tALE, tLHD cycles are

ignored and EBI can access data continuously within one read or write command. There will be dummy cycle between each access command. The timing waveform is shown as Figure 6.31-11.

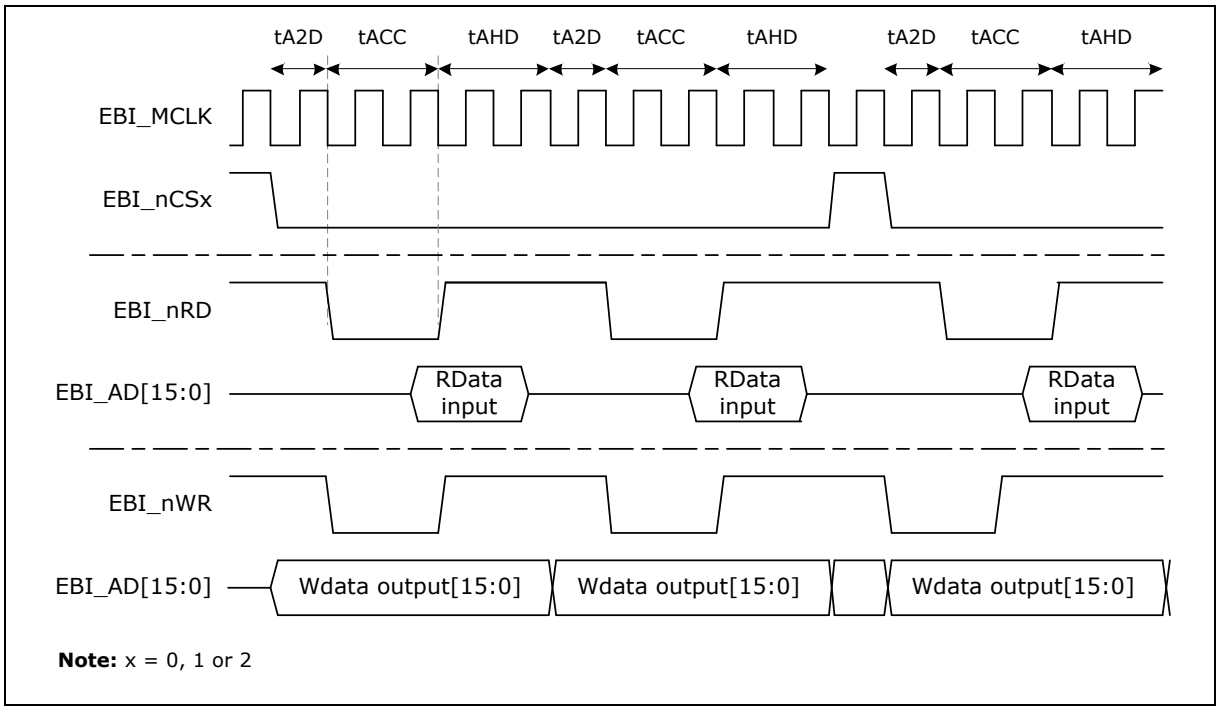


Figure 6.31-11 Timing Control Waveform for Continuous Data Access Mode

6.31.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
EBI Base Address: **EBI_BA = 0x4001_0000** **EBI non-secure base address is EBI_BA + 0x1000_0000.**				
EBI_CTL0	EBI_BA+0x00	R/W	External Bus Interface Bank0 Control Register	0x0000_0000
EBI_TCTL0	EBI_BA+0x04	R/W	External Bus Interface Bank0 Timing Control Register	0x0000_0000
EBI_CTL1	EBI_BA+0x10	R/W	External Bus Interface Bank1 Control Register	0x0000_0000
EBI_TCTL1	EBI_BA+0x14	R/W	External Bus Interface Bank1 Timing Control Register	0x0000_0000
EBI_CTL2	EBI_BA+0x20	R/W	External Bus Interface Bank2 Control Register	0x0000_0000
EBI_TCTL2	EBI_BA+0x24	R/W	External Bus Interface Bank2 Timing Control Register	0x0000_0000

6.31.7 Register Description

External Bus Interface Control Register (EBI_CTLx)

Register	Offset	R/W	Description	Reset Value
EBI_CTL0	EBI_BA+0x00	R/W	External Bus Interface Bank0 Control Register	0x0000_0000
EBI_CTL1	EBI_BA+0x10	R/W	External Bus Interface Bank1 Control Register	0x0000_0000
EBI_CTL2	EBI_BA+0x20	R/W	External Bus Interface Bank2 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							WBUFEN
23	22	21	20	19	18	17	16
Reserved					TALE		
15	14	13	12	11	10	9	8
Reserved					MCLKDIV		
7	6	5	4	3	2	1	0
Reserved			CACCESS	ADSEPEN	CSPOLINV	DW16	EN

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24]	**WBUFEN**	**EBI Write Buffer Enable Bit** 0 = EBI write buffer Disabled. 1 = EBI write buffer Enabled. **Note:** This bit is only available in EBI_CTL0 register.
[23:19]	**Reserved**	Reserved.
[18:16]	**TALE**	**Extend Time of ALE** The EBI_ALE high pulse period (tALE) to latch the address can be controlled by TALE. tALE = (TALE + 1) * EBI_MCLK. **Note:** This field is only available in EBI_CTL0 register.
[15:11]	**Reserved**	Reserved.
[10:8]	**MCLKDIV**	**External Output Clock Divider** The frequency of EBI output clock (MCLK) is controlled by MCLKDIV as follow: 000 = HCLK/1. 001 = HCLK/2. 010 = HCLK/4. 011 = HCLK/8. 100 = HCLK/16. 101 = HCLK/32. 110 = HCLK/64. 111 = HCLK/128.

[7:5]	**Reserved**	Reserved.
[4]	**CACCESS**	**Continuous Data Access Mode** When continuous access mode enabled, the tASU, tALE and tLHD cycles are bypass for continuous data transfer request. 0 = Continuous data access mode Disabled. 1 = Continuous data access mode Enabled.
[3]	**ADSEPEN**	**EBI Address/Data Bus Separate Mode Enable Bit** 0 = Address/Data Bus Separate Mode Disabled. 1 = Address/Data Bus Separate Mode Enabled.
[2]	**CSPOLINV**	**Chip Select Pin Polar Inverse** This bit defines the active level of EBI chip select pin (EBI_nCS). 0 = Chip select pin (EBI_nCS) is active low. 1 = Chip select pin (EBI_nCS) is active high.
[1]	**DW16**	**EBI Data Width 16-bit Select** This bit defines if the EBI data width is 8-bit or 16-bit. 0 = EBI data width is 8-bit. 1 = EBI data width is 16-bit.
[0]	**EN**	**EBI Enable Bit** This bit is the functional enable bit for EBI. 0 = EBI function Disabled. 1 = EBI function Enabled.

External Bus Interface Timing Control Register (EBI_TCTLx)

Register	Offset	R/W	Description	Reset Value
EBI_TCTL0	EBI_BA+0x04	R/W	External Bus Interface Bank0 Timing Control Register	0x0000_0000
EBI_TCTL1	EBI_BA+0x14	R/W	External Bus Interface Bank1 Timing Control Register	0x0000_0000
EBI_TCTL2	EBI_BA+0x24	R/W	External Bus Interface Bank2 Timing Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				R2R			
23	22	21	20	19	18	17	16
WAHDOFF	RAHDOFF	Reserved					
15	14	13	12	11	10	9	8
W2X				Reversed	TAHD		
7	6	5	4	3	2	1	0
TACC					Reserved		

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:24]	**R2R**	**Idle Cycle Between Read-to-read** This field defines the number of R2R idle cycle. R2R idle cycle = (R2R * EBI_MCLK). When read action is finished and the next action is going to read, R2R idle cycle is inserted and EBI_nCS return to idle state.
[23]	**WAHDOFF**	**Access Hold Time Disable Control When Write** 0 = Data Access Hold Time (tAHD) during EBI writing Enabled. 1 = Data Access Hold Time (tAHD) during EBI writing Disabled.
[22]	**RAHDOFF**	**Access Hold Time Disable Control When Read** 0 = Data Access Hold Time (tAHD) during EBI reading Enabled. 1 = Data Access Hold Time (tAHD) during EBI reading Disabled.
[21:16]	**Reserved**	Reserved.
[15:12]	**W2X**	**Idle Cycle After Write** This field defines the number of W2X idle cycle. W2X idle cycle = (W2X * EBI_MCLK). When write action is finished, W2X idle cycle is inserted and EBI_nCS return to idle state.
[11]	**Reserved**	Reserved.
[10:8]	**TAHD**	**EBI Data Access Hold Time** TAHD defines data access hold time (tAHD). tAHD = (TAHD + 1) * EBI_MCLK.

[7:3]	**TACC**	**EBI Data Access Time** TACC defines data access time (tACC). tACC = (TACC + 1) * EBI_MCLK.
[2:0]	**Reserved**	Reserved.

6.32 USB 1.1 Device Controller (USBD)

6.32.1 Overview

There is one set of USB 2.0 full-speed device controller and transceiver in this device. It is compliant with USB 2.0 full-speed device specification and supports Control/Bulk/Interrupt/Isochronous transfer types.

In this device controller, there are two main interfaces: the APB bus and USB bus which comes from the USB PHY transceiver. For the APB bus, the CPU can program control registers through it. There are 1 Kbytes internal SRAM as data buffer in this controller. For IN or OUT transfer, it is necessary to write data to SRAM or read data from SRAM through the APB interface or SIE. User needs to set the effective starting address of SRAM for each endpoint buffer through buffer segmentation register (USBD_BUFSEGx).

There are 12 endpoints in this controller. Each of the endpoint can be configured as IN or OUT endpoint. All the operations including Control, Bulk, Interrupt and Isochronous transfer are implemented in this block. The block of “Endpoint Control” is also used to manage the data sequential synchronization, endpoint states, current start address, transaction status, and data buffer status for each endpoint.

There are four different interrupt events in this controller. They are the no-event-wake-up, device plug-in or plug-out event, USB events, like IN ACK and OUT ACK, etc, and BUS events, like suspend and resume, etc. Any event will cause an interrupt, and users just need to check the related event flags in interrupt event status register (USBD_INTSTS) to acknowledge what kind of interrupt occurring, and then check the related USB Endpoint Status Register (USBD_EPSTS0 and USBD_EPSTS1) to acknowledge what kind of event occurring in this endpoint.

A software-disconnect function is also supported for this USB controller. It is used to simulate the disconnection of this device from the host. If user enables SE0 bit (USBD_SE0), the USB controller will force the output of USB_D+ and USB_D- to level low and its function is disabled. After disabling the SE0 bit, host will enumerate the USB device again.

For more information on the Universal Serial Bus, please refer to *Universal Serial Bus Specification Revision 1.1.*

6.32.2 Features

- Compliant with USB 2.0 Full-Speed specification
- Provides 1 interrupt vector with 4 different interrupt events (NEVWK, VBDET, USB and BUS)
- Supports Control/Bulk/Interrupt/Isochronous transfer type
- Supports suspend function when no bus activity existing for 3ms
- Supports 12 endpoints for configurable Control/Bulk/Interrupt/Isochronous transfer types and maximum 1 Kbytes buffer size
- Provides remote wake-up capability

6.32.3 Block Diagram

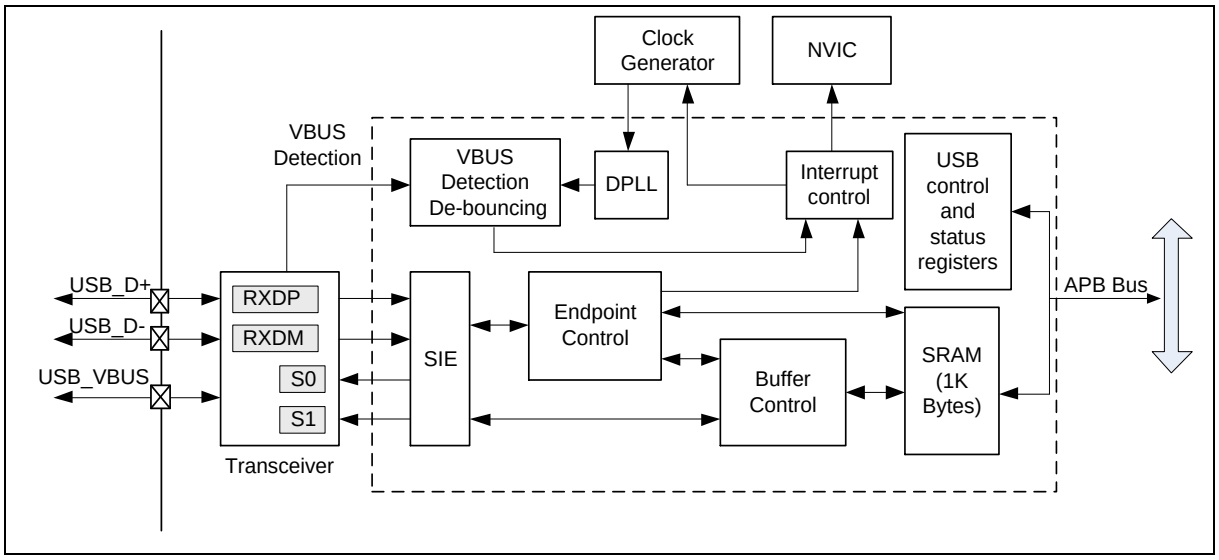


Figure 6.32-1 USB Block Diagram

6.32.4 Basic Configuration

The role of USB frame is determined by USBROLE (SYS_USBPHY[1:0]). The internal USB 3.3V LDO can be enabled by OTGPHYEN (SYS_USBPHY[8]). These two configurations are write-protection bits. Before writing to these bits, user must disable the register protection function. Refer to the description of SYS_REGLCTL register for details. The USBD clock source is derived from PLL or HIRC48 by setting USBSEL (CLK_CLKSEL0[8]). If the clock source is selected from PLL, user has to set the PLL related configurations before USB device controller is enabled. If the clock source is selected from HIRC48, user has to set the HIRC48 related configurations, set the HIRC48EN (CLK_PWRCTL[18]) bit to enable HIRC48 and wait HIRC48 stable by observing HIRC48STB(CLK_STATUS[6]). User has to set the USBDCKEN (CLK_APBCLK0[27]) bit to enable USBD clock and 4-bit pre-scaler USBDIV (CLK_CLKDIV0[7:4]) to generate the proper USBD clock rate.

6.32.5 Functional Description

6.32.5.1 Serial Interface Engine (SIE)

The SIE is the front-end of the device controller and handles most of the USB packet protocol. The SIE typically comprehends signaling up to the transaction level. The functions that it handles could include:

- Packet recognition and transaction sequencing
- SOP, EOP, RESET, RESUME signal detection/generation
- Clock/Data separation
- NRZI Data encoding/decoding and bit-stuffing
- CRC generation and checking (for Token and Data)
- Packet ID (PID) generation and checking/decoding
- Serial-Parallel/Parallel-Serial conversion

6.32.5.2 Endpoint Control

This controller supports 12 endpoints. Each of the endpoint can be configured as Control, Bulk, Interrupt, or Isochronous transfer type. All the operations including Control, Bulk, Interrupt and Isochronous transfer are implemented in this block. It is also used to manage the data sequential

synchronization, endpoint state control, current endpoint start address, current transaction status, and data buffer status in each endpoint.

6.32.5.3 Digital Phase Lock Loop (DPLL)

The bit rate of USB data is 12 MHz. The DPLL uses the 48 MHz which comes from the clock controller to lock the input data RXDP and RXDM. The 12 MHz bit rate clock is also converted from DPLL.

6.32.5.4 VBUS Detection De-bouncing

A USB device may be plugged-in or plugged-out from the USB host. To monitor the state of a USB device when it is detached from the USB host, the device controller provides hardware de-bouncing for USB VBUS detection interrupt to avoid bounce problems on USB plug-in or plug-out. VBUS detection interrupt appears about 10ms later than USB plug-in or plug-out. User can acknowledge USB plug-in/plug-out by reading USBD_VBUSDET register. The VBUSDET flag represents the current state on the bus without de-bouncing. If VBUSDET is 1, it means the USB cable is plugged-in. If user polls the flag to check USB state, software de-bouncing must be added if needed.

6.32.5.5 Interrupt Control

This USB provides 1 interrupt vector with 4 interrupt events (NEVWK, VBDET, USB and BUS). The NEVWK event occurs after waking up the system from Power-down mode (The Power-down mode function is defined in system power-down control register, CLK_PWRCTL). The VBDET event is used for USB plug-in or plug-out. The USB event notifies users of some USB requests, such as IN ACK and OUT ACK. And the BUS event notifies users of some bus events, such as suspend and resume. The related bits must be set in the interrupt enable register (USBD_INTEN) of USB Device Controller to enable USB interrupts.

NEVWK interrupt is only presented when no the other USB interrupt events happened more than 20ms after the chip is waked up from Power-down mode. After the chip enters Power-down mode, any change on USB_VBUS, USB_D+ and USB_D- can wake up this chip if USB wake-up function is enabled. If this change is not intentionally, no interrupt but NEVWK interrupt will occur. After waking up by USB, this interrupt will occur when no the other USB interrupt events are presented for more than 20ms. Figure 6.32-2 shows the control flow of wake-up interrupt.

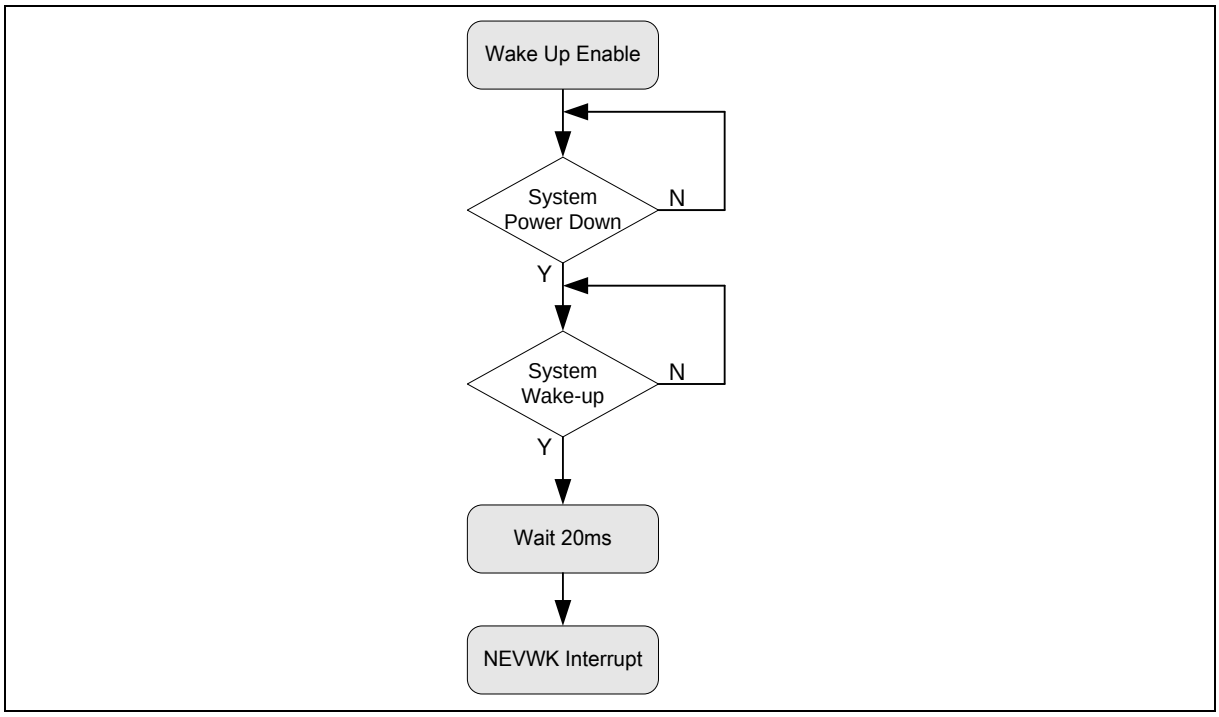


Figure 6.32-2 NEVWK Interrupt Operation Flow

The USB interrupt is used to notify users of any USB event on the bus, and user can read EPSTS (USBD_EPSTS0 and USBD_EPSTS1) and EPEVT11~0 (USBD_INTSTS[27:16]) to take necessary responses.

Same as USB interrupt, BUS interrupt notifies users of some bus events, like USB reset, suspend, time-out, and resume. A user can read USBD_ATTR to acknowledge bus events.

6.32.5.6 Power Saving

User can write 0 to USBD_ATTR[4] to disable PHY under special circumstances, like suspend, to conserve power.

6.32.5.7 Buffer Control

There is 1 Kbytes SRAM in the controller and the 12 endpoints share this buffer. User shall configure each endpoint's effective starting address in the buffer segmentation register before the USB function active. The "Buffer Control" block is used to control each endpoint's effective starting address and its SRAM size is defined in the USBD_MXPLDx register.

Figure 6.32-3 depicts the starting address for each endpoint according the content of USBD_BUFSEGx and USBD_MXPLDx registers. If the USBD_BUFSEG0 is programmed as 0x08h and USBD_MXPLD0 is set as 0x40h, the SRAM size of endpoint 0 starts from USBD_BA+0x108h and ends in USBD_BA+0x148h.

Note: The USBD SRAM base is USBD_BA+0x100h.

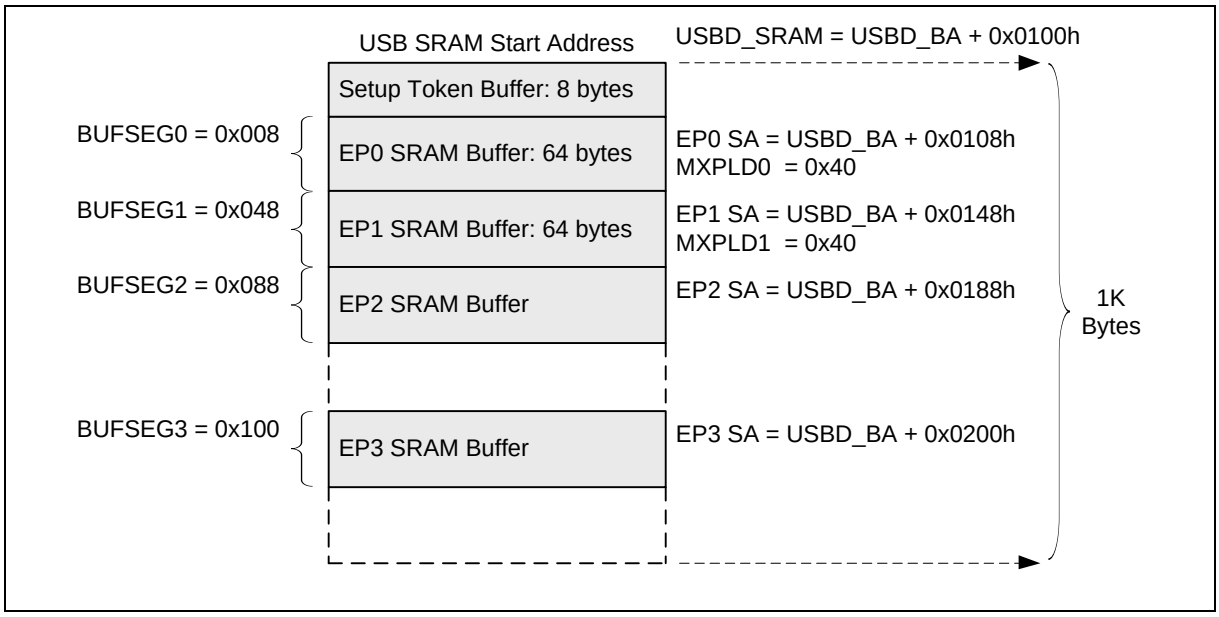


Figure 6.32-3 Endpoint SRAM Structure

6.32.5.8 Handling Transactions with USB Device Peripheral

User can use interrupt or polling USBD_INTSTS to monitor the USB transactions. When transactions occur, USBD_INTSTS will be set by hardware and send an interrupt request to CPU (if related interrupt enabled), or user can polling USBD_INTSTS to get these events without interrupt. The following is the control flow with interrupt enabled.

When USB host has requested data from a device controller, user needs to prepare related data in the specified endpoint buffer in advance. After buffering the required data, user needs to write the actual

data length in the specified USBD_MXPLDx register. Once this register is written, the internal signal “In_Rdy” will be asserted and the buffering data will be transmitted immediately after receiving associated IN token from Host. Note that after transferring the specified data, the signal “In_Rdy” will de-assert automatically by hardware.

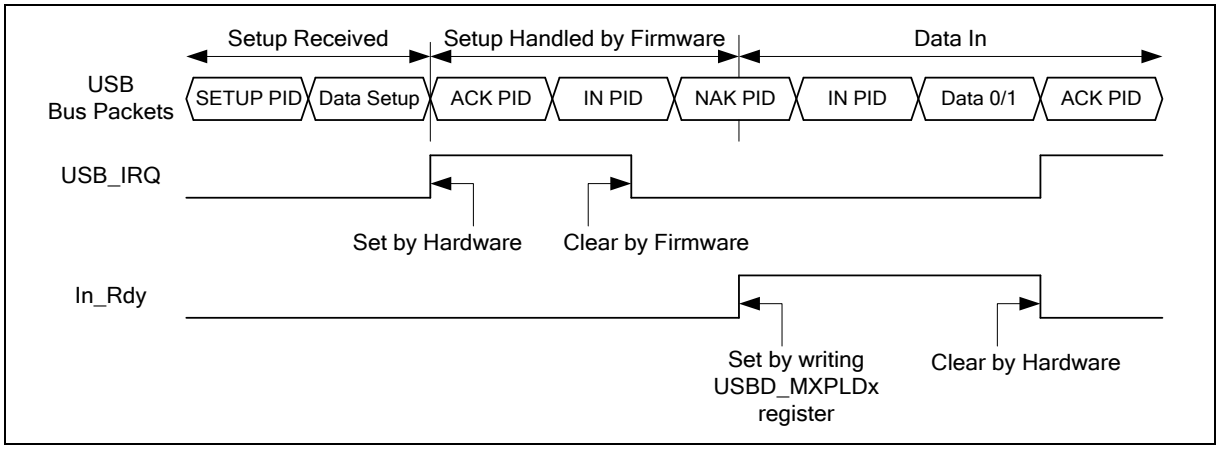


Figure 6.32-4 Setup Transaction Followed by Data IN Transaction

Alternatively, when USB host wants to transmit data to the OUT endpoint in the device controller, hardware will buffer these data to the specified endpoint buffer. After this transaction is completed, hardware will record the data length in specified USBD_MXPLDx register and de-assert the internal signal “Out_Rdy”. This will avoid hardware accepting next transaction until user moves out the current data in the related endpoint buffer. Once users have processed this transaction, the specified USBD_MXPLDx register needs to be written by firmware to assert the signal “Out_Rdy” again to accept the next transaction.

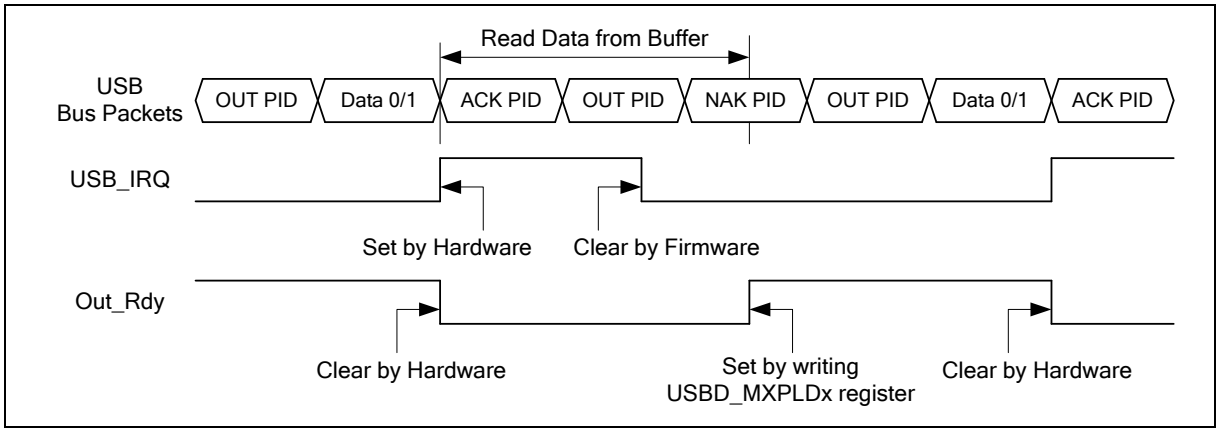


Figure 6.32-5 Data Out Transfer

6.32.5.9 Link Power Management (LPM)

Link Power Management (LPM) which is similar to the suspend/resume function, but has transitional latencies of tens of microseconds between power states (instead of three to greater than 20 milliseconds latencies of the USB2.0 suspend/resume).

Through the LPM mechanism Host lets device state fastly from an enable state (called L0), to a new sleep state (called L1). For detailed definition of L0 and L1 state, see Table 6.32-1. The register USBD_ATTR and USBD_LPMATTR can let user know the current power state for LPM mechanism.

LPM State	Description
L0 (On)	In this state, the port is enabled for propagation of transaction signaling traffic. A port in L0 is either actively transmitting or receiving data (L0-Active) or able to do so but not currently transmitting or receiving information (L0-Idle). While in this state Start-of-Frame (SOF) packets are issued by the host at a rate corresponding to the speed of the client device.
L1 (Sleep)	L1 is similar to L2 (below) but supports finer granularity in use. When in L1, the line state is identical to L2. Entry to L1 is started by a request to a hub or host port to transition to L1. A LPM transaction is sent to the downstream device. The requested transition can only occur if the device response with an ACK handshake. Exit from L1 is via remote wake, resume signaling, reset signaling or disconnect. L1 does not impose any specific power draw requirements (from VBUS) on the attached device as L2 does. Either the host or device can initiate resume signaling when in L1. Although the signaling levels of resume are the same as L2, the duration of the signaling and transitional latencies associated with the L1 to L0 transition are much shorter.
L2 (Suspend)	This is the formalized name for USB 2.0 Suspend, Entry to L2 is nominally triggered by a command to a hub or host port to transition to suspend. The device discovers the suspend condition via observing 3ms of inactivity. The resultant line state is either Low or Full-speed idle. L2 also imposes power draw requirements (from VBUS) on the attached device. Exit from this state is via remote wake, resume signaling, reset signaling or disconnect.
L3 (Off)	In this state, the port is not capable of performing any data signaling. It corresponds to the powered-off, disconnected, and disabled states.

Table 6.32-1 USB Link Power Management (Lx) States

For the state transaction process, please refer to Figure 6.32-6, and for more information on the USB Link Power Management (LPM), please refer to USB2.0 Link Power Management ECN(Engineering Change Notice).

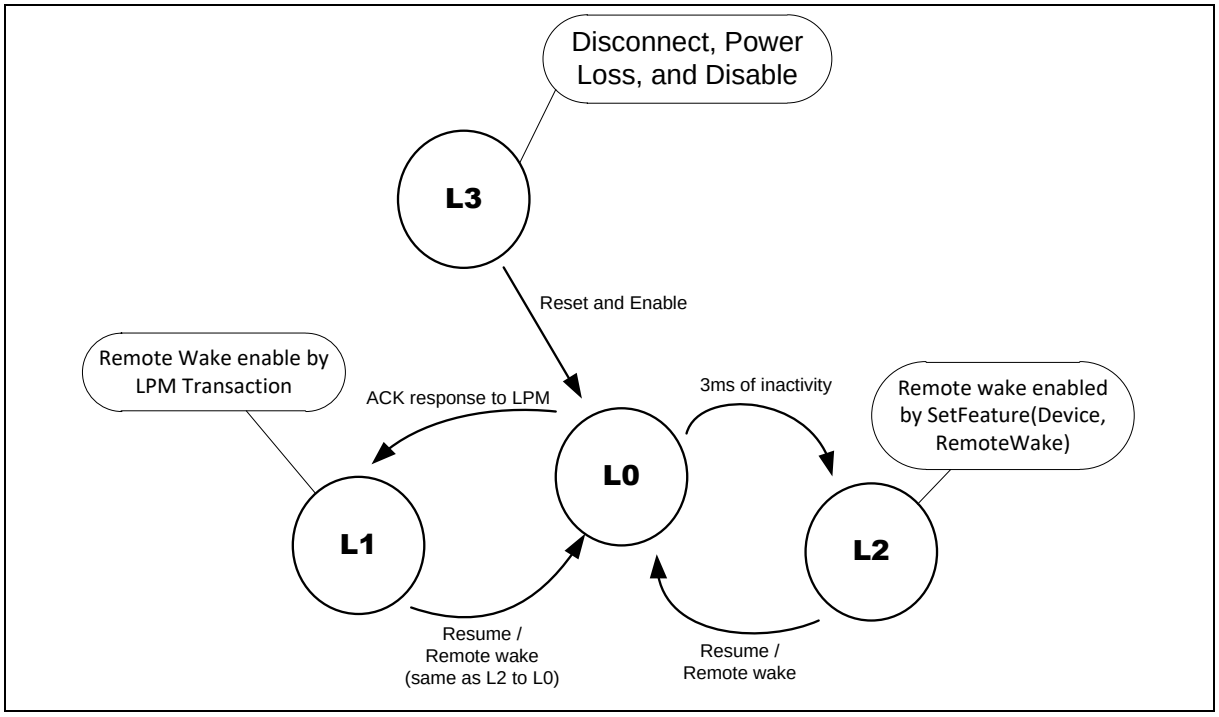


Figure 6.32-6 LPM State Transition Diagram

6.32.5.10 Double Buffer Transfer

When user sets double buffer to continuously transfer or receive data, hardware will automatically toggle endpoint buffer to access data after transaction is finished. User has much time to process the

last transaction data and set the next transaction data to reduce the chances that the device replies NAK to the host.

There are 12 endpoints buffer. User can set single buffer + double buffer * 2 = 12, and set double buffer by “DBEN”, ” DBTGACTIVE” in USB_CFGx. The double buffer function supports IN, OUT and ISO transaction.

The USB device receives endpoint number and token to decide which endpoint buffer will be used. If setting double buffer (DBEN = 1), actived endpoint (DBTGACTIVE =1) will transfer or receive data.

When double buffer is enabled, DBTGACTIVE will be toggled automatically by hardware after successful transfer, and DSQSYNC will remain stable.

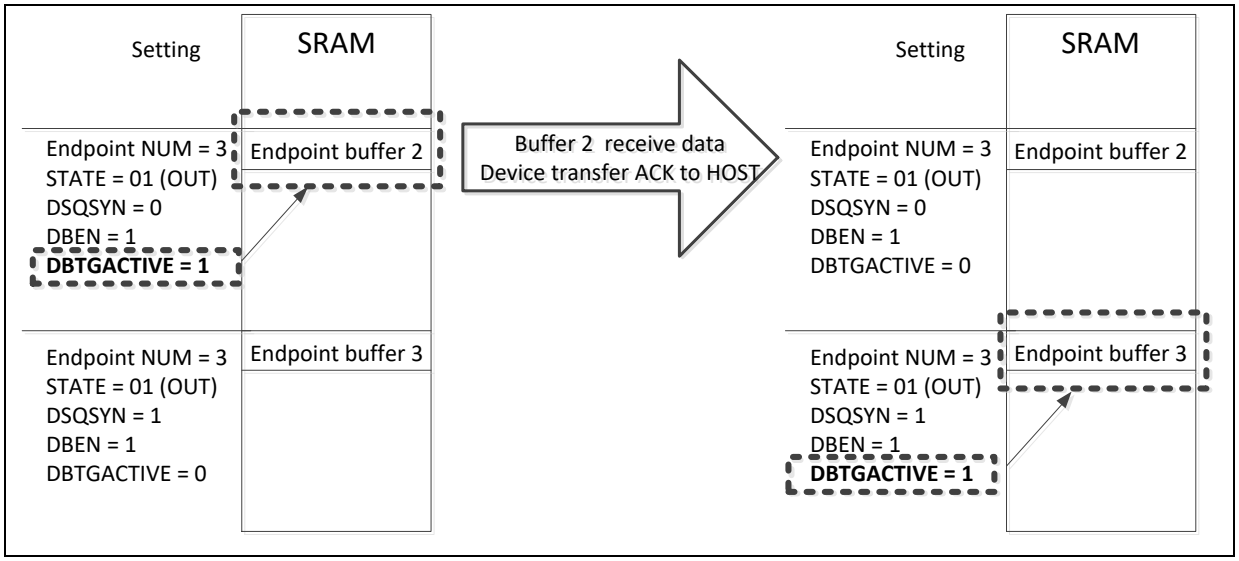


Figure 6.32-7 Double Buffer Change Process

6.32.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
USBD Base Address: **USBD_BA = 0x400C_0000** **USBD non-secure base address is USBD_BA + 0x1000_0000.**				
USBD_INTEN	USBD_BA+0x000	R/W	USB Device Interrupt Enable Register	0x0000_0000
USBD_INTSTS	USBD_BA+0x004	R/W	USB Device Interrupt Event Status Register	0x0000_0000
USBD_FADDR	USBD_BA+0x008	R/W	USB Device Function Address Register	0x0000_0000
USBD_EPSTS	USBD_BA+0x00C	R	USB Device Endpoint Status Register	0x0000_0000
USBD_ATTR	USBD_BA+0x010	R/W	USB Device Bus Status and Attribution Register	0x0000_0040
USBD_VBUSDET	USBD_BA+0x014	R	USB Device VBUS Detection Register	0x0000_0000
USBD_STBUFSEG	USBD_BA+0x018	R/W	SETUP Token Buffer Segmentation Register	0x0000_0000
USBD_EPSTS0	USBD_BA+0x020	R	USB Device Endpoint Status Register 0	0x0000_0000
USBD_EPSTS1	USBD_BA+0x024	R	USB Device Endpoint Status Register 1	0x0000_0000
USBD_LPMATTR	USBD_BA+0x088	R	USB LPM Attribution Register	0x0000_0000
USBD_FN	USBD_BA+0x08C	R	USB Frame Number Register	0x0000_0XXX
USBD_SE0	USBD_BA+0x090	R/W	USB Device Drive SE0 Control Register	0x0000_0001
USBD_BUFSEG0	USBD_BA+0x500	R/W	Endpoint 0 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD0	USBD_BA+0x504	R/W	Endpoint 0 Maximal Payload Register	0x0000_0000
USBD_CFG0	USBD_BA+0x508	R/W	Endpoint 0 Configuration Register	0x0000_0000
USBD_CFGP0	USBD_BA+0x50C	R/W	Endpoint 0 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG1	USBD_BA+0x510	R/W	Endpoint 1 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD1	USBD_BA+0x514	R/W	Endpoint 1 Maximal Payload Register	0x0000_0000
USBD_CFG1	USBD_BA+0x518	R/W	Endpoint 1 Configuration Register	0x0000_0000
USBD_CFGP1	USBD_BA+0x51C	R/W	Endpoint 1 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG2	USBD_BA+0x520	R/W	Endpoint 2 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD2	USBD_BA+0x524	R/W	Endpoint 2 Maximal Payload Register	0x0000_0000
USBD_CFG2	USBD_BA+0x528	R/W	Endpoint 2 Configuration Register	0x0000_0000
USBD_CFGP2	USBD_BA+0x52C	R/W	Endpoint 2 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG3	USBD_BA+0x530	R/W	Endpoint 3 Buffer Segmentation Register	0x0000_0000

USBD_MXPLD3	USBD_BA+0x534	R/W	Endpoint 3 Maximal Payload Register	0x0000_0000
USBD_CFG3	USBD_BA+0x538	R/W	Endpoint 3 Configuration Register	0x0000_0000
USBD_CFGP3	USBD_BA+0x53C	R/W	Endpoint 3 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG4	USBD_BA+0x540	R/W	Endpoint 4 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD4	USBD_BA+0x544	R/W	Endpoint 4 Maximal Payload Register	0x0000_0000
USBD_CFG4	USBD_BA+0x548	R/W	Endpoint 4 Configuration Register	0x0000_0000
USBD_CFGP4	USBD_BA+0x54C	R/W	Endpoint 4 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG5	USBD_BA+0x550	R/W	Endpoint 5 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD5	USBD_BA+0x554	R/W	Endpoint 5 Maximal Payload Register	0x0000_0000
USBD_CFG5	USBD_BA+0x558	R/W	Endpoint 5 Configuration Register	0x0000_0000
USBD_CFGP5	USBD_BA+0x55C	R/W	Endpoint 5 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG6	USBD_BA+0x560	R/W	Endpoint 6 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD6	USBD_BA+0x564	R/W	Endpoint 6 Maximal Payload Register	0x0000_0000
USBD_CFG6	USBD_BA+0x568	R/W	Endpoint 6 Configuration Register	0x0000_0000
USBD_CFGP6	USBD_BA+0x56C	R/W	Endpoint 6 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG7	USBD_BA+0x570	R/W	Endpoint 7 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD7	USBD_BA+0x574	R/W	Endpoint 7 Maximal Payload Register	0x0000_0000
USBD_CFG7	USBD_BA+0x578	R/W	Endpoint 7 Configuration Register	0x0000_0000
USBD_CFGP7	USBD_BA+0x57C	R/W	Endpoint 7 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG8	USBD_BA+0x580	R/W	Endpoint 8 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD8	USBD_BA+0x584	R/W	Endpoint 8 Maximal Payload Register	0x0000_0000
USBD_CFG8	USBD_BA+0x588	R/W	Endpoint 8 Configuration Register	0x0000_0000
USBD_CFGP8	USBD_BA+0x58C	R/W	Endpoint 8 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG9	USBD_BA+0x590	R/W	Endpoint 9 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD9	USBD_BA+0x594	R/W	Endpoint 9 Maximal Payload Register	0x0000_0000
USBD_CFG9	USBD_BA+0x598	R/W	Endpoint 9 Configuration Register	0x0000_0000
USBD_CFGP9	USBD_BA+0x59C	R/W	Endpoint 9 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG10	USBD_BA+0x5A0	R/W	Endpoint 10 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD10	USBD_BA+0x5A4	R/W	Endpoint 10 Maximal Payload Register	0x0000_0000
USBD_CFG10	USBD_BA+0x5A8	R/W	Endpoint 10 Configuration Register	0x0000_0000

USBD_CFGP10	USBD_BA+0x5AC	R/W	Endpoint 10 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG11	USBD_BA+0x5B0	R/W	Endpoint 11 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD11	USBD_BA+0x5B4	R/W	Endpoint 11 Maximal Payload Register	0x0000_0000
USBD_CFG11	USBD_BA+0x5B8	R/W	Endpoint 11 Configuration Register	0x0000_0000
USBD_CFGP11	USBD_BA+0x5BC	R/W	Endpoint 11 Set Stall and Clear In/Out Ready Control Register	0x0000_0000

Memory Type	**Address**	**Size**	**Description**
USBD_BA = 0x400C_0000			
USBD_SRAM	USBD_BA+0x100 ~ USBD_BA+0x4FF	1024 Bytes	The SRAM is used for the entire endpoints buffer. Refer to section 6.32.5.7 for the endpoint SRAM structure and its description.

6.32.7 Register Description

USB Interrupt Enable Register (USBD_INTEN)

Register	Offset	R/W	Description	Reset Value
USBD_INTEN	USBD_BA+0x000	R/W	USB Device Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
INNAKEN	**Reserved**						**WKEN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			**SOFIEN**	**NEVWKIEN**	**VBDETIEN**	**USBIEN**	**BUSIEN**

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**INNAKEN**	**Active NAK Function and Its Status in IN Token** 0 = When device responds NAK after receiving IN token, IN NAK status will not be updated to USBD_EPSTS0 and USBD_EPSTS1 register, so that the USB interrupt event will not be asserted. 1 = IN NAK status will be updated to USBD_EPSTS0 and USBD_EPSTS1 register and the USB interrupt event will be asserted, when the device responds NAK after receiving IN token.
[14:9]	**Reserved**	Reserved.
[8]	**WKEN**	**Wake-up Function Enable Bit** 0 = USB wake-up function Disabled. 1 = USB wake-up function Enabled.
[7:5]	**Reserved**	Reserved.
[4]	**SOFIEN**	**Start of Frame Interrupt Enable Bit** 0 = SOF Interrupt Disabled. 1 = SOF Interrupt Enabled.
[3]	**NEVWKIEN**	**USB No-event-wake-up Interrupt Enable Bit** 0 = No-event-wake-up Interrupt Disabled. 1 = No-event-wake-up Interrupt Enabled.
[2]	**VBDETIEN**	**VBUS Detection Interrupt Enable Bit** 0 = VBUS detection Interrupt Disabled. 1 = VBUS detection Interrupt Enabled.
[1]	**USBIEN**	**USB Event Interrupt Enable Bit** 0 = USB event interrupt Disabled. 1 = USB event interrupt Enabled.

[0]	**BUSIEN**	**Bus Event Interrupt Enable Bit** 0 = BUS event interrupt Disabled. 1 = BUS event interrupt Enabled.

USB Interrupt Event Status Register (USBD_INTSTS)

Register	Offset	R/W	Description	Reset Value
USBD_INTSTS	USBD_BA+0x004	R/W	USB Device Interrupt Event Status Register	0x0000_0000

31	30	29	28	27	26	25	24
SETUP	Reserved			EPEVT11	EPEVT10	EPEVT9	EPEVT8
23	22	21	20	19	18	17	16
EPEVT7	EPEVT6	EPEVT5	EPEVT4	EPEVT3	EPEVT2	EPEVT1	EPEVT0
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			SOFIF	NEVWKIF	VBDETIF	USBIF	BUSIF

Bits	Description	
[31]	SETUP	**Setup Event Status** 0 = No Setup event. 1 = Setup event occurred, cleared by writing 1 to USBD_INTSTS[31].
[30:28]	Reserved	Reserved.
[27]	EPEVT11	**Endpoint 11's USB Event Status** 0 = No event occurred in endpoint 11. 1 = USB event occurred on Endpoint 11. Check USBD_EPSTS1[15:12] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[27] or USBD_INTSTS[1].
[26]	EPEVT10	**Endpoint 10's USB Event Status** 0 = No event occurred in endpoint 10. 1 = USB event occurred on Endpoint 10. Check USBD_EPSTS1[11:8] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[26] or USBD_INTSTS[1].
[25]	EPEVT9	**Endpoint 9's USB Event Status** 0 = No event occurred in endpoint 9. 1 = USB event occurred on Endpoint 9. Check USBD_EPSTS1[7:4] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[25] or USBD_INTSTS[1].
[24]	EPEVT8	**Endpoint 8's USB Event Status** 0 = No event occurred in endpoint 8. 1 = USB event occurred on Endpoint 8. Check USBD_EPSTS1[3:0] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[24] or USBD_INTSTS[1].
[23]	EPEVT7	**Endpoint 7's USB Event Status** 0 = No event occurred in endpoint 7. 1 = USB event occurred on Endpoint 7. Check USBD_EPSTS0[31:28] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[23] or USBD_INTSTS[1].
[22]	EPEVT6	**Endpoint 6's USB Event Status** 0 = No event occurred in endpoint 6.

		1 = USB event occurred on Endpoint 6. Check USBD_EPSTS0[27:24] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[22] or USBD_INTSTS[1].
[21]	**EPEVT5**	**Endpoint 5's USB Event Status** 0 = No event occurred in endpoint 5. 1 = USB event occurred on Endpoint 5. Check USBD_EPSTS0[23:20] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[21] or USBD_INTSTS[1].
[20]	**EPEVT4**	**Endpoint 4's USB Event Status** 0 = No event occurred in endpoint 4. 1 = USB event occurred on Endpoint 4. Check USBD_EPSTS0[19:16] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[20] or USBD_INTSTS[1].
[19]	**EPEVT3**	**Endpoint 3's USB Event Status** 0 = No event occurred in endpoint 3. 1 = USB event occurred on Endpoint 3. Check USBD_EPSTS0[15:12] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[19] or USBD_INTSTS[1].
[18]	**EPEVT2**	**Endpoint 2's USB Event Status** 0 = No event occurred in endpoint 2. 1 = USB event occurred on Endpoint 2. Check USBD_EPSTS0[11:8] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[18] or USBD_INTSTS[1].
[17]	**EPEVT1**	**Endpoint 1's USB Event Status** 0 = No event occurred in endpoint 1. 1 = USB event occurred on Endpoint 1, check USBD_EPSTS0[7:4] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[17] or USBD_INTSTS[1].
[16]	**EPEVT0**	**Endpoint 0's USB Event Status** 0 = No event occurred in endpoint 0. 1 = USB event occurred on Endpoint 0. Check USBD_EPSTS0[3:0] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[16] or USBD_INTSTS[1].
[15:5]	**Reserved**	Reserved.
[4]	**SOFIF**	**Start of Frame Interrupt Status** 0 = SOF event did not occur. 1 = SOF event occurred, cleared by writing 1 to USBD_INTSTS[4].
[3]	**NEVWKIF**	**No-event-wake-up Interrupt Status** 0 = NEVWK event did not occur. 1 = No-event-wake-up event occurred, cleared by writing 1 to USBD_INTSTS[3].
[2]	**VBDETIF**	**VBUS Detection Interrupt Status** 0 = There is no attached/detached event in the USB. 1 = There is attached/detached event in the USB bus and it is cleared by writing 1 to USBD_INTSTS[2].
[1]	**USBIF**	**USB Event Interrupt Status** The USB event includes the SETUP Token, IN Token, OUT ACK, ISO IN, or ISO OUT events in the bus. 0 = No USB event occurred. 1 = USB event occurred. Check EPSTS0~11[3:0] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[1] or SETUP (USBD_INTSTS[31]).
[0]	**BUSIF**	**BUS Interrupt Status** The BUS event means that there is one of the suspense or the resume function in the bus. 0 = No BUS event occurred. 1 = Bus event occurred. Check USBD_ATTR[3:0] to know which kind of bus event was occurred, cleared by writing 1 to USBD_INTSTS[0].

USB Device Function Address Register (USBD_FADDR)

A 7-bit value is used as the address of a device on the USB BUS.

Register	Offset	R/W	Description	Reset Value
USBD_FADDR	USBD_BA+0x008	R/W	USB Device Function Address Register	0x0000_0000

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

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6:0]	**FADDR**	USB device function address

USB Endpoint Status Register (USBD_EPSTS)

Register	Offset	R/W	Description	Reset Value
USBD_EPSTS	USBD_BA+0x00C	R	USB Device Endpoint Status Register	0x0000_0000

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**OV**	**Overrun** It indicates that the received data is over the maximum payload number or not. 0 = No overrun. 1 = Out Data is more than the Max Payload in MXPLD register or the Setup Data is more than 8 bytes.
[6:0]	**Reserved**	Reserved.

USB Bus Status and Attribution Register (USBD_ATTR)

Register	Offset	R/W	Description	Reset Value
USBD_ATTR	USBD_BA+0x010	R/W	USB Device Bus Status and Attribution Register	0x0000_0040

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		L1RESUME	L1SUSPEND	LPMACK	BYTEM	Reserved	DPPUEN
7	6	5	4	3	2	1	0
USBEN	Reserved	RWAKEUP	PHYEN	TOUT	RESUME	SUSPEND	USBRST

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**L1RESUME**	**LPM L1 Resume (Read Only)** 0 = Bus no LPM L1 state resume. 1 = LPM L1 state resume from LPM L1 state suspend.
[12]	**L1SUSPEND**	**LPM L1 Suspend (Read Only)** 0 = Bus no L1 state suspend. 1 = This bit is set by the hardware when LPM command to enter the L1 state is successfully received and acknowledged.
[11]	**LPMACK**	**LPM Token Acknowledge Enable Bit** The NYET/ACK will be returned only on a successful LPM transaction if no errors in both the EXT token and the LPM token and a valid bLinkState = 0001 (L1) is received, else ERROR and STALL will be returned automatically, respectively. 0= The valid LPM Token will be NYET. 1= The valid LPM Token will be ACK.
[10]	**BYTEM**	**CPU Access USB SRAM Size Mode Selection** 0 = Word mode: The size of the transfer from CPU to USB SRAM can be Word only. 1 = Byte mode: The size of the transfer from CPU to USB SRAM can be Byte only.
[9]	**Reserved**	Reserved.
[8]	**DPPUEN**	**Pull-up Resistor on USB_DP Enable Bit** 0 = Pull-up resistor in USB_D+ bus Disabled. 1 = Pull-up resistor in USB_D+ bus Active.
[7]	**USBEN**	**USB Controller Enable Bit** 0 = USB Controller Disabled. 1 = USB Controller Enabled.
[6]	**Reserved**	Reserved.

[5]	**RWAKEUP**	**Remote Wake-up** 0 = Release the USB bus from K state. 1 = Force USB bus to K (USB_D+ low, USB_D- high) state, used for remote wake-up.
[4]	**PHYEN**	**PHY Transceiver Function Enable Bit** 0 = PHY transceiver function Disabled. 1 = PHY transceiver function Enabled.
[3]	**TOUT**	**Time-out Status (Read Only)** 0 = No time-out. 1 = No Bus response more than 18 bits time.
[2]	**RESUME**	**Resume Status (Read Only)** 0 = No bus resume. 1 = Resume from suspend.
[1]	**SUSPEND**	**Suspend Status (Read Only)** 0 = Bus no suspend. 1 = Bus idle more than 3ms, either cable is plugged-out or host is sleeping.
[0]	**USBRST**	**USB Reset Status (Read Only)** 0 = Bus no reset. 1 = Bus reset when SE0 (single-ended 0) more than 2.5us.

USB Device VBUS Detection Register (USBD_VBUSDET)

Register	Offset	R/W	Description	Reset Value
USBD_VBUSDET	USBD_BA+0x014	R	USB Device VBUS Detection Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**VBUSDET**	**Device VBUS Detection** 0 = Controller is not attached to the USB host. 1 = Controller is attached to the USB host.

USB SETUP Token Buffer Segmentation Register (USBD_STBUFSEG)

Register	Offset	R/W	Description	Reset Value
USBD_STBUFSEG	USBD_BA+0x018	R/W	SETUP Token Buffer Segmentation Register	0x0000_0000

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

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8:3]	**STBUFSEG**	**SETUP Token Buffer Segmentation** It is used to indicate the offset address for the SETUP token with the USB Device SRAM starting address. The effective starting address is USBD_SRAM address + {STBUFSEG, 3'b000} Where the USBD_SRAM address = USBD_BA+0x100h. **Note:** It is used for SETUP token only.
[2:0]	**Reserved**	Reserved.

USB Endpoint Status Register 0 (USBD_EPSTS0)

Register	Offset	R/W	Description	Reset Value
USBD_EPSTS0	USBD_BA+0x020	R	USB Device Endpoint Status Register 0	0x0000_0000

31	30	29	28	27	26	25	24
EPSTS7				EPSTS6			
23	22	21	20	19	18	17	16
EPSTS5				EPSTS4			
15	14	13	12	11	10	9	8
EPSTS3				EPSTS2			
7	6	5	4	3	2	1	0
EPSTS1				EPSTS0			

Bits	Description	
[31:28]	**EPSTS7**	**Endpoint 7 Status** These bits are used to indicate the current status of this endpoint 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[27:24]	**EPSTS6**	**Endpoint 6 Status** These bits are used to indicate the current status of this endpoint 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[23:20]	**EPSTS5**	**Endpoint 5 Status** These bits are used to indicate the current status of this endpoint 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.

[19:16]	**EPSTS4**	**Endpoint 4 Status** These Bits Are Used To Indicate The Current Status Of This Endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous Transfer End.
[15:12]	**EPSTS3**	**Endpoint 3 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[11:8]	**EPSTS2**	**Endpoint 2 Status** These bits are used to indicate the current status of this endpoint 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[7:4]	**EPSTS1**	**Endpoint 1 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[3:0]	**EPSTS0**	**Endpoint 0 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.

USB Endpoint Status Register 1 (USBD_EPSTS1)

Register	Offset	R/W	Description	Reset Value
USBD_EPSTS1	USBD_BA+0x024	R	USB Device Endpoint Status Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
EPSTS11				**EPSTS10**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EPSTS9				**EPSTS8**			

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:12]	**EPSTS11**	**Endpoint 11 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[11:8]	**EPSTS10**	**Endpoint 10 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[7:4]	**EPSTS9**	**Endpoint 9 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[3:0]	**EPSTS8**	**Endpoint 8 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK.

		0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.

USB LPM Attribution Register (USBD_LPMATTR)

Register	Offset	R/W	Description	Reset Value
USBD_LPMATTR	USBD_BA+0x088	R	USB LPM Attribution Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							**LPMRWAKUP**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LPMBESL				**LPMLINKSTS**			

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**LPMRWAKUP**	**LPM Remote Wake-up** This bit contains the bRemoteWake value received with last ACK LPM Token.
[7:4]	**LPMBESL**	**LPM Best Effort Service Latency** These bits contain the BESL value received with last ACK LPM Token. 0000 = 125us. 0001 = 150us. 0010 = 200us. 0011 = 300us. 0100 = 400us. 0101 = 500us. 0110 = 1000us. 0111 = 2000us. 1000 = 3000us. 1001 = 4000us. 1010 = 5000us. 1011 = 6000us. 1100 = 7000us. 1101 = 8000us. 1110 = 9000us. 1111 = 10000us.
[3:0]	**LPMLINKSTS**	**LPM Link State** These bits contain the bLinkState received with last ACK LPM Token. 0000 = Reserve. 0001 = L1 (Sleep). 0010 – 1111 = Reserve.

USB Frame Number Register (USBD_FN)

Register	Offset	R/W	Description	Reset Value
USBD_FN	USBD_BA+0x08C	R	USB Frame Number Register	0x0000_0XXX

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

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10:0]	**FN**	**Frame Number** These bits contain the 11-bits frame number in the last received SOF packet.

USB Drive SE0 Register (USBD_SE0)

Register	Offset	R/W	Description	Reset Value
USBD_SE0	USBD_BA+0x090	R/W	USB Device Drive SE0 Control Register	0x0000_0001

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**SE0**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**SE0**	**Drive Single Ended Zero in USB Bus** The Single Ended Zero (SE0) is when both lines (USB_D+ and USB_D-) are being pulled low. 0 = Normal operation. 1 = Force USB PHY transceiver to drive SE0.

USB Buffer Segmentation Register (USB_BUFSEGx)

Register	Offset	R/W	Description	Reset Value
USBD_BUFSEG0	USBD_BA+0x500	R/W	Endpoint 0 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG1	USBD_BA+0x510	R/W	Endpoint 1 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG2	USBD_BA+0x520	R/W	Endpoint 2 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG3	USBD_BA+0x530	R/W	Endpoint 3 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG4	USBD_BA+0x540	R/W	Endpoint 4 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG5	USBD_BA+0x550	R/W	Endpoint 5 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG6	USBD_BA+0x560	R/W	Endpoint 6 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG7	USBD_BA+0x570	R/W	Endpoint 7 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG8	USBD_BA+0x580	R/W	Endpoint 8 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG9	USBD_BA+0x590	R/W	Endpoint 9 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG10	USBD_BA+0x5A0	R/W	Endpoint 10 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG11	USBD_BA+0x5B0	R/W	Endpoint 11 Buffer Segmentation Register	0x0000_0000

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

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8:3]	**BUFSEG**	**Endpoint Buffer Segmentation** It is used to indicate the offset address for each endpoint with the USB SRAM starting address. The effective starting address of the endpoint is USBD_SRAM address + {BUFSEG, 3'b000} Where the USBD_SRAM address = USBD_BA+0x100h. Refer to the section 6.32.5.7 for the endpoint SRAM structure and its description.
[2:0]	**Reserved**	Reserved.

USB Maximal Payload Register (USB_MXPLDx)

Register	Offset	R/W	Description	Reset Value
USBD_MXPLD0	USBD_BA+0x504	R/W	Endpoint 0 Maximal Payload Register	0x0000_0000
USBD_MXPLD1	USBD_BA+0x514	R/W	Endpoint 1 Maximal Payload Register	0x0000_0000
USBD_MXPLD2	USBD_BA+0x524	R/W	Endpoint 2 Maximal Payload Register	0x0000_0000
USBD_MXPLD3	USBD_BA+0x534	R/W	Endpoint 3 Maximal Payload Register	0x0000_0000
USBD_MXPLD4	USBD_BA+0x544	R/W	Endpoint 4 Maximal Payload Register	0x0000_0000
USBD_MXPLD5	USBD_BA+0x554	R/W	Endpoint 5 Maximal Payload Register	0x0000_0000
USBD_MXPLD6	USBD_BA+0x564	R/W	Endpoint 6 Maximal Payload Register	0x0000_0000
USBD_MXPLD7	USBD_BA+0x574	R/W	Endpoint 7 Maximal Payload Register	0x0000_0000
USBD_MXPLD8	USBD_BA+0x584	R/W	Endpoint 8 Maximal Payload Register	0x0000_0000
USBD_MXPLD9	USBD_BA+0x594	R/W	Endpoint 9 Maximal Payload Register	0x0000_0000
USBD_MXPLD10	USBD_BA+0x5A4	R/W	Endpoint 10 Maximal Payload Register	0x0000_0000
USBD_MXPLD11	USBD_BA+0x5B4	R/W	Endpoint 11 Maximal Payload Register	0x0000_0000

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

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8:0]	**MXPLD**	**Maximal Payload** Define the data length which is transmitted to host (IN token) or the actual data length which is received from the host (OUT token). It also used to indicate that the endpoint is ready to be transmitted in IN token or received in OUT token. (1) When the register is written by CPU, For IN token, the value of MXPLD is used to define the data length to be transmitted and indicate the data buffer is ready. For OUT token, it means that the controller is ready to receive data from the host and the value of MXPLD is the maximal data length comes from host. (2) When the register is read by CPU,

		For IN token, the value of MXPLD is indicated by the data length be transmitted to host. For OUT token, the value of MXPLD is indicated the actual data length receiving from host. **Note:** Once MXPLD is written, the data packets will be transmitted/received immediately after IN/OUT token arrived.

USB Configuration Register (USB_CFGx)

Register	Offset	R/W	Description	Reset Value
USBD_CFG0	USBD_BA+0x508	R/W	Endpoint 0 Configuration Register	0x0000_0000
USBD_CFG1	USBD_BA+0x518	R/W	Endpoint 1 Configuration Register	0x0000_0000
USBD_CFG2	USBD_BA+0x528	R/W	Endpoint 2 Configuration Register	0x0000_0000
USBD_CFG3	USBD_BA+0x538	R/W	Endpoint 3 Configuration Register	0x0000_0000
USBD_CFG4	USBD_BA+0x548	R/W	Endpoint 4 Configuration Register	0x0000_0000
USBD_CFG5	USBD_BA+0x558	R/W	Endpoint 5 Configuration Register	0x0000_0000
USBD_CFG6	USBD_BA+0x568	R/W	Endpoint 6 Configuration Register	0x0000_0000
USBD_CFG7	USBD_BA+0x578	R/W	Endpoint 7 Configuration Register	0x0000_0000
USBD_CFG8	USBD_BA+0x588	R/W	Endpoint 8 Configuration Register	0x0000_0000
USBD_CFG9	USBD_BA+0x598	R/W	Endpoint 9 Configuration Register	0x0000_0000
USBD_CFG10	USBD_BA+0x5A8	R/W	Endpoint 10 Configuration Register	0x0000_0000
USBD_CFG11	USBD_BA+0x5B8	R/W	Endpoint 11 Configuration Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**DBEN**	**DBTGACTIVE**	**CSTALL**	**Reserved**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DSQSYNC	**STATE**		**ISOCH**	**EPNUM**			

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11]	**DBEN**	**Double Buffer Enable** 0 = Single buffer mode. 1 = Double buffer mode.
[10]	**DBTGACTIVE**	**Double Buffer Toggle Active Bit** 0 = Inactive in double buffer mode. 1 = Active in double buffer mode. When DBEN = 1 and endpoint successful transfer, DBTGACTIVE of each double buffer will be toggled automatically by hardware.

[9]	**CSTALL**	**Clear STALL Response** 0 = Disable the device to clear the STALL handshake in setup stage. 1 = Clear the device to response STALL handshake in setup stage.
[8]	**Reserved**	Reserved.
[7]	**DSQSYNC**	**Data Sequence Synchronization** 0 = DATA0 PID. 1 = DATA1 PID. For IN token, DSQSYNC specify DATA0 or DATA1 PID in transfer data packet For OUT token, DSQSYNC specify DATA0 or DATA1 PID in received data packet. DSQSYNC will be toggled automatically by hardware when IN or OUT token transfer successfully in single buffer mode, but won't be toggled in double buffer mode. **Note 1:** When double buffer is enabled, hardware will automatically write 0 to DSQSYNC with active double buffer and write 1 to DSQSYNC with inactive double buffer. **Note 2:** It won't be toggled by hardware when DBEN = 1. USB data toggle will be guaranteed by changing endpoint.
[6:5]	**STATE**	**Endpoint State** 00 = Endpoint Disabled. 01 = Out endpoint. 10 = IN endpoint. 11 = Undefined.
[4]	**ISOCH**	**Isochronous Endpoint** This bit is used to set the endpoint as Isochronous endpoint, no handshake. 0 = No Isochronous endpoint. 1 = Isochronous endpoint.
[3:0]	**EPNUM**	**Endpoint Number** These bits are used to define the endpoint number of the current endpoint.

USB Extra Configuration Register (USB_CFGPx)

Register	Offset	R/W	Description	Reset Value
USBD_CFGP0	USBD_BA+0x50C	R/W	Endpoint 0 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP1	USBD_BA+0x51C	R/W	Endpoint 1 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP2	USBD_BA+0x52C	R/W	Endpoint 2 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP3	USBD_BA+0x53C	R/W	Endpoint 3 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP4	USBD_BA+0x54C	R/W	Endpoint 4 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP5	USBD_BA+0x55C	R/W	Endpoint 5 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP6	USBD_BA+0x56C	R/W	Endpoint 6 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP7	USBD_BA+0x57C	R/W	Endpoint 7 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP8	USBD_BA+0x58C	R/W	Endpoint 8 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP9	USBD_BA+0x59C	R/W	Endpoint 9 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP10	USBD_BA+0x5AC	R/W	Endpoint 10 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP11	USBD_BA+0x5BC	R/W	Endpoint 11 Set Stall and Clear In/Out Ready Control Register	0x0000_0000

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

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**SSTALL**	**Set STALL** 0 = Disable the device to response STALL. 1 = Set the device to respond STALL automatically.
[0]	**CLRRDY**	**Clear Ready** When the USBD_MXPLDx register is set by user, it means that the endpoint is ready to transmit or receive data. If the user wants to disable this transaction before the transaction start, users can set this bit to 1 to disable it and it is automatically cleared to 0. For IN token, write '1' to clear the IN token had ready to transmit the data to USB. For OUT token, write '1' to clear the OUT token had ready to receive the data from USB.

		This bit is written 1 only and is always 0 when it is read back.

6.33 USB 1.1 Host Controller (USBH)

6.33.1 Overview

This chip is equipped with a USB 1.1 Host Controller (USBH) that supports Open Host Controller Interface (OpenHCI, OHCI) Specification, a register-level description of a host controller, to manage the devices and data transfer of Universal Serial Bus (USB).

The USBH supports an integrated Root Hub with a USB port, a DMA for real-time data transfer between system memory and USB bus, port power control and port overcurrent detection.

The USBH is responsible for detecting the connect and disconnect of USB devices, managing data transfer, collecting status and activity of USB bus, providing power control and detecting overcurrent of attached USB devices.

6.33.2 Features

- Compliant with Universal Serial Bus (USB) Specification Revision 1.1.
- Supports Open Host Controller Interface (OpenHCI) Specification Revision 1.0.
- Supports both full-speed (12Mbps) and low-speed (1.5Mbps) USB devices.
- Supports Control, Bulk, Interrupt and Isochronous transfers.
- Supports an integrated Root Hub.
- Supports a USB host port shared with USB device (OTG function).
- Supports port power control and port overcurrent detection.
- Supports DMA for real-time data transfer.

6.33.3 Block Diagram

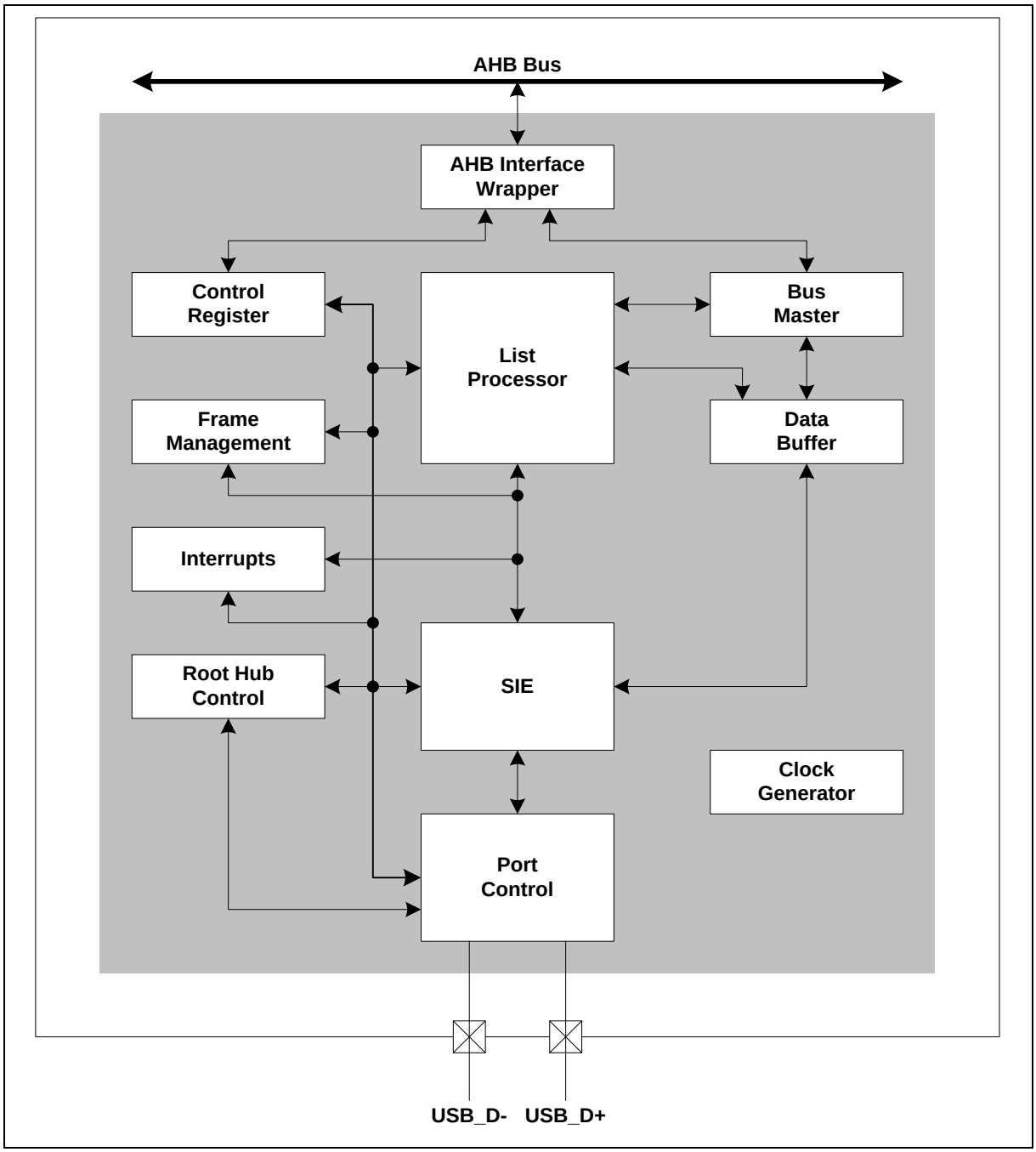


Figure 6.33-1 USB 1.1 Host Controller Block Diagram

6.33.4 Basic Configuration

6.33.4.1 *USBH OHCI Controller Basic Configuration*

- Clock Source Configuration
  - Select the source of USBH engine clock on USBSEL (CLK_CLKSEL0[8]).
  - Select the clock divider number of USBH engine clock on USBDIV (CLK_CLKDIV0[7:4]).
  - Enable USBH engine clock in USBHCKEN (CLK_AHBCLK[16]).
  - Select HCLK clock source to be more than or equal to 24 MHz.
- Reset Configuration
  - Reset USBH controller in USBHRST (SYS_IPRST0[4]).
- Pin Configuration

Group	Pin Name	GPIO	MFP
USB	USB_D+	PA.14	MFP14
	USB_D-	PA.13	MFP14
	USB_OTG_ID	PA.15	MFP14
	USB_VBUS	PA.12	MFP14
	USB_VBUS_EN	PB.6, PB.15	MFP14
	USB_VBUS_ST	PB.7, PB.14, PD.4	MFP14

6.33.5 Functional Description

6.33.5.1 *OHCI Controller*

AHB Interface

The OpenHCI Host Controller is connected to the system by the AHB bus. The design requires both master and slave bus operations. As a master, the Host Controller is responsible for running cycles on the AHB bus to access EDs and TDs as well as transferring data between memory and the local data buffer. As a slave, the Host Controller monitors the cycles on the AHB bus and determines when to respond to these cycles. Configuration and non-real-time control access to the Host Controller operational registers are through the AHB bus slave interface.

Host Controller

The host controller includes 5 functional blocks, including List Processing, Frame Management, Interrupt Processing, Host Controller Bus Master and Data Buffer.

The List Processor manages the data structures from the Host Controller Driver and coordinates all activity within the Host Controller.

The Frame Management is responsible for managing the frame specific tasks required by the USB specification and the OpenHCI specification. These tasks are:

- Management of the OpenHCI frame specific Operational Registers
- Operation of the Largest Data Packet Counter.
- Performing frame qualifications on USB Transaction requests to the SIE.
- Generate SOF token requests to the SIE.

Interrupts are the communication method for HC-initiated communication with the Host Controller Driver. There are several events that may trigger an interrupt from the Host Controller. Each specific event sets a specific bit in the HcInterruptStatus register.

The Host Controller Bus Master is the central block in the data path. The Host Controller Bus Master coordinates all access to the AHB Interface. There are two sources of bus mastering within Host Controller: the List Processor and the Data Buffer Engine.

The Data Buffer serves as the data interface between the Bus Master and the SIE. It is a combination of a 64-byte latched based bi-directional asynchronous FIFO and a single DWORD AHB Holding Register.

USB Interface

The USB interface includes the integrated Root Hub with an USB port, Port 1 as well as the Serial Interface Engine (SIE) and USB clock generator. The interface combines responsibility for executing bus transactions requested by the HC as well as the hub and port management specified by USB.

The SIE is responsible for managing all transactions to the USB. It controls the bus protocol, packet generation/extraction, data parallel-to-serial conversion, CRC coding, bit stuffing, and NRZI encoding. All transactions on the USB are requested from the List Processor and Frame Manager.

The Root Hub is a collection of ports that are individually controlled and a hub that maintains control/status over functions common to all ports.

6.33.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
USBH Base Address: **USBH_BA = 0x4000_9000**				
HcRevision	USBH_BA+0x000	R	Host Controller Revision Register	0x0000_0110
HcControl	USBH_BA+0x004	R/W	Host Controller Control Register	0x0000_0000
HcCommandStatus	USBH_BA+0x008	R/W	Host Controller Command Status Register	0x0000_0000
HcInterruptStatus	USBH_BA+0x00C	R/W	Host Controller Interrupt Status Register	0x0000_0000
HcInterruptEnable	USBH_BA+0x010	R/W	Host Controller Interrupt Enable Register	0x0000_0000
HcInterruptDisable	USBH_BA+0x014	R/W	Host Controller Interrupt Disable Register	0x0000_0000
HcHCCA	USBH_BA+0x018	R/W	Host Controller Communication Area Register	0x0000_0000
HcPeriodCurrentED	USBH_BA+0x01C	R/W	Host Controller Period Current ED Register	0x0000_0000
HcControlHeadED	USBH_BA+0x020	R/W	Host Controller Control Head ED Register	0x0000_0000
HcControlCurrentED	USBH_BA+0x024	R/W	Host Controller Control Current ED Register	0x0000_0000
HcBulkHeadED	USBH_BA+0x028	R/W	Host Controller Bulk Head ED Register	0x0000_0000
HcBulkCurrentED	USBH_BA+0x02C	R/W	Host Controller Bulk Current ED Register	0x0000_0000
HcDoneHead	USBH_BA+0x030	R/W	Host Controller Done Head Register	0x0000_0000
HcFmInterval	USBH_BA+0x034	R/W	Host Controller Frame Interval Register	0x0000_2EDF
HcFmRemaining	USBH_BA+0x038	R	Host Controller Frame Remaining Register	0x0000_0000
HcFmNumber	USBH_BA+0x03C	R	Host Controller Frame Number Register	0x0000_0000
HcPeriodicStart	USBH_BA+0x040	R/W	Host Controller Periodic Start Register	0x0000_0000
HcLSThreshold	USBH_BA+0x044	R/W	Host Controller Low-speed Threshold Register	0x0000_0628
HcRhDescriptorA	USBH_BA+0x048	R/W	Host Controller Root Hub Descriptor A Register	0x0100_0001
HcRhDescriptorB	USBH_BA+0x04C	R/W	Host Controller Root Hub Descriptor B Register	0x0000_0000
HcRhStatus	USBH_BA+0x050	R/W	Host Controller Root Hub Status Register	0x0000_0000
HcRhPortStatus1	USBH_BA+0x058	R/W	Host Controller Root Hub Port Status [1]	0x0000_0000
HcPhyControl	USBH_BA+0x200	R/W	Host Controller PHY Control Register	0x080F_0000
HcMiscControl	USBH_BA+0x204	R/W	Host Controller Miscellaneous Control Register	0x0000_0000

6.33.7 Register Description

Host Controller Revision Register (HcRevision)

Register	Offset	R/W	Description	Reset Value
HcRevision	USBH_BA+0x000	R	Host Controller Revision Register	0x0000_0110

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**REV**	**Revision Number** Indicates the Open HCI Specification revision number implemented by the Hardware. Host Controller supports 1.1 specification. (X.Y = XYh).

Host Controller Control Register (HcControl)

Register	Offset	R/W	Description	Reset Value
HcControl	USBH_BA+0x004	R/W	Host Controller Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HCFS		**BLE**	**CLE**	**IE**	**PLE**	**CBSR**	

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:6]	**HCFS**	**Host Controller Functional State** This field sets the Host Controller state. The Controller may force a state change from USBSUSPEND to USBRESUME after detecting resume signaling from a downstream port. States are: 00 = USBRESET. 01 = USBRESUME. 10 = USBOPERATIONAL. 11 = USBSUSPEND.
[5]	**BLE**	**Bulk List Enable Bit** 0 = Processing of the Bulk list after next SOF (Start-Of-Frame) Disabled. 1 = Processing of the Bulk list in the next frame Enabled.
[4]	**CLE**	**Control List Enable Bit** 0 = Processing of the Control list after next SOF (Start-Of-Frame) Disabled. 1 = Processing of the Control list in the next frame Enabled.
[3]	**IE**	**Isochronous List Enable Bit** Both IE and PLE (HcControl[2]) high enables Host Controller to process the Isochronous list. Either IE or PLE (HcControl[2]) is low disables Host Controller to process the Isochronous list. 0 = Processing of the Isochronous list after next SOF (Start-Of-Frame) Disabled. 1 = Processing of the Isochronous list in the next frame Enabled, if the PLE (HcControl[2]) is high, too.
[2]	**PLE**	**Periodic List Enable Bit** When set, this bit enables processing of the Periodic (interrupt and isochronous) list. The Host Controller checks this bit prior to attempting any periodic transfers in a frame. 0 = Processing of the Periodic (Interrupt and Isochronous) list after next SOF (Start-Of-Frame) Disabled. 1 = Processing of the Periodic (Interrupt and Isochronous) list in the next frame Enabled. **Note:** To enable the processing of the Isochronous list, user has to set both PLE and IE (HcControl[3]) high.

[1:0]	**CBSR**	**Control Bulk Service Ratio** This specifies the service ratio between Control and Bulk EDs. Before processing any of the non-periodic lists, HC must compare the ratio specified with its internal count on how many nonempty Control EDs have been processed, in determining whether to continue serving another Control ED or switching to Bulk EDs. The internal count will be retained when crossing the frame boundary. In case of reset, HCD is responsible for restoring this value. 00 = Number of Control EDs over Bulk EDs served is 1:1. 01 = Number of Control EDs over Bulk EDs served is 2:1. 10 = Number of Control EDs over Bulk EDs served is 3:1. 11 = Number of Control EDs over Bulk EDs served is 4:1.

Host Controller Command Status Register (HcCommandStatus)

Register	Offset	R/W	Description	Reset Value
HcCommandStatus	USBH_BA+0x008	R/W	Host Controller Command Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						SOC	
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					BLF	CLF	HCR

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17:16]	**SOC**	**Schedule Overrun Count (Read Only)** These bits are incremented on each scheduling overrun error. It is initialized to 00b and wraps around at 11b. This will be incremented when a scheduling overrun is detected even if SO (HcInterruptStatus[0]) has already been set.
[15:3]	**Reserved**	Reserved.
[2]	**BLF**	**Bulk List Filled** Set high to indicate there is an active TD on the Bulk list. This bit may be set by either software or the Host Controller and cleared by the Host Controller each time it begins processing the head of the Bulk list. 0 = No active TD found or Host Controller begins to process the head of the Bulk list. 1 = An active TD added or found on the Bulk list.
[1]	**CLF**	**Control List Filled** Set high to indicate there is an active TD on the Control List. It may be set by either software or the Host Controller and cleared by the Host Controller each time it begins processing the head of the Control List. 0 = No active TD found or Host Controller begins to process the head of the Control list. 1 = An active TD added or found on the Control list.
[0]	**HCR**	**Host Controller Reset** This bit is set to initiate the software reset of Host Controller. This bit is cleared by the Host Controller, upon completed of the reset operation. This bit, when set, didn't reset the Root Hub and no subsequent reset signaling be asserted to its downstream ports. 0 = Host Controller is not in software reset state. 1 = Host Controller is in software reset state.

Host Controller Interrupt Status Register (HcInterruptStatus)

Register	Offset	R/W	Description	Reset Value
HcInterruptStatus	USBH_BA+0x00C	R/W	Host Controller Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**RHSC**	**FNO**	**Reserved**	**RD**	**SF**	**WDH**	**SO**

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**RHSC**	**Root Hub Status Change** This bit is set when the content of HcRhStatus or the content of HcRhPortStatus1 register has changed. 0 = The content of HcRhStatus and the content of HcRhPortStatus1 register didn't change. 1 = The content of HcRhStatus or the content of HcRhPortStatus1 register has changed. **Note:** This bit is cleared by writing '1Fh' to HcRhPortStatus1[20:16].
[5]	**FNO**	**Frame Number Overflow** This bit is set when bit 15 of Frame Number changes from 1 to 0 or from 0 to 1. 0 = The bit 15 of Frame Number didn't change. 1 = The bit 15 of Frame Number changes from 1 to 0 or from 0 to 1. **Note:** This bit is cleared by writing 1 to it.
[4]	**Reserved**	Reserved.
[3]	**RD**	**Resume Detected** Set when Host Controller detects resume signaling on a downstream port. 0 = No resume signaling detected on a downstream port. 1 = Resume signaling detected on a downstream port. **Note:** This bit is cleared by writing 1 to it.
[2]	**SF**	**Start of Frame** Set when the Frame Management functional block signals a 'Start of Frame' event. Host Control generates a SOF token at the same time. 0 = Not the start of a frame. 1 = Indicate the start of a frame and Host Controller generates a SOF token. **Note:** This bit is cleared by writing 1 to it.

[1]	**WDH**	**Write Back Done Head** Set after the Host Controller has written HcDoneHead to HccaDoneHead. Further updates of the HccaDoneHead will not occur until this bit has been cleared. 0 = Host Controller didn't update HccaDoneHead. 1 = Host Controller has written HcDoneHead to HccaDoneHead. **Note:** This bit is cleared by writing 1 to it.
[0]	**SO**	**Scheduling Overrun** Set when the List Processor determines a Schedule Overrun has occurred. 0 = Schedule Overrun didn't occur. 1 = Schedule Overrun has occurred. **Note:** This bit is cleared by writing 1 to it.

Host Controller Interrupt Enable Register (HcInterruptEnable)

Register	Offset	R/W	Description	Reset Value
HcInterruptEnable	USBH_BA+0x010	R/W	Host Controller Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
MIE	**Reserved**						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**RHSC**	**FNO**	**Reserved**	**RD**	**SF**	**WDH**	**SO**

Bits	Description	
[31]	**MIE**	**Master Interrupt Enable Bit** This bit is a global interrupt enable. A write of '1' allows interrupts to be enabled via the specific enable bits listed above. Write Operation: 0 = No effect. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Enabled if the corresponding bit in HcInterruptEnable is high. Read Operation: 0 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Disabled even if the corresponding bit in HcInterruptEnable is high. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Enabled if the corresponding bit in HcInterruptEnable is high.
[30:7]	**Reserved**	Reserved.
[6]	**RHSC**	**Root Hub Status Change Enable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Enabled. Read Operation: 0 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Disabled. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Enabled.

[5]	**FNO**	**Frame Number Overflow Enable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to FNO (HcInterruptStatus[5]) Enabled. Read Operation: 0 = Interrupt generation due to FNO (HcInterruptStatus[5]) Disabled. 1 = Interrupt generation due to FNO (HcInterruptStatus[5]) Enabled.
[4]	**Reserved**	Reserved.
[3]	**RD**	**Resume Detected Enable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to RD (HcInterruptStatus[3]) Enabled. Read Operation: 0 = Interrupt generation due to RD (HcInterruptStatus[3]) Disabled. 1 = Interrupt generation due to RD (HcInterruptStatus[3]) Enabled.
[2]	**SF**	**Start of Frame Enable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to SF (HcInterruptStatus[2]) Enabled. Read Operation: 0 = Interrupt generation due to SF (HcInterruptStatus[2]) Disabled. 1 = Interrupt generation due to SF (HcInterruptStatus[2]) Enabled.
[1]	**WDH**	**Write Back Done Head Enable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to WDH (HcInterruptStatus[1]) Enabled. Read Operation: 0 = Interrupt generation due to WDH (HcInterruptStatus[1]) Disabled. 1 = Interrupt generation due to WDH (HcInterruptStatus[1]) Enabled.
[0]	**SO**	**Scheduling Overrun Enable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to SO (HcInterruptStatus[0]) Enabled. Read Operation: 0 = Interrupt generation due to SO (HcInterruptStatus[0]) Disabled. 1 = Interrupt generation due to SO (HcInterruptStatus[0]) Enabled.

Host Controller Interrupt Disable Register (HcInterruptDisable)

Register	Offset	R/W	Description	Reset Value
HcInterruptDisable	USBH_BA+0x014	R/W	Host Controller Interrupt Disable Register	0x0000_0000

31	30	29	28	27	26	25	24
MIE	Reserved						
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved	RHSC	FNO	Reserved	RD	SF	WDH	SO

Bits	Description	
[31]	**MIE**	**Master Interrupt Disable Bit** Global interrupt disable. Writing '1' to disable all interrupts. Write Operation: 0 = No effect. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Disabled if the corresponding bit in HcInterruptEnable is high. Read Operation: 0 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Disabled even if the corresponding bit in HcInterruptEnable is high. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Enabled if the corresponding bit in HcInterruptEnable is high.
[30:7]	**Reserved**	Reserved.
[6]	**RHSC**	**Root Hub Status Change Disable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Disabled. Read Operation: 0 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Disabled. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Enabled.

[5]	**FNO**	**Frame Number Overflow Disable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to FNO (HcInterruptStatus[5]) Disabled. Read Operation: 0 = Interrupt generation due to FNO (HcInterruptStatus[5]) Disabled. 1 = Interrupt generation due to FNO (HcInterruptStatus[5]) Enabled.
[4]	**Reserved**	Reserved.
[3]	**RD**	**Resume Detected Disable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to RD (HcInterruptStatus[3]) Disabled. Read Operation: 0 = Interrupt generation due to RD (HcInterruptStatus[3]) Disabled. 1 = Interrupt generation due to RD (HcInterruptStatus[3]) Enabled.
[2]	**SF**	**Start of Frame Disable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to SF (HcInterruptStatus[2]) Disabled. Read Operation: 0 = Interrupt generation due to SF (HcInterruptStatus[2]) Disabled. 1 = Interrupt generation due to SF (HcInterruptStatus[2]) Enabled.
[1]	**WDH**	**Write Back Done Head Disable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to WDH (HcInterruptStatus[1]) Disabled. Read Operation: 0 = Interrupt generation due to WDH (HcInterruptStatus[1]) Disabled. 1 = Interrupt generation due to WDH (HcInterruptStatus[1]) Enabled.
[0]	**SO**	**Scheduling Overrun Disable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to SO (HcInterruptStatus[0]) Disabled. Read Operation: 0 = Interrupt generation due to SO (HcInterruptStatus[0]) Disabled. 1 = Interrupt generation due to SO (HcInterruptStatus[0]) Enabled.

Host Controller Communication Area Register (HcHCCA)

Register	Offset	R/W	Description	Reset Value
HcHCCA	USBH_BA+0x018	R/W	Host Controller Communication Area Register	0x0000_0000

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

Bits	Description	
[31:8]	**HCCA**	**Host Controller Communication Area** Pointer to indicate the base address of the Host Controller Communication Area (HCCA).
[7:0]	**Reserved**	Reserved.

Host Controller Period Current ED Register (HcPeriodCurrentED)

Register	Offset	R/W	Description	Reset Value
HcPeriodCurrentED	USBH_BA+0x01C	R/W	Host Controller Period Current ED Register	0x0000_0000

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

Bits	Description	
[31:4]	**PCED**	**Periodic Current ED** Pointer to indicate the physical address of the current Isochronous or Interrupt Endpoint Descriptor.
[3:0]	**Reserved**	Reserved.

Host Controller Control Head ED Register (HcControlHeadED)

Register	Offset	R/W	Description	Reset Value
HcControlHeadED	USBH_BA+0x020	R/W	Host Controller Control Head ED Register	0x0000_0000

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

Bits	Description	
[31:4]	**CHED**	**Control Head ED** Pointer to indicate the physical address of the first Endpoint Descriptor of the Control list.
[3:0]	**Reserved**	Reserved.

Host Controller Control Current ED Register (HcControlCurrentED)

Register	Offset	R/W	Description	Reset Value
HcControlCurrentED	USBH_BA+0x024	R/W	Host Controller Control Current ED Register	0x0000_0000

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

Bits	Description	
[31:4]	**CCED**	**Control Current Head ED** Pointer to indicate the physical address of the current Endpoint Descriptor of the Control list.
[3:0]	**Reserved**	Reserved.

Host Controller Bulk Head ED Register (HcBulkHeadED)

Register	Offset	R/W	Description	Reset Value
HcBulkHeadED	USBH_BA+0x028	R/W	Host Controller Bulk Head ED Register	0x0000_0000

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

Bits	Description	
[31:4]	**BHED**	**Bulk Head ED** Pointer to indicate the physical address of the first Endpoint Descriptor of the Bulk list.
[3:0]	**Reserved**	Reserved.

Host Controller Bulk Current ED Register (HcBulkCurrentED)

Register	Offset	R/W	Description	Reset Value
HcBulkCurrentED	USBH_BA+0x02C	R/W	Host Controller Bulk Current ED Register	0x0000_0000

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

Bits	Description	
[31:4]	**BCED**	**Bulk Current Head ED** Pointer to indicate the physical address of the current Endpoint Descriptor of the Bulk list.
[3:0]	**Reserved**	Reserved.

Host Controller Done Head Register (HcDoneHead)

Register	Offset	R/W	Description	Reset Value
HcDoneHead	USBH_BA+0x030	R/W	Host Controller Done Head Register	0x0000_0000

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

Bits	Description	
[31:4]	**DH**	**Done Head** Pointer to indicate the physical address of the last completed Transfer Descriptor that was added to the Done queue.
[3:0]	**Reserved**	Reserved.

Host Controller Frame Interval Register (HcFmInterval)

Register	Offset	R/W	Description	Reset Value
HcFmInterval	USBH_BA+0x034	R/W	Host Controller Frame Interval Register	0x0000_2EDF

31	30	29	28	27	26	25	24
FIT	**Reserved**	**FSMPS**					
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
FSMPS							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**FI**					
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FI							

Bits	Description	
[31]	**FIT**	**Frame Interval Toggle** This bit is toggled by Host Controller Driver when it loads a new value into FI (HcFmInterval[13:0]). 0 = Host Controller Driver didn't load new value into FI (HcFmInterval[13:0]). 1 = Host Controller Driver loads a new value into FI (HcFmInterval[13:0]).
[30]	**Reserved**	Reserved.
[29:16]	**FSMPS**	**FS Largest Data Packet** This field specifies a value that is loaded into the Largest Data Packet Counter at the beginning of each frame.
[15:14]	**Reserved**	Reserved.
[13:0]	**FI**	**Frame Interval** This field specifies the length of a frame as (bit times - 1). For 12,000 bit times in a frame, a value of 11,999 is stored here.

Host Controller Frame Remaining Register (HcFmRemaining)

Register	Offset	R/W	Description	Reset Value
HcFmRemaining	USBH_BA+0x038	R	Host Controller Frame Remaining Register	0x0000_0000

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

Bits	Description	
[31]	**FRT**	**Frame Remaining Toggle** This bit is loaded from the FIT (HcFmInterval[31]) whenever FR (HcFmRemaining[13:0]) reaches 0.
[30:14]	**Reserved**	Reserved.
[13:0]	**FR**	**Frame Remaining** When the Host Controller is in the USBOPERATIONAL state, this 14-bit field decrements each 12 MHz clock period. When the count reaches 0, (end of frame) the counter reloads with Frame Interval. In addition, the counter loads when the Host Controller transitions into USBOPERATIONAL.

Host Controller Frame Number Register (HcFmNumber)

Register	Offset	R/W	Description	Reset Value
HcFmNumber	USBH_BA+0x03C	R	Host Controller Frame Number Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**FN**	**Frame Number** This 16-bit incrementing counter field is incremented coincident with the re-load of FR (HcFmRemaining[13:0]). The count rolls over from 'FFFFh' to '0h.'

Host Controller Periodic Start Register (HcPeriodicStart)

Register	Offset	R/W	Description	Reset Value
HcPeriodicStart	USBH_BA+0x040	R/W	Host Controller Periodic Start Register	0x0000_0000

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

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13:0]	**PS**	**Periodic Start** This field contains a value used by the List Processor to determine where in a frame the Periodic List processing must begin.

Host Controller Low-speed Threshold Register (HcLSThreshold)

Register	Offset	R/W	Description	Reset Value
HcLSThreshold	USBH_BA+0x044	R/W	Host Controller Low-speed Threshold Register	0x0000_0628

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

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**LST**	**Low-speed Threshold** This field contains a value which is compared to the FR (HcFmRemaining[13:0]) field prior to initiating a Low-speed transaction. The transaction is started only if FR (HcFmRemaining[13:0]) >= this field. The value is calculated by Host Controller Driver with the consideration of transmission and setup overhead.

Host Controller Root Hub Descriptor A Register (HcRhDescriptorA)

Register	Offset	R/W	Description	Reset Value
HcRhDescriptorA	USBH_BA+0x048	R/W	Host Controller Root Hub Descriptor A Register	0x0100_0001

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved			NOCP	OCPM	Reserved		PSM
7	6	5	4	3	2	1	0
NDP							

Bits	Description	
[31:13]	**Reserved**	Reserved.
[12]	**NOCP**	**No Overcurrent Protection** This bit describes how the overcurrent status for the Root Hub ports reported. 0 = Overcurrent status is reported. 1 = Overcurrent status is not reported.
[11]	**OCPM**	**Overcurrent Protection Mode** This bit describes how the overcurrent status for the Root Hub ports reported. This bit is only valid when NOCP (HcRhDescriptorA[12]) is cleared. 0 = Global overcurrent. 1 = Individual overcurrent.
[10:9]	**Reserved**	Reserved.
[8]	**PSM**	**Power Switching Mode** This bit is used to specify how the power switching of the Root Hub ports is controlled. 0 = Global switching. 1 = Individual switching.
[7:0]	**NDP**	**Number Downstream Ports** USB host control supports two downstream ports and only one port is available in this series of chip. **Note:** NDP = 1 in this series of chip

Host Controller Root Hub Descriptor B Register (HcRhDescriptorB)

Register	Offset	R/W	Description	Reset Value
HcRhDescriptorB	USBH_BA+0x04C	R/W	Host Controller Root Hub Descriptor B Register	0x0000_0000

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

Bits	Description	
[31:16]	**PPCM**	**Port Power Control Mask** Global power switching. This field is only valid if Power Switching Mode is set (individual port switching). When set, the port only responds to individual port power switching commands (Set/Clear Port Power). When cleared, the port only responds to global power switching commands (Set/Clear Global Power). 0 = Port power controlled by global power switching. 1 = Port power controlled by port power switching. **Note:** PPCM[15:2] and PPCM[0] are reserved.
[15:0]	**Reserved**	Reserved.

Host Controller Root Hub Status Register (HcRhStatus)

Register	Offset	R/W	Description	Reset Value
HcRhStatus	USBH_BA+0x050	R/W	Host Controller Root Hub Status Register	0x0000_0000

31	30	29	28	27	26	25	24
CRWE	Reserved						
23	22	21	20	19	18	17	16
Reserved						OCIC	LPSC
15	14	13	12	11	10	9	8
DRWE	Reserved						
7	6	5	4	3	2	1	0
Reserved						OCI	LPS

Bits	Description	
[31]	**CRWE**	**Clear Remote Wake-up Enable Bit** This bit is use to clear DRWE (HcRhStatus[15]). This bit is always read as 0. Write Operation: 0 = No effect. 1 = Clear DRWE (HcRhStatus[15]).
[30:18]	**Reserved**	Reserved.
[17]	**OCIC**	**Overcurrent Indicator Change** This bit is set by hardware when a change has occurred in OCI (HcRhStatus[1]). Write 1 to clear this bit to 0. 0 = OCI (HcRhStatus[1]) didn't change. 1 = OCI (HcRhStatus[1]) changed.
[16]	**LPSC**	**Set Global Power** In global power mode (PSM (HcRhDescriptorA[8]) = 0), this bit is written to one to enable power to all ports. This bit is always read as 0. Write Operation: 0 = No effect. 1 = Set global power.
[15]	**DRWE**	**Device Remote Wakeup Enable Bit** This bit controls if port's Connect Status Change as a remote wake-up event. Write Operation: 0 = No effect. 1 = Connect Status Change as a remote wake-up event Enabled. Read Operation: 0 = Connect Status Change as a remote wake-up event Disabled. 1 = Connect Status Change as a remote wake-up event Enabled.

[14:2]	**Reserved**	Reserved.
[1]	**OCI**	**Overcurrent Indicator (Read Only)** This bit reflects the state of the overcurrent status pin. This field is only valid if NOCP (HcRhDescriptorA[12]) and OCPM (HcRhDescriptorA[11]) are cleared. 0 = No overcurrent condition. 1 = Overcurrent condition.
[0]	**LPS**	**Clear Global Power** In global power mode (PSM (HcRhDescriptorA[8]) = 0), this bit is written to one to clear all ports' power. This bit is always read as 0. Write Operation: 0 = No effect. 1 = Clear global power.

Host Controller Root Hub Port Status [1] (HcRhPortStatus1)

Register	Offset	R/W	Description	Reset Value
HcRhPortStatus1	USBH_BA+0x058	R/W	Host Controller Root Hub Port Status [1]	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved			PRSC	OCIC	PSSC	PESC	CSC
15	14	13	12	11	10	9	8
Reserved						LSDA	PPS
7	6	5	4	3	2	1	0
Reserved			PRS	POCI	PSS	PES	CCS

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20]	**PRSC**	**Port Reset Status Change** This bit indicates that the port reset signal has completed. Write 1 to clear this bit to 0. 0 = Port reset is not complete. 1 = Port reset is complete.
[19]	**OCIC**	**Port Overcurrent Indicator Change** This bit is set when POCI (HcRhPortStatus1[3]) changes. Write 1 to clear this bit to 0. 0 = POCI (HcRhPortStatus1[3]) didn't change. 1 = POCI (HcRhPortStatus1[3]) changed.
[18]	**PSSC**	**Port Suspend Status Change** This bit indicates the completion of the selective resume sequence for the port. Write 1 to clear this bit to 0. 0 = Port resume is not complete. 1 = Port resume is complete.
[17]	**PESC**	**Port Enable Status Change** This bit indicates that the port has been disabled (PES (HcRhPortStatus1[1]) cleared) due to a hardware event. Write 1 to clear this bit to 0. 0 = PES (HcRhPortStatus1[1]) didn't change. 1 = PES (HcRhPortStatus1[1]) changed.

[16]	**CSC**	**Connect Status Change** This bit indicates connect or disconnect event has been detected (CCS (HcRhPortStatus1[0]) changed). Write 1 to clear this bit to 0. 0 = No connect/disconnect event (CCS (HcRhPortStatus1[0]) didn't change). 1 = Hardware detection of connect/disconnect event (CCS (HcRhPortStatus1[0]) changed).
[15:10]	**Reserved**	Reserved.
[9]	**LSDA**	**Low Speed Device Attached (Read) or Clear Port Power (Write)** This bit defines the speed (and bus idle) of the attached device. It is only valid when CCS (HcRhPortStatus1[0]) is set. This bit is also used to clear port power. Write Operation: 0 = No effect. 1 = Clear PPS (HcRhPortStatus1[8]). Read Operation: 0 = Full Speed device. 1 = Low-speed device.
[8]	**PPS**	**Port Power Status (Read) or Set Port Power (Write)** This bit reflects the power state of the port regardless of the power switching mode. Write Operation: 0 = No effect. 1 = Port Power Enabled. Read Operation: 0 = Port power is Disabled. 1 = Port power is Enabled.
[7:5]	**Reserved**	Reserved.
[4]	**PRS**	**Port Reset Status (Read) or Set Port Reset (Write)** This bit reflects the reset state of the port. Write Operation: 0 = No effect. 1 = Set port reset. Read Operation 0 = Port reset signal is not active. 1 = Port reset signal is active.
[3]	**POCI**	**Port Overcurrent Indicator (Read) or Clear Port Suspend (Write)** This bit reflects the state of the overcurrent status pin dedicated to this port. This field is only valid if NOCP (HcRhDescriptorA[12]) is cleared and OCPM (HcRhDescriptorA[11]) is set. This bit is also used to initiate the selective result sequence for the port. Write Operation: 0 = No effect. 1 = Clear port suspend. Read Operation: 0 = No overcurrent condition. 1 = Overcurrent condition.

[2]	**PSS**	**Port Suspend Status (Read) or Set Port Suspend (Write)** This bit indicates the port is suspended Write Operation: 0 = No effect. 1 = Set port suspend. Read Operation: 0 = Port is not suspended. 1 = Port is selectively suspended.
[1]	**PES**	**Port Enable Status (Read) or Set Port Enable (Write)** Write Operation: 0 = No effect. 1 = Set port enable. Read Operation: 0 = Port Disabled. 1 = Port Enabled.
[0]	**CCS**	**Current Connect Status (Read) or Clear Port Enable (Write)** Write Operation: 0 = No effect. 1 = Clear port enable. Read Operation: 0 = No device connected. 1 = Device connected.

Host Controller PHY Control Register (HcPhyControl)

Register	Offset	R/W	Description	Reset Value
HcPhyControl	USBH_BA+0x200	R/W	Host Controller PHY Control Register	0x080F_0000

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

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27]	**STBYEN**	**USB Transceiver Standby Enable Bit** This bit controls if USB transceiver could enter the standby mode to reduce power consumption. 0 = The USB transceiver would never enter the standby mode. 1 = The USB transceiver will enter standby mode while port is in power off state (port power is inactive).
[26:0]	**Reserved**	Reserved.

Host Controller Miscellaneous Control Register (HcMiscControl)

Register	Offset	R/W	Description	Reset Value
HcMiscControl	USBH_BA+0x204	R/W	Host Controller Miscellaneous Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							DPRT1
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			PPCAL	OCAL	Reserved	ABORT	Reserved

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**DPRT1**	**Disable Port 1** This bit controls if the connection between USB host controller and transceiver of port 1 is disabled. If the connection is disabled, the USB host controller will not recognize any event of USB bus. Set this bit high, the transceiver of port 1 will also be forced into the standby mode no matter what USB host controller operation is. 0 = The connection between USB host controller and transceiver of port 1 Enabled. 1 = The connection between USB host controller and transceiver of port 1 Disabled and the transceiver of port 1 will also be forced into the standby mode.
[15:5]	**Reserved**	Reserved.
[4]	**PPCAL**	**Port Power Control Active Low** This bit controls the polarity of port power control to external power IC. 0 = Port power control is high active. 1 = Port power control is low active.
[3]	**OCAL**	**Overcurrent Active Low** This bit controls the polarity of overcurrent flag from external power IC. 0 = Overcurrent flag is high active. 1 = Overcurrent flag is low active.
[2]	**Reserved**	Reserved.
[1]	**ABORT**	**AHB Bus Error Response** This bit indicates there is an Error response received in AHB bus. 0 = No Error response received. 1 = Error response received. **Note:** This bit is cleared by writing 1 to it.
[0]	**Reserved**	Reserved.

6.34 USB On-The-Go (OTG)

6.34.1 Overview

The OTG controller interfaces to USB PHY and USB controllers which consist of a USB 1.1 host controller and a USB 2.0 FS device controller. The OTG controller supports HNP and SRP protocols defined in the “On-The-Go and Embedded Host Supplement to the USB 2.0 Revision 2.0 Specification”.

USB frame, including USB host, USB device, and OTG controller, can be configured as Host-only, Device-only, ID dependent or OTG device mode defined in USBROLE (SYS_USBPHY[1:0]). In Host-only mode, USB frame acts as USB host. USB frame can support both full-speed and low-speed transfer. In Device-only mode, USB frame acts as USB device. USB frame only supports full-speed transfer. In ID dependent mode, USB frame can be USB Host or USB device depending on USB_ID pin state. In OTG device mode, the role of USB frame depends on the definition of OTG specification. USB frame only supports full-speed transfer when OTG device acts as a peripheral.

6.34.2 Features

- Built in USB PHY
- Configurable to operate as:
  - Host-only
  - Device-only
  - ID dependent: The role of USB frame is only dependent on USB_ID pin value--as USB Host (USB_ID pin is low) or USB Device (USB_ID pin is high). Not support HNP or SRP protocol.
  - OTG device: dependent on USB_ID pin status to be A-device (USB_ID pin is low) or B-device (USB_ID pin is high). Support HNP and SRP protocols.

6.34.3 Block Diagram

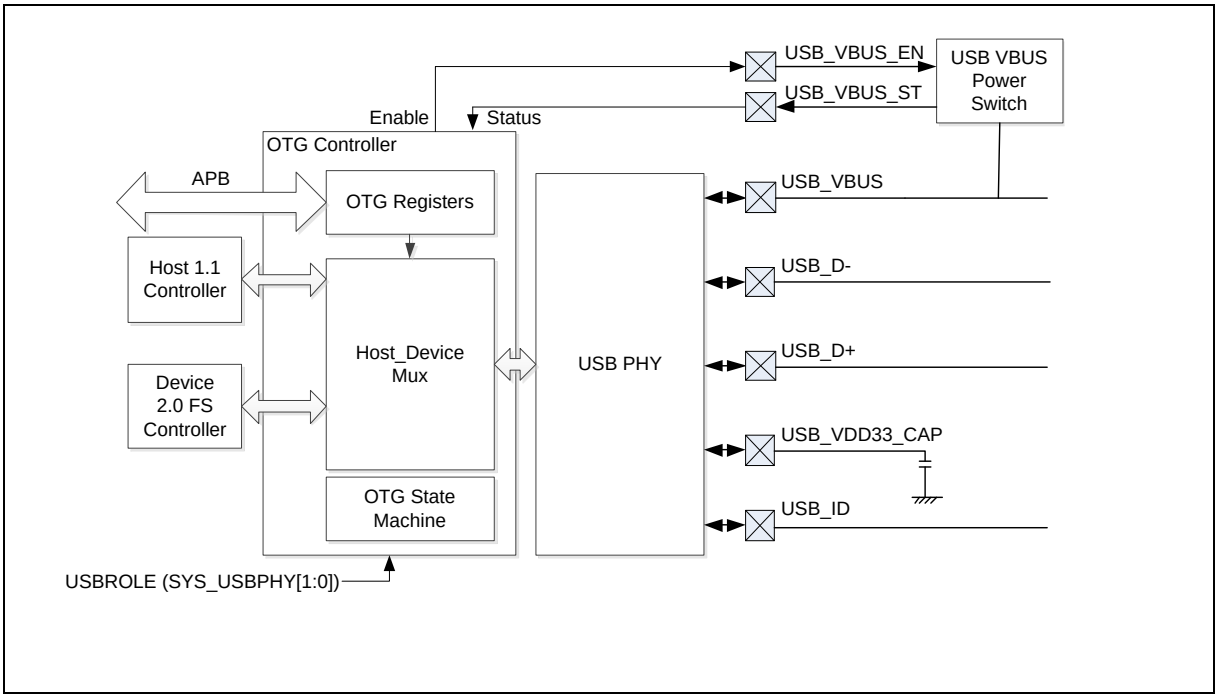


Figure 6.34-1 USB OTG Block Diagram

6.34.4 Basic Configuration

The OTG peripheral clock can be enabled by OTGCKEN (CLK_APBCLK0[26]). The role of USB frame is determined by USBROLE (SYS_USBPHY[1:0]). This configuration is write-protection bits. Before writing to these bits, user must disable the register protection function. Refer to the description of SYS_REGLCTL register for details. USB_VBUS_EN and USB_VBUS_ST pin functions are configured in SYS_GPB_MFPL, SYS_GPB_MFPH or SYS_GPD_MFPL registers.

6.34.5 Functional Description

The role of USB frame depends on the setting of USBROLE (SYS_USBPHY[1:0]) and USB_ID pin status. The USBROLE configuration has precedence over USB_ID pin status. User can configure the OTG controller to USB Host mode, USB Device mode, ID dependent mode or OTG device mode. In USB Host mode, the host controller will interact with USB PHY directly. In USB Device mode, the device controller will interact with USB PHY directly. In these cases, the OTG controller is used simply as a multiplexer. In ID dependent mode, USB_ID pin status will decide USB frame to act as USB host or USB device. If the USB_ID pin is FALSE state (low level), USB frame will act as USB host. If the USB_ID pin is TRUE state (high level), USB frame will act as USB device. In OTG device mode, the OTG controller will handle OTG HNP and SRP protocols. If the USB_ID pin is FALSE state (low level), the OTG controller will act as an OTG A-device. If the USB_ID pin is TRUE state (high level), the OTG controller will act as an OTG B-device.

6.34.5.1 The Role of USB Frame

USB Device Mode

When USBROLE (SYS_USBPHY[1:0]) is set to 0, USB frame acts as USB device. The USB host function is not available.

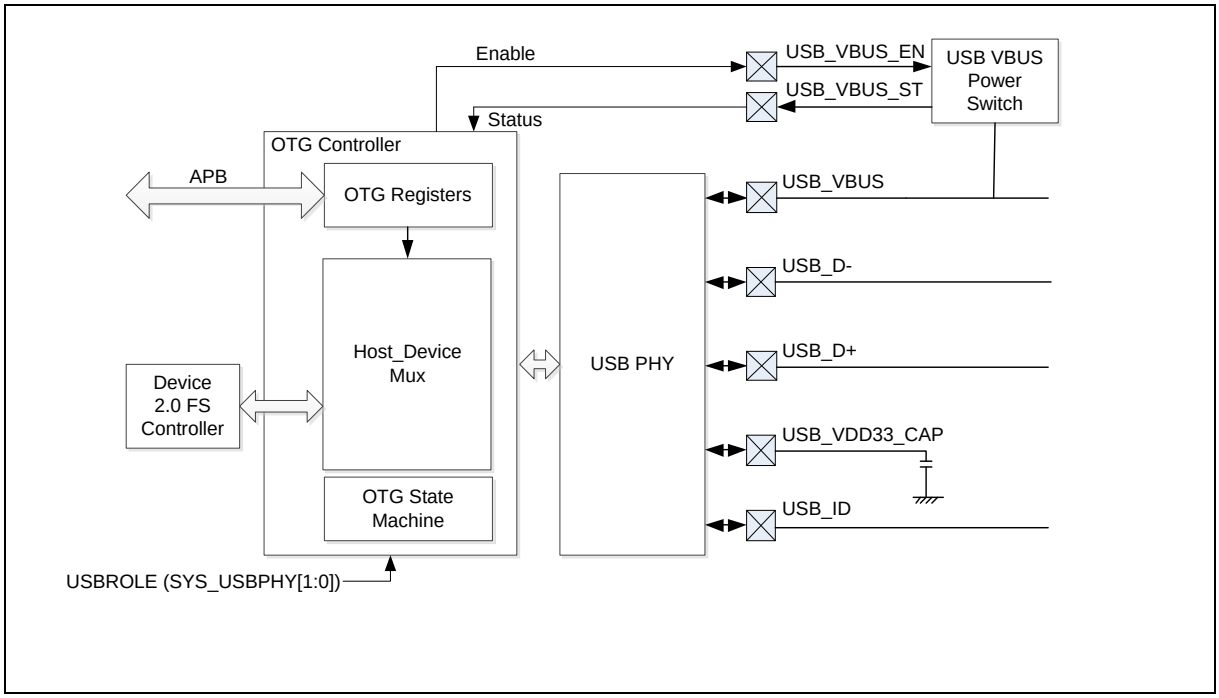


Figure 6.34-2 USB Device Mode

USB Host Mode

When USBROLE (SYS_USBPHY[1:0]) is set to 1, USB frame acts as USB host. The USB device function is not available.

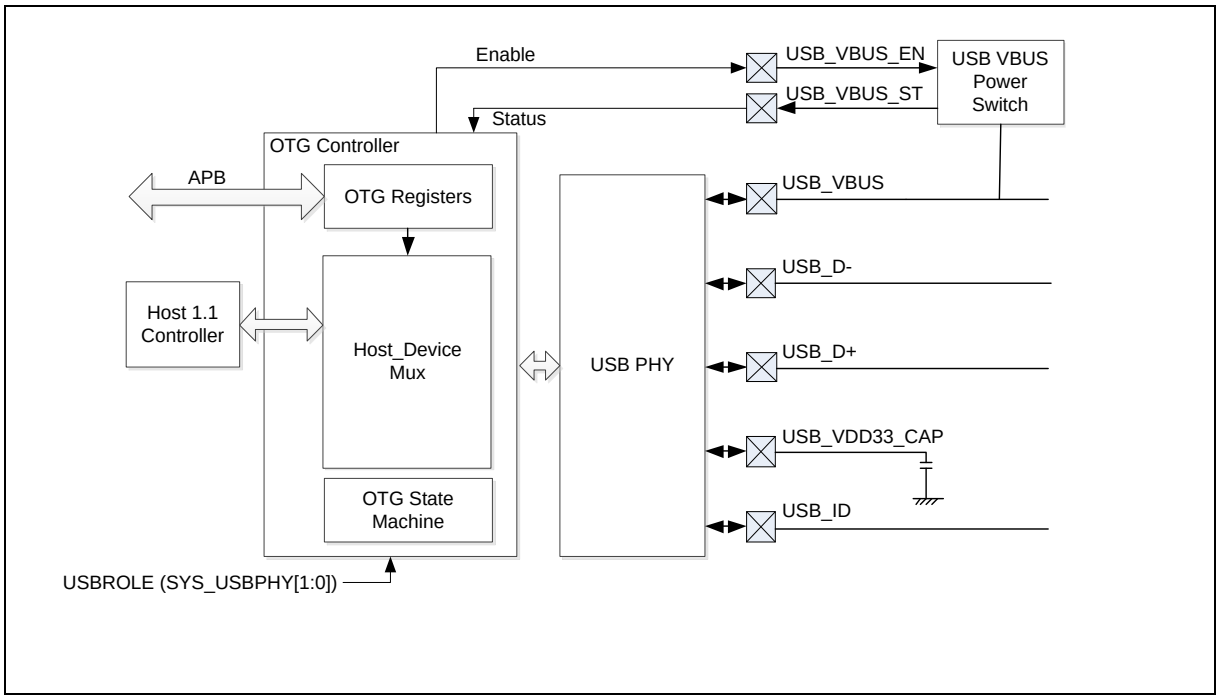


Figure 6.34-3 USB Host Mode

ID Dependent Mode

When USBROLE (SYS_USBPHY[1:0]) is set to 2, the role of USB frame depends on USB_ID pin status. The ID detection function can be enabled by set IDDETEN (OTG_PHYCTL[1]) to 1. The USB_ID pin status reflects on IDSTS (OTG_STATUS[1]). When USB frame acts as USB host (USB_ID pin is low level), the function is the same as USB Host mode. When USB frame acts as USB device (USB_ID pin is high level), the function is the same as USB Device mode.

OTG Device Mode

When USBROLE (SYS_USBPHY[1:0]) is set to 3, the role of USB frame depends on USB_ID pin status. The ID detection function can be enabled by set IDDETEN (OTG_PHYCTL[1]) to 1. The USB_ID pin status reflects on IDSTS (OTG_STATUS[1]). When USB_ID pin status is low level, the OTG controller acts as OTG A-device. When USB_ID pin status is high level, the OTG controller acts as OTG B-device. Please refer to OTG specification to get detail behavior of A-device and B-device.

6.34.5.2 Session Request Protocol (SRP)

When the USB frame is configured as OTG device mode, OTG controller supports SRP to conserve power. Refer to OTG specification for details of SRP.

A-Device Session Request Protocol

1. A-device turns off USB bus power to conserve power. B-device recognizes such condition by checking VBUS status.
2. B-device requests A-device to supply USB bus power by data line pulsing when B-device wants to connect to A-device.
3. A-device recognizes USB bus power request through checking SRPDETIF (OTG_INTSTS[13]).
4. A-device starts to drive VBUS by setting BUSREQ (OTG_CTL[1]) to 1 once SRPDETIF (OTG_INTSTS[13]) is set to 1 by hardware. If VBUS reaches valid level in specific time interval and B-device is connected, A-device will become USB host, HOSTIF (OTG_INTSTS[7]) will be set to 1. If VBUS cannot reach valid level in specific time interval, it means overcurrent condition occurs. Then VBUS error bit, VBEIF (OTG_INTSTS[1]), will be set to 1.

B-device Session Request Protocol

1. A-device turns off USB bus power to conserve power. B-device recognizes such condition by checking VBUSVLD (OTG_STATUS[5]).
2. B-device can request A-device to supply USB bus power by setting BUSREQ(OTG_CTL[1]) to 1.
3. B-device will generate data line pulsing as defined in OTG specification.
4. A-device will start to drive VBUS after detecting data line pulsing and B-device can recognize such condition by checking VBUSVLD (OTG_STATUS[5]). If A-device drives VBUS to valid level in specific time interval, B-device becomes USB peripheral, PDEVIF (OTG_INTSTS[6]) will be set to 1. If A-device does not drive VBUS to valid level in specific time interval, SRP failure flag, SRPFIF (OTG_INTSTS[2]), will be set to 1 and B-device will go to idle state defined in OTG specification.

6.34.5.3 Host Negotiation Protocol (HNP)

When the USB frame is configured as OTG device mode, the host function can be transferred between two directly connected OTG device without changing the cable connection. Refer to OTG specification for details of HNP.

A-device Host Negotiation Protocol

1. A-Host defined in OTG specification sends SetFeature b_hnp_enable command to enable B-device HNP capability. B-device responses ACK to indicate B-device supports HNP. User

needs to set HNPREQEN (OTG_CTL[2]) to 1 to enable HNP protocol.

2. A-Host goes to a_suspend state by setting BUSREQ (OTG_CTL[1]) to 0 and put USB bus into J-state (USB_D+ high and USB_D- low) when A-Host has finished all desired operations.
3. A-Host becomes A-Peripheral if A-Host detects B-peripheral disconnected by checking USB_D+ and USB_D- low in specific time interval. If A-Host cannot detect B-Peripheral disconnected in specific time interval, A-Host will back to idle state.

B-device Host Negotiation Protocol

1. After B-Peripheral receives SetFeature b_hnp_enable command successfully, user enables B-peripheral HNP function by setting HNPREQEN (OTG_CTL[2]) to 1.
2. User sets BUSREQ (OTG_CTL[1]) to 1 after detecting USB bus in J-state (USB_D+ high and USB_D- low). Then USB_D+ pull high resistor will be removed to cause USB disconnect state (USB_D+ low and USB_D- low).
3. If B-device detects A-device is connected (USB_D+ high) in specific time interval, B-device will become B-Host. If B-device cannot detect A-device is connected (USB_D+ high) in specific time interval, HNP failure flag, HNPFIF (OTG_INTSTS[3]), will be set to 1.

6.34.6 Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
OTG Base Address: **OTG_BA = 0x4004_D000** **OTG non-secure base address is OTG_BA + 0x1000_0000.**				
OTG_CTL	OTG_BA+0x00	R/W	OTG Control Register	0x0000_0000
OTG_PHYCTL	OTG_BA+0x04	R/W	OTG PHY Control Register	0x0000_0000
OTG_INTEN	OTG_BA+0x08	R/W	OTG Interrupt Enable Register	0x0000_0000
OTG_INTSTS	OTG_BA+0x0C	R/W	OTG Interrupt Status Register	0x0000_0000
OTG_STATUS	OTG_BA+0x10	R	OTG Status Register	0x0000_0006

6.34.7 Register Description

OTG Control Register (OTG_CTL)

Register	Offset	R/W	Description	Reset Value
OTG_CTL	OTG_BA+0x00	R/W	OTG Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		WKEN	OTGEN	Reserved	HNPREQEN	BUSREQ	VBUSDROP

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**WKEN**	**OTG ID Pin Wake-up Enable Bit** 0 = OTG ID pin status change wake-up function Disabled. 1 = OTG ID pin status change wake-up function Enabled.
[4]	**OTGEN**	**OTG Function Enable Bit** User needs to set this bit to enable OTG function while USB frame configured as OTG device. When USB frame is not configured as OTG device, this bit must be low. 0 = OTG function Disabled. 1 = OTG function Enabled.
[3]	**Reserved**	Reserved.
[2]	**HNPREQEN**	**OTG HNP Request Enable Bit** When USB frame as A-device, set this bit when A-device allows to process HNP protocol—A-device changes role from Host to Peripheral. This bit will be cleared when OTG state changes from a_suspend to a_peripheral or goes back to a_idle state. When USB frame as B-device, set this bit after the OTG A-device successfully sends a SetFeature (b_hnp_enable) command to the OTG B-device to start role change—B-device changes role from Peripheral to Host. This bit will be cleared when OTG state changes from b_peripheral to b_wait_acon or goes back to b_idle state. 0 = HNP request Disabled. 1 = HNP request Enabled (A-device can change role from Host to Peripheral or B-device can change role from Peripheral to Host). **Note:** Refer to OTG specification to get a_suspend, a_peripheral, a_idle and b_idle state.
[1]	**BUSREQ**	**OTG Bus Request** If OTG A-device wants to do data transfers via USB bus, setting this bit will drive VBUS high to detect USB device connection. If user won't use the bus any more, clearing this bit will drop VBUS to save power. This bit will be cleared when A-device goes to A_wait_vfall state. This bit will be also cleared if VBUSDROP (OTG_CTL[0]) bit is set or IDSTS (OTG_STATUS[1]) changed.

Bits	Description	
		If user of an OTG-B Device wants to request VBUS, setting this bit will run SRP protocol. This bit will be cleared if SRP failure (OTG A-device does not provide VBUS after B-device issues SRP in specified interval, defined in OTG specification). This bit will be also cleared if VBUSDROP (OTG_CTL[0]) bit is set or IDSTS (OTG_STATUS[1]) changed. 0 = Not launch VBUS in OTG A-device or not request SRP in OTG B-device. 1 = Launch VBUS in OTG A-device or request SRP in OTG B-device.
[0]	**VBUSDROP**	**Drop VBUS Control** If user application running on this OTG A-device wants to conserve power, set this bit to drop VBUS. BUSREQ (OTG_CTL[1]) will be also cleared no matter A-device or B-device. 0 = Not drop the VBUS. 1 = Drop the VBUS.

OTG PHY Control Register (OTG_PHYCTL)

Register	Offset	R/W	Description	Reset Value
OTG_PHYCTL	OTG_BA+0x04	R/W	OTG PHY Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		VBSTSPOL	VBENPOL	Reserved		IDDETEN	OTGPHYEN

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**VBSTSPOL**	**Off-chip USB VBUS Power Switch Status Polarity** The polarity of off-chip USB VBUS power switch valid signal depends on the selected component. A USB_VBUS_ST pin is used to monitor the valid signal of the off-chip USB VBUS power switch. Set this bit as following according to the polarity of off-chip USB VBUS power switch. 0 = The polarity of off-chip USB VBUS power switch valid status is high. 1 = The polarity of off-chip USB VBUS power switch valid status is low.
[4]	**VBENPOL**	**Off-chip USB VBUS Power Switch Enable Polarity** The OTG controller will enable off-chip USB VBUS power switch to provide VBUS power when need. A USB_VBUS_EN pin is used to control the off-chip USB VBUS power switch. The polarity of enabling off-chip USB VBUS power switch (high active or low active) depends on the selected component. Set this bit as following according to the polarity of off-chip USB VBUS power switch. 0 = The off-chip USB VBUS power switch enable is active high. 1 = The off-chip USB VBUS power switch enable is active low.
[3:2]	**Reserved**	Reserved.
[1]	**IDDETEN**	**ID Detection Enable Bit** 0 = Detect ID pin status Disabled. 1 = Detect ID pin status Enabled.
[0]	**OTGPHYEN**	**OTG PHY Enable Bit** When USB frame is configured as either OTG device or ID dependent, user needs to set this bit before using OTG function. If device is configured as neither OTG device nor ID dependent, this bit is “don’t care”. 0 = OTG PHY Disabled. 1 = OTG PHY Enabled.

OTG Interrupt Enable Register (OTG_INTEN)

Register	Offset	R/W	Description	Reset Value
OTG_INTEN	OTG_BA+0x08	R/W	OTG Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		SRPDETIEN	Reserved	SECHGIEN	VBCHGIEN	AVLDCHGIEN	BVLDCHGIEN
7	6	5	4	3	2	1	0
HOSTIEN	PDEVIEN	IDCHGIEN	GOIDLEIEN	HNPFIEN	SRPFIEN	VBEIEN	ROLECHGIEN

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**SRPDETIEN**	**SRP Detected Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[12]	**Reserved**	Reserved.
[11]	**SECHGIEN**	**SESSEND Status Changed Interrupt Enable Bit** If this bit is set to 1 and SESSEND (OTG_STATUS[2]) status is changed from high to low or from low to high, an interrupt will be asserted. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[10]	**VBCHGIEN**	**VBUSVLD Status Changed Interrupt Enable Bit** If this bit is set to 1 and VBUSVLD (OTG_STATUS[5]) status is changed from high to low or from low to high, an interrupt will be asserted. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[9]	**AVLDCHGIEN**	**A-Device Session Valid Status Changed Interrupt Enable Bit** If this bit is set to 1 and AVLD (OTG_STATUS[4]) status is changed from high to low or from low to high, an interrupt will be asserted. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[8]	**BVLDCHGIEN**	**B-device Session Valid Status Changed Interrupt Enable Bit** If this bit is set to 1 and BVLD (OTG_STATUS[3]) status is changed from high to low or from low to high, an interrupt will be asserted. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[7]	**HOSTIEN**	**Act As Host Interrupt Enable Bit**

Bits	Description	
		If this bit is set to 1 and the device is changed as a host, an interrupt will be asserted. 0 = This device as a host interrupt Disabled. 1 = This device as a host interrupt Enabled.
[6]	**PDEVIEN**	**Act As Peripheral Interrupt Enable Bit** If this bit is set to 1 and the device is changed as a peripheral, an interrupt will be asserted. 0 = This device as a peripheral interrupt Disabled. 1 = This device as a peripheral interrupt Enabled.
[5]	**IDCHGIEN**	**IDSTS Changed Interrupt Enable Bit** If this bit is set to 1 and IDSTS (OTG_STATUS[1]) status is changed from high to low or from low to high, an interrupt will be asserted. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[4]	**GOIDLEIEN**	**OTG Device Goes to IDLE State Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note:** Going to idle state means going to a_idle or b_idle state. Please refer to A-device state diagram and B-device state diagram in OTG specification.
[3]	**HNPFIEN**	**HNP Fail Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[2]	**SRPFIEN**	**SRP Fail Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[1]	**VBEIEN**	**VBUS Error Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note:** VBUS error means going to a_vbus_err state. Please refer to A-device state diagram in OTG specification.
[0]	**ROLECHGIEN**	**Role (Host or Peripheral) Changed Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled.

OTG Interrupt Status Register (OTG_INTSTS)

Register	Offset	R/W	Description	Reset Value
OTG_INTSTS	OTG_BA+0x0C	R/W	OTG Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		SRPDETIF	Reserved	SECHGIF	VBCHGIF	AVLDCHGIF	BVLDCHGIF
7	6	5	4	3	2	1	0
HOSTIF	PDEVIF	IDCHGIF	GOIDLEIF	HNPFIF	SRPFIF	VBEIF	ROLECHGIF

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**SRPDETIF**	**SRP Detected Interrupt Status** 0 = SRP not detected. 1 = SRP detected. **Note:** Write 1 to clear this status.
[12]	**Reserved**	Reserved.
[11]	**SECHGIF**	**SESSEND State Change Interrupt Status** 0 = SESSEND (OTG_STATUS[2]) not toggled. 1 = SESSEND (OTG_STATUS[2]) from high to low or from low to high. **Note:** Write 1 to clear this flag.
[10]	**VBCHGIF**	**VBUSVLD State Change Interrupt Status** 0 = VBUSVLD (OTG_STATUS[5]) not toggled. 1 = VBUSVLD (OTG_STATUS[5]) from high to low or from low to high. **Note:** Write 1 to clear this status.
[9]	**AVLDCHGIF**	**A-Device Session Valid State Change Interrupt Status** 0 = AVLD (OTG_STATUS[4]) not toggled. 1 = AVLD (OTG_STATUS[4]) from high to low or low to high. **Note:** Write 1 to clear this status.
[8]	**BVLDCHGIF**	**B-device Session Valid State Change Interrupt Status** 0 = BVLD (OTG_STATUS[3]) not toggled. 1 = BVLD (OTG_STATUS[3]) from high to low or low to high. **Note:** Write 1 to clear this status.
[7]	**HOSTIF**	**Act As Host Interrupt Status** 0 = This device does not act as a host. 1 = This device acts as a host.

Bits	Description	
		Note: Write 1 to clear this flag.
[6]	**PDEVIF**	**Act As Peripheral Interrupt Status** 0 = This device does not act as a peripheral. 1 = This device acts as a peripheral. **Note:** Write 1 to clear this flag.
[5]	**IDCHGIF**	**ID State Change Interrupt Status** 0 = IDSTS (OTG_STATUS[1]) not toggled. 1 = IDSTS (OTG_STATUS[1]) from high to low or from low to high. **Note:** Write 1 to clear this flag.
[4]	**GOIDLEIF**	**OTG Device Goes to IDLE Interrupt Status** Flag is set if the OTG device transfers from non-idle state to idle state. The OTG device will be neither a host nor a peripheral. 0 = OTG device does not go back to idle state (a_idle or b_idle). 1 = OTG device goes back to idle state (a_idle or b_idle). **Note 1:** Going to idle state means going to a_idle or b_idle state. Please refer to OTG specification. **Note 2:** Write 1 to clear this flag.
[3]	**HNPFIF**	**HNP Fail Interrupt Status** When A-device has granted B-device to be host and USB bus is in SE0 (both USB_D+ and USB_D- low) state, this bit will be set when A-device does not connect after specified interval expires. 0 = A-device connects to B-device before specified interval expires. 1 = A-device does not connect to B-device before specified interval expires. **Note:** Write 1 to clear this flag.
[2]	**SRPFIF**	**SRP Fail Interrupt Status** After initiating SRP, an OTG B-device will wait for the OTG A-device to drive VBUS high at least TB_SRP_FAIL minimum, defined in OTG specification. This flag is set when the OTG B-device does not get VBUS high after this interval. 0 = OTG B-device gets VBUS high before this interval. 1 = OTG B-device does not get VBUS high before this interval. **Note:** Write 1 to clear this flag.
[1]	**VBEIF**	**VBUS Error Interrupt Status** This bit will be set when voltage on VBUS cannot reach a minimum valid threshold 4.4V within a maximum time of 100ms after OTG A-device starting to drive VBUS high. 0 = OTG A-device drives VBUS over threshold voltage before this interval expires. 1 = OTG A-device cannot drive VBUS over threshold voltage before this interval expires. **Note:** Write 1 to clear this flag and recover from the VBUS error state.
[0]	**ROLECHGIF**	**OTG Role Change Interrupt Status** This flag is set when the role of an OTG device changed from a host to a peripheral, or changed from a peripheral to a host while USB_ID pin status does not change. 0 = OTG device role not changed. 1 = OTG device role changed. **Note:** Write 1 to clear this flag.

OTG Functional Status Register (OTG_STATUS)

Register	Offset	R/W	Description	Reset Value
OTG_STATUS	OTG_BA+0x10	R	OTG Status Register	0x0000_0006

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
ASHOST	ASPERI	VBUSVLD	AVLD	BVLD	SESSEND	IDSTS	OVERCUR

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**ASHOST**	**As Host Status** When OTG acts as Host, this bit is set. 0 = OTG not as Host. 1 = OTG as Host.
[6]	**ASPERI**	**As Peripheral Status** When OTG acts as peripheral, this bit is set. 0 = OTG not as peripheral. 1 = OTG as peripheral.
[5]	**VBUSVLD**	**VBUS Valid Status** When VBUS is larger than 4.7V, this bit will be set to 1. 0 = VBUS is not valid. 1 = VBUS is valid.
[4]	**AVLD**	**A-Device Session Valid Status** 0 = A-device session is not valid. 1 = A-device session is valid.
[3]	**BVLD**	**B-device Session Valid Status** 0 = B-device session is not valid. 1 = B-device session is valid.
[2]	**SESSEND**	**Session End Status** When VBUS voltage is lower than 0.4V, this bit will be set to 1. Session end means no meaningful power on VBUS. 0 = Session is not end. 1 = Session is end.

Bits	Description	
[1]	**IDSTS**	**USB_ID Pin State of Mini-/Micro- Plug** 0 = Mini-A/Micro-A plug is attached. 1 = Mini-B/Micro-B plug is attached.
[0]	**OVERCUR**	**Overcurrent Condition** The voltage on VBUS cannot reach a minimum VBUS valid threshold, 4.4V minimum, within a maximum time of 100ms after OTG A-device drives VBUS high. 0 = OTG A-device drives VBUS successfully. 1 = OTG A-device cannot drives VBUS high in this interval.

6.35 CRC Controller (CRC)

6.35.1 Overview

The Cyclic Redundancy Check (CRC) generator can perform CRC calculation with four common polynomials CRC-CCITT, CRC-8, CRC-16, and CRC-32 settings.

6.35.2 Features

- Supports four common polynomials CRC-CCITT, CRC-8, CRC-16, and CRC-32
  - CRC-CCITT: $X^{16} + X^{12} + X^5 + 1$
  - CRC-8: $X^8 + X^2 + X + 1$
  - CRC-16: $X^{16} + X^{15} + X^2 + 1$
  - CRC-32: $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Programmable seed value
- Supports programmable order reverse setting for input data and CRC checksum
- Supports programmable 1's complement setting for input data and CRC checksum
- Supports 8/16/32-bit of data width
  - 8-bit write mode: 1-AHB clock cycle operation
  - 16-bit write mode: 2-AHB clock cycle operation
  - 32-bit write mode: 4-AHB clock cycle operation
- Supports using PDMA to write data to perform CRC operation

6.35.3 Block Diagram

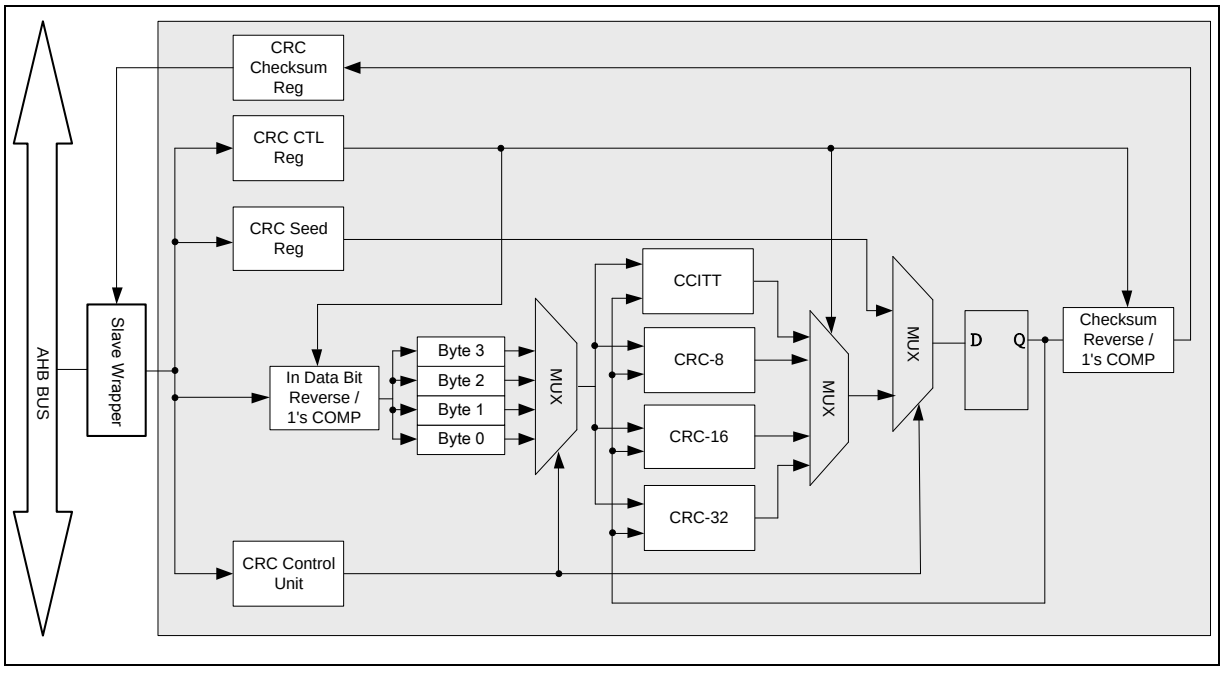


Figure 6.35-1 CRC Generator Block Diagram

6.35.4 Basic Configuration

- Clock Source Configuration
  - Enable CRC peripheral clock in CRCCKEN (CLK_AHBCLK[7]).
- Reset Configuration
  - Reset CRC controller in CRCRST (SYS_IPRST0[7]).

6.35.5 Functional Description

CRC generator can perform CRC calculation with four common polynomial settings. The operation polynomial includes CRC-CCITT, CRC-8, CRC-16 and CRC-32; User can choose the CRC operation polynomial mode by setting CRCMODE[1:0] (CRC_CTL[31:30] CRC Polynomial Mode).

The following is a program sequence example.

1. Enable CRC generator by setting CRCEN (CRC_CTL[0] CRC Channel Enable Bit).
2. Initial setting for CRC calculation.
   1) Configure 1's complement for CRC checksum by setting CHKSFMT (CRC_CTL[27] Checksum 1's Complement).
   2) Configure bit order reverse for CRC checksum by setting CHKSREV (CRC_CTL[25] Checksum Bit Order Reverse). The funcitonal block is also shown in Figure 6.35-2 CHECKSUM Bit Order Reverse Functional Block
   3) Configure 1's complement for CRC write data by setting DATFMT (CRC_CTL[26] Write Data 1's Complement).
   4) Configure bit order reverse for CRC write data per byte by setting DATREV (CRC_CTL[24] Write Data Bit Order Reverse). The functional block is also shown in Figure 6.35-3.
3. Perform CHKSINIT (CRC_CTL[1] Checksum Initialization) to load the initial checksum value from CRC_SEED register value.
4. Write data to CRC_DAT register to calculate CRC checksum.
5. Get the CRC checksum result by reading CRC_CHECKSUM register.

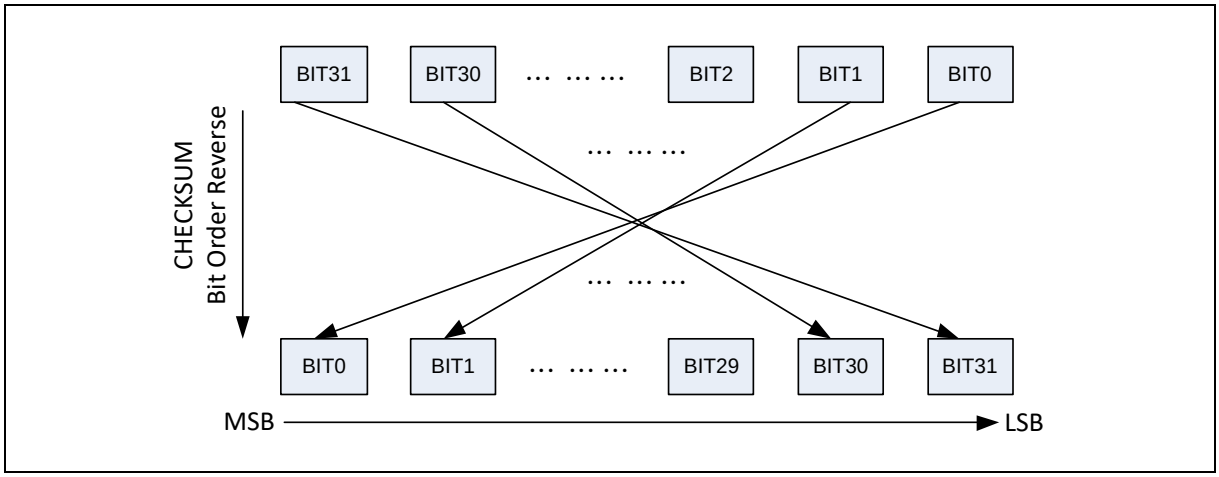


Figure 6.35-2 CHECKSUM Bit Order Reverse Functional Block

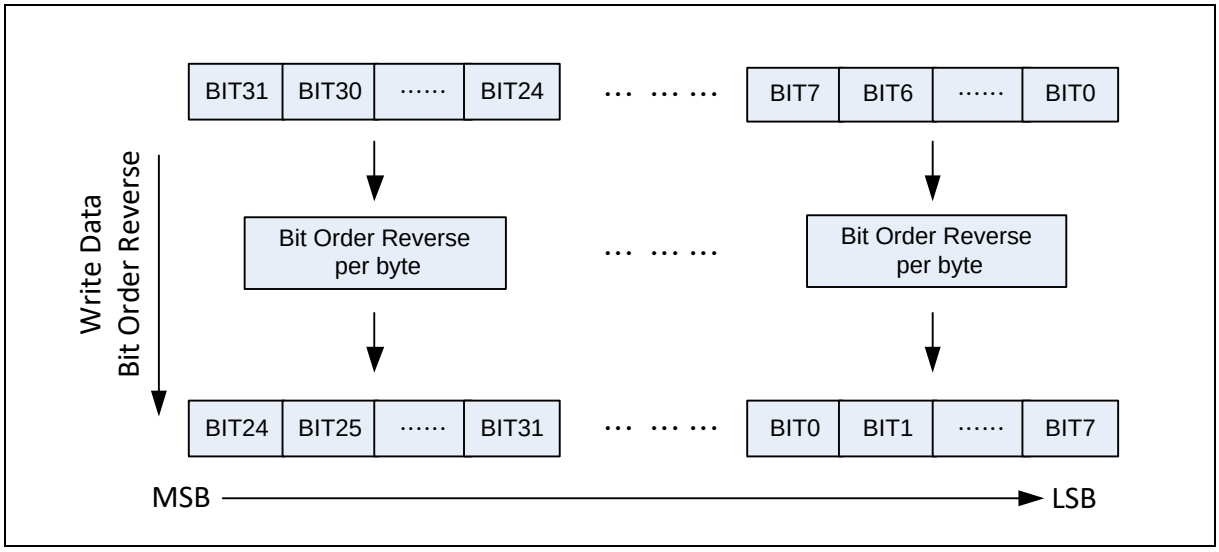


Figure 6.35-3 Write Data Bit Order Reverse Functional Block

6.35.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
CRC Base Address: **CRC_BA = 0x4003_1000** **CRC non-secure base address is CRC_BA+0x1000_0000**				
CRC_CTL	CRC_BA+0x00	R/W	CRC Control Register	0x2000_0000
CRC_DAT	CRC_BA+0x04	R/W	CRC Write Data Register	0x0000_0000
CRC_SEED	CRC_BA+0x08	R/W	CRC Seed Register	0xFFFF_FFFF
CRC_CHECKSUM	CRC_BA+0x0C	R	CRC Checksum Register	0xFFFF_FFFF

6.35.7 Register Description

CRC Control Register (CRC_CTL)

Register	Offset	R/W	Description	Reset Value
CRC_CTL	CRC_BA+0x00	R/W	CRC Control Register	0x2000_0000

31	30	29	28	27	26	25	24
CRCMODE		DATLEN		CHKSFMT	DATFMT	CHKSREV	DATREV
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						CHKSINIT	CRCEN

Bits	Description	
[31:30]	**CRCMODE**	**CRC Polynomial Mode** This field indicates the CRC operation polynomial mode. 00 = CRC-CCITT Polynomial mode. 01 = CRC-8 Polynomial mode. 10 = CRC-16 Polynomial mode. 11 = CRC-32 Polynomial mode.
[29:28]	**DATLEN**	**CPU Write Data Length** This field indicates the write data length. 00 = Data length is 8-bit mode. 01 = Data length is 16-bit mode. 1x = Data length is 32-bit mode. **Note:** When the write data length is 8-bit mode, the valid data in CRC_DAT register is only DATA[7:0] bits; if the write data length is 16-bit mode, the valid data in CRC_DAT register is only DATA[15:0].
[27]	**CHKSFMT**	**Checksum 1's Complement** This bit is used to enable the 1's complement function for checksum result in CRC_CHECKSUM register. 0 = 1's complement for CRC checksum Disabled. 1 = 1's complement for CRC checksum Enabled.
[26]	**DATFMT**	**Write Data 1's Complement** This bit is used to enable the 1's complement function for write data value in CRC_DAT register. 0 = 1's complement for CRC writes data in Disabled. 1 = 1's complement for CRC writes data in Enabled.

[25]	**CHKSREV**	**Checksum Bit Order Reverse** This bit is used to enable the bit order reverse function for checksum result in CRC_CHECKSUM register. 0 = Bit order reverse for CRC checksum Disabled. 1 = Bit order reverse for CRC checksum Enabled. **Note:** If the checksum result is 0xDD7B0F2E, the bit order reverse for CRC checksum is 0x74F0DEBB.
[24]	**DATREV**	**Write Data Bit Order Reverse** This bit is used to enable the bit order reverse function per byte for write data value in CRC_DAT register. 0 = Bit order reversed for CRC write data in Disabled. 1 = Bit order reversed for CRC write data in Enabled (per byte). **Note:** If the write data is 0xAABBCCDD, the bit order reverse for CRC write data in is 0x55DD33BB.
[23:2]	**Reserved**	Reserved.
[1]	**CHKSINIT**	**Checksum Initialization** 0 = No effect. 1 = Initial checksum value by auto reload CRC_SEED register value to CRC_CHECKSUM register value. **Note:** This bit will be cleared automatically.
[0]	**CRCEN**	**CRC Channel Enable Bit** 0 = No effect. 1 = CRC operation Enabled.

CRC Write Data Register (CRC_DAT)

Register	Offset	R/W	Description	Reset Value
CRC_DAT	CRC_BA+0x04	R/W	CRC Write Data Register	0x0000_0000

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

Bits	Description	
[31:0]	DATA	**CRC Write Data Bits** User can write data directly by CPU mode or use PDMA function to write data to this field to perform CRC operation. **Note:** When the write data length is 8-bit mode, the valid data in CRC_DAT register is only DATA[7:0] bits; if the write data length is 16-bit mode, the valid data in CRC_DAT register is only DATA[15:0].

CRC Seed Register (CRC_SEED)

Register	Offset	R/W	Description	Reset Value
CRC_SEED	CRC_BA+0x08	R/W	CRC Seed Register	0xFFFF_FFFF

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

Bits	Description	
[31:0]	SEED	**CRC Seed Value** This field indicates the CRC seed value. **Note:** This field will be reloaded as checksum initial value (CRC_CHECKSUM register) after perform CHKSINIT (CRC_CTL[1]).

CRC Checksum Register (CRC_CHECKSUM)

Register	Offset	R/W	Description	Reset Value
CRC_CHECKSUM	CRC_BA+0x0C	R	CRC Checksum Register	0xFFFF_FFFF

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

Bits	Description	
[31:0]	CHECKSUM	**CRC Checksum Results** This field indicates the CRC checksum result. **Note:** Data in CRC_CHECKSUM register has different length when user chooses different operation polynomial modes. For example: If final checksum result is 0x12 in CRC-8 polynomial mode, the CHECKSUM[31:0] value will be read as 0x12121212, only CHECKSUM[7:0] is valid in this mode. If final checksum result is 0x1234 in CRC-CCITT or CRC-16 mode, the CHECKSUM[31:0] value will be read as 0x12341234, only CHECKSUM[15:0] is valid in this mode. And the CHECKSUM[31:0] is valid for CRC-32 mode.

6.36 Cryptographic Accelerator (CRYPTO)

6.36.1 Overview

The Crypto (Cryptographic Accelerator) includes a secure pseudo random number generator (PRNG) core and supports AES, SHA/HMAC, RSA, and ECC algorithms.

The PRNG core supports 128, 163, 192, 224, 233, 255, 256, 283, 384, 409, 512, 521 and 571 bits random number generation. (283~571 bits are only generated for Key Store.)

The AES accelerator is an implementation fully compliant with the AES (Advance Encryption Standard) encryption and decryption algorithm. The AES accelerator supports ECB, CBC, CFB, OFB, CTR, CBC-CS1, CBC-CS2, CBC-CS3, CCM and GCM mode.

The SHA accelerator is an implementation fully compliant with the SM3, SHA-160, SHA-224, SHA-256, SHA-384, SHA-512 and corresponding HMAC (Keyed-Hash Message Authentication Code) algorithms.

The ECC accelerator is an implementation fully compliant with elliptic curve cryptography by using polynomial basis in binary field and prime filed.

The RSA accelerator is an implementation fully compliant with RSA cryptography, CRT decryption algorithm and side-channel attack countermeasures algorithm.

The Crypto can get key from key store and/or put key to key store determined by the function of each accelerator.

6.36.2 Features

- PRNG
  - Supports 128, 163, 192, 224, 233, 255, 256, 283, 384, 409, 512, 521 and 571 bits random number generation (283~571 bits only generated for Key Store)
  - Able to take the true random number seed from TRNG
- AES
  - Supports FIPS NIST 197
  - Supports SP800-38A and addendum
  - Supports 128, 192, and 256 bits key
  - Supports both encryption and decryption
  - Supports ECB, CBC, CFB, OFB, CTR, CBC-CS1, CBC-CS2 and CBC-CS3 modes
  - Supports CCM mode, GCM mode and GHASH function
  - Supports SM4 block cipher algorithm
  - Supports key expander
  - Supports one technique to improve side-channel attack protection ability
- SHA
  - Supports FIPS NIST 180, 180-2, 180-4
  - Supports SHA-160, SHA-224, SHA-256, SHA-384 and SHA-512
  - Supports SM3 Cryptographic Hash Algorithm
- ECC

- Supports both prime field GF(p) and binary filed GF(2^m)
- Supports NIST P-192, P-224, P-256, P-384, and P-521
- Supports NIST B-163, B-233, B-283, B-409, and B-571
- Supports NIST K-163, K-233, K-283, K-409, and K-571
- Supports Curve25519
- Supports Public Key Cryptographic Algorithm SM2 Based on Elliptic Curves
- Supports point multiplication, addition and doubling operations in GF(p) and GF(2^m)
- Supports modulus division, multiplication, addition and subtraction operations in GF(p)
- Supports three techniques to improve side-channel attack protection ability

- RSA
  - Supports both encryption and decryption with 1024, 2048, 3072 and 4096 bits
  - Supports CRT decryption with 2048, 3072 and 4096 bits
  - Supports three techniques to improve side-channel attack protection ability

6.36.3 Block Diagram

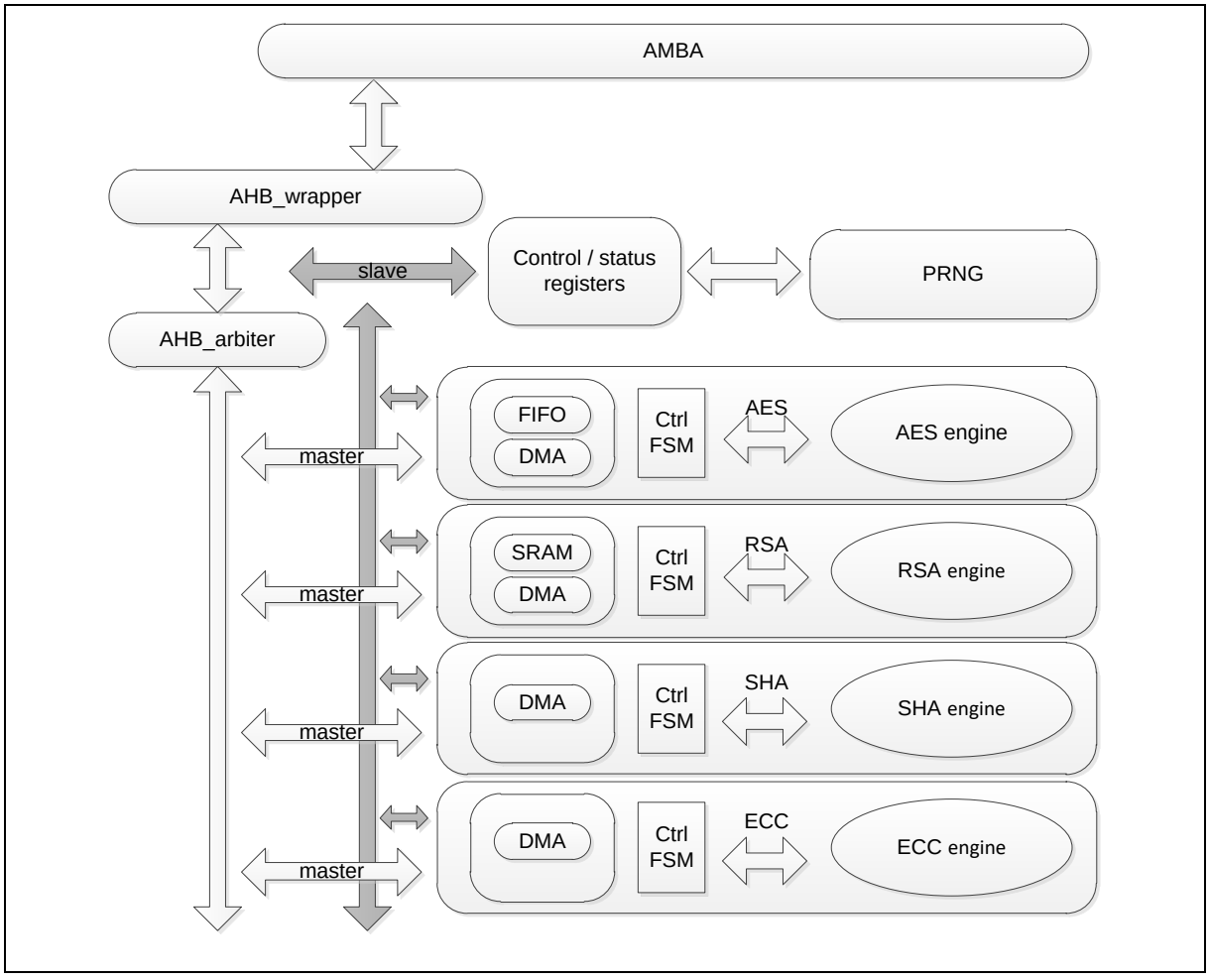


Figure 6.36-1 Cryptographic Accelerator Block Diagram

6.36.4 Basic Configuration

- Clock Source Configuration
  - Enable CRYPTO peripheral clock in CRPTCKEN(CLK_AHBCLK[12])
- Reset Configuration
  - Reset CRYPTO controller in CRPTRST(SYS_IPRST0[12])
- Power Switch Configuration
  - Enable CRYPTO power switch in CRPTPWREN(SYS_PSWCTL[12])

6.36.5 Functional Description

The cryptographic accelerator includes a secure pseudo random number generator (PRNG) core and supports AES, SHA/HMAC, RSA, and ECC algorithms. The accelerator can be used in different data security applications, such as secure communications that need cryptographic protection and integrity.

1. The PRNG core supports 128, 163, 192, 224, 233, 255, 256, 283, 384, 409, 512, 521 and 571 bits random number generation configured by KEYSZ. (283~571 bits are only generated for Key Store.)
2. The AES accelerator is a fully compliant implementation of the AES (Advance Encryption

Standard) encryption and decryption algorithm. The AES accelerator supports ECB, CBC, CFB, OFB, CTR, CBC-CS1, CBC-CS2, CBC-CS3, CCM and GCM mode. Besides, SM4 block cipher algorithm is also implemented in this accelerator The AES accelerator provides the DMA function to reduce the CPU intervention, and supports three burst lengths, sixteen-words, eight-words, and four-words.

3. The SHA/HMAC accelerator is a fully compliant implementation of the SHA1 (i.e., SHA-160), SHA2 (i.e, SHA-224, SHA-256, SHA-384, SHA-512) and corresponding HMAC algorithm. Besides, SM3 Cryptographic Hash Algorithm is also implemented in this accelerator. The SHA/HMAC accelerator also supports the DMA function to reduce the CPU intervention. It supports three burst lengths, sixteen-words, eight-words, and four-words.
4. The ECC accelerator is a fully compliant implementation of the prime field GF(p) and binary field GF(2^m) algorithm in polynomial basis. The prime field GF(p) supports NIST P-192, P-224, P-256, P-384, P-521 and Curve25519. The binary field GF(2^m) supports NIST B-163, B-233, B-283, B-409, B-571 and NIST K-163, K-233, K-283, K-409 and K-571. Besides, Public Key Cryptographic Algorithm SM2 is also implemented in this accelerator.
5. The RSA accelerator is a fully compliant implementation of RSA-1024, RSA-2048, RSA-3072 and RSA-4096. The CRT mode supports RSA-2048, RSA-3072 and RSA-4096.

Software can control the data flow by enabling the CRYPTO_INTEN, and monitor the accelerator status by checking the CRYPTO_INTSTS status register. When any engine operation error or buffer error happened, the corresponding error flag will be set to 1 and informed to CPU if error interrupt enable bit is set to 1. If users want to detail error condition, software can check status flag register of each engine.

Engine	Error Interrupt Enable Bit	Error Interrupt Flag	Error Conditions
PRNG	PRNGEIEN (CRYPTO_INTEN[17])	PRNGEIF (CRYPTO_INTSTS[17])	KSERR/KCTLERR
AES	AESEIEN (CRYPTO_INTEN[1])	AESEIF (CRYPTO_INTSTS[1])	KSERR/INBUFERR/OUTBUFERR/BUSERR
SHA/HMAC	HMACEIEN (CRYPTO_INTEN[25])	HMACEIF (CRYPTO_INTSTS[25])	KSERR/DMAERR/BUSERR
ECC	ECCEIEN (CRYPTO_INTEN[23])	ECCEIF (CRYPTO_INTSTS[23])	KSERR/BUSERR
RSA	RSAEIEN (CRYPTO_INTEN[31])	RSAEIF (CRYPTO_INTSTS[31])	KSERR/CTLERR/BUSERR

Table 6.36-1 Each Engine Error Conditions and Error Flag

Secure Mode Switch

Once the secure state of Crypto is changed, all registers that users can access will be cleared to be reset value or 0. Thus, user must make sure that all engines are idle to protect the computation of Crypto before switching the secure state of Crypto. To avoid Crypto being in the non-secure state with too long time, users can force all engines to be idle by setting the stop bit of all engines to be 1 when they are busy. Then, user can switch the secure state of Crypto safely only when the busy registers of all engines are 0.

The cryptographic accelerator supports the following features to enhance the performance.

DMA Mode

Once DMA source address register, destination address register, and byte count register are configured by CPU, moving data from and to accelerator is done by DMA logic totally. This mode can off-load the loading from the CPU. The cryptographic accelerator embeds one hardware DMA channel for AES engine, one hardware DMA channel for SHA/HMAC engine, one hardware DMA channel for RSA engine, and one hardware DMA channel for ECC engine. The RSA engine only supports DMA mode.

Engine	DMA Enable Bit
AES	DMAEN (CRYPTO_AES_CTL[7])
SHA/HMAC	DMAEN (CRYPTO_HMAC_CTL[7])
ECC	DMAEN (CRYPTO_ECC_CTL[7])

Table 6.36-2 DMA Enable Bit Table

DMA Cascade Mode

In the case that the data SRAM resource is tight, or another peripheral is scheduled to switch, the data source or sink needs an update, while the setting for the accelerator operation is planned to be kept. In this mode, software can update DMA source address register, destination address register, and byte count register during a cascade operation, without finishing the accelerator operation. Bacause AES-CCM and AES-GCM operations require storing more imformation to continue the operation, AES and SHA/HMAC provide FBOUT and FBIN bits in their control register to trigger DMA engine to automatically output and input all required information according to their DMA feedback address register (CRYPTO_AES_FBADDR or CRYPTO_HMAC_FBADDR).

Engine	DMA Cascade Bit
AES	DMACSCAD (CRYPTO_AES_CTL[6])
SHA/HMAC	DMACSCAD (CRYPTO_HMAC_CTL[6])

Table 6.36-3 DMA Cascade Bit Table

Non-DMA Mode

In the case that the input data is small in size, DMA mode is not preferred. This mode can reduce the processing time for the accelerator, since no DMA related register needs a configuration, and no latency in DMA logic is introduced. Input data is fed to cryptographic engine by writing to data input register. The RSA engine doses not support Non-DMA mode.

Channel Expansion Mode

In this mode, several virtual channels are feasible in AES mode. The intermediate data (or called feedback information) from feedback registers (CRYPTO_AES_FDBCKx) should be stored temporarily in data SRAM, and switched to another configuration setting of accelerator operation that includes operational mode, encryption/decryption, key, key size, IV, and other parameters. Once switching back, the intermediate data from feedback registers should be written to initial vectors (CRYPTO_AES_IVx) for the accelerator to continue the operation with the original configuration setting. Finally, user must configure AES control register (CRYPTO_AES_CTL) to complete accelerator operation. Note that, in ECB mode, there is no need to move the intermediate data from feedback registers to IV. Becase the feedback information of CCM and GCM mode are 18 words, above method are not suitable for CCM and GCM mode. To switch channels and continue the operation in CCM and GCM, AES provides FBOUT and FBIN bits in CRYPTO_AES_CTL to trigger AES DMA engine to automatically output and input all feedback information according to DMA feedback address register (CRYPTO_AES_FBADDR).

6.36.5.1 PRNG (Pseudo Random Number Generator)

The PRNG block diagram is depicted in Figure 6.36-2. The core supports 128, 163, 192, 224, 233, 255, 256, 283, 384, 409, 512, 521 and 571 bits random number generation configured by KEYSZ(CRYPTO_PRNG_CTL[5:2]). (283~571 bits are only generated for Key Store.)

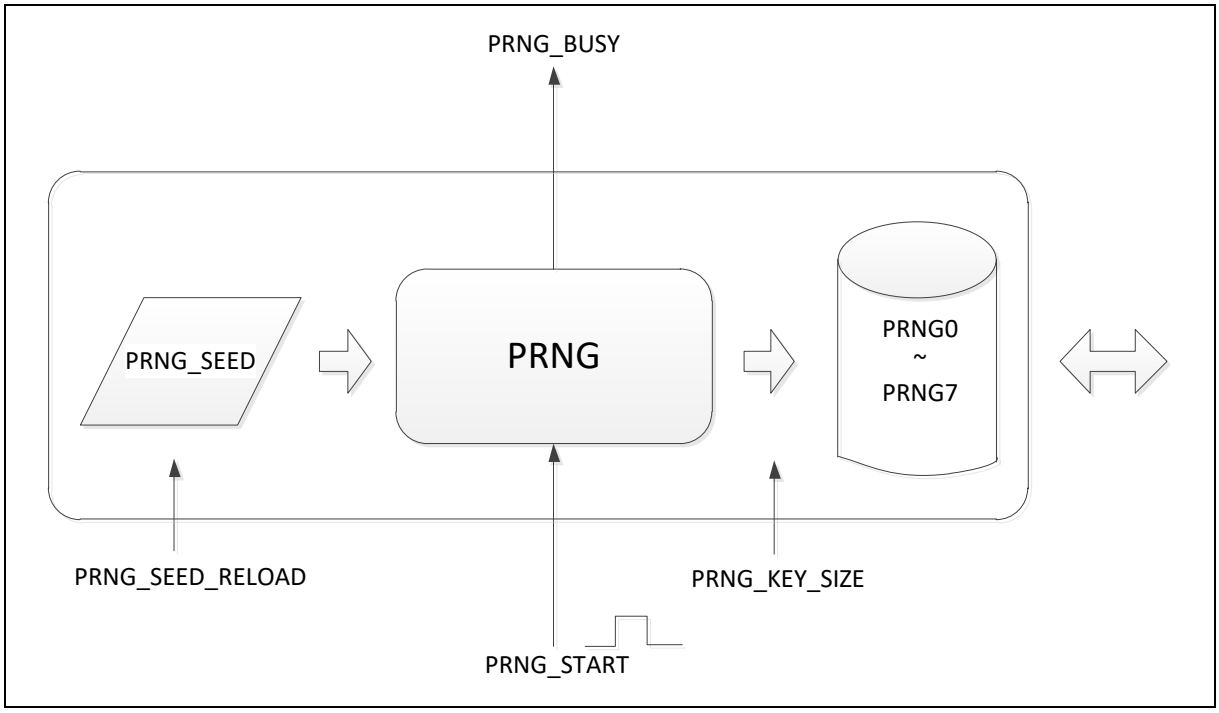


Figure 6.36-2 PRNG Function Diagram

Programming steps to get the **pseudo** random number are depicted below.

1. Check the BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0.
2. Initialize PRNG parameters. Configure KEYSZ (CRYPTO_PRNG_CTL[5:2]), and write a random seed to CRYPTO_PRNG_SEED. Note that CRYPTO_PRNG_SEED should be initialized since it is not initialized as the chip powers up.
3. Configure PRNG control register CRYPTO_PRNG_CTL for key size(KEYSZ), seed reload(SEEDRLD), and PRNG start(START).
4. Software checks BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0, or waits for the PRNGIF (CRYPTO_INTSTS[16]) (must enable PRNGIEN (CRYPTO_INTEN[16])). Then software can read the output random numbers (KEY) from CRYPTO_PRNG_KEY0 ~ CRYPTO_PRNG_KEY7.

Programming steps to get the **pseudo** random number to **key store** are depicted below.

5. Check the BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0.
6. Initialize PRNG parameters. Configure KEYSZ (CRYPTO_PRNG_CTL[5:2]), and write a random seed to CRYPTO_PRNG_SEED. Note that CRYPTO_PRNG_SEED should be initialized since it is not initialized as the chip powers up.
7. Configure PRNG key control register CRYPTO_PRNG_KSCTL for key owner (OWNER), trusted key (TRUST), key destination (DST), ECDSA selection bit (ECDSA) and ECDH selection bit (ECDH) in Key Store.
8. Configure PRNG control register CRYPTO_PRNG_CTL for key size(KEYSZ), seed reload(SEEDRLD), and PRNG start(START).
9. Software checks BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0, or waits for the PRNGIF (CRYPTO_INTSTS[16]) (must enable PRNGIEN (CRYPTO_INTEN[16])). Then software can read the output random numbers (KEY) from CRYPTO_PRNG_KEY0 ~

CRYPTO_PRNG_KEY7 or read NUMBER (CRYPTO_PRNG_KSSTS[4:0]).

Programming steps to get the random number with the true random number seed from TRNG (including TRNG generate the true random number seed steps) are depicted below. When SEEDGEN (TRNG_CTL[8]) is set to 1, users cannot read the data from TRNG Data Register.

1. Select TRNG peripheral clock frequency in CLKP (TRNG_CTL[5:2])
2. Enable ACT (TRNG_ACT[7]) bit
3. Wait for READY (TRNG_CTL[7]) bit to be set to 1
4. Enable SEEDGEN (TRNG_CTL[8]) bit
5. Wait for SEEDRDY (TRNG_CTL[9]) bit to be set to 1 to get 32 bits true random number seed
6. Check the BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0
7. Set SEEDSEL(CRYPTO_PRNG_CTL[6]) to 1.
8. Wait for SEEDSRC(CRYPTO_PRNG_CTL[7]) to be set to 1.
9. Configure PRNG control register CRYPTO_PRNG_CTL for key size(KEYSZ), seed reload(SEEDRLD), and PRNG start(START).
10. Software checks BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0, or waits for the PRNGIF (CRYPTO_INTSTS[16]) (PRNGIEN (CRYPTO_INTEN[16]) must be enabled). Then software can read the output random numbers (KEY) from CRYPTO_PRNG_KEY0 ~ CRYPTO_PRNG_KEY7.

Programming steps to get the random number with the true random number seed from TRNG (including TRNG generate the true random number seed steps) to **key store** are depicted below. When SEEDGEN (TRNG_CTL[8]) is set to 1, users cannot read the data from TRNG Data Register.

1. Select TRNG peripheral clock frequency in CLKP (TRNG_CTL[5:2])
2. Enable ACT (TRNG_ACT[7]) bit
3. Wait for READY (TRNG_CTL[7]) bit to be set to 1
4. Enable SEEDGEN (TRNG_CTL[8]) bit
5. Wait for SEEDRDY (TRNG_CTL[9]) bit to be set to 1 to get 32 bits true random number seed
6. Check the BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0
7. Set SEEDSEL(CRYPTO_PRNG_CTL[6]) to 1.
8. Wait for SEEDSRC(CRYPTO_PRNG_CTL[7]) to be set to 1.
9. Configure PRNG key control register CRYPTO_PRNG_KSCTL for key owner (OWNER), trusted key (TRUST), key destination (DST), ECDSA selection bit (ECDSA) and ECDH selection bit (ECDH) in Key Store.
10. Configure PRNG control register CRYPTO_PRNG_CTL for key size(KEYSZ), seed reload(SEEDRLD), and PRNG start(START).
11. Software checks BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0, or waits for the PRNGIF (CRYPTO_INTSTS[16]) (must enable PRNGIEN (CRYPTO_INTEN[16])). Then software can read the output random numbers (KEY) from CRYPTO_PRNG_KEY0 ~ CRYPTO_PRNG_KEY7 or read NUMBER (CRYPTO_PRNG_KSSTS[4:0]).

Some Notices of PRNG:

10. If users want to get the next (not first time) true random number seed from TRNG, they should set SEEDSEL(CRYPTO_PRNG_CTL[6]) bit to 0 before step 4 (Enable SEEDGEN (TRNG_CTL[8]) bit).

11. If users do not generate key for key store (CRYPTO_PRNG_KSCTL[21] is '0'), PRNG cannot generate more than 256 bits key (CRYPTO_PRNG_CTL[5:2]).
12. If users want to write key for ECDSA or ECDH to key store, PRNG seed must be from TRNG(CRYPTO_PRNG_CTL[7] must be '1') and key must be written to the SRAM of key store (WSDST, CRYPTO_PRNG_KSCTL[23:22] must be set to '00'). Otherwise, KCTLERR(CRYPTO_PRNG_KSSTS[16]) will become '1'.
13. If users want to write key for ECDSA or ECDH to key store, key must be in the interval [1, n-1] (the parameter n is from ECC). The value of n cannot be 0 or 1, otherwise, PRNG will always keep busy.

6.36.5.2 AES (Advanced Encryption Standard)

The AES accelerator supports not only AES algorithm from U.S NIST but also SM4 block cipher algorithm from CHINA state cryptography administration. This accelerator provides 10 block cipher modes includes ECB, CBC, CFB, OFB, CTR, CBC-CS1, CBC-CS2, CBC-CS3, CCM and GCM mode. The detail of above block cipher modes are described below.

Electronic Codebook Mode

The Electronic Codebook (ECB) mode is a confidentiality mode that features the assignment of a fixed ciphertext block to each plaintext block, for a given key. It is analogous to the assignment of code words in a codebook.

In ECB encryption, each block of the plaintext is applied to the forward cipher function $CIPH_k$ directly and independently. The resulting sequence of output blocks is the ciphertext. In ECB dencryption, each block of the ciphertext is applied to the inverse cipher function $CIPH^{-1}_k$ directly and independently. The resulting sequence of output blocks is the plaintext.

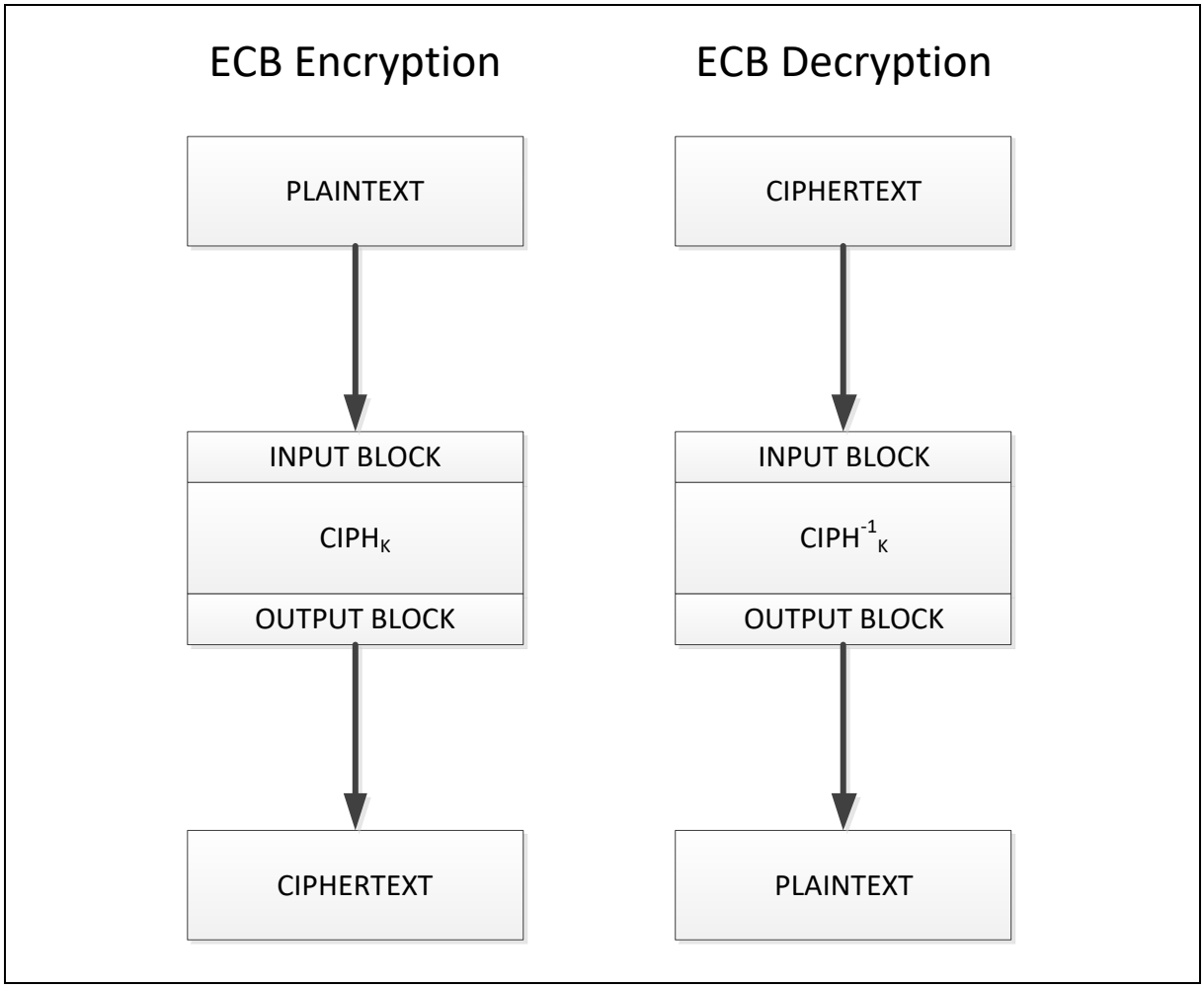


Figure 6.36-3 Electronic Codebook Mode

In ECB mode, any given plaintext block always gets encrypted to the same ciphertext block under a given key. If this property is undesirable in a particular application, the ECB mode should not be used.

Cipher Block Chaining Mode

The Cipher Block Chaining (CBC) mode is a confidentiality mode whose encryption process features the combining chaining of the plaintext blocks with the previous ciphertext blocks. The CBC mode requires an initialization vector (IV) to combine with the first plaintext block. The IV does not need to be secret, but it must be unpredictable.

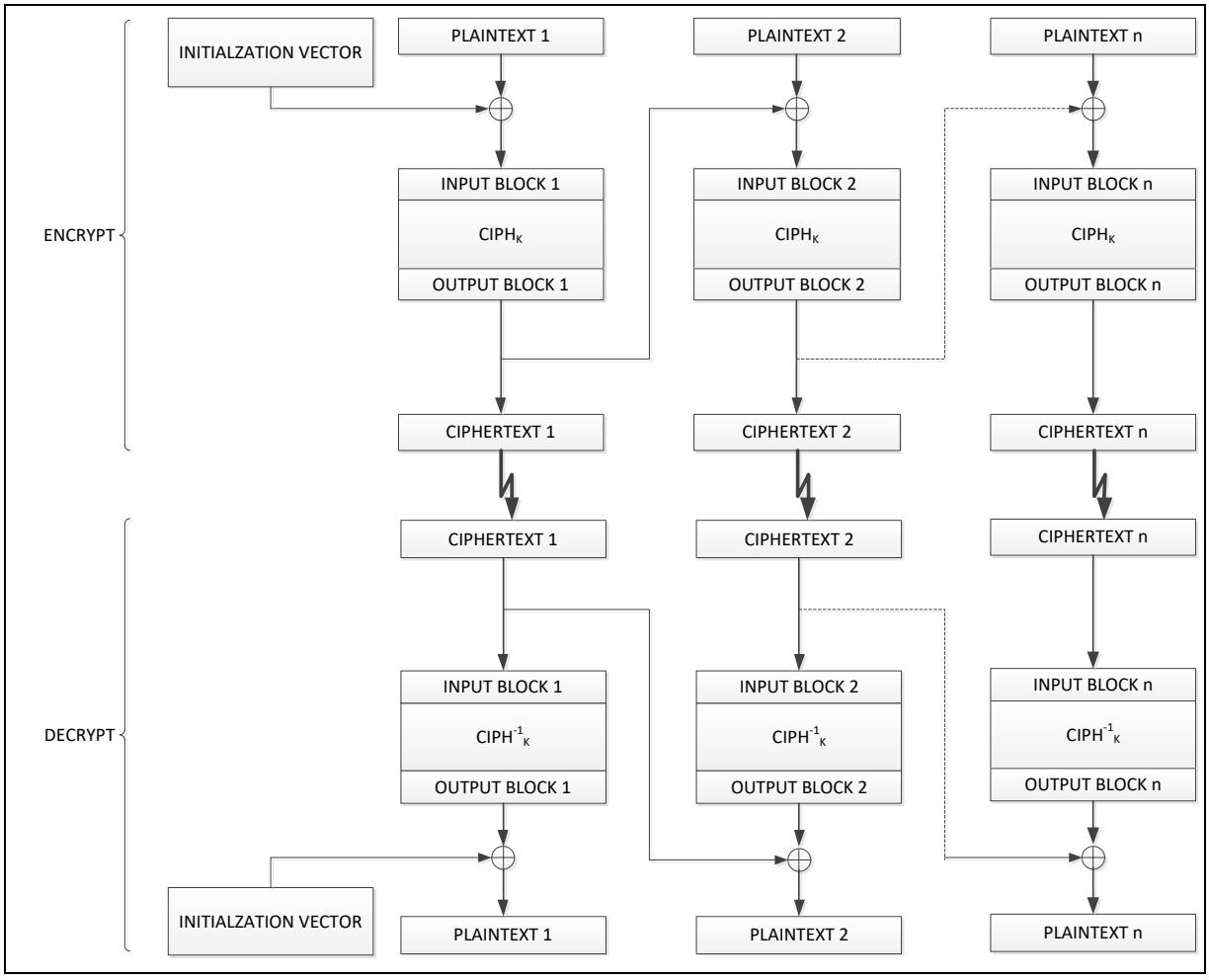


Figure 6.36-4 Cipher Block Chaining Mode

Cipher Feedback Mode (CFB)

The Cipher Feedback (CFB) mode is a confidentiality mode that features the feedback of successive ciphertext segments into the input blocks of the forward cipher to generate output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. The CFB mode requires an IV as the initial input block. The IV needs not be secret, but it must be unpredictable. The AES only supports 128-bit segment length CFB mode.

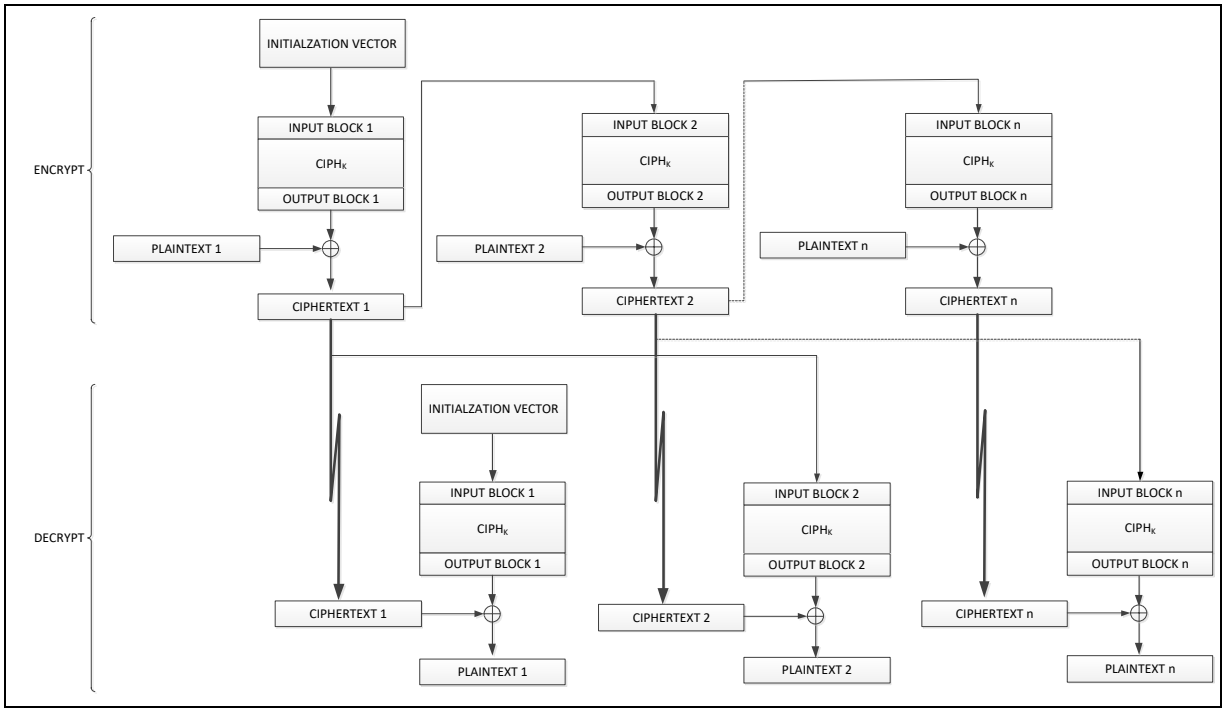


Figure 6.36-5 Cipher Feedback Mode

Output Feedback Mode

The Output Feedback (OFB) mode is a confidentiality mode that features the iteration of the forward cipher on an IV to generate a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. The OFB mode requires that the IV is a nonce, i.e., the IV must be unique for each execution of the mode under the given key.

The OFB mode requires a unique IV for every message that is ever encrypted under the given key. If, contrary to this requirement, the same IV is used for the encryption of more than one message, then the confidentiality of those messages may be compromised. Confidentiality may be similarly compromised if any of the input blocks to the forward cipher function for the encryption of a message is designated as the IV for the encryption of another message under the given key.

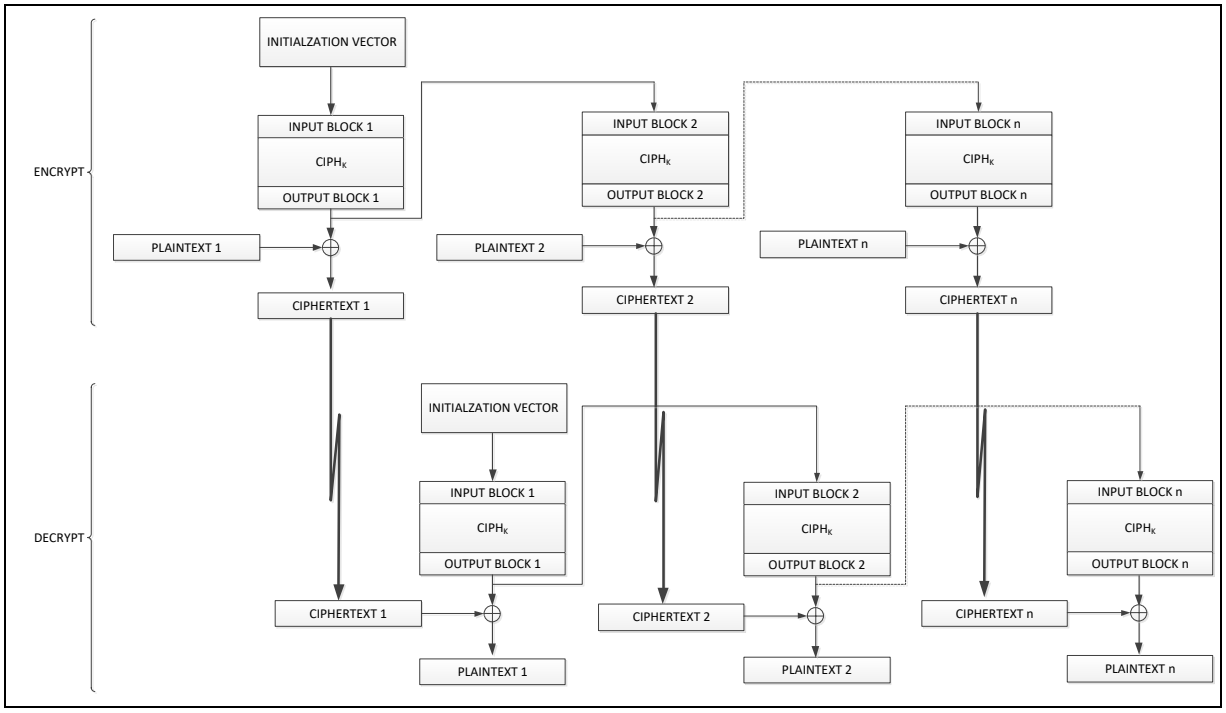


Figure 6.36-6 Output Feedback Mode

Counter Mode (CTR)

The Counter (CTR) mode is a confidentiality mode that features the application of the forward cipher to a set of input blocks, called counters, to produce a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. The sequence of counters must have the property that each block in the sequence is different from every other block. This condition is not restricted to a single message: across all of the messages that are encrypted under the given key, all of the counters must be distinct.

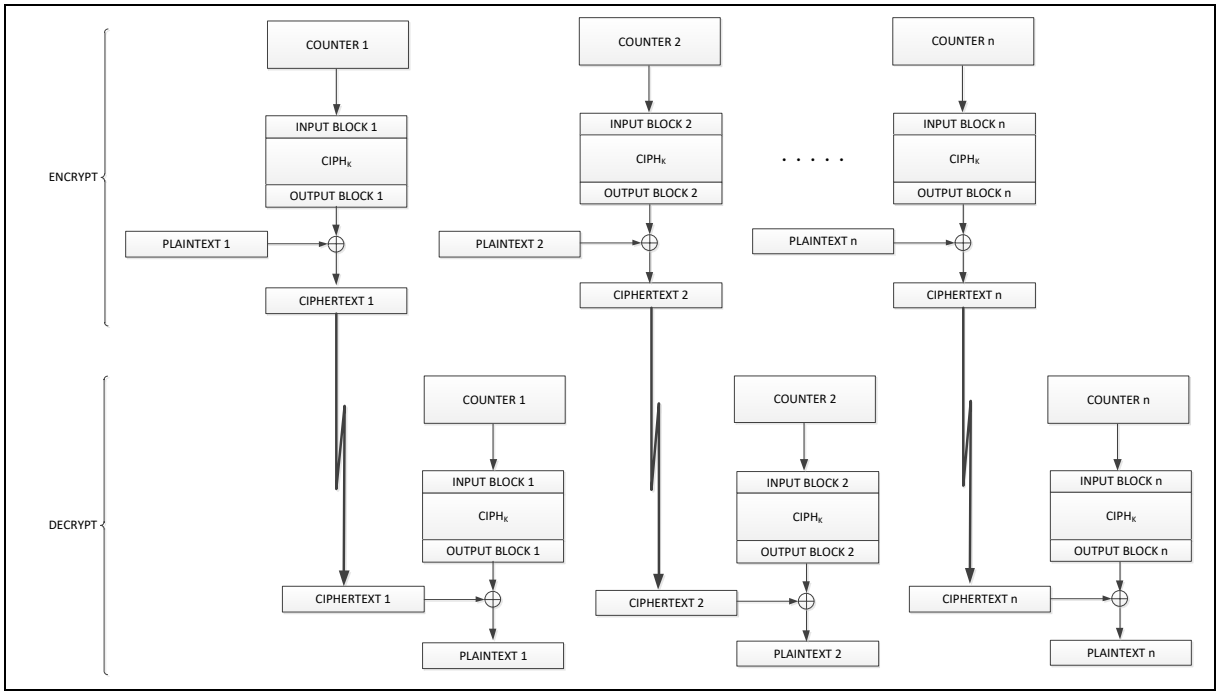


Figure 6.36-7 Counter Mode

CBC Ciphertext-Stealing 1 Mode (CBC-CS1)

Figure 6.36-8 illustrates the CBC-CS1-Encrypt algorithm for the case that P_n^* is a partial block. The cryptographic accelerator would append P_n^* with '0' to form a complete block P_n.

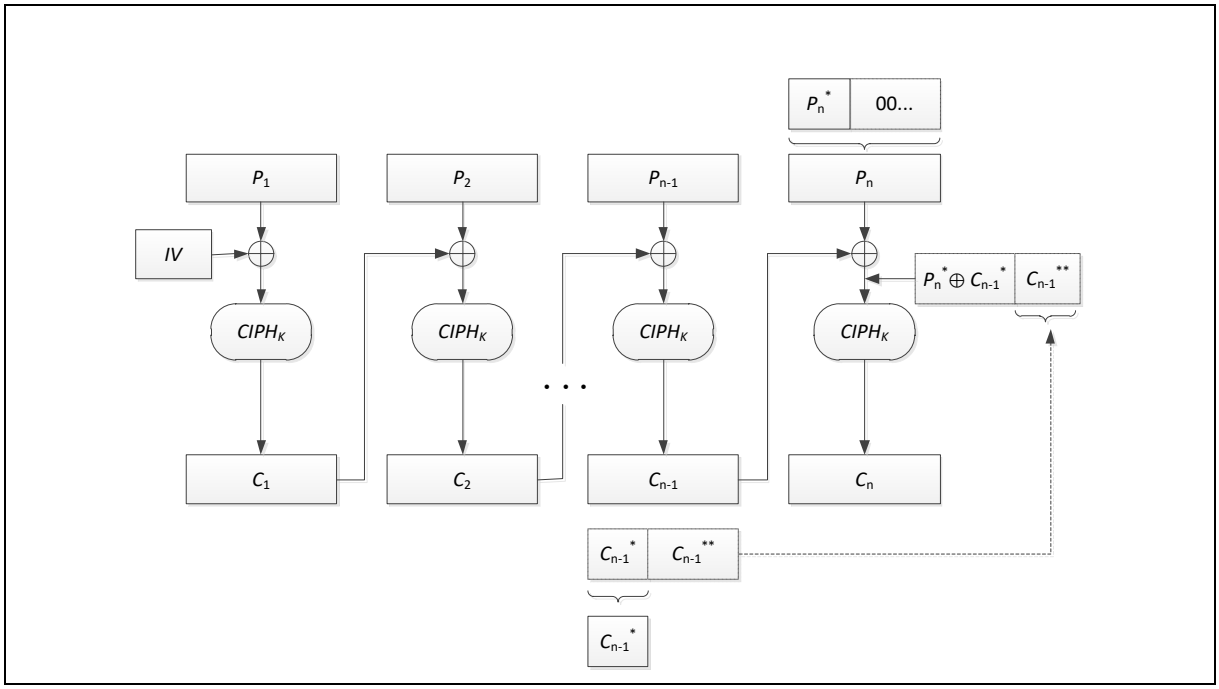


Figure 6.36-8 CBC-CS1 Encryption

Figure 6.36-9 illustrates the CBC-CS1-Decrypt algorithm for the case that C_{n-1}^* is a partial block.

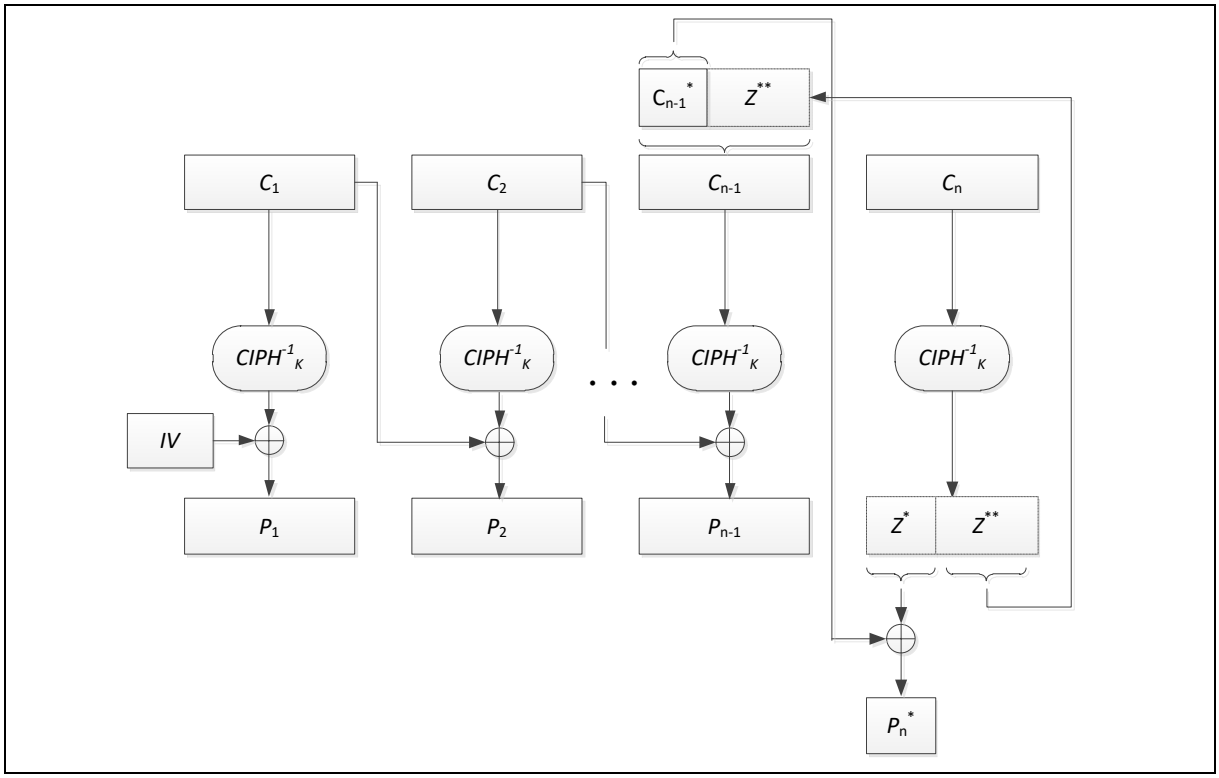


Figure 6.36-9 CBC-CS1 Decryption

CBC Ciphertext-Stealing 2 Mode (CBC-CS2)

When P_n^* is a partial block, then CBC-CS2-Encrypt and CBC-CS1-Encrypt differ only in the ordering of C_{n-1}^* and C_n.

CBC Ciphertext-Stealing 3 Mode (CBC-CS3)

C_{n-1}^* and C_n are unconditionally swapped, i.e., even when C_{n-1}^* is a complete block; therefore, CBC-CS3 is not strictly an extension of CBC mode. In the other case, i.e., when C_{n-1}^* is a nonempty partial block, CBC-CS3-Encrypt is equivalent to CBC-CS2-Encrypt.

Counter with CBC-MAC Mode (CCM)

The Counter with Cipher Block Chaining-Message Authentication Coder (CCM) mode is an algorithm for authenticated encryption/decryption with associated data. It provides the assurance of the confidentiality of data via CTR mode and the authenticity of the confidential data via CBC-MAC function. The input data of AES CCM includes plaintext/ciphertext (denoted P/C in CCM) in encryption/decryption, additional associated data (denoted A in CCM) and nonce (denoted N in CCM). The output data of GCM are C/P in encryption/decryption. In CCM, the authentication tag (denoted T in CCM) is encrpted in the last block of ciphertext C and decrypted from the same block. Figure 6.36-10 illustrates the CCM encryption and decryption flow. User can refer the document "Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality" to obtain the definition of all symbols. As shown in Figure 6.36-10, the CCM encryption and decryption has four main steps. To simplify harware architecture, this accerator only supports step 2 of encryption, step 3 of encryption, step 1 of decryption and step 3 of decryption. Thus, the input blocks structure of AES CCM in this accelerator are restricted to the following rules:

1. Input blocks of encryption = $\{B_0 \| \ldots \| B_r \| B_{r-m} \| B_{r-m+1} \| \ldots \| B_r\}$; where B_0 to B_r are generated from Formatting function.
2. Input blocks of decryption = $\{B_0 \| \ldots \| B_r \| B_{r-m} \| B_{r-m+1} \| \ldots \| B_r\}$; where B_0 to B_{r-m} are generated from Formatting function and B_{r-m+1} to B_r is replaced by the ciphertext C.

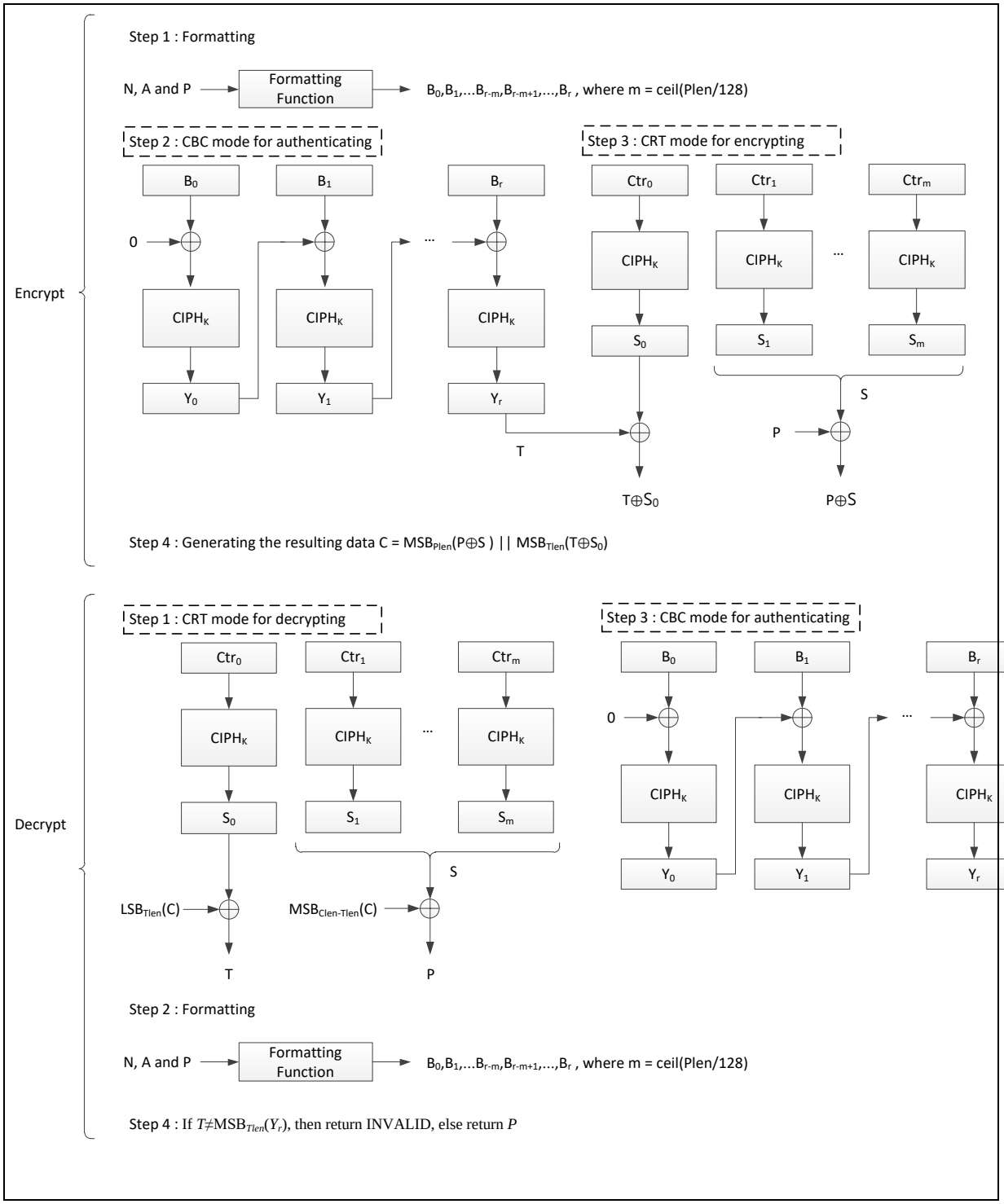


Figure 6.36-10 CCM Encryption and Decryption Flow

The output blocks structure of AES CCM are defined by the following rules:

3. Output blocks of encryption = { MSB_{Plen} (P⊕S) || (m*128-Plen) bits redundant data || T⊕S_0}

4.Output blocks of decryption = { MSB_{Plen} (C⊕S) || (m*128-Plen) bits redundant data || Y_r⊕S_0}

Both input and output blocks in CCM must be block alignment (128 bits), except for the last input block in the plaintext or ciphertext. Before starting every AES CCM operation, users have to write the byte count of A (i.e., (r-m+1)*16 bytes in Figure 6.36-10) and P (i.e., Plen/8 bytes in Figure 6.36-10) to the registers CRPT_AES_GCM_ACNT0/1 and CRPT_AES_GCM_PCNT0/1, respectively. Afterward, user must write initial vectors (i.e., Ctr_0 in Figure 6.36-10) to registers CRYPTO_AES_IV0 ~ CRYPTO_AES_IV3. After finishing CCM mode, users can obtain the ciphertext (i.e., MSB_{Plen}(P⊕S)) or plaintext (i.e., MSB_{Plen}(C⊕S)) with an authentication tag (i.e., T⊕S_0 or Y_r⊕S_0). In the end of CCM decryption, user can compare the last output block (i.e., MSB_{Tlen}(T⊕S_0) with the last input block of ciphertext C (i.e., MSB_{Tlen}(C_{m+1})) to make sure the authenticity of the confidential data in step 4. When adopting the DMA cascade mode in CCM mode, user must use DMA automatically feedback function by controlling the FBOUT and FBIN bits in CRYPTO_AES_CTL to trigger DMA engine to automatically write and read all required information according to the DMA feedback address register (i.e., CRYPTO_AES_FBADDR). Note that user must allocate 18 words continued memory space from the address CRYPTO_AES_FBADDR at least.

Galois/Counter Mode (GCM)

The Galois/Counter (GCM) mode is an algorithm for authenticated encryption/decryption with associated data. It provides the assurance of the confidentiality of data via CTR mode and the authenticity of the confidential data via GHASH function (i.e., binary Galois finite field hash function). The input data of GCM includes plaintext/ciphertext (denoted P/C in GCM) in encryption/decryption, additional authenticated data(denoted A in GCM) and initialization vector (denoted IV in GCM). The output data of GCM are C/P in encryption/decryption and an authentication tag (denoted T in GCM). Figure 6.36-11 illustrates the GCM encryption/decryption algorithm. User can refer to the document “Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC” to obtain the definition of all symbols in Figure 6.36-11, where the rectangle with gray color means input blocks, and the rectangle with dotted line represents output blocks. The input blocks structure of AES GCM for this cryptographic accelerator are restricted to the following rules:

1. Input blocks = {IV || 0^{31} || 1 || A || 0^{V} || P/C || 0^{U}} ; if len(IV) == 96
2. Input blocks = {IV || 0^{S} || 0^{64} || len(IV) || A || 0^{V} || P/C || 0^{U}} ; if len(IV) != 96

The output blocks structure of AES GCM are defined by the following rule

3. Output blocks = {C/P || U bits redundant data || T/T’}

Both input and output blocks in GCM must be block alignment (128 bits). Before starting every AES GCM operation, users have to write the byte count of IV, A and P to the registers CRPT_AES_GCM_IVCNT0/1, CRPT_AES_GCM_ACNT0/1 and CRPT_AES_GCM_PCNT0/1, respectively. Besides, this cryptographic accelerator also provides the GHASH function of AES GCM. The input blocks structure of AES GHASH are restricted to the block alignment. After finishing AES GHASH operation, cryptographic accelerator will output a hash value (128-bits). In the end of GCM decryption, user can compare the value of T’ with the one of T to make sure the authenticity of the confidential data. When adopting the DMA cascade mode in GCM mode, user must use DMA automatically feedback function by controlling the FBOUT and FBIN bits in CRYPTO_AES_CTL to trigger DMA engine to automatically write and read all required information according to the DMA feedback address register (i.e., CRYPTO_AES_FBADDR). Note that user must allocate 18 words continued memory space at least from the address CRYPTO_AES_FBADDR.

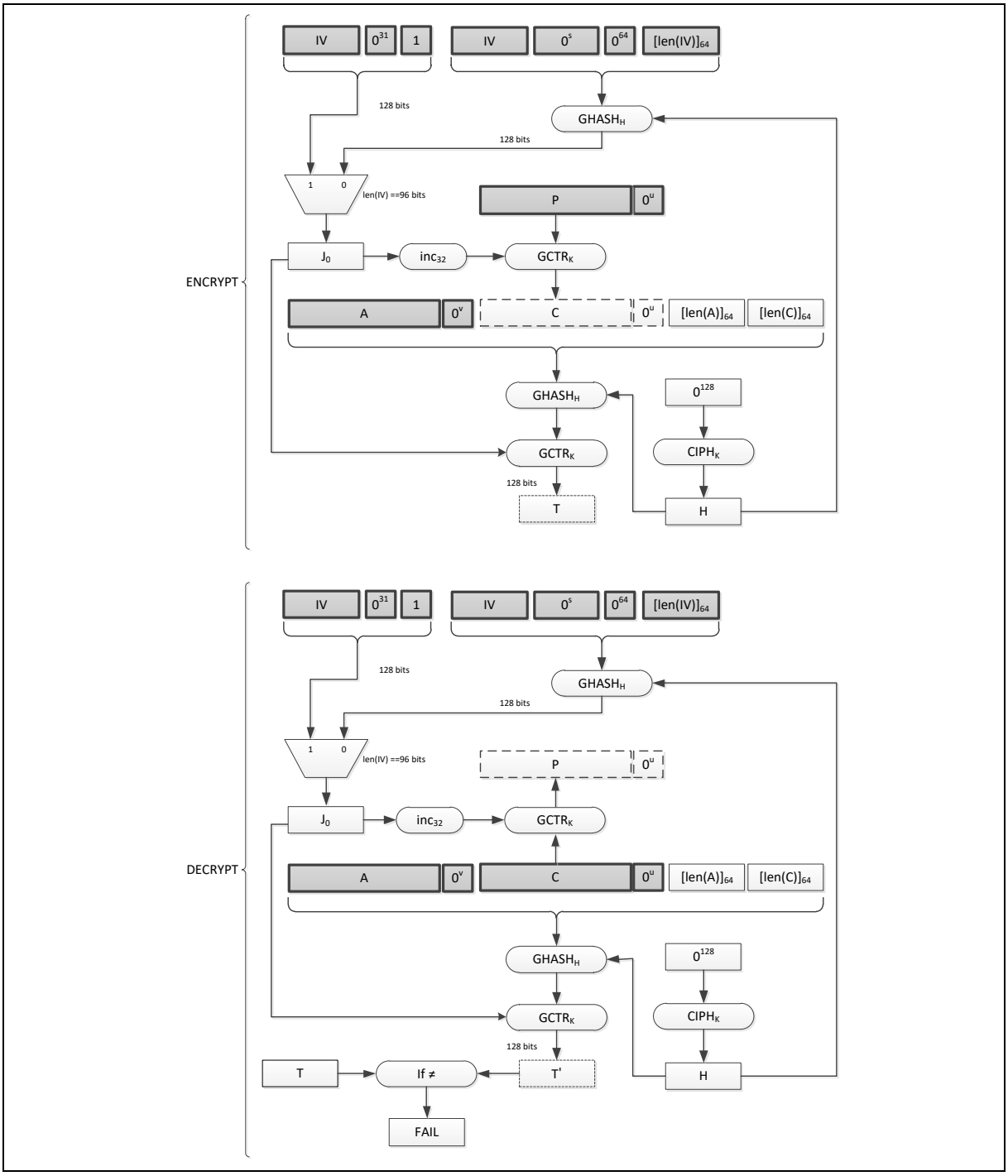


Figure 6.36-11 GCM encryption and decryption flow

Refer to the following programming steps for how to program the AES related registers.

AES DMA Mode Programming Flow

1. Write 1 to AESIEN (CRYPTO_INTEN[0]) to enable AES interrupt if needed.
2. Program AES key information to register CRYPTO_AES_KSCTL when key is from key store IP.

3. Program AES key to registers CRYPTO_AES_KEY0 ~ CRYPTO_AES_KEY7 when AES key is not from key store IP.
4. Program initial vectors to registers CRYPTO_AES_IV0 ~ CRYPTO_AES_IV3.
5. Program the total byte count of IV, A and P to the registers CRPT_AES_GCM_IVCNT0/1, CRPT_AES_GCM_ACNT0/1 and CRPT_AES_GCM_PCNT0/1 in AES CCM or GCM, respectively. Note that IV is not necessary for AES CCM.
6. Program DMA source address to register CRYPTO_AES_SADDR.
7. Program DMA destination address to register CRYPTO_AES_DADDR.
8. Program DMA byte count to register CRYPTO_AES_CNT.
9. Program DMA feedback address to register CRYPTO_AES_FBADDR (18 words memory space must be reserved for feedback information) if DMA automatically feedback function is needed.
10. Program AES control register CRYPTO_AES_CTL for FBOUT and FBIN to trigger AES DMA engine to write and read feedback information according to CRYPTO_AES_FBADDR if needed.
11. Configure AES control register CRYPTO_AES_CTL for key protection(KEYPRT), DMA input/output swap(INSWAP/OUTSWAP), encryption/decryption(ENCRYPTO), operational mode(OPMODE), DMA mode, DMA cascade control and key size(KEYSZ).
12. Write input data to DMA source address with selected DMA byte count.
13. Write 1 to START(CRYPTO_AES_CTL[0]) to start AES encryption/decryption.
14. Wait for the AES interrupt flag AESIF (CRYPTO_INTSTS[0]) be set or AES busy flag (CRYPTO_AES_STS[0]) to be set to zero.
15. Read output data from DMA destination address with selected DMA byte count.
16. Repeat step 6 to step 12 until all data is processed if DMACSCAD (CRYPTO_AES_CTL[6]) enabled.

AES Non-DMA Mode Programming Flow

1. Write 1 to AESIEN (CRYPTO_INTEN[0]) to enable AES interrupt if needed.
2. Program AES key information to register CRYPTO_AES_KSCTL when key is from key store IP.
3. Program AES key to register CRYPTO_AES_KEY0 ~ CRYPTO_AES_KEY7 when AES key is not from key store IP.
4. Program initial vectors to register CRYPTO_AES_IV0 ~ CRYPTO_AES_IV3.
5. Program the total byte count of IV, A and P to registers CRPT_AES_GCM_IVCNT0/1, CRPT_AES_GCM_ACNT0/1 and CRPT_AES_GCM_PCNT0/1 in AES CCM and GCM, respectively. Note that IV is not necessary for AES CCM.
6. Configure AES control register CRYPTO_AES_CTL for key protection(KEYPRT), input/output swap(INSWAP/OUTSWAP), encryption/decryption(ENCRYPTO), operational mode(OPMODE), DMA cascade control and key size(KEYSZ).
7. Write 1 to START(CRYPTO_AES_CTL[0]) to start AES encryption/decryption.
8. Polling INBUFFULL(CRYPTO_AES_STS[9]) and OUTBUFEMPTY(CRYPTO_AES_STS[16]). If INBUFFULL(CRYPTO_AES_STS[9]) is 0, write 32 bits input data to CRYPTO_AES_DATIN.If OUTBUFEMPTY(CRYPTO_AES_STS[16]) is 0, read 32 bits data from CRYPTO_AES_DATOUT.

9. Repeat step 8 until 128 bits data (16 bytes) is written to and read from AES engine.
10. Write 1 to DMALAST(CRYPTO_AES_CTL[5]) if current operation is the last operation.
11. Write data byte count of last operation to register CRYPTO_AES_CNT if current operation is the last operation.
12. Repeat steps 7 to step 11 until all data is processed.

6.36.5.3 SHA (Secure Hash Algorithm)

The SHA accelerator supports not only SHA1 and SHA2 algorithm from U.S NIST but also SM3 cryptographic hash algorithm from CHINA state cryptography administration.

User can refer to the following steps to understand how to program the SHA/HMAC related registers.

SHA DMA Mode Programming Flow

1. Write 1 to HMACIEN (CRYPTO_INTEN[24]) to enable SHA interrupt if needed.
2. Configure SHA control register CRYPTO_HMAC_CTL for SHA engine input/output data swap(INSWAP/OUTSWAP), DMA mode(DMAEN), and SHA operation mode(OPMODE). Clear HMACEN (CRYPTO_HMAC_CTL[4]) to select SHA mode.
3. This step is only for cascade mode. Write 1 to DMACSCAD (CRYPTO_HMAC_CTL[6]) and DMAFIRST (CRYPTO_HMAC_CTL[4]) in the first time of cascade mode computation. Write 1 to DMACSCAD (CRYPTO_HMAC_CTL[6]) and DMALAST (CRYPTO_HMAC_CTL[5]) in the final time of cascade mode computation. In other cases of cascade mode computation, only write 1 to DMACSCAD(CRYPTO_HMAC_CTL[6]).
4. This step is only for manual cascade mode. Write the feedback information of the last cascade mode computation to feedback registers (CRYPTO_HMAC_FDBCK0~53 for SHA1/2).
5. This step is only for automatic cascade mode by DMA automatically feedback function. Program DMA feedback address to register CRYPTO_HMAC_FBADDR for reading or writing feedback information in SRAM (user must reserve 54 words space for SHA1/2 feedback information).
6. This step is only for automatic cascade mode by DMA automatically feedback function. Program HMAC control register CRYPTO_HMAC_CTL for FBOUT and FBIN to trigger HMAC DMA engine to output and input feedback information according to CRYPTO_HMAC_FBADDR.
7. Program DMA source address to register CRYPTO_HMAC_SADDR.
8. Program DMA byte count to register CRYPTO_HMAC_DMACNT.
9. Write input data to DMA source address with selected DMA byte count.
10. Write 1 to START(CRYPTO_HMAC_CTL[0]) to start SHA encryption.
11. Wait for the SHA interrupt flag HMACIF (CRYPTO_INTSTS[24]) be set.
12. Read output digest (SHA-160: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST4, SHA-224: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST6, SHA-256: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST7, SHA-384: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST11, SHA-512: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST15).

Note: The KEYCNT(CRYPTO_HMAC_KEYCNT[31:0]) keeps the byte count of key that SHA engine operates. In SHA with DMA cascade mode, the CRYPTO_HMAC_DMACNT must keep block size alignment, except for the final time of cascade mode computation. The block size of HMAC-SHA-512 and HMAC-SHA-384 are 128 bytes; others are 64 bytes. After finishing every cascase mode computation, user needs to store the feedback information from all HMAC feedback registers if the

further computation will cascade the current one.

SHA Non-DMA Mode Programming Flow

1. Configure SHA control register CRYPTO_HMAC_CTL for SHA engine input/output data swap(INSWAP/OUTSWAP) and SHA operation mode(OPMODE). Clear HMACEN (CRYPTO_HMAC_CTL[4]) to select SHA mode.
2. If it is the last input word, set DMALAST(CRYPTO_HMAC_CTL[5]).
3. Write 1 to START(CRYPTO_HMAC_CTL[0]) to start SHA encryption.
4. Wait for the SHA data input request DATINREQ(CRYPTO_HMAC_STS[16]) be set.
5. Write one word of input data to CRYPTO_HMAC_DATIN.
6. Repeat step 2 to 5 until all inut words are written into SHA engine.
7. Wait for the BUSY (CRYPTO_HMAC_STS[0]) be cleared.
8. Read output digest (SHA-160: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST4, SHA-224: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST6, SHA-256: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST7, SHA-384: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST11, SHA-512: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST15).

Note: The KEYCNT(CRYPTO_HMAC_KEYCNT[31:0]) keeps the byte count of key that SHA engine operates.

6.36.5.4 HMAC (Keyed-Hash Message Authentication Code)

The Keyed-Hash Message Authentication Code is a specific construction for calculating a message authentication code involving a cryptographic hash function in combination with a secret cryptographic key. Any cryptographic hash function, such as SHA-256, may be used in the calculation of an HMAC; the resulting MAC algorithm is termed HMAC-SHA-256 accordingly.User can refer to the following steps to understand how to program the SHA/HMAC related registers.

HMAC DMA Mode Programming Flow

1. Write 1 to HMACIEN(CRYPTO_INTEN[24]) to enable HMAC interrupt if needed.
2. Program HMAC key information to register CRPT_HMAC_KSCTL when key is from key store IP.
3. Configure SHA/HMAC control register CRYPTO_HMAC_CTL for HMAC engine input/output data swap(INSWAP/OUTSWAP), DMA mode(DMAEN), and HMAC operation mode(OPMODE). Set HMACEN(CRYPTO_HMAC_CTL[4]) to select HMAC mode.
4. This step is only for cascade mode. Write 1 to DMACSCAD (CRYPTO_HMAC_CTL[6]) and DMAFIRST (CRYPTO_HMAC_CTL[4]) in the first time of cascade mode computation. Write 1 to DMACSCAD (CRYPTO_HMAC_CTL[6]) and DMALAST (CRYPTO_HMAC_CTL[5]) in the final time of cascade mode computation. In other cases of cascade mode computation, only write 1 to DMACSCAD(CRYPTO_HMAC_CTL[6]).
5. This step is only for manual cascade mode. Write the feedback information of the last cascade mode computation to feedback registers (CRYPTO_HMAC_FDBCKx)
6. This step is only for automatic cascade mode by DMA automatically feedback function. Program DMA feedback address to register CRYPTO_HMAC_FBADDR for reading or writing feedback information in SRAM (user must reserve 54 words space for feedback information).
7. This step is only for automatic cascade mode by DMA automatically feedback function. Program HMAC control register CRYPTO_HMAC_CTL for FBOUT and FBIN to trigger HMAC DMA engine to output and input feedback information according to

CRYPTO_HMAC_FBADDR.

8. Program DMA source address to register CRYPTO_HMAC_SADDR.
9. Program DMA byte count to register CRYPTO_HMAC_DMACNT.
10. Write input data to DMA source address with selected DMA byte count.
11. Write 1 to START(CRYPTO_HMAC_CTL[0]) to start HMAC encryption.
12. Wait for the HMAC interrupt flag HMACIF(CRYPTO_INTSTS[24]) to be set.
13. Read output digest (HMAC-SHA-160: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST4, HMAC-SHA-224: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST6, HMAC-SHA-256: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST7, HMAC-SHA-384: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST11, HMAC-SHA-512: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST15).

Note: The KEYCNT(CRYPTO_HMAC_KEYCNT[31:0]) keeps the byte count of key that HMAC engine operates. In HMAC DMA with cascade mode, the CRYPTO_HMAC_DMACNT must keep block size alignment, except for the final time of cascade mode computation. Do not merge the key and data in the same block when executing HMAC DMA with cascade mode computation. If the KEYCNT is more than one block size and not block size alignment, user must padding 0x0 in the end of key to ensure that its length is block size alignment. In above case, user must add the length of padding bits to the CRYPTO_HMAC_DMACNT, not to KEYCNT. For example, user wants to execute a HMAC-SHA-256 (block size is 64 bytes) with DMA cascade mode computation, 96 bytes key size and 96 bytes data size. In this case, user must allocate two block size memory space for HMAC key (not contiguous memory allocation). The first block is the 64 bytes of HMAC key[511:0] and the second one is 32 bytes of HMAC key [767:512] and 32 bytes zero padding. If the first cascade computation is only for HMAC key, user must set CRYPTO_HMAC_DMACNT to 0x80 and KEYCNT to 0x60. After finishing every cascase mode computation, user needs to store the feedback information from all HMAC feedback registers if the further computation will cascade the current one. The size of HMAC key in key store must be block size (i.e., HMAC-SHA-512 and HMAC-SHA-384 are 128 bytes; others are 64 bytes).

HMAC Non-DMA Mode Programming Flow

1. Configure SHA/HMAC control register CRYPTO_HMAC_CTL for SHA/HMAC engine input/output data swap(INSWAP/OUTSWAP) and HMAC operation mode(OPMODE). Set HMACEN(CRYPTO_HMAC_CTL[4]) to select HMAC mode.
2. If it is the last input word, set DMALAST(CRYPTO_HMAC_CTL[5]).
3. Program HMAC key information to register CRPT_HMAC_KSCTL.
4. Write 1 to START(CRYPTO_HMAC_CTL[0]) to start HMAC encryption.
5. Wait for the HMAC data input request DATINREQ(CRYPTO_HMAC_STS[16]) to be set.
6. Write one word of input data to CRYPTO_HMAC_DATIN.
7. Repeat step 2 to 5 until all inut words are written into SHA engine.
8. Wait for the BUSY (CRYPTO_HMAC_STS[0]) to be cleared.
9. Read output digest (HMAC-SHA-160: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST4, HMAC-SHA-224: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST6, HMAC-SHA-256: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST7, HMAC-SHA-384: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST11, HMAC-SHA-512: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST15).

Note: The KEYCNT(CRYPTO_HMAC_KEYCNT[31:0]) keeps the byte count of key that HMAC engine operates. The maxinum size of HMAC key in key store is block size (i.e., HMAC-SHA-512 and HMAC-SHA-384 are 128 bytes; others are 64 bytes).

6.36.5.5 ECC (Elliptic Curve Cryptography)

Elliptic Curve Cryptography (ECC) is a famous approach of public-key cryptosystems. Recently, many protocols and applications utilize the algebraic cyclic group characters of elliptic curves over finite field to build cryptographic systems. All points of an elliptic curve will follow the formula of elliptic curve : $y^2 \equiv x^3+A*x+B \pmod N$ or $B*y^2 \equiv x^3+A*x^2+x \pmod N$ in GF(p) and $y^2+x*y \equiv x^3+A*x^2+B \pmod N$ in GF(2^m). Thus, ECC accelerator also can support SM2 algorithm from CHINA state cryptography administration. Figure 6.27-10 exhibits the main hierarchy chart of ECC applications. The often appeared parameters and corresponding registers are shown in Table 6.36-4.

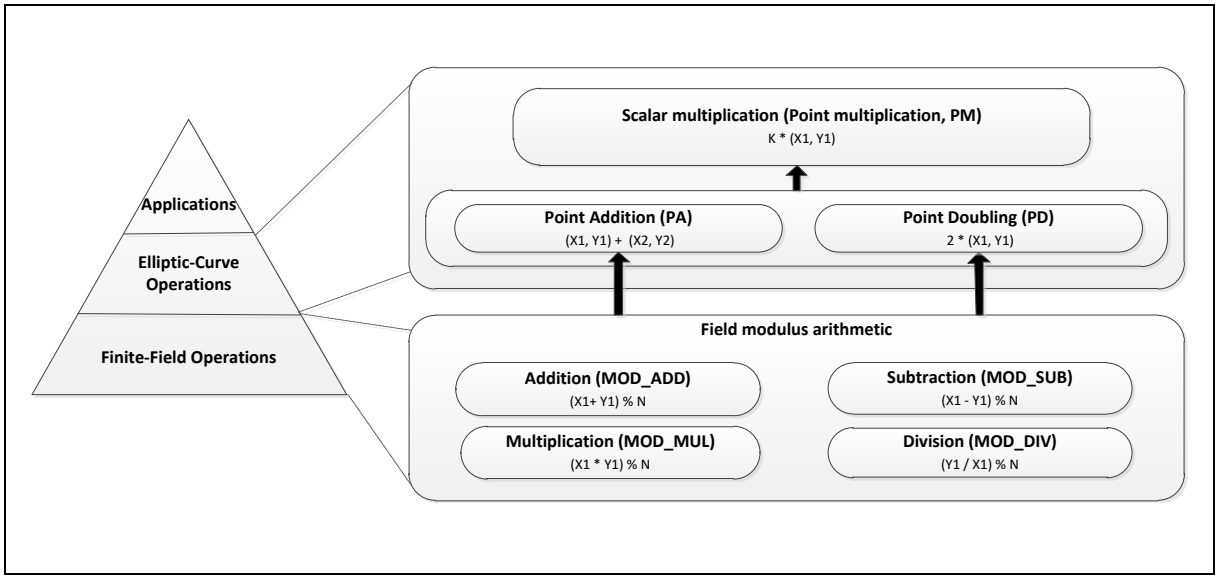


Figure 6.36-12 Main Hierarchy Chart of ECC

Parameter	Description	Corresponding Register
X1	The x-coordinate of point1	CRYPTO_ECC_X1_00~ CRYPTO_ECC_X1_17
Y1	The y-coordinate of point1	CRYPTO_ECC_Y1_00~ CRYPTO_ECC_Y1_17
X2	The x-coordinate of point2	CRYPTO_ECC_X2_00~ CRYPTO_ECC_X2_17
Y2	The y-coordinate of point2	CRYPTO_ECC_Y2_00~ CRYPTO_ECC_Y2_17
A	The curve parameter A	CRYPTO_ECC_A_00~ CRYPTO_ECC_A_17
B	The curve parameter B	CRYPTO_ECC_B_00~ CRYPTO_ECC_B_17
N	The curve parameter N	CRYPTO_ECC_N_00~ CRYPTO_ECC_N_17
M	The curve length	CRYPTO_ECC_CTL[31:22]
K	The scalar constant	CRYPTO_ECC_K_00~ CRYPTO_ECC_K_17

Table 6.36-4 ECC Parameters and Corresponding Registers Table

Scalar multiplication (point multiplication) is the core operation in ECC applications. The computation of scalar multiplication is composed of point addition and point doubling operations. Moreover, there are many finite field modulus arithmetic operations in the formula of point addition and doubling operation. To accelerate ECC applications, an elliptic curve cryptographic accelerator can be used to process not only three point operations in both GF(p) and GF(2^m) but also four modulus operations in GF(p). Before starting ECC accelerator, user must provide the required input data of ECC operation

include point coordinates (X1, Y1, X2, Y2), curve parameters (A, B, N, M) and scalar data (K) in Table 6.36-5. The mark "√" means that the input data is necessary for this operation. The detail definition of input data and the corresponding registers in the ECC accelerator are exhibited in the next section Register Map. When executing point multiplication (PM) with side-channel attack protection (SCPA = CRYPTO_ECC_CTL[14] =1), users must write the order value of elliptic curve into the registers from CRYPTO_ECC_X2_00 to CRYPTO_ECC_X2_17. To bring the side-channel protection of ECC into full play, it is suggested that user must execute "TRNG generate the true random number seed steps" (please refer to TRNG DTS) at least once before starting the first ECC operation after every CRYPTO or system reset.

After ECC accelerator finished, all point operations will generate a output point includes x-coordinate in the registers from CRYPTO_ECC_X1_00 to CRYPTO_ECC_X1_17 and y-coordinate in the registers from CRYPTO_ECC_Y1_00 to CRYPTO_ECC_Y1_17. In all modulus operations, ECC accelerator will only produce a output result in the registers from CRYPTO_ECC_X1_00 to CRYPTO_ECC_X1_17.

Operation	PM	PA	PD	MOD_DIV	MOD_MUL	MOD_ADD	MOD_SUB	ECDSAS
ECCOP[1:0]	00	10	11	01	01	01	01	01
MODOP[1:0]	XX	XX	XX	00	01	10	11	XX
ECDSAS	X	X	X	X	X	X	X	1
X1	√	√	√	√	√	√	√	√
Y1	√	√	√	√	√	√	√	√
X2		√						√
Y2		√						√
A	√	√	√					
B	√		√					
N	√	√	√	√	√	√	√	√
M	√	√	√		√			√
K	√							

Table 6.36-5 Required Input Data of Various Operations

User can refer to the following steps to understand how to program the ECC related registers.

ECC DMA Mode Programming Flow

1. Write 1 to ECCIEN(CRYPTO_INTEN[22]) to enable ECC interrupt if needed.
2. Program DMA source address to register CRYPTO_ECC_SADDR.
3. Program DMA destination address to register CRYPTO_ECC_DADDR.
4. Program DMA word count to register CRYPTO_ECC_WORDCNT.
5. Program the starting register address of all input data in Table 6.36-5 that will update to the register CRYPTO_ECC_STARTRE.G.
6. Write input data to DMA source address with selected DMA word count.
7. Program ECC key information to register CRYPTO_ECC_KSCTL and CRYPTO_ECC_KSXY when key is form key store IP.
8. Configure ECC control register CRYPTO_ECC_CTL for ECC accelerator, such as the start

signal of ECC accelerator(START), DMA mode enable signal(DMAEN), field selection(FSEL), point operation mode(ECCOP), modulus operation mode(MODOP), curve selection(CSEL), side-channel attack protection(SCAP), the control signals fo all input data registers(LDA, LDB, LDN, LDK, LDP1, LDP2), and the key length of elliptic curve(CURVEM).

9. Wait for the ECC interrupt flag ECCIF(CRYPTO_INTSTS[22]) to be set.
10. Read output data and then clear ECC interrupt flag ECCIF.

ECC Non-DMA Mode Programming Flow

1. Write all necessary input data in the corresponding registers according to in Table 6.36-5, such as CURVEA, CURVEB, CURVEN, SCALARK.
2. Program ECC key information to register CRYPTO_ECC_KSCTL and CRYPTO_ECC_KSXY whe key is from key store IP.
3. Configure ECC control register CRYPTO_ECC_CTL for ECC accelerator, such as the start signal of ECC accelerator (START), field selection (FSEL), point operation mode (ECCOP), modulus operation mode (MODOP), curve selection (CSEL), side-channel attack protection (SCAP), the control signals fo all input data registers (LDA, LDB, LDN, LDK, LDP1, LDP2), and the key length of elliptic curve(CURVEM).
4. Wait for the BUSY (CRYPTO_ECC_STS[0]) to be cleared.
5. Read output digest and then clear ECC interrupt flag ECCIF.

Some Notices of ECC Accelerator

1. The key length support of ECC accelerator is from 163 to 571 bits.
2. All input and output data must be positive. (If the input data is negative, it must be added N).
3. The irreducible polynomial of GF(2^m) must adopt the smallest one from HP. Please refer to Table 6.36-6 (Reference from HP, Table of Low-Weight Binary Irreducible Polynomials, HPL-98-135, August, 1998).

	2,1	**3,1**	**4,1**	**5,2**
6,1	7,1	8,4,3,1	9,1	10,3
11,2	12,3	13,4,3,1	14,5	15,1
16,5,3,1	17,3	18,3	19,5,2,1	20,3
21,2	22,1	23,5	24,4,3,1	25,3
26,4,3,1	27,5,2,1	28,1	29,2	30,1
31,3	32,7,3,2	33,10	34,7	35,2
36,9	37,6,4,1	38,6,5,1	39,4	40,5,4,3
41,3	42,7	43,6,4,3	44,5	45,4,3,1
46,1	47,5	48,5,3,2	49,9	50,4,3,2
51,6,3,1	52,3	53,6,2,1	54,9	55,7
56,7,4,2	57,4	58,19	59,7,4,2	60,1
61,5,2,1	62,29	63,1	64,4,3,1	65,18
66,3	67,5,2,1	68,9	69,6,5,2	70,5,3,1

71,6	72,10,9,3	73,25	74,35	75,6,3,1
76,21	77,6,5,2	78,6,5,3	79,9	80,9,4,2
81,4	82,8,3,1	83,7,4,2	84,5	85,8,2,1
86,21	87,13	88,7,6,2	89,38	90,27
91,8,5,1	92,21	93,2	94,21	95,11
96,10,9,6	97,6	98,11	99,6,3,1	100,15
101,7,6,1	102,29	103,9	104,4,3,1	105,4
106,15	107,9,7,4	108,17	109,5,4,2	110,33
111,10	112,5,4,3	113,9	114,5,3,2	115,8,7,5
116,4,2,1	117,5,2,1	118,33	119,8	120,4,3,1
121,18	122,6,2,1	123,2	124,19	125,7,6,5
126,21	127,1	128,7,2,1	129,5	130,3
131,8,3,2	132,17	133,9,8,2	134,57	135,11
136,5,3,2	137,21	138,8,7,1	139,8,5,3	140,15
141,10,4,1	142,21	143,5,3,2	144,7,4,2	145,52
146,71	147,14	148,27	149,10,9,7	150,53
151,3	152,6,3,2	153,1	154,15	155,62
156,9	157,6,5,2	158,8,6,5	159,31	160,5,3,2
161,18	162,27	163,7,6,3	164,10,8,7	165,9,8,3
166,37	167,6	168,15,3,2	169,34	170,11
171,6,5,2	172,1	173,8,5,2	174,13	175,6
176,11,3,2	177,8	178,31	179,4,2,1	180,3
181,7,6,1	182,81	183,56	184,9,8,7	185,24
186,11	187,7,6,5	188,6,5,2	189,6,5,2	190,8,7,6
191,9	192,7,2,1	193,15	194,87	195,8,3,2
196,3	197,9,4,2	198,9	199,34	200,5,3,2
201,14	202,55	203,8,7,1	204,27	205,9,5,2
206,10,9,5	207,43	208,9,3,1	209,6	210,7
211,11,10,8	212,105	213,6,5,2	214,73	215,23
216,7,3,1	217,45	218,11	219,8,4,1	220,7
221,8,6,2	222,5,4,2	223,33	224,9,8,3	225,32
226,10,7,3	227,10,9,4	228,113	229,10,4,1	230,8,7,6
231,26	232,9,4,2	233,74	234,31	235,9,6,1
236,5	237,7,4,1	238,73	239,36	240,8,5,3

241,70	242,95	243,8,5,1	244,111	245,6,4,1
246,11,2,1	247,82	248,15,14,10	249,35	250,103
251,7,4,2	252,15	253,46	254,7,2,1	255,52
256,10,5,2	257,12	258,71	259,10,6,2	260,15
261,7,6,4	262,9,8,4	263,93	264,9,6,2	265,42
266,47	267,8,6,3	268,25	269,7,6,1	270,53
271,58	272,9,3,2	273,23	274,67	275,11,10,9
276,63	277,12,6,3	278,5	279,5	280,9,5,2
281,93	282,35	283,12,7,5	284,53	285,10,7,5
286,69	287,71	288,11,10,1	289,21	290,5,3,2
291,12,11,5	292,37	293,11,6,1	294,33	295,48
296,7,3,2	297,5	298,11,8,4	299,11,6,4	300,5
301,9,5,2	302,41	303,1	304,11,2,1	305,102
306,7,3,1	307,8,4,2	308,15	309,10,6,4	310,93
311,7,5,3	312,9,7,4	313,79	314,15	315,10,9,1
316,63	317,7,4,2	318,45	319,36	320,4,3,1
321,31	322,67	323,10,3,1	324,51	325,10,5,2
326,10,3,1	327,34	328,8,3,1	329,50	330,99
331,10,6,2	332,89	333,2	334,5,2,1	335,10,7,2
336,7,4,1	337,55	338,4,3,1	339,16,10,7	340,45
341,10,8,6	342,125	343,75	344,7,2,1	345,22
346,63	347,11,10,3	348,103	349,6,5,2	350,53
351,34	352,13,11,6	353,69	354,99	355,6,5,1
356,10,9,7	357,11,10,2	358,57	359,68	360,5,3,2
361,7,4,1	362,63	363,8,5,3	364,9	365,9,6,5
366,29	367,21	368,7,3,2	369,91	370,139
371,8,3,2	372,111	373,8,7,2	374,8,6,5	375,16
376,8,7,5	377,41	378,43	379,10,8,5	380,47
381,5,2,1	382,81	383,90	384,12,3,2	385,6
386,83	387,8,7,1	388,159	389,10,9,5	390,9
391,28	392,13,10,6	393,7	394,135	395,11,6,5
396,25	397,12,7,6	398,7,6,2	399,26	400,5,3,2
401,152	402,171	403,9,8,5	404,65	405,13,8,2
406,141	407,71	408,5,3,2	409,87	410,10,4,3

411,12,10,3	412,147	413,10,7,6	414,13	415,102
416,9,5,2	417,107	418,199	419,15,5,4	420,7
421,5,4,2	422,149	423,25	424,9,7,2	425,12
426,63	427,11,6,5	428,105	429,10,8,7	430,14,6,1
431,120	432,13,4,3	433,33	434,12,11,5	435,12,9,5
436,165	437,6,2,1	438,65	439,49	440,4,3,1
441,7	442,7,5,2	443,10,6,1	444,81	445,7,6,4
446,105	447,73	448,11,6,4	449,134	450,47
451,16,10,1	452,6,5,4	453,15,6,4	454,8,6,1	455,38
456,18,9,6	457,16	458,203	459,13,5,2	460,19
461,7,6,1	462,73	463,93	464,19,18,13	465,31
466,14,11,6	467,11,6,1	468,27	469,9,5,2	470,9
471,1	472,11,3,2	473,200	474,191	475,9,8,4
476,9	477,16,15,7	478,121	479,104	480,15,9,6
481,138	482,9,6,5	483,9,6,4	484,105	485,17,16,6
486,81	487,94	488,4,3,1	489,83	490,219
491,11,6,3	492,7	493,10,5,3	494,17	495,76
496,16,5,2	497,78	498,155	499,11,6,5	500,27
501,5,4,2	502,8,5,4	503,3	504,15,14,6	505,156
506,23	507,13,6,3	508,9	509,8,7,3	510,69
511,10	512,8,5,2	513,26	514,67	515,14,7,4
516,21	517,12,10,2	518,33	519,79	520,15,11,2
521,32	522,39	523,13,6,2	524,167	525,6,4,1
526,97	527,47	528,11,6,2	529,42	530,10,7,3
531,10,5,4	532,1	533,4,3,2	534,161	535,8,6,2
536,7,5,3	537,94	538,195	539,10,5,4	540,9
541,13,10,4	542,8,6,1	543,16	544,8,3,1	545,122
546,8,2,1	547,13,7,4	548,10,5,3	549,16,4,3	550,193
551,135	552,19,16,9	553,39	554,10,8,7	555,10,9,4
556,153	557,7,6,5	558,73	559,34	560,11,9,6
561,71	562,11,4,2	563,14,7,3	564,163	565,11,6,1
566,153	567,28	568,15,7,6	569,77	570,67
571,10,5,2	572,12,8,1	573,10,6,4	574,13	575,146
576,13,4,3	577,25	578,23,22,16	579,12,9,7	580,237

Table 6.36-6 Low-Weight Binary Irreducible Polynomials

4. Only when START and DMAEN (CRYPTO_ECC_CTL[0] and [7]) are assigned to 1 simultaneously, ECC DMA mode will be active.
5. When ECC engine is active (i.e., BUSY is 1 and DMABUSY (CRYPTO_ECC_STS[1]) is 0), user cannot modify all input data registers (CRYPTO_ECC_X1_00 ~ CRYPTO_ECC_K_17).
6. If user wants to stop ECC accelerator, please configure the STOP (CRYPTO_ECC_CTL[1]) to 1. **Note:** To avoid the transmission error of the next operation, BUSY signal will not be cleared immediately until the action of DMA is done.
7. The modulus operation is not supported for binary field.
8. The input data of modulus multiplication and division operation for PF must be less than N.
9. K is private key, so this register is write only.
10. When side-channel attack protection (i.e., CRYPTO_ECC_CTL[14] =1) is active, users must write the order value of elliptic curve into the registers from CRYPTO_ECC_X2_00 to CRYPTO_ECC_X2_17 beforehand.

The following describes the method of application about Modular operations, key pair generation, ECDSA, ECDSA with key store, ECDH and ECDH with key store.

*6.36.5.6 **Modular Operations***

Modular addition in prime field: X1 = X1 + Y1 (mod N)

1. Write the *modulus* to N register.
2. Write the summand to X1 register and the addend to Y1 register.
3. Set MODOP(CRYPTO_ECC_CTL[12:11]) to 10
4. Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
5. Set FSEL(CRYPTO_ECC_CTL[8]) to 1
6. Set START(CRYPTO_ECC_CTL[0]) to 1
7. Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
8. Read result from X1 registers

Modular subtraction in prime field: X1 = X1 - Y1 (mod N)

1. Write the *modulus* to N register.
2. Write the minuend to X1 register and the subtrahend to Y1 register.
3. Set MODOP(CRYPTO_ECC_CTL[12:11]) to 11
4. Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
5. Set FSEL(CRYPTO_ECC_CTL[8]) to 1
6. Set START(CRYPTO_ECC_CTL[0]) to 1
7. Wait for BUSY(CRYPTO_ECC_STS[0]) be cleared
8. Read result from X1 registers

Modular multiplication in prime field: X1 = X1 * Y1 (mod N)

1. Write the *modulus* to N register and the bit length of multiplication to register M.
2. Write the multiplicand to X1 register and the multiplicator to Y1 register.

3. Set CURVEM (CRYPTO_ECC_CTL[31:22]) to key length
4. Set MODOP(CRYPTO_ECC_CTL[12:11]) to 01
5. Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
6. Set FSEL(CRYPTO_ECC_CTL[8]) to 1
7. Set START(CRYPTO_ECC_CTL[0]) to 1
8. Wait for BUSY(CRYPTO_ECC_STS[0]) be cleared
9. Read result from X1 registers

Modular division in prime field: X1 = Y1 / X1 (mod N)

1. Write the *modulus* to N register.
2. Write the divisor to X1 register and the dividend to Y1 register.
3. Set MODOP(CRYPTO_ECC_CTL[12:11]) to 00
4. Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
5. Set FSEL(CRYPTO_ECC_CTL[8]) to 1
6. Set START(CRYPTO_ECC_CTL[0]) to 1
7. Wait for BUSY(CRYPTO_ECC_STS[0]) be cleared
8. Read result from X1 registers

6.36.5.7 ***Key Pair Generation***

Public key generation function: Q = dG (mod N)

1. Write the curve parameter A, B, N, and curve length M to corresponding registers according to Table 6.36-4.
2. Write the point G(x, y) to X1, Y1 registers according to Table 6.36-4 or proram the key information of G(x, y) to register CRYPTO_ECC_KSXY and set RSRCXY (CRYPTO_ECC_KSXY[5]) to 1 for updating X1, Y1 registers from key store
3. Write the private key d to K register according to Table 6.36-4 or proram the key information of d to register CRYPTO_ECC_KSCTL and set RSRCK (CRYPTO_ECC_KSCTL[5]) to 1 for updating K register from key store.
4. Set CURVEM (CRYPTO_ECC_CTL[31:22]) to key length
5. Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00
6. Set FSEL(CRYPTO_ECC_CTL[8]) according to used curve of prime field or binary field
7. Set START(CRYPTO_ECC_CTL[0]) to 1
8. Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
9. Read public key Q from X1, Y1 registers

6.36.5.8 ***Elliptic Curve Digital Signature Algorithm (ECDSA)***

ECDSA signature generation steps:

1. Calculate e = HASH(m), where HASH is a cryptographic hashing algorithm, (i.e. SHA-160)
   1) Use SHA to calculate e
   2) Keep the leftmost L_{order} bits of e, where L_{order} is the bit length of order n

2. Select a random integer k form [1, n-1]
    1) Note that n is order, not prime modulus or irreducible polynomial function
3. Compute r = x1 (mod n), where (x1, y1) = k * G. If r = 0, go to step 2
    1) Write the curve parameter A, B, N and curve length M to corresponding registers according to Table 6.36-4
    2) Write the prime modulus or irreducible polynomial function to N registers according to Table 6.36-4
    3) Write the point G(x, y) to X1, Y1 registers according to Table 6.36-4
    4) Write the random integer k to K register according to Table 6.36-4
    5) Set CURVEM (CRYPTO_ECC_CTL[31:22]) to key length
    6) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00
    7) Set FSEL(CRYPTO_ECC_CTL[8]) according to used curve of prime field or binary field
    8) Set START(CRYPTO_ECC_CTL[0]) to 1
    9) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
    10) Write the curve order and curve length to N, M registers according to Table 6.36-4
    11) Write 0x0 to Y1 registers
    12) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
    13) Set MOPOP(CRYPTO_ECC_CTL[12:11]) to 10
    14) Set START(CRYPTO_ECC_CTL[0]) to 1
    15) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
    16) Read X1 registers to get r
4. Compute $s = k^{-1} \times (e + d \times r)(\text{mod } n)$. If s = 0, go to step 2
    1) Write the curve order to N registers according to Table 6.36-4
    2) Write the k to X1 register and the d to Y1 register according to Table 6.36-4
    3) Write the r to X2 register and the e to Y2 register according to Table 6.36-4
    4) Set CURVEM (CRYPTO_ECC_CTL[31:22]) to key length
    5) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
    6) Set ECDSAS(CRYPTO_ECC_CTL[4]) to 1
    7) Set START(CRYPTO_ECC_CTL[0]) to 1
    8) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
    9) Read X1 registers to get s
5. The signature is the pair (r, s)

ECDSA signature verification steps:

1. Verify that r and s are integers in the interval [1, n-1]. If not, the signature is invalid
2. Compute e = HASH (m), where HASH is the hashing algorithm in signature generation
    1) Use SHA to calculate e
    2) Keep the leftmost L_{order} bits of e, where L_{order} is the bit length of order n

3. Compute $w = s^{-1} \pmod{n}$
   1) Write the curve order to N registers according to Table 6.36-4
   2) Write s to X1 registers according to Table 6.36-4
   3) Write 0x1 to Y1 registers according to Table 6.36-4
   4) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
   5) Set MOPOP(CRYPTO_ECC_CTL[12:11]) to 00
   6) Set FSEL(CRYPTO_ECC_CTL[8]) to 1
   7) Set START(CRYPTO_ECC_CTL[0]) to 1
   8) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
   9) Read X1 registers to get w
4. Compute $u1 = e \times w \pmod{n}$ and $u2 = r \times w \pmod{n}$
   1) Write the curve order and curve length to N, M registers according to Table 6.36-4
   2) Write e, w to X1, Y1 registers
   3) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
   4) Set MOPOP(CRYPTO_ECC_CTL[12:11]) to 01
   5) Set START(CRYPTO_ECC_CTL[0]) to 1
   6) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
   7) Read X1 registers to get u1
   8) Write the curve order and curve length to N, M registers according to Table 6.36-4
   9) Write r, w to X1, Y1 registers
   10) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
   11) Set MOPOP(CRYPTO_ECC_CTL[12:11]) to 01
   12) Set START(CRYPTO_ECC_CTL[0]) to 1
   13) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
   14) Read X1 registers to get u2
5. Compute X' (x1', y1') = u1 * G + u2 * Q
   1) Write the curve parameter A, B, N, and curve length M to corresponding registers according to Table 6.36-4
   2) Write the point G(x, y) to X1, Y1 registers
   3) Write u1 to K registers
   4) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00
   5) Set START(CRYPTO_ECC_CTL[0]) to 1
   6) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
   7) Read X1, Y1 registers to get u1*G
   8) Write the curve parameter A, B, N, and curve length M to corresponding registers according to Table 6.36-4
   9) Write the publick key Q(x,y) to X1, Y1 registers

10) Write u2 to K registers
11) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00
12) Set START(CRYPTO_ECC_CTL[0]) to 1
13) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
14) Write the curve parameter A, B, N, and curve length M to corresponding registers according to Table 6.36-4
15) Write the result data u1*G to X2, Y2 registers
16) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 10
17) Set START(CRYPTO_ECC_CTL[0]) to 1
18) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
19) Read X1, Y1 registers to get X'(x1', y1')
20) Write the curve order and curve length to N, M registers according to Table 6.36-4
21) Write x1' to X1 registers
22) Write 0x0 to Y1 registers
23) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
24) Set MOPOP(CRYPTO_ECC_CTL[12:11]) to 10
25) Set START(CRYPTO_ECC_CTL[0]) to 1
26) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
27) Read X1 registers to get x1' (mod n)

6. The signature is valid if x1' = r, otherwise it is invalid.

*6.36.5.9 **Elliptic Curve Digital Signature Algorithm (ECDSA) With Key Store***

ECDSA signature generation steps:

1. Calculate e = HASH(m), where HASH is a cryptographic hashing algorithm, (i.e. SHA-160)
   1) Use SHA to calculate e
   2) Keep the leftmost L_{order} bits of e, where L_{order} is the bit length of order n
2. Select a random integer k form [1, n-1] via PRNG with TRNG and store in the key store
   1) Note that n is order, not prime modulus or irreducible polynomial function.
   2) Write the curve order n to ECC N registers for checking PNRG key.
   3) Check the BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0
   4) Configure PRNG key control register CRYPTO_PRNG_KSCTL. Set OWNER to 0x100, WSDST to 0x0, WDST to 0x1, ECDSA to 0x1 and PRIV to the user defined value in Key Store
   5) Configure PRNG control register CRYPTO_PRNG_CTL for KEYSZ to key length, SEEDSEL to 0x1 and SEEDSRC to 0x1, SEEDRLD to 0x1, and START to 0x1.
   6) Software checks BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0, or waits for the PRNGIF (CRYPTO_INTSTS[16]) (must enable PRNGIEN (CRYPTO_INTEN[16])). Then software can read the NUM (CRYPTO_PRNG_KSSTS[4:0]) of k.
3. Compute r = x1 (mod n), where (x1, y1) = k * G. If r = 0, go to step 2

1) Write the curve parameter A, B, N and curve length M to corresponding registers according to Table 6.36-4
2) Write the prime modulus or irreducible polynomial function to N registers according to Table 6.36-4
3) Write the point G(x, y) to X1 and Y1 registers according to Table 6.36-4 or proram the key information of G(x, y) to registe CRYPTO_ECC_KSXY and set RSRCXY (CRYPTO_ECC_KSXY[5]) to 1 for updating X1, Y1 registers from key store.
4) Write the key information of k to register CRYPTO_ECC_KSCTL and set RSRCK (CRYPTO_ECC_KSCTL[5]) to 1 for updating K register from key store
5) Set CURVEM (CRYPTO_ECC_CTL[31:22]) to key length
6) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00
7) Set FSEL(CRYPTO_ECC_CTL[8]) according to used curve of prime field or binary field
8) Set ECDSAR(CRYPTO_ECC_CTL[5]) to 1
9) Set START(CRYPTO_ECC_CTL[0]) to 1
10) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
11) Write the curve order and curve length to N, M registers according to Table 6.36-4
12) Write 0x0 to Y1 registers
13) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
14) Set MOPOP(CRYPTO_ECC_CTL[12:11]) to 10
15) Set START(CRYPTO_ECC_CTL[0]) to 1
16) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
17) Read X1 registers to get r

4. Compute $s = k^{-1} \times (e + d \times r) (\mathrm{mod}\ n)$. If s = 0, go to step 2
   1) Write the curve order to N registers according to Table 6.36-4
   2) Write RSSRCY (CRYPTO_ECC_KSXY[15:14]) and NUMY (CRYPTO_ECC_KSXY[12:8]) according to the key information of d
   3) Write RSRCXY (CRYPTO_ECC_KSXY[5]) to 1
   4) Write RSSRCX (CRYPTO_ECC_KSXY[7:6]) and NUMX (CRYPTO_ECC_KSXY[4:0]) according to the key information of k
   5) Write the r to X2 registers and the e to Y2 registers according to Table 6.36-4
   6) Set CURVEM (CRYPTO_ECC_CTL[31:22]) to key length
   7) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
   8) Set ECDSAS(CRYPTO_ECC_CTL[4]) to 1
   9) Set START(CRYPTO_ECC_CTL[0]) to 1
   10) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
   11) Read X1 registers to get s
5. The signature is the pair (r, s).

ECDSA signature verification steps:

1. Verify that r and s are integers in the interval [1, n-1]. If not, the signature is invalid
2. Compute e = HASH (m), where HASH is the hashing algorithm in signature generation
    1) Use SHA to calculate e
    2) Keep the leftmost L_{order} bits of e, where L_{order} is the bit length of order n
3. Compute $w = s^{-1} \pmod n$
    1) Write the curve order to N registers according to Table 6.36-4
    2) Write s to X1 registers according to Table 6.36-4
    3) Write 0x1 to Y1 registers according to Table 6.36-4
    4) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
    5) Set MOPOP(CRYPTO_ECC_CTL[12:11]) to 00
    6) Set FSEL(CRYPTO_ECC_CTL[8]) to 1
    7) Set START(CRYPTO_ECC_CTL[0]) to 1
    8) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
    9) Read X1 registers to get w
4. Compute $u1 = e \times w \pmod n$ and $u2 = r \times w \pmod n$
    1) Write the curve order and curve length to N, M registers according to Table 6.36-4
    2) Write e, w to X1, Y1 registers
    3) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
    4) Set MOPOP(CRYPTO_ECC_CTL[12:11]) to 01
    5) Set START(CRYPTO_ECC_CTL[0]) to 1
    6) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
    7) Read X1 registers to get u1
    8) Write the curve order and curve length to N, M registers according to Table 6.36-4
    9) Write r, w to X1, Y1 registers
    10) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
    11) Set MOPOP(CRYPTO_ECC_CTL[12:11]) to 01
    12) Set START(CRYPTO_ECC_CTL[0]) to 1
    13) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
    14) Read X1 registers to get u2
5. Compute X' (x1', y1') = u1 * G + u2 * Q
    1) Write the curve parameter A, B, N, and curve length M to corresponding registers according to Table 6.36-4
    2) Write the point G(x, y) to X1 and Y1 registers according to Table 6.36-4 or proram the key information of G(x, y) to register CRYPTO_ECC_KSXY and set RSRCXY (CRYPTO_ECC_KSXY[5]) to 1 for updating X1, Y1 registers from key store
    3) Write u1 to K registers
    4) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00

5) Set START(CRYPTO_ECC_CTL[0]) to 1
6) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
7) Read X1, Y1 registers to get u1*G
8) Write the curve parameter A, B, N, and curve length M to corresponding registers according to Table 6.36-4
9) Write the publick key Q(x,y) to X1 and Y1 registers according to Table 6.36-4 or proram the key information of Q(x, y) to register CRYPTO_ECC_KSXY and set RSRCXY (CRYPTO_ECC_KSXY[5]) to 1 for updating X1, Y1 registers from key store
10) Write u2 to K registers
11) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00
12) Set START(CRYPTO_ECC_CTL[0]) to 1
13) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
14) Write the curve parameter A, B, N, and curve length M to corresponding registers according to Table 6.36-4
15) Write the result data u1*G to X2, Y2 registers
16) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 10
17) Set START(CRYPTO_ECC_CTL[0]) to 1
18) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
19) Read X1, Y1 registers to get X'(x1', y1')
20) Write the curve order and curve length to N, M registers according to Table 6.36-4
21) Write x1' to X1 registers
22) Write 0x0 to Y1 registers
23) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
24) Set MOPOP(CRYPTO_ECC_CTL[12:11]) to 10
25) Set START(CRYPTO_ECC_CTL[0]) to 1
26) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
27) Read X1 registers to get x1' (mod n)

6. The signature is valid if x1' = r, otherwise it is invalid.

Elliptic Curve Diffie-Hellman

Share secret generation function: Z is the x-coordinate of Q where Q = dG (mod N)

1. Write the curve parameter A, B, N, and curve length M to corresponding registers according to Table 6.36-4.
2. Write the public key of receiving party G(x, y) to X1, Y1 registers according to Table 6.36-4.
3. Write (cofactor h * my private key d) to K register according to Table 6.36-4.
   1) h=1 in P-192, P-224, P-256, P-384 and P-521
   2) h=2 in B-163, B-233, B-283, B-409, B-571 and K-163
   3) h=4 in K-233, K-283, K-409 and K-571
4. Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00

5. Set FSEL(CRYPTO_ECC_CTL[8]) according to used curve of prime field or binary field
6. Set START(CRYPTO_ECC_CTL[0]) to 1
7. Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
8. Read public key Q from X1, Y1 registers

Hash-based key derivation function: DerivedKeyingMaterial = KDF(Z, OtherInput)

Step1 For i = 1 to reps, where reps = ceil((the length of Z, OtherInput)/hash length)

1. Write the curve parameter A, B, N, and curve length M to corresponding registers according to Table 6.36-4.
2. Write the public key of receiving party G(x, y) to X1, Y1 registers according to Table 6.36-4.

Step 2 Return DerivedKeyingMaterial = $Hash_1$ || $Hash_2$ || … || $Hash_{reps}$

Note:

1. For the details of OtherInput please refer to the page 46 in NIST SP 800-56A, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography”.
2. For the details of cofactor please refer to “RECOMMENDED ELLIPTIC CURVES FOR FEDERAL GOVERNMENT USE”.

Elliptic Curve Diffie-Hellman With Key Store

Exchange a share key via RNG, ECC and key store.

1. Generate privated key d_A via PRNG with TRNG to key store.
   1) Write the curve order n to ECC N registers for checking PNRG key.
   2) Check the BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0
   3) Configure PRNG key control register CRYPTO_PRNG_KSCTL. Set OWNER to 0x100, WSDST to 0x0, WDST to 0x1, ECDH to 0x1 and PRIV to the user defined value in Key Store
   4) Configure PRNG control register CRYPTO_PRNG_CTL forKEYSZ to key length, SEEDSEL to 0x1 and SEEDSRC to 0x1, SEEDRLD to 0x1, and START to 0x1.
   5) Software checks BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0, or waits for the PRNGIF (CRYPTO_INTSTS[16]) (must enable PRNGIEN (CRYPTO_INTEN[16])). Then software can read the NUMBER (CRYPTO_PRNG_KSSTS[4:0]) of d_A.
2. Generate public key Q_A according to d_A via ECC and key store
   1) Write the curve parameter A, B, N and curve length M to corresponding registers according to Table 6.36-4
   2) Write the prime modulus or irreducible polynomial function to N registers according to Table 6.36-4
   3) Write the public point G(x, y) to X1, Y1 registers according to Table 6.36-4 or proram the key information of G(x, y) to register CRYPTO_ECC_KSXY and set RSRCXY (CRYPTO_ECC_KSXY[5]) to 1 for updating X1, Y1 registers from key store
   4) Program the key information of dA to register CRYPTO_ECC_KSCTL. Set the RSRCK to

0x1 and the NUMK to the NUM (CRYPTO_PRNG_KSSTS[4:0]) of dA. Set the RSSRCKto the WSDST (CRYPTO_PRNG_KSCTL[23:22]). Set others bits to 0x0.

5) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00

6) Set FSEL(CRYPTO_ECC_CTL[8]) according to used curve of prime field or binary field

7) Set START(CRYPTO_ECC_CTL[0]) to 1

8) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared

9) Read X1 and Y1 registers to get public key Q_A.

3. Send our public key Q_A to the opposite side and receive their public key Q_B.
4. Generate the share key according to d_A and Q_B via ECC and key store

1) Write the curve parameter A, B, N and curve length M to corresponding registers according to Table 6.36-4

2) Write the prime modulus or irreducible polynomial function to N registers according to Table 6.36-4

3) Write the public key QB(x, y) to X1, Y1 registers according to Table 6.36-4 or proram the key information of QB(x, y) to register CRYPTO_ECC_KSXY and set RSRCXY (CRYPTO_ECC_KSXY[5]) to 1 for updating X1, Y1 registers from key store

4) Program ECC key information to register CRYPTO_ECC_KSCTL. Set the RSRCK to 0x1 and the NUMK to the NUMBER (CRYPTO_PRNG_KSSTS[4:0]) of dA. Set the RSSRCK to the WSDST (CRYPTO_PRNG_KSCTL[23:22]). Set RSRCXY to 0x0. Set PRIV to the user defined value. Set ECDH to 0x1. Set XY, WDST, WSDST and OWNER to the user defined value.

5) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00

6) Set FSEL(CRYPTO_ECC_CTL[8]) according to used curve of prime field or binary field

7) Set START(CRYPTO_ECC_CTL[0]) to 1

8) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared

9) Read NUM(CRPT_ECC_KSSTS[4:0]) to get the number of share key in key store.

Note:

5. To generate the share key with key store support saftly, it is suggested that user should make a new random number instead of the adpted private key when executing ECDH with key store every time.

6. When setting ECDH (CRYPTO_PRNG_KSCTL[19]) to generate a new d_A of ECDH flow by PRNG, the maxinum read times of d_A key is 2 in key store. The first time is for step 2 and the second one is for step 4. Thus, ECC will output the result of point multilplication in X1 and Y1 register or the DMA destination address only when the d_A key is read for the first time. When the d_A key is read second time, the only destination of point multilplication is key store.

7. It is suggested that user can erase the d_A key in key store after generating the share key.

6.36.5.10 RSA (Rivest, Shamir and Adleman) Cryptography Withardwareithout CRT (Chinese Remainder Theorem) and SCA (Side-Channel Attack)

RSA is the first asymmetric cryptosystem (public-key cryptosystems) designed and named by Ron Rivest, Adi Shamir and Leonard Adleman from MIT in 1977. In the past years, many protocols and applications utilize the complexity of the prime factoring problem in the large semi-prime number to build cryptographic systems. The main computation of encryption and description in RSA is modulus

exponentiation operation with very large bit length (as least 2048 bits from NIST's suggestion). To accelerate RSA applications, an RSA accelerator can be used to quickly compute the complex modulus exponentiation operation.

Chinese Remainder Theorem (CRT) is a theorem of number theory, which is widely used for computing with large integers. As it allows replacing a computation for which one knows a bound on the size of the result by several similar computations on small integers. If users want to repeatedly execute decryption with the same private key, they can switch normal RSA decryption mode to Chinese Remainder Theorem mode to improve performance.

Side-channel attack (SCA) is any attack based on information gained from the implementation of a computer system, rather than weaknesses in the implemented algorithm itself. Then, power consumption analysis is a method of side-channel attack most often used. If users want to against simple power analysis (SPA) and differential power analysis (DPA), they can use the side channel attack protection.

Before starting RSA accelerator with non-CRT operation, users must provide the required input data of modulus exponentiation operation include the base of exponentiation (abbreviated to M) in the memory address SADDR0, the base of modulus operation (abbrevated to N) in the memory address SADDR1 and the exponent of exponentiation (abbreviated to E) in the memory address SADDR2. Moreover, users still reserve a memory address DADDR for storing the Intermediate temporary value from RSA accelerator. After finishing non-CRT operation, RSA will store the result of modulus exponentiation operation in the memory address DADDR.

Before starting RSA accelerator with CRT operation, users must provide the required input data of modulus exponentiation operation include the base of exponentiation (abbreviated to M) in the memory address SADDR0, the base of modulus operation (abbrevated to N) in the memory address SADDR1 and the exponent of exponentiation (abbreviated to E) in the memory address SADDR2, and the factors of modulus operation (abbreviated to p and abbreviated to q) in memory address SADDR3 and SADDR4. Moreover, user still reserve seven memory address MADDR0~MADDR5 and DADDR for storing these intermediate temporary value from RSA accelerator. After finishing CRT operation, RSA will store the result of modulus exponentiation operation in the memory address DADDR. If RSA accelerator execute CRT operation with key from key store, key number of these intermediate temporary value (MADDR0~MADDR5) are stored in Key Number2~7 of CRYPTO_RSA_KSSTS0 and CRYPTO_RSA_KSSTS1.

However, if users want to execute side channel attack (SCA) protection and non-CRT mode, users must provide the required input data of modulus exponentiation operation include the base of exponentiation (abbreviated to M) in the memory address SADDR0, the base of modulus operation (abbrevated to N) in the memory address SADDR1 and the exponent of exponentiation (abbreviated to E) in the memory address SADDR2, and the factors of modulus operation (abbreviated to p and abbreviated to q) in memory address SADDR3 and SADDR4. Moreover, user still reserve two memory address DADDR and MADDR6 for storing these intermediate temporary value from RSA accelerator. After finishing non-CRT operation, RSA will store the result of modulus exponentiation operation in the memory address DADDR.

If users want to decrypt repeatedly with the same key, they can execute CRT bypass mode(CRTBYP) after the first time CRT decryption, only need to provide the new required input data M in the memory address SADDR0. But users cannot change the data in the memory address SADDR1~ SADDR4(key, N, Key Number0~1 in CRYPTO_RSA_KSSTS0) and MADDR2~MADDR5(Key Number4~7 in CRYPTO_RSA_KSSTS1). If key and data are stored in key store, users also cannot erase or revoke them. Otherwise, CRT bypass mode will be failed. Moreover, they cannot execute CRTBYP in non-CRT mode.

RSA input data definition: M<N, input data (the base of exponentiation in the memory address SADDR0, abbreviated to M) must be less than modulus (the base of modulus operation in the memory address SADDR1, abbrevated to N).

Users need to make sure that the memory space from SADDR0~SADDR2, MADDR0~MADDR1(Key

Number2~3 in CRYPTO_RSA_KSSTS0), MADDR4~MADDR5(Key Number6~7 in CRYPTO_RSA_KSSTS1) memory address to its address +key length are reserved for RSA accelerator. And the memory space from SADDR3~SADDR4(Key Number0~1 in CRYPTO_RSA_KSSTS0), MADDR2~MADDR3(Key Number4~5 in CRYPTO_RSA_KSSTS1) memory address to its address +half key length are reserved for RSA accelerator. If executing side channel attack(SCA) protection in non-CRT mode, users should make sure that the memory space from MADDR6 to its address +key length+128-bits are reserved for RSA accelerator. (**Note:** The data endianness is little-endian. If the data bit-width is smaller than the keeping memory space, users should write zero to not enough part. For example, if key length is 8-bits and the data just 4-bits(1011), users should write 0000_1011 to the memory.)

After starting RSA accelerator operation, users cannot change input data in the memory address SADDR0~ SADDR4, MADDR0~MADDR6 and DADDR until the operation is finishing. Otherwise, users will get wrong result of modulus exponentiation operation.

Users need to allocate the space which size is 1 × key length (for p, q) in SRAM of key store in advance before executing the SCA protection mode with key store. If users want to execute SCA protection and no-CRT mode, users need to allocate another space which size is 1 × key length (for blind private key) in SRAM of key store, and it should be set un-readable. (If users want to execute RSA with key store, users need to operate these register: CRYPTO_RSA_KSCTL, CRYPTO_RSA_KSSTS0 and CRYPTO_RSA_KSSTS1.)

Users need to allocate the space which size is 6 × key length in SRAM of key store in advance before executing the CRT mode with key store. (middle data only read from / write to the SRAM of key store.)

If users want to execute a new encryption or decryption with RSA accelerator, they must rewrite input data (the base of exponentiation in the memory address SADDR0, abbreviated to M). Otherwise, they may get wrong input message and wrong answer.

To bring the side-channel protection of RSA into full play, it is suggested that user must execute “TRNG generate the true random number seed steps” (please refer to TRNG DTS) at least once before starting the first RSA operation after every CRYPTO or system reset.

The RSA accelerator will execute correctly in the support situation. The support situation as shown in Table 6.36-7. If users use the error combination, CTLERR(CRYPTO_RSA_STS[17]) will be set to 1 (even user unset the START(CRYPTO_RSA_CTL[0])), and RSA accelerator will not start in the unsupported situation.

CRT: Chinese remainder theorem; CRT_BYP: CRT bypass; SCAP: side channel attack protection; xxx = { SCAP, CRT_BYP, CRT }, 1 = enable, 0 = disable. (‘✓’means RSA available, ‘×’ means RSA unavailable and CTLERR(CRYPTO_RSA_STS[17]) will become to ‘1’)

KEY LENGTH	Normal (Xxx=000)	CRT (Xxx=001)	CRT_BYP (Xxx=010)	SCAP (Xxx=100)	CRT + CRT_BYP (Xxx=011)	CRT + SCAP (Xxx=101)	CRT_BYP + SCA (Xxx=110)	CRT + SCAP + CRT_BYP (Xxx=111)
1024	✓	×	×	✓	×	×	×	×
2048	✓	✓	×	✓	✓	✓	×	✓
3072	✓	✓	×	✓	✓	✓	×	✓
4096	✓	✓	×	✓	✓	✓	×	✓

Table 6.36-7 RSA Accelerator Support Situation

RSA DMA Mode Programming Flow

1. Write 1 to RSAIEN(CRYPTO_INTEN [30]) to enable RSA interrupt if needed.
2. Write input data to DMA source address.
3. Program DMA source address to register CRYPTO_RSA_SADDR0~ SADDR4.
4. If executing SCA protection and no-CRT mode, program DMA middle address to register CRYPTO_RSA_MADDR6.
5. Program DMA destination address to register CRYPTO_RSA_DADDR.
6. Configure RSA control register CRYPTO_RSA_CTL for RSA accelerator, such as the start signal of RSA accelerator(START), the stop signal of RSA accelerator(STOP), CRT enable control(CRT), CRT bypass enable control (CRTBYP), side channel attack protection enable control (SCAP), and the key length of RSA(KEYLENG).
7. Wait for the RSA interrupt flag RSAIF(CRYPTO_INTSTS[30]) to be set.
8. Read output data and then clear RSA interrupt flag RSAIF.

RSA CRT DMA Mode Programming Flow

1. Write 1 to RSAIEN(CRYPTO_INTEN [30]) to enable RSA interrupt if needed.
2. Write input data to DMA source address.
3. Program DMA source address to register CRYPTO_RSA_SADDR0~ SADDR4.
4. Program DMA middle address to register CRYPTO_RSA_MADDR0~ MADDR5.
5. Program DMA destination address to register CRYPTO_RSA_DADDR.
6. Configure RSA control register CRYPTO_RSA_CTL for RSA accelerator, such as the start signal of RSA accelerator(START), the stop signal of RSA accelerator(STOP), CRT enable control(CRT), CRT bypass enable control (CRTBYP), side channel attack protection enable control (SCAP), and the key length of RSA(KEYLENG).
7. Wait for the RSA interrupt flag RSAIF(CRYPTO_INTSTS[30]) to be set.
8. Read output data and then clear RSA interrupt flag RSAIF.

RSA DMA Mode with Key Store Programming Flow

1. Write 1 to RSAIEN(CRYPTO_INTEN [30]) to enable RSA interrupt if needed.
2. Write message, N and public key to DMA, and write others data (private key, p, q) to Key Store.
3. If executing SCA protection and no-CRT mode, allocate the space SRAM of Key Store which is for exponent blind key in advance before executing the SCA protection and no-CRT mode with Key Store. It's OWNER (KS_METADATA[18:16]) bits should be set '010' and it also should be set un-readable(KS_METADATA[2] is set to '0').
4. Configure RSA key control register CRYPTO_RSA_KSCTL for Key Store Source (RSSRC), key Source (RSRC), exponent blind key number of key (BKNUM) and key number of key (NUM) in Key Store.
5. Program DMA source address to register CRYPTO_RSA_SADDR0~ SADDR2.
6. Configure RSA key status register CRYPTO_RSA_KSSTS0 for each key number of others data (NUM0~1) in Key Store.
7. Program DMA destination address to register CRYPTO_RSA_DADDR.
8. Configure RSA control register CRYPTO_RSA_CTL for RSA accelerator, such as the start signal of RSA accelerator(START), the stop signal of RSA accelerator(STOP), CRT enable control(CRT), CRT bypass enable control (CRTBYP), side channel attack protection enable

control (SCAP), and the key length of RSA(KEYLENG).

9. Wait for the RSA interrupt flag RSAIF(CRYPTO_INTSTS[30]) to be set.
10. Read output data and then clear RSA interrupt flag RSAIF.

RSA CRT DMA Mode with Key Store Programming Flow

1. Write 1 to RSAIEN(CRYPTO_INTEN [30]) to enable RSA interrupt if needed.
2. Write message, N and public key to DMA, and write others data (private key, p, q) to Key Store.
3. Allocate the space SRAM of Key Store which is for Number2~7 (in CRYPTO_RSA_KSSTS0 and CRYPTO_RSA_KSSTS1) in advance before executing the CRT mode with Key Store. (middle data only read from / write to the SRAM of Key Store)
4. Configure RSA key control register CRYPTO_RSA_KSCTL for Key Store Source (RSSRC), key Source (RSRC) and key number of key (NUM) in Key Store.
5. Program DMA source address to register CRYPTO_RSA_SADDR0~ SADDR2.
6. Configure RSA key status register CRYPTO_RSA_KSSTS0, CRYPTO_RSA_KSSTS1 for each key number of others data (Key Number0~7) in Key Store.
7. Program DMA destination address to register CRYPTO_RSA_DADDR.
8. Configure RSA control register CRYPTO_RSA_CTL for RSA accelerator, such as the start signal of RSA accelerator(START), the stop signal of RSA accelerator(STOP), CRT enable control(CRT), CRT bypass enable control (CRTBYP), side channel attack protection enable control (SCAP), and the key length of RSA(KEYLENG).
9. Wait for the RSA interrupt flag RSAIF(CRYPTO_INTSTS[30]) to be set.
10. Read output data and then clear RSA interrupt flag RSAIF.

Some Notices of RSA Accelerator

11. The key length support of RSA accelerator for both non-CRT encryption and decryption is 1024, 2048, 3072 and 4096 bits. The key length support of RSA accelerator for CRT decryption is 2048, 3072 and 4096 bits.
12. All input and output data must be positive.
13. When RSA engine is active (i.e., BUSY is 1 and DMABUSY (CRYPTO_RSA_STS[1]) is 0), user cannot modify any input and output data registers.
14. If user wants to stop RSA accelerator, please configures the STOP (CRYPTO_RSA_CTL[1]) to 1. Note that: To avoid the transmission error of the next operation, BUSY signal will not be cleared immediately until the action of DMA is done.
15. User must rewrite input data (the base of exponentiation in the memory address SADDR0, abbreviated to M) in every encryption or decryption.
16. If user wants to execute CRT bypass mode(CRTBYP) after the first time CRT decryption mode, they cannot change the data in the memory address SADDR1~ SADDR4(key, N, Key Number0~1 in CRYPTO_RSA_KSSTS0) and MADDR2~MADDR5(Key Number4~7 in CRYPTO_RSA_KSSTS1). If key and data are stored in key store, user also cannot erase or revoke them. Otherwise, CRT bypass mode will be failed.

6.36.6 Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
CRYPTO Base Address: **CRYPTO_BA = 0x4003_2000** **CRYPTO non-secure base address is CRYPTO_BA + 0x1000_0000**				
CRYPTO_INTEN	CRYPTO_BA+0x000	R/W	Crypto Interrupt Enable Control Register	0x0000_0000
CRYPTO_INTSTS	CRYPTO_BA+0x004	R/W	Crypto Interrupt Flag	0x0000_0000
CRYPTO_PRNG_CTL	CRYPTO_BA+0x008	R/W	PRNG Control Register	0x0000_0000
CRYPTO_PRNG_SEED	CRYPTO_BA+0x00C	W	Seed for PRNG	Undefined
CRYPTO_PRNG_KEY0	CRYPTO_BA+0x010	R	PRNG Generated Key0	Undefined
CRYPTO_PRNG_KEY1	CRYPTO_BA+0x014	R	PRNG Generated Key1	Undefined
CRYPTO_PRNG_KEY2	CRYPTO_BA+0x018	R	PRNG Generated Key2	Undefined
CRYPTO_PRNG_KEY3	CRYPTO_BA+0x01C	R	PRNG Generated Key3	Undefined
CRYPTO_PRNG_KEY4	CRYPTO_BA+0x020	R	PRNG Generated Key4	Undefined
CRYPTO_PRNG_KEY5	CRYPTO_BA+0x024	R	PRNG Generated Key5	Undefined
CRYPTO_PRNG_KEY6	CRYPTO_BA+0x028	R	PRNG Generated Key6	Undefined
CRYPTO_PRNG_KEY7	CRYPTO_BA+0x02C	R	PRNG Generated Key7	Undefined
CRYPTO_PRNG_STS	CRYPTO_BA+0x030	R	PRNG Status Register	0x0000_0000
CRYPTO_AES_FDBCK0	CRYPTO_BA+0x050	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000
CRYPTO_AES_FDBCK1	CRYPTO_BA+0x054	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000
CRYPTO_AES_FDBCK2	CRYPTO_BA+0x058	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000
CRYPTO_AES_FDBCK3	CRYPTO_BA+0x05C	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000
CRYPTO_AES_GCM_IVCNT0	CRYPTO_BA+0x080	R/W	AES GCM IV Byte Count Register 0	0x0000_0000
CRYPTO_AES_GCM_IVCNT1	CRYPTO_BA+0x084	R/W	AES GCM IV Byte Count Register 1	0x0000_0000
CRYPTO_AES_GCM_ACNT0	CRYPTO_BA+0x088	R/W	AES GCM A Byte Count Register 0	0x0000_0000
CRYPTO_AES_GCM_ACNT1	CRYPTO_BA+0x08C	R/W	AES GCM A Byte Count Register 1	0x0000_0000
CRYPTO_AES_GCM_PCNT0	CRYPTO_BA+0x090	R/W	AES GCM P Byte Count Register 0	0x0000_0000
CRYPTO_AES_GCM_PCNT1	CRYPTO_BA+0x094	R/W	AES GCM P Byte Count Register 1	0x0000_0000
CRYPTO_AES_FBADDR	CRYPTO_BA+0x0A0	R/W	AES DMA Feedback Address Register	0x0000_0000
CRYPTO_AES_CTL	CRYPTO_BA+0x100	R/W	AES Control Register	0x0000_0000
CRYPTO_AES_STS	CRYPTO_BA+0x104	R	AES Engine Flag	0x0001_0100
CRYPTO_AES_DATIN	CRYPTO_BA+0x108	R/W	AES Engine Data Input Port Register	0x0000_0000

CRYPTO_AES_DATOUT	CRYPTO_BA+0x10C	R	AES Engine Data Output Port Register	0x0000_0000
CRYPTO_AES_KEY0	CRYPTO_BA+0x110	R/W	AES Key Word 0 Register	0x0000_0000
CRYPTO_AES_KEY1	CRYPTO_BA+0x114	R/W	AES Key Word 1 Register	0x0000_0000
CRYPTO_AES_KEY2	CRYPTO_BA+0x118	R/W	AES Key Word 2 Register	0x0000_0000
CRYPTO_AES_KEY3	CRYPTO_BA+0x11C	R/W	AES Key Word 3 Register	0x0000_0000
CRYPTO_AES_KEY4	CRYPTO_BA+0x120	R/W	AES Key Word 4 Register	0x0000_0000
CRYPTO_AES_KEY5	CRYPTO_BA+0x124	R/W	AES Key Word 5 Register	0x0000_0000
CRYPTO_AES_KEY6	CRYPTO_BA+0x128	R/W	AES Key Word 6 Register	0x0000_0000
CRYPTO_AES_KEY7	CRYPTO_BA+0x12C	R/W	AES Key Word 7 Register	0x0000_0000
CRYPTO_AES_IV0	CRYPTO_BA+0x130	R/W	AES Initial Vector Word 0 Register	0x0000_0000
CRYPTO_AES_IV1	CRYPTO_BA+0x134	R/W	AES Initial Vector Word 1 Register	0x0000_0000
CRYPTO_AES_IV2	CRYPTO_BA+0x138	R/W	AES Initial Vector Word 2 Register	0x0000_0000
CRYPTO_AES_IV3	CRYPTO_BA+0x13C	R/W	AES Initial Vector Word 3 Register	0x0000_0000
CRYPTO_AES_SADDR	CRYPTO_BA+0x140	R/W	AES DMA Source Address Register	0x0000_0000
CRYPTO_AES_DADDR	CRYPTO_BA+0x144	R/W	AES DMA Destination Address Register	0x0000_0000
CRYPTO_AES_CNT	CRYPTO_BA+0x148	R/W	AES Byte Count Register	0x0000_0000
CRYPTO_HMAC_CTL	CRYPTO_BA+0x300	R/W	SHA/HMAC Control Register	0x0000_0000
CRYPTO_HMAC_STS	CRYPTO_BA+0x304	R	SHA/HMAC Status Flag	0x0000_0000
CRYPTO_HMAC_DGST0	CRYPTO_BA+0x308	R	SHA/HMAC Output Feedback Data 0	0x0000_0000
CRYPTO_HMAC_DGST1	CRYPTO_BA+0x30C	R	SHA/HMAC Output Feedback Data 1	0x0000_0000
CRYPTO_HMAC_DGST2	CRYPTO_BA+0x310	R	SHA/HMAC Output Feedback Data 2	0x0000_0000
CRYPTO_HMAC_DGST3	CRYPTO_BA+0x314	R	SHA/HMAC Output Feedback Data 3	0x0000_0000
CRYPTO_HMAC_DGST4	CRYPTO_BA+0x318	R	SHA/HMAC Output Feedback Data 4	0x0000_0000
CRYPTO_HMAC_DGST5	CRYPTO_BA+0x31C	R	SHA/HMAC Output Feedback Data 5	0x0000_0000
CRYPTO_HMAC_DGST6	CRYPTO_BA+0x320	R	SHA/HMAC Output Feedback Data 6	0x0000_0000
CRYPTO_HMAC_DGST7	CRYPTO_BA+0x324	R	SHA/HMAC Output Feedback Data 7	0x0000_0000
CRYPTO_HMAC_DGST8	CRYPTO_BA+0x328	R	SHA/HMAC Output Feedback Data 8	0x0000_0000
CRYPTO_HMAC_DGST9	CRYPTO_BA+0x32C	R	SHA/HMAC Output Feedback Data 9	0x0000_0000
CRYPTO_HMAC_DGST10	CRYPTO_BA+0x330	R	SHA/HMAC Output Feedback Data 10	0x0000_0000
CRYPTO_HMAC_DGST11	CRYPTO_BA+0x334	R	SHA/HMAC Output Feedback Data 11	0x0000_0000
CRYPTO_HMAC_DGST12	CRYPTO_BA+0x338	R	SHA/HMAC Output Feedback Data 12	0x0000_0000
CRYPTO_HMAC_DGST13	CRYPTO_BA+0x33C	R	SHA/HMAC Output Feedback Data 13	0x0000_0000
CRYPTO_HMAC_DGST14	CRYPTO_BA+0x340	R	SHA/HMAC Output Feedback Data 14	0x0000_0000
CRYPTO_HMAC_DGST15	CRYPTO_BA+0x344	R	SHA/HMAC Output Feedback Data 15	0x0000_0000

CRYPTO_HMAC_KEYCNT	CRYPTO_BA+0x348	R/W	SHA/HMAC Key Byte Count Register	0x0000_0000
CRYPTO_HMAC_SADDR	CRYPTO_BA+0x34C	R/W	SHA/HMAC DMA Source Address Register	0x0000_0000
CRYPTO_HMAC_DMACNT	CRYPTO_BA+0x350	R/W	SHA/HMAC Byte Count Register	0x0000_0000
CRYPTO_HMAC_DATIN	CRYPTO_BA+0x354	R/W	SHA/HMAC Engine Non-dMA Mode Data Input Port Register	0x0000_0000
CRYPTO_HMAC_FDBCK0	CRYPTO_BA+0x358	R/W	SHA/HMAC Output Feedback Data 0 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK1	CRYPTO_BA+0x35C	R/W	SHA/HMAC Output Feedback Data 1 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK2	CRYPTO_BA+0x360	R/W	SHA/HMAC Output Feedback Data 2 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK3	CRYPTO_BA+0x364	R/W	SHA/HMAC Output Feedback Data 3 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK4	CRYPTO_BA+0x368	R/W	SHA/HMAC Output Feedback Data 4 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK5	CRYPTO_BA+0x36C	R/W	SHA/HMAC Output Feedback Data 5 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK6	CRYPTO_BA+0x370	R/W	SHA/HMAC Output Feedback Data 6 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK7	CRYPTO_BA+0x374	R/W	SHA/HMAC Output Feedback Data 7 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK8	CRYPTO_BA+0x378	R/W	SHA/HMAC Output Feedback Data 8 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK9	CRYPTO_BA+0x37C	R/W	SHA/HMAC Output Feedback Data 9 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK10	CRYPTO_BA+0x380	R/W	SHA/HMAC Output Feedback Data 10 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK11	CRYPTO_BA+0x384	R/W	SHA/HMAC Output Feedback Data 11 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK12	CRYPTO_BA+0x388	R/W	SHA/HMAC Output Feedback Data 12 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK13	CRYPTO_BA+0x38C	R/W	SHA/HMAC Output Feedback Data 13 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK14	CRYPTO_BA+0x390	R/W	SHA/HMAC Output Feedback Data 14 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK15	CRYPTO_BA+0x394	R/W	SHA/HMAC Output Feedback Data 15 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK16	CRYPTO_BA+0x398	R/W	SHA/HMAC Output Feedback Data 16 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK17	CRYPTO_BA+0x39C	R/W	SHA/HMAC Output Feedback Data 17 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK18	CRYPTO_BA+0x3A0	R/W	SHA/HMAC Output Feedback Data 18 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK19	CRYPTO_BA+0x3A4	R/W	SHA/HMAC Output Feedback Data 19 After SHA/HMAC Operation	0x0000_0000

CRYPTO_HMAC_FDBCK20	CRYPTO_BA+0x3A8	R/W	SHA/HMAC Output Feedback Data 20 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK21	CRYPTO_BA+0x3AC	R/W	SHA/HMAC Output Feedback Data 21 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK22	CRYPTO_BA+0x3B0	R/W	SHA/HMAC Output Feedback Data 22 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK23	CRYPTO_BA+0x3B4	R/W	SHA/HMAC Output Feedback Data 23 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK24	CRYPTO_BA+0x3B8	R/W	SHA/HMAC Output Feedback Data 24 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK25	CRYPTO_BA+0x3BC	R/W	SHA/HMAC Output Feedback Data 25 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK26	CRYPTO_BA+0x3C0	R/W	SHA/HMAC Output Feedback Data 26 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK27	CRYPTO_BA+0x3C4	R/W	SHA/HMAC Output Feedback Data 27 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK28	CRYPTO_BA+0x3C8	R/W	SHA/HMAC Output Feedback Data 28 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK29	CRYPTO_BA+0x3CC	R/W	SHA/HMAC Output Feedback Data 29 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK30	CRYPTO_BA+0x3D0	R/W	SHA/HMAC Output Feedback Data 30 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK31	CRYPTO_BA+0x3D4	R/W	SHA/HMAC Output Feedback Data 31 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK32	CRYPTO_BA+0x3D8	R/W	SHA/HMAC Output Feedback Data 32 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK33	CRYPTO_BA+0x3DC	R/W	SHA/HMAC Output Feedback Data 33 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK34	CRYPTO_BA+0x3E0	R/W	SHA/HMAC Output Feedback Data 34 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK35	CRYPTO_BA+0x3E4	R/W	SHA/HMAC Output Feedback Data 35 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK36	CRYPTO_BA+0x3E8	R/W	SHA/HMAC Output Feedback Data 36 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK37	CRYPTO_BA+0x3EC	R/W	SHA/HMAC Output Feedback Data 37 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK38	CRYPTO_BA+0x3F0	R/W	SHA/HMAC Output Feedback Data 38 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK39	CRYPTO_BA+0x3F4	R/W	SHA/HMAC Output Feedback Data 39 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK40	CRYPTO_BA+0x3F8	R/W	SHA/HMAC Output Feedback Data 40 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK41	CRYPTO_BA+0x3FC	R/W	SHA/HMAC Output Feedback Data 41 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK42	CRYPTO_BA+0x400	R/W	SHA/HMAC Output Feedback Data 42 After SHA/HMAC Operation	0x0000_0000

CRYPTO_HMAC_FDBCK43	CRYPTO_BA+0x404	R/W	SHA/HMAC Output Feedback Data 43 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK44	CRYPTO_BA+0x408	R/W	SHA/HMAC Output Feedback Data 44 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK45	CRYPTO_BA+0x40C	R/W	SHA/HMAC Output Feedback Data 45 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK46	CRYPTO_BA+0x410	R/W	SHA/HMAC Output Feedback Data 46 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK47	CRYPTO_BA+0x414	R/W	SHA/HMAC Output Feedback Data 47 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK48	CRYPTO_BA+0x418	R/W	SHA/HMAC Output Feedback Data 48 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK49	CRYPTO_BA+0x41C	R/W	SHA/HMAC Output Feedback Data 49 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK50	CRYPTO_BA+0x420	R/W	SHA/HMAC Output Feedback Data 50 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK51	CRYPTO_BA+0x424	R/W	SHA/HMAC Output Feedback Data 51 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK52	CRYPTO_BA+0x428	R/W	SHA/HMAC Output Feedback Data 52 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK53	CRYPTO_BA+0x42C	R/W	SHA/HMAC Output Feedback Data 53 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FBADDR	CRYPTO_BA+0x4FC	R/W	SHA/HMAC DMA Feedback Address Register	0x0000_0000
CRYPTO_ECC_CTL	CRYPTO_BA+0x800	R/W	ECC Control Register	0x0000_0000
CRYPTO_ECC_STS	CRYPTO_BA+0x804	R	ECC Status Register	0x0000_0000
CRYPTO_ECC_X1_00	CRYPTO_BA+0x808	R/W	ECC the X-coordinate Word0 of the First Point	0x0000_0000
CRYPTO_ECC_X1_01	CRYPTO_BA+0x80C	R/W	ECC the X-coordinate Word1 of the First Point	0x0000_0000
CRYPTO_ECC_X1_02	CRYPTO_BA+0x810	R/W	ECC the X-coordinate Word2 of the First Point	0x0000_0000
CRYPTO_ECC_X1_03	CRYPTO_BA+0x814	R/W	ECC the X-coordinate Word3 of the First Point	0x0000_0000
CRYPTO_ECC_X1_04	CRYPTO_BA+0x818	R/W	ECC the X-coordinate Word4 of the First Point	0x0000_0000
CRYPTO_ECC_X1_05	CRYPTO_BA+0x81C	R/W	ECC the X-coordinate Word5 of the First Point	0x0000_0000
CRYPTO_ECC_X1_06	CRYPTO_BA+0x820	R/W	ECC the X-coordinate Word6 of the First Point	0x0000_0000
CRYPTO_ECC_X1_07	CRYPTO_BA+0x824	R/W	ECC the X-coordinate Word7 of the First Point	0x0000_0000
CRYPTO_ECC_X1_08	CRYPTO_BA+0x828	R/W	ECC the X-coordinate Word8 of the First Point	0x0000_0000
CRYPTO_ECC_X1_09	CRYPTO_BA+0x82C	R/W	ECC the X-coordinate Word9 of the First Point	0x0000_0000
CRYPTO_ECC_X1_10	CRYPTO_BA+0x830	R/W	ECC the X-coordinate Word10 of the First Point	0x0000_0000
CRYPTO_ECC_X1_11	CRYPTO_BA+0x834	R/W	ECC the X-coordinate Word11 of the First Point	0x0000_0000
CRYPTO_ECC_X1_12	CRYPTO_BA+0x838	R/W	ECC the X-coordinate Word12 of the First Point	0x0000_0000
CRYPTO_ECC_X1_13	CRYPTO_BA+0x83C	R/W	ECC the X-coordinate Word13 of the First Point	0x0000_0000
CRYPTO_ECC_X1_14	CRYPTO_BA+0x840	R/W	ECC the X-coordinate Word14 of the First Point	0x0000_0000

CRYPTO_ECC_X1_15	CRYPTO_BA+0x844	R/W	ECC the X-coordinate Word15 of the First Point	0x0000_0000
CRYPTO_ECC_X1_16	CRYPTO_BA+0x848	R/W	ECC the X-coordinate Word16 of the First Point	0x0000_0000
CRYPTO_ECC_X1_17	CRYPTO_BA+0x84C	R/W	ECC the X-coordinate Word17 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_00	CRYPTO_BA+0x850	R/W	ECC the Y-coordinate Word0 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_01	CRYPTO_BA+0x854	R/W	ECC the Y-coordinate Word1 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_02	CRYPTO_BA+0x858	R/W	ECC the Y-coordinate Word2 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_03	CRYPTO_BA+0x85C	R/W	ECC the Y-coordinate Word3 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_04	CRYPTO_BA+0x860	R/W	ECC the Y-coordinate Word4 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_05	CRYPTO_BA+0x864	R/W	ECC the Y-coordinate Word5 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_06	CRYPTO_BA+0x868	R/W	ECC the Y-coordinate Word6 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_07	CRYPTO_BA+0x86C	R/W	ECC the Y-coordinate Word7 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_08	CRYPTO_BA+0x870	R/W	ECC the Y-coordinate Word8 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_09	CRYPTO_BA+0x874	R/W	ECC the Y-coordinate Word9 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_10	CRYPTO_BA+0x878	R/W	ECC the Y-coordinate Word10 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_11	CRYPTO_BA+0x87C	R/W	ECC the Y-coordinate Word11 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_12	CRYPTO_BA+0x880	R/W	ECC the Y-coordinate Word12 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_13	CRYPTO_BA+0x884	R/W	ECC the Y-coordinate Word13 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_14	CRYPTO_BA+0x888	R/W	ECC the Y-coordinate Word14 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_15	CRYPTO_BA+0x88C	R/W	ECC the Y-coordinate Word15 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_16	CRYPTO_BA+0x890	R/W	ECC the Y-coordinate Word16 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_17	CRYPTO_BA+0x894	R/W	ECC the Y-coordinate Word17 of the First Point	0x0000_0000
CRYPTO_ECC_X2_00	CRYPTO_BA+0x898	R/W	ECC the X-coordinate Word0 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_01	CRYPTO_BA+0x89C	R/W	ECC the X-coordinate Word1 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_02	CRYPTO_BA+0x8A0	R/W	ECC the X-coordinate Word2 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_03	CRYPTO_BA+0x8A4	R/W	ECC the X-coordinate Word3 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_04	CRYPTO_BA+0x8A8	R/W	ECC the X-coordinate Word4 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_05	CRYPTO_BA+0x8AC	R/W	ECC the X-coordinate Word5 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_06	CRYPTO_BA+0x8B0	R/W	ECC the X-coordinate Word6 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_07	CRYPTO_BA+0x8B4	R/W	ECC the X-coordinate Word7 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_08	CRYPTO_BA+0x8B8	R/W	ECC the X-coordinate Word8 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_09	CRYPTO_BA+0x8BC	R/W	ECC the X-coordinate Word9 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_10	CRYPTO_BA+0x8C0	R/W	ECC the X-coordinate Word10 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_11	CRYPTO_BA+0x8C4	R/W	ECC the X-coordinate Word11 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_12	CRYPTO_BA+0x8C8	R/W	ECC the X-coordinate Word12 of the Second Point	0x0000_0000

CRYPTO_ECC_X2_13	CRYPTO_BA+0x8CC	R/W	ECC the X-coordinate Word13 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_14	CRYPTO_BA+0x8D0	R/W	ECC the X-coordinate Word14 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_15	CRYPTO_BA+0x8D4	R/W	ECC the X-coordinate Word15 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_16	CRYPTO_BA+0x8D8	R/W	ECC the X-coordinate Word16 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_17	CRYPTO_BA+0x8DC	R/W	ECC the X-coordinate Word17 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_00	CRYPTO_BA+0x8E0	R/W	ECC the Y-coordinate Word0 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_01	CRYPTO_BA+0x8E4	R/W	ECC the Y-coordinate Word1 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_02	CRYPTO_BA+0x8E8	R/W	ECC the Y-coordinate Word2 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_03	CRYPTO_BA+0x8EC	R/W	ECC the Y-coordinate Word3 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_04	CRYPTO_BA+0x8F0	R/W	ECC the Y-coordinate Word4 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_05	CRYPTO_BA+0x8F4	R/W	ECC the Y-coordinate Word5 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_06	CRYPTO_BA+0x8F8	R/W	ECC the Y-coordinate Word6 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_07	CRYPTO_BA+0x8FC	R/W	ECC the Y-coordinate Word7 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_08	CRYPTO_BA+0x900	R/W	ECC the Y-coordinate Word8 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_09	CRYPTO_BA+0x904	R/W	ECC the Y-coordinate Word9 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_10	CRYPTO_BA+0x908	R/W	ECC the Y-coordinate Word10 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_11	CRYPTO_BA+0x90C	R/W	ECC the Y-coordinate Word11 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_12	CRYPTO_BA+0x910	R/W	ECC the Y-coordinate Word12 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_13	CRYPTO_BA+0x914	R/W	ECC the Y-coordinate Word13 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_14	CRYPTO_BA+0x918	R/W	ECC the Y-coordinate Word14 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_15	CRYPTO_BA+0x91C	R/W	ECC the Y-coordinate Word15 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_16	CRYPTO_BA+0x920	R/W	ECC the Y-coordinate Word16 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_17	CRYPTO_BA+0x924	R/W	ECC the Y-coordinate Word17 of the Second Point	0x0000_0000
CRYPTO_ECC_A_00	CRYPTO_BA+0x928	R/W	ECC the Parameter CURVEA Word0 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_01	CRYPTO_BA+0x92C	R/W	ECC the Parameter CURVEA Word1 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_02	CRYPTO_BA+0x930	R/W	ECC the Parameter CURVEA Word2 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_03	CRYPTO_BA+0x934	R/W	ECC the Parameter CURVEA Word3 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_04	CRYPTO_BA+0x938	R/W	ECC the Parameter CURVEA Word4 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_05	CRYPTO_BA+0x93C	R/W	ECC the Parameter CURVEA Word5 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_06	CRYPTO_BA+0x940	R/W	ECC the Parameter CURVEA Word6 of Elliptic Curve	0x0000_0000

CRYPTO_ECC_A_07	CRYPTO_BA+0x944	R/W	ECC the Parameter CURVEA Word7 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_08	CRYPTO_BA+0x948	R/W	ECC the Parameter CURVEA Word8 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_09	CRYPTO_BA+0x94C	R/W	ECC the Parameter CURVEA Word9 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_10	CRYPTO_BA+0x950	R/W	ECC the Parameter CURVEA Word10 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_11	CRYPTO_BA+0x954	R/W	ECC the Parameter CURVEA Word11 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_12	CRYPTO_BA+0x958	R/W	ECC the Parameter CURVEA Word12 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_13	CRYPTO_BA+0x95C	R/W	ECC the Parameter CURVEA Word13 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_14	CRYPTO_BA+0x960	R/W	ECC the Parameter CURVEA Word14 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_15	CRYPTO_BA+0x964	R/W	ECC the Parameter CURVEA Word15 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_16	CRYPTO_BA+0x968	R/W	ECC the Parameter CURVEA Word16 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_17	CRYPTO_BA+0x96C	R/W	ECC the Parameter CURVEA Word17 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_00	CRYPTO_BA+0x970	R/W	ECC the Parameter CURVEB Word0 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_01	CRYPTO_BA+0x974	R/W	ECC the Parameter CURVEB Word1 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_02	CRYPTO_BA+0x978	R/W	ECC the Parameter CURVEB Word2 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_03	CRYPTO_BA+0x97C	R/W	ECC the Parameter CURVEB Word3 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_04	CRYPTO_BA+0x980	R/W	ECC the Parameter CURVEB Word4 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_05	CRYPTO_BA+0x984	R/W	ECC the Parameter CURVEB Word5 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_06	CRYPTO_BA+0x988	R/W	ECC the Parameter CURVEB Word6 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_07	CRYPTO_BA+0x98C	R/W	ECC the Parameter CURVEB Word7 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_08	CRYPTO_BA+0x990	R/W	ECC the Parameter CURVEB Word8 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_09	CRYPTO_BA+0x994	R/W	ECC the Parameter CURVEB Word9 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_10	CRYPTO_BA+0x998	R/W	ECC the Parameter CURVEB Word10 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_11	CRYPTO_BA+0x99C	R/W	ECC the Parameter CURVEB Word11 of Elliptic Curve	0x0000_0000

CRYPTO_ECC_B_12	CRYPTO_BA+0x9A0	R/W	ECC the Parameter CURVEB Word12 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_13	CRYPTO_BA+0x9A4	R/W	ECC the Parameter CURVEB Word13 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_14	CRYPTO_BA+0x9A8	R/W	ECC the Parameter CURVEB Word14 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_15	CRYPTO_BA+0x9AC	R/W	ECC the Parameter CURVEB Word15 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_16	CRYPTO_BA+0x9B0	R/W	ECC the Parameter CURVEB Word16 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_17	CRYPTO_BA+0x9B4	R/W	ECC the Parameter CURVEB Word17 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_00	CRYPTO_BA+0x9B8	R/W	ECC the Parameter CURVEN Word0 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_01	CRYPTO_BA+0x9BC	R/W	ECC the Parameter CURVEN Word1 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_02	CRYPTO_BA+0x9C0	R/W	ECC the Parameter CURVEN Word2 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_03	CRYPTO_BA+0x9C4	R/W	ECC the Parameter CURVEN Word3 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_04	CRYPTO_BA+0x9C8	R/W	ECC the Parameter CURVEN Word4 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_05	CRYPTO_BA+0x9CC	R/W	ECC the Parameter CURVEN Word5 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_06	CRYPTO_BA+0x9D0	R/W	ECC the Parameter CURVEN Word6 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_07	CRYPTO_BA+0x9D4	R/W	ECC the Parameter CURVEN Word7 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_08	CRYPTO_BA+0x9D8	R/W	ECC the Parameter CURVEN Word8 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_09	CRYPTO_BA+0x9DC	R/W	ECC the Parameter CURVEN Word9 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_10	CRYPTO_BA+0x9E0	R/W	ECC the Parameter CURVEN Word10 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_11	CRYPTO_BA+0x9E4	R/W	ECC the Parameter CURVEN Word11 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_12	CRYPTO_BA+0x9E8	R/W	ECC the Parameter CURVEN Word12 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_13	CRYPTO_BA+0x9EC	R/W	ECC the Parameter CURVEN Word13 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_14	CRYPTO_BA+0x9F0	R/W	ECC the Parameter CURVEN Word14 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_15	CRYPTO_BA+0x9F4	R/W	ECC the Parameter CURVEN Word15 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_16	CRYPTO_BA+0x9F8	R/W	ECC the Parameter CURVEN Word16 of Elliptic Curve	0x0000_0000

CRYPTO_ECC_N_17	CRYPTO_BA+0x9FC	R/W	ECC the Parameter CURVEN Word17 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_K_00	CRYPTO_BA+0xA00	W	ECC the Scalar SCALARK Word0 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_01	CRYPTO_BA+0xA04	W	ECC the Scalar SCALARK Word1 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_02	CRYPTO_BA+0xA08	W	ECC the Scalar SCALARK Word2 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_03	CRYPTO_BA+0xA0C	W	ECC the Scalar SCALARK Word3 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_04	CRYPTO_BA+0xA10	W	ECC the Scalar SCALARK Word4 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_05	CRYPTO_BA+0xA14	W	ECC the Scalar SCALARK Word5 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_06	CRYPTO_BA+0xA18	W	ECC the Scalar SCALARK Word6 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_07	CRYPTO_BA+0xA1C	W	ECC the Scalar SCALARK Word7 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_08	CRYPTO_BA+0xA20	W	ECC the Scalar SCALARK Word8 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_09	CRYPTO_BA+0xA24	W	ECC the Scalar SCALARK Word9 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_10	CRYPTO_BA+0xA28	W	ECC the Scalar SCALARK Word10 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_11	CRYPTO_BA+0xA2C	W	ECC the Scalar SCALARK Word11 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_12	CRYPTO_BA+0xA30	W	ECC the Scalar SCALARK Word12 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_13	CRYPTO_BA+0xA34	W	ECC the Scalar SCALARK Word13 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_14	CRYPTO_BA+0xA38	W	ECC the Scalar SCALARK Word14 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_15	CRYPTO_BA+0xA3C	W	ECC the Scalar SCALARK Word15 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_16	CRYPTO_BA+0xA40	W	ECC the Scalar SCALARK Word16 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_17	CRYPTO_BA+0xA44	W	ECC the Scalar SCALARK Word17 of Point Multiplication	0x0000_0000
CRYPTO_ECC_SADDR	CRYPTO_BA+0xA48	R/W	ECC DMA Source Address Register	0x0000_0000
CRYPTO_ECC_DADDR	CRYPTO_BA+0xA4C	R/W	ECC DMA Destination Address Register	0x0000_0000
CRYPTO_ECC_STARTREG	CRYPTO_BA+0xA50	R/W	ECC Starting Address of Updated Registers	0x0000_0000
CRYPTO_ECC_WORDCNT	CRYPTO_BA+0xA54	R/W	ECC DMA Word Count	0x0000_0000
CRYPTO_RSA_CTL	CRYPTO_BA+0xB00	R/W	RSA Control Register	0x0000_0000
CRYPTO_RSA_STS	CRYPTO_BA+0xB04	R	RSA Status Register	0x0000_0000

CRYPTO_RSA_SADDR0	CRYPTO_BA+0xB08	R/W	RSA DMA Source Address Register0	0x0000_0000
CRYPTO_RSA_SADDR1	CRYPTO_BA+0xB0C	R/W	RSA DMA Source Address Register1	0x0000_0000
CRYPTO_RSA_SADDR2	CRYPTO_BA+0xB10	R/W	RSA DMA Source Address Register2	0x0000_0000
CRYPTO_RSA_SADDR3	CRYPTO_BA+0xB14	R/W	RSA DMA Source Address Register3	0x0000_0000
CRYPTO_RSA_SADDR4	CRYPTO_BA+0xB18	R/W	RSA DMA Source Address Register4	0x0000_0000
CRYPTO_RSA_DADDR	CRYPTO_BA+0xB1C	R/W	RSA DMA Destination Address Register	0x0000_0000
CRYPTO_RSA_MADDR0	CRYPTO_BA+0xB20	R/W	RSA DMA Middle Address Register0	0x0000_0000
CRYPTO_RSA_MADDR1	CRYPTO_BA+0xB24	R/W	RSA DMA Middle Address Register1	0x0000_0000
CRYPTO_RSA_MADDR2	CRYPTO_BA+0xB28	R/W	RSA DMA Middle Address Register2	0x0000_0000
CRYPTO_RSA_MADDR3	CRYPTO_BA+0xB2C	R/W	RSA DMA Middle Address Register3	0x0000_0000
CRYPTO_RSA_MADDR4	CRYPTO_BA+0xB30	R/W	RSA DMA Middle Address Register4	0x0000_0000
CRYPTO_RSA_MADDR5	CRYPTO_BA+0xB34	R/W	RSA DMA Middle Address Register5	0x0000_0000
CRYPTO_RSA_MADDR6	CRYPTO_BA+0xB38	R/W	RSA DMA Middle Address Register6	0x0000_0000
CRYPTO_PRNG_KSCTL	CRYPTO_BA+0xF00	W	PRNG Key Control Register	0x0000_0000
CRYPTO_PRNG_KSSTS	CRYPTO_BA+0xF04	R	PRNG Key Status Register	0x0000_0000
CRYPTO_AES_KSCTL	CRYPTO_BA+0xF10	W	AES Key Control Register	0x0000_0000
CRYPTO_HMAC_KSCTL	CRYPTO_BA+0xF30	W	HMAC Key Control Register	0x0000_0000
CRYPTO_ECC_KSCTL	CRYPTO_BA+0xF40	W	ECC Key Control Register	0x0000_0000
CRYPTO_ECC_KSSTS	CRYPTO_BA+0xF44	R	ECC Key Status Register	0x0000_0000
CRYPTO_ECC_KSXY	CRYPTO_BA+0xF48	W	ECC XY Number Register	0x0000_0000
CRYPTO_RSA_KSCTL	CRYPTO_BA+0xF50	W	RSA Key Control Register	0x0000_0000
CRYPTO_RSA_KSSTS0	CRYPTO_BA+0xF54	R/W	RSA Key Status Register 0	0x0000_0000
CRYPTO_RSA_KSSTS1	CRYPTO_BA+0xF58	R/W	RSA Key Status Register 1	0x0000_0000

6.36.7 Register Description

6.36.7.1 Crypto Register

CRYPTO Interrupt Enable Control Register (CRYPTO_INTEN)

Register	Offset	R/W	Description	Reset Value
CRYPTO_INTEN	CRYPTO_BA+0x000	R/W	Crypto Interrupt Enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
RSAEIEN	RSAIEN	Reserved				HMACEIEN	HMACIEN
23	22	21	20	19	18	17	16
ECCEIEN	ECCIEN	Reserved				PRNGEIEN	PRNGIEN
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						AESEIEN	AESIEN

Bits	Description	
[31]	RSAEIEN	**RSA Error Interrupt Enable Bit** 0 = RSA error interrupt flag Disabled. 1 = RSA error interrupt flag Enabled.
[30]	RSAIEN	**RSA Interrupt Enable Bit** 0 = RSA interrupt Disabled. 1 = RSA interrupt Enabled.
[29:26]	Reserved	Reserved.
[25]	HMACEIEN	**SHA/HMAC Error Interrupt Enable Bit** 0 = SHA/HMAC error interrupt flag Disabled. 1 = HMAC error interrupt flag Enabled.
[24]	HMACIEN	**SHA/HMAC Interrupt Enable Bit** 0 = SHA/HMAC interrupt Disabled. 1 = SHA/HMAC interrupt Enabled. **Note:** In DMA mode, an interrupt will be triggered when amount of data set in HMAC_DMA_CNT is fed into the SHA/HMAC engine. In Non-DMA mode, an interrupt will be triggered when the SHA/HMAC engine finishes the operation.
[23]	ECCEIEN	**ECC Error Interrupt Enable Bit** 0 = ECC error interrupt flag Disabled. 1 = ECC error interrupt flag Enabled.

[22]	**ECCIEN**	**ECC Interrupt Enable Bit** 0 = ECC interrupt Disabled. 1 = ECC interrupt Enabled. **Note:** In DMA mode, an interrupt will be triggered when amount of data set in ECC_DMA_CNT is fed into the ECC engine. In Non-DMA mode, an interrupt will be triggered when the ECC engine finishes the operation.
[21:18]	**Reserved**	Reserved.
[17]	**PRNGEIEN**	**PRNG Error Flag Enable Bit** 0 = PRNG error interrupt flag Disabled. 1 = PRNG error interrupt flag Enabled.
[16]	**PRNGIEN**	**PRNG Interrupt Enable Bit** 0 = PRNG interrupt Disabled. 1 = PRNG interrupt Enabled.
[15:2]	**Reserved**	Reserved.
[1]	**AESEIEN**	**AES Error Flag Enable Bit** 0 = AES error interrupt flag Disabled. 1 = AES error interrupt flag Enabled.
[0]	**AESIEN**	**AES Interrupt Enable Bit** 0 = AES interrupt Disabled. 1 = AES interrupt Enabled. **Note:** In DMA mode, an interrupt will be triggered when amount of data set in AES_DMA_CNT is fed into the AES engine. In Non-DMA mode, an interrupt will be triggered when the AES engine finishes the operation.

CRYPTO Interrupt Flag Register (CRYPTO_INTSTS)

Register	Offset	R/W	Description	Reset Value
CRYPTO_INTSTS	CRYPTO_BA+0x004	R/W	Crypto Interrupt Flag	0x0000_0000

31	30	29	28	27	26	25	24
RSAEIF	RSAIF	Reserved				HMACEIF	HMACIF
23	22	21	20	19	18	17	16
ECCEIF	ECCIF	Reserved				PRNGEIF	PRNGIF
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						AESEIF	AESIF

Bits	Description	
[31]	**RSAEIF**	**RSA Error Interrupt Flag** This register includes operating and setting error. The detail flag is shown in CRYPTO_RSA_STS register. This bit is cleared by writing 1, and it has no effect by writing 0. 0 = No RSA error. 1 = RSA error interrupt.
[30]	**RSAIF**	**RSA Finish Interrupt Flag** This bit is cleared by writing 1, and it has no effect by writing 0. 0 = No RSA interrupt. 1 = RSA operation done interrupt.
[29:26]	**Reserved**	Reserved.
[25]	**HMACEIF**	**SHA/HMAC Error Flag** This register includes operating and setting error. The detail flag is shown in CRYPTO_HMAC_STS register. 0 = No SHA/HMAC error. 1 = SHA/HMAC error interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.
[24]	**HMACIF**	**SHA/HMAC Finish Interrupt Flag** 0 = No SHA/HMAC interrupt. 1 = SHA/HMAC operation done interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.
[23]	**ECCEIF**	**ECC Error Flag** This register includes operating and setting error. The detail flag is shown in CRYPTO_ECC_STS register. 0 = No ECC error. 1 = ECC error interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.

[22]	**ECCIF**	**ECC Finish Interrupt Flag** 0 = No ECC interrupt. 1 = ECC operation done interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.
[21:18]	**Reserved**	Reserved.
[17]	**PRNGEIF**	**PRNG Error Flag** 0 = No PRNG error. 1 = PRNG key generation error interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.
[16]	**PRNGIF**	**PRNG Finish Interrupt Flag** 0 = No PRNG interrupt. 1 = PRNG key generation done interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.
[15:2]	**Reserved**	Reserved.
[1]	**AESEIF**	**AES Error Flag** 0 = No AES error. 1 = AES encryption/decryption error interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.
[0]	**AESIF**	**AES Finish Interrupt Flag** 0 = No AES interrupt. 1 = AES encryption/decryption done interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.

6.36.7.2 PRNG Register

PRNG Control Register (CRYPTO_PRNG_CTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_PRNG_CTL	CRYPTO_BA+0x008	R/W	PRNG Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							BUSY
7	6	5	4	3	2	1	0
SEEDSRC	SEEDSEL	KEYSZ				SEEDRLD	START

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**BUSY**	**PRNG Busy (Read Only)** 0 = PRNG engine is idle. 1 = Indicate that the PRNG engine is generating CRYPTO_PRNG_KEYx.
[7]	**SEEDSRC**	**Seed Source (Read Only)** 0 = Seed is from PRNG. 1 = Seed is from TRNG. (not from CRYPTO_PRNG_SEED) **Note:** This bit is cleared to '0' when SEEDSEL is 0.
[6]	**SEEDSEL**	**Seed Select** This bit can be set to 1 only after SEEDRDY (TRNG_CTL[9]) bit become to 1. 0 = Select the seed which is from PRNG. 1 = Select the seed which is from TRNG (not from CRYPTO_PRNG_SEED).

[5:2]	**KEYSZ**	**PRNG Generate Key Size** 0000 = 128 bits. 0001 = 163 bits. 0010 = 192 bits. 0011 = 224 bits. 0100 = 233 bits. 0101 = 255 bits. 0110 = 256 bits. 0111 = 283 bits (only for KS). 1000 = 384 bits (only for KS). 1001 = 409 bits (only for KS). 1010 = 512 bits (only for KS). 1011 = 521 bits (only for KS). 1100 = 571 bits (only for KS). 1101 = Reserved. 1110 = Reserved. 1111 = Reserved. **Note:** 283~571 bits are only generated for Key Store.
[1]	**SEEDRLD**	**Reload New Seed for PRNG Engine** 0 = Generating key based on the current seed. 1 = Reload new seed.
[0]	**START**	**Start PRNG Engine** 0 = Stop PRNG engine. 1 = Generate new key and store the new key to register CRYPTO_PRNG_KEYx, which will be cleared when the new key is generated.

PRNG Seed Register (CRYPTO_PRNG_SEED)

Register	Offset	R/W	Description	Reset Value
CRYPTO_PRNG_SEED	CRYPTO_BA+0x00C	W	Seed for PRNG	Undefined

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

Bits	Description	
[31:0]	SEED	**Seed for PRNG (Write Only)** The bits store the seed for PRNG engine. **Note:** In TRNG+PRNG mode, the seed is from TRNG engine, and it will not be stored in this register.

PRNG Key x Register (CRYPTO_PRNG_KEYx)

Register	Offset	R/W	Description	Reset Value
CRYPTO_PRNG_KEY0	CRYPTO_BA+0x010	R	PRNG Generated Key0	Undefined
CRYPTO_PRNG_KEY1	CRYPTO_BA+0x014	R	PRNG Generated Key1	Undefined
CRYPTO_PRNG_KEY2	CRYPTO_BA+0x018	R	PRNG Generated Key2	Undefined
CRYPTO_PRNG_KEY3	CRYPTO_BA+0x01C	R	PRNG Generated Key3	Undefined
CRYPTO_PRNG_KEY4	CRYPTO_BA+0x020	R	PRNG Generated Key4	Undefined
CRYPTO_PRNG_KEY5	CRYPTO_BA+0x024	R	PRNG Generated Key5	Undefined
CRYPTO_PRNG_KEY6	CRYPTO_BA+0x028	R	PRNG Generated Key6	Undefined
CRYPTO_PRNG_KEY7	CRYPTO_BA+0x02C	R	PRNG Generated Key7	Undefined

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

Bits	Description	
[31:0]	**KEY**	**Store PRNG Generated Key (Read Only)** The bits store the key that is generated by PRNG.

PRNG Status Register (CRYPTO_PRNG_STS)

Register	Offset	R/W	Description	Reset Value
CRYPTO_PRNG_STS	CRYPTO_BA+0x030	R	PRNG Status Register	0x0000_0000

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

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**KSERR**	**PRNG Access Key Store Error Flag** 0 = No error. 1 = Access key store fail.
[16]	**KCTLERR**	**PRNG Key Control Register Error Flag** 0 = No error. 1 = PRNG key control error. When PRNG execute ECDSA or ECDH, but PRNG seed not from TRNG or key is not written to the SRAM of key store (WSDST, CRYPTO_PRNG_KSCTL[23:22] is not equal to '00').
[15:1]	**Reserved**	Reserved.
[0]	**BUSY**	**PRNG Busy Flag** 0 = PRNG engine is idle. 1 = Indicate that the PRNG engine is generating CRYPTO_PRNG_KEYx.

6.36.7.3 AES Register

AES Feedback x Register (CRYPTO_AES_FDBCKx)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_FDBCK0	CRYPTO_BA+0x050	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000
CRYPTO_AES_FDBCK1	CRYPTO_BA+0x054	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000
CRYPTO_AES_FDBCK2	CRYPTO_BA+0x058	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000
CRYPTO_AES_FDBCK3	CRYPTO_BA+0x05C	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000

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

Bits	Description	
[31:0]	**FDBCK**	**AES Feedback Information** The feedback value is 128 bits in size. The AES engine uses the data from CRYPTO_AES_FDBCKx as the data inputted to CRYPTO_AES_IVx for the next block in DMA cascade mode. The AES engine outputs feedback information for IV in the next block's operation. Software can use this feedback information to implement more than four DMA channels. Software can store that feedback value temporarily. After switching back, fill the stored feedback value to CRYPTO_AES_IVx in the same channel operation, and then continue the operation with the original setting.

AES GCM IV Byte Count Register 0 (CRYPTO_AES_GCM_IVCNT0)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_GCM_IVCNT0	CRYPTO_BA+0x080	R/W	AES GCM IV Byte Count Register 0	0x0000_0000

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

Bits	Description	
[31:0]	CNT	**AES GCM IV Byte Count** The bit length of IV is 64 bits for AES GCM mode. The CRYPTO_AES_GCM_IVCNT0 keeps the low weight byte count of initial vector (i.e., len(IV)[34:3]) of AES GCM mode and can be read and written.

AES GCM IV Byte Count Register 1 (CRYPTO_AES_GCM_IVCNT1)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_GCM_IVCNT1	CRYPTO_BA+0x084	R/W	AES GCM IV Byte Count Register 1	0x0000_0000

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

Bits	Description	
[31:29]	Reserved	Reserved.
[28:0]	CNT	**AES GCM IV Byte Count** The bit length of IV is 64 bits for AES GCM mode. The CRYPTO_AES_GCM_IVCNT1 keeps the high weight byte count of initial vector (i.e., len(IV)[64:35]) of AES GCM mode and can be read and written.

AES GCM A Byte Count Register 0 (CRYPTO_AES_GCM_ACNT0)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_GCM_ACNT0	CRYPTO_BA+0x088	R/W	AES GCM A Byte Count Register 0	0x0000_0000

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

Bits	Description	
[31:0]	CNT	**AES GCM a Byte Count** The bit length of A is 64 bits for AES GCM mode. The CRYPTO_AES_GCM_ACNT0 keeps the low weight byte count of the additional authenticated data (i.e., len(A)[34:3]) of AES GCM mode and can be read and written.

AES GCM A Byte Count Register 1 (CRYPTO_AES_GCM_ACNT1)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_GCM_ACNT1	CRYPTO_BA+0x08C	R/W	AES GCM A Byte Count Register 1	0x0000_0000

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

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:0]	**CNT**	**AES GCM a Byte Count** The bit length of A is 64 bits for AES GCM mode. The CRYPTO_AES_GCM_ACNT0 keeps the high weight byte count of the additional authenticated data (i.e., len(A)[63:35]) of AES GCM mode and can be read and written.

AES GCM P Byte Count Register 0 (CRYPTO_AES_GCM_PCNT0)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_GCM_PCNT0	CRYPTO_BA+0x090	R/W	AES GCM P Byte Count Register 0	0x0000_0000

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

Bits	Description	
[31:0]	CNT	**AES GCM P Byte Count** The bit length of Por C is 39 bits for AES GCM mode. The CRYPTO_AES_GCM_PCNT0 keeps the low weight byte count of the plaintext or ciphertext (i.e., len(P)[34:3] or len(C)[34:3]) of AES GCM mode and can be read and written.

AES GCM P Byte Count Register 1 (CRYPTO_AES_GCM_PCNT1)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_GCM_PCNT1	CRYPTO_BA+0x094	R/W	AES GCM P Byte Count Register 1	0x0000_0000

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

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:0]	**CNT**	**AES GCM P Byte Count** The bit length of Por C is 39 bits for AES GCM mode. The CRYPTO_AES_GCM_PCNT1 keeps the high weight byte count of the plaintext or ciphertext (i.e., len(P)[38:35] or len(C)[38:35]) of AES GCM mode and can be read and written. The bit length of Por C is 64 bits for AES CCM mode. The CRYPTO_AES_GCM_PCNT1 keeps the high weight byte count of the plaintext or ciphertext (i.e., len(P)[63:35] or len(C)[63:35]) of AES CCM mode and can be read and written.

AES DMA Feedback Address Register (CRYPTO_AES_FBADDR)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_FBADDR	CRYPTO_BA+0x0A0	R/W	AES DMA Feedback Address Register	0x0000_0000

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

Bits	Description	
[31:0]	FBADDR	**AES DMA Feedback Address** In DMA cascade mode, software can update DMA feedback address register for automatically reading and writing feedback values via DMA.The FBADDR keeps the feedback address of the feedback data for the next cascade operation. Based on the feedback address, the AES accelerator can read thefeedback data of the last cascade opeation from SRAM memory space and write thefeedback data of the current cascade opeation to SRAM memory space. The start of feedback address should be located at word boundary. In other words, bit 1 and 0 of FBADDR are ignored. FBADDR can be read and written. In DMA mode, software can update the next CRYPTO_AES_FBADDR before triggering START.

AES Control Register (CRYPTO_AES_CTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_CTL	CRYPTO_BA+0x100	R/W	AES Control Register	0x0000_0000

31	30	29	28	27	26	25	24
KEYPRT	KEYUNPRT					KINSWAP	KOUTSWAP
23	22	21	20	19	18	17	16
INSWAP	OUTSWAP	FBOUT	FBIN	Reserved		SM4EN	ENCRYPTO
15	14	13	12	11	10	9	8
OPMODE							
7	6	5	4	3	2	1	0
DMAEN	DMACSCAD	DMALAST	Reserved	KEYSZ		STOP	START

Bits	Description	
[31]	**KEYPRT**	**Protect Key** Read as a flag to reflect KEYPRT. 0 = No effect. 1 = Protect the content of the AES key from reading. The return value for reading CRYPTO_AES_KEYx is not the content of the registers CRYPTO_AES_KEYx. Once it is set, it can be cleared by asserting KEYUNPRT, and the key content would be cleared as well.
[30:26]	**KEYUNPRT**	**Unprotect Key** Writing 0 to CRYPTO_AES_CTL[31] and “10110” to CRYPTO_AES_CTL[30:26] is to unprotect the AES key. The KEYUNPRT can be read and written. When it is written as the AES engine is operating, BUSY flag is 1, there would be no effect on KEYUNPRT.
[25]	**KINSWAP**	**AES Engine Input Key and Initial Vector Swap** 0 = Keep the original order. 1 = The order that CPU feeds key and initial vector to the accelerator will be changed from {byte3, byte2, byte1, byte0} to {byte0, byte1, byte2, byte3}.
[24]	**KOUTSWAP**	**AES Engine Output Key, Initial Vector and Feedback Swap** 0 = Keep the original order. 1 = The order that CPU reads key, initial vector and feeback from the accelerator will be changed from {byte3, byte2, byte1, byte0} to {byte0, byte1, byte2, byte3}.
[23]	**INSWAP**	**AES Engine Input Data Swap** 0 = Keep the original order. 1 = The order that CPU feeds data to the accelerator will be changed from {byte3, byte2, byte1, byte0} to {byte0, byte1, byte2, byte3}.
[22]	**OUTSWAP**	**AES Engine Output Data Swap** 0 = Keep the original order. 1 = The order that CPU reads data from the accelerator will be changed from {byte3, byte2, byte1, byte0} to {byte0, byte1, byte2, byte3}.

[21]	**FBOUT**	**Feedback Output From AES Via DMA Automatically** 0 = Disable DMA automatical feedback output function. 1 = Enable DMA automatical feedback output function when DMAEN = 1.
[20]	**FBIN**	**Feedback Input to AES Via DMA Automatically** 0 = Disable DMA automatical feedback input function. 1 = Enable DMA automatical feedback input function. when DMAEN = 1.
[19:18]	**Reserved**	Reserved.
[17]	**SM4EN**	**SM4 Engine Enable** 0 = Enable AES engine. 1 = Enable SM4 engine.
[16]	**ENCRYPTO**	**AES Encryption/Decryption** 0 = AES engine executes decryption operation. 1 = AES engine executes encryption operation.
[15:8]	**OPMODE**	**AES Engine Operation Modes** 0x00 = ECB (Electronic Codebook Mode) 0x01 = CBC (Cipher Block Chaining Mode). 0x02 = CFB (Cipher Feedback Mode). 0x03 = OFB (Output Feedback Mode). 0x04 = CTR (Counter Mode). 0x10 = CBC-CS1 (CBC Ciphertext-Stealing 1 Mode). 0x11 = CBC-CS2 (CBC Ciphertext-Stealing 2 Mode). 0x12 = CBC-CS3 (CBC Ciphertext-Stealing 3 Mode). 0x20 = GCM (Galois/Counter Mode). 0x21 = GHASH (Galois Hash Function). 0x22 = CCM (Counter with CBC-MAC Mode).
[7]	**DMAEN**	**AES Engine DMA Enable Bit** 0 = AES DMA engine Disabled. The AES engine operates in Non-DMA mode. The data need to be written in CRYPTO_AES_DATIN. 1 = AES_DMA engine Enabled. The AES engine operates in DMA mode, and data movement from/to the engine is done by DMA logic.
[6]	**DMACSCAD**	**AES Engine DMA with Cascade Mode** 0 = DMA cascade function Disabled. 1 = In DMA cascade mode, software can update DMA source address register, destination address register, and byte count register during a cascade operation, without finishing the accelerator operation.
[5]	**DMALAST**	**AES Last Block** In DMA mode, this bit must be set as beginning the last DMA cascade round. In Non-DMA mode, this bit must be set when feeding in the last block of data in ECB, CBC, CTR, OFB, and CFB mode, and feeding in the (last-1) block of data at CBC-CS1, CBC-CS2, and CBC-CS3 mode. This bit is always 0 when it is read back. Must be written again once START is triggered.
[4]	**Reserved**	Reserved.

[3:2]	**KEYSZ**	**AES Key Size** This bit defines three different key size for AES operation. 2'b00 = 128 bits key. 2'b01 = 192 bits key. 2'b10 = 256 bits key. 2'b11 = Reserved. If the AES accelerator is operating and the corresponding flag BUSY is 1, updating this register has no effect. **Note:** When SM4EN=1, the key size of AES must be 128.
[1]	**STOP**	**AES Engine Stop** 0 = No effect. 1 = Stop AES engine. **Note:** This bit is always 0 when it is read back.
[0]	**START**	**AES Engine Start** 0 = No effect. 1 = Start AES engine. BUSY flag will be set. **Note:** This bit is always 0 when it is read back.

AES Status Flag Register (CRYPTO_AES_STS)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_STS	CRYPTO_BA+0x104	R	AES Engine Flag	0x0001_0100

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		KSERR	BUSERR	Reserved	OUTBUFERR	OUTBUFFULL	OUTBUFEMPTY
15	14	13	12	11	10	9	8
Reserved			CNTERR	Reserved	INBUFERR	INBUFFULL	INBUFEMPTY
7	6	5	4	3	2	1	0
Reserved							BUSY

Bits	Description	
[31:22]	**Reserved**	Reserved.
[21]	**KSERR**	**AES Engine Access Key Store Error Flag** 0 = No error. 1 = Access error will stop AES engine.
[20]	**BUSERR**	**AES DMA Access Bus Error Flag** 0 = No error. 1 = Bus error will stop DMA operation and AES engine.
[19]	**Reserved**	Reserved.
[18]	**OUTBUFERR**	**AES Out Buffer Error Flag** 0 = No error. 1 = Error happens when getting the result from AES engine.
[17]	**OUTBUFFULL**	**AES Out Buffer Full Flag** 0 = AES output buffer is not full. 1 = AES output buffer is full, and software needs to get data from CRYPTO_AES_DATOUT. Otherwise, the AES engine will be pending since the output buffer is full.
[16]	**OUTBUFEMPTY**	**AES Out Buffer Empty** 0 = AES output buffer is not empty. There are some valid data kept in output buffer. 1 = AES output buffer is empty. Software cannot get data from CRYPTO_AES_DATOUT. Otherwise, the flag OUTBUFERR will be set to 1 since the output buffer is empty.
[15:13]	**Reserved**	Reserved.
[12]	**CNTERR**	**CRYPTO_AES_CNT Setting Error** 0 = No error in CRYPTO_AES_CNT setting. 1 = CRYPTO_AES_CNT is 0 if DMAEN (CRYPTO_AES_CTL[7]) is enabled.
[11]	**Reserved**	Reserved.

[10]	**INBUFERR**	**AES Input Buffer Error Flag** 0 = No error. 1 = Error happens during feeding data to the AES engine.
[9]	**INBUFFULL**	**AES Input Buffer Full Flag** 0 = AES input buffer is not full. Software can feed the data into the AES engine. 1 = AES input buffer is full. Software cannot feed data to the AES engine. Otherwise, the flag INBUFERR will be set to 1.
[8]	**INBUFEMPTY**	**AES Input Buffer Empty** 0 = There are some data in input buffer waiting for the AES engine to process. 1 = AES input buffer is empty. Software needs to feed data to the AES engine. Otherwise, the AES engine will be pending to wait for input data.
[7:1]	**Reserved**	Reserved.
[0]	**BUSY**	**AES Engine Busy** 0 = The AES engine is idle or finished. 1 = The AES engine is under processing.

AES Data Input Port Register (CRYPTO_AES_DATIN)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_DATIN	CRYPTO_BA+0x108	R/W	AES Engine Data Input Port Register	0x0000_0000

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

Bits	Description	
[31:0]	DATIN	**AES Engine Input Port** CPU feeds data to AES engine through this port by checking CRYPTO_AES_STS. Feed data as INBUFFULL is 0.

AES Data Output Port Register (CRYPTO_AES_DATOUT)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_DATOUT	CRYPTO_BA+0x10C	R	AES Engine Data Output Port Register	0x0000_0000

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

Bits	Description	
[31:0]	DATOUT	**AES Engine Output Port** CPU gets results from the AES engine through this port by checking CRYPTO_AES_STS. Get data as OUTBUFEMPTY is 0.

AES Key Word x Register (CRYPTO_AES_KEYx)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_KEY0	CRYPTO_BA+0x110	R/W	AES Key Word 0 Register	0x0000_0000
CRYPTO_AES_KEY1	CRYPTO_BA+0x114	R/W	AES Key Word 1 Register	0x0000_0000
CRYPTO_AES_KEY2	CRYPTO_BA+0x118	R/W	AES Key Word 2 Register	0x0000_0000
CRYPTO_AES_KEY3	CRYPTO_BA+0x11C	R/W	AES Key Word 3 Register	0x0000_0000
CRYPTO_AES_KEY4	CRYPTO_BA+0x120	R/W	AES Key Word 4 Register	0x0000_0000
CRYPTO_AES_KEY5	CRYPTO_BA+0x124	R/W	AES Key Word 5 Register	0x0000_0000
CRYPTO_AES_KEY6	CRYPTO_BA+0x128	R/W	AES Key Word 6 Register	0x0000_0000
CRYPTO_AES_KEY7	CRYPTO_BA+0x12C	R/W	AES Key Word 7 Register	0x0000_0000

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

Bits	Description	
[31:0]	**KEY**	**CRYPTO_AES_KEYx** The KEY keeps the security key for AES operation. x = 0, 1..7. The security key for AES accelerator can be 128, 192, or 256 bits and four, six, or eight 32-bit registers are to store each security key. {CRYPTO_AES_KEY3, CRYPTO_AES_KEY2, CRYPTO_AES_KEY1, CRYPTO_AES_KEY0} stores the 128-bit security key for AES operation. {CRYPTO_AES_KEY5, CRYPTO_AES_KEY4, CRYPTO_AES_KEY3, CRYPTO_AES_KEY2, CRYPTO_AES_KEY1, CRYPTO_AES_KEY0} stores the 192-bit security key for AES operation. {CRYPTO_AES_KEY7, CRYPTO_AES_KEY6, CRYPTO_AES_KEY5, CRYPTO_AES_KEY4, CRYPTO_AES_KEY3, CRYPTO_AES_KEY2, CRYPTO_AES_KEY1, CRYPTO_AES_KEY0} stores the 256-bit security key for AES operation.

AES Initial Vector Word x Register (CRYPTO_AES_IVx)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_IV0	CRYPTO_BA+0x130	R/W	AES Initial Vector Word 0 Register	0x0000_0000
CRYPTO_AES_IV1	CRYPTO_BA+0x134	R/W	AES Initial Vector Word 1 Register	0x0000_0000
CRYPTO_AES_IV2	CRYPTO_BA+0x138	R/W	AES Initial Vector Word 2 Register	0x0000_0000
CRYPTO_AES_IV3	CRYPTO_BA+0x13C	R/W	AES Initial Vector Word 3 Register	0x0000_0000

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

Bits		Description
[31:0]	**IV**	**AES Initial Vectors** x = 0, 1..3. Four initial vectors (CRYPTO_AES_IV0, CRYPTO_AES_IV1, CRYPTO_AES_IV2, and CRYPTO_AES_IV3) are for AES operating in CBC, CFB, and OFB mode. Four registers (CRYPTO_AES_IV0, CRYPTO_AES_IV1, CRYPTO_AES_IV2, and CRYPTO_AES_IV3) act as Nonce counter when the AES engine is operating in CTR mode.

AES DMA Source Address Register (CRYPTO_AES_SADDR)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_SADDR	CRYPTO_BA+0x140	R/W	AES DMA Source Address Register	0x0000_0000

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

Bits	Description	
[31:0]	SADDR	**AES DMA Source Address** The AES accelerator supports DMA function to transfer the plain text between SRAM memory space and embedded FIFO. The SADDR keeps the source address of the data buffer where the source text is stored. Based on the source address, the AES accelerator can read the plain text (encryption) / cipher text (descryption) from SRAM memory space and do AES operation. The start of source address should be located at word boundary. In other words, bit 1 and 0 of SADDR are ignored. SADDR can be read and written. Writing to SADDR while the AES accelerator is operating does not affect the current AES operation. But the value of SADDR will be updated later on. Consequently, software can prepare the DMA source address for the next AES operation. In DMA mode, software can update the next CRYPTO_AES_SADDR before triggering START. The value of CRYPTO_AES_SADDR and CRYPTO_AES_DADDR can be the same.

AES DMA Destination Address Register (CRYPTO_AES_DADDR)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_DADDR	CRYPTO_BA+0x144	R/W	AES DMA Destination Address Register	0x0000_0000

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

Bits	Description	
[31:0]	DADDR	**AES DMA Destination Address** The AES accelerator supports DMA function to transfer the cipher text between SRAM memory space and embedded FIFO. The DADDR keeps the destination address of the data buffer where the engine output's text will be stored. Based on the destination address, the AES accelerator can write the cipher text (encryption) / plain text (decryption) back to SRAM memory space after the AES operation is finished. The start of destination address should be located at word boundary. In other words, bit 1 and 0 of DADDR are ignored. DADDR can be read and written. Writing to DADDR while the AES accelerator is operating does not affect the current AES operation. But the value of DADDR will be updated later on. Consequently, software can prepare the destination address for the next AES operation. In DMA mode, software can update the next CRYPTO_AES_DADDR before triggering START. The value of CRYPTO_AES_SADDR and CRYPTO_AES_DADDR can be the same.

AES Byte Count Register (CRYPTO_AES_CNT)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_CNT	CRYPTO_BA+0x148	R/W	AES Byte Count Register	0x0000_0000

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

Bits	Description	
[31:0]	CNT	**AES Byte Count** The CRYPTO_AES_CNT keeps the byte count of source text that is for the AES engine operating in DMA mode. The CRYPTO_AES_CNT is 32-bit and the maximum of byte count is 4G bytes. CRYPTO_AES_CNT can be read and written. Writing to CRYPTO_AES_CNT while the AES accelerator is operating does not affect the current AES operation. But the value of CRYPTO_AES_CNT will be updated later on. Consequently, software can prepare the byte count of data for the next AES operation. According to CBC-CS1, CBC-CS2, and CBC-CS3 standard, the count of operation data must be more than 16 bytes. Operations that are qual or less than one block will output unexpected result. In Non-DMA ECB, CBC, CFB, OFB, CTR, CCM and GCM mode, CRYPTO_AES_CNT must be set as byte count for the last block of data before feeding in the last block of data. In Non-DMA CBC-CS1, CBC-CS2, and CBC-CS3 mode, CRYPTO_AES_CNT must be set as byte count for the last two blocks of data before feeding in the last two blocks of data. In AES GCM mode without DMA cascade function, the value of CRYPTO_AES_CNT is equal to the total value of {CRYPTO_AES_GCM_IVCNT1, CRYPTO_AES_GCM_IVCNT0}, {CRYPTO_AES_GCM_ACNT1, CRYPTO_AES_GCM_ACNT0} and {CRYPTO_AES_GCM_PCNT1, CRYPTO_AES_GCM_PCNT0}. In AES GCM mode with DMA cascade function, the value of CRYPTO_AES_CNT represents the byte count of source text in this cascade function. Thus, the value of CRYPTO_AES_CNT is less than or equal to the total value of {CRYPTO_AES_GCM_IVCNT1, CRYPTO_AES_GCM_IVCNT0}, {CRYPTO_AES_GCM_ACNT1, CRYPTO_AES_GCM_ACNT0} and {CRYPTO_AES_GCM_PCNT1, CRYPTO_AES_GCM_PCNT0} and must be block alignment. In AES CCM mode without DMA cascade function, the value of CRYPTO_AES_CNT is equal to the total value of {CRYPTO_AES_GCM_ACNT1, CRYPTO_AES_GCM_ACNT0} and {CRYPTO_AES_GCM_PCNT1, CRYPTO_AES_GCM_PCNT0}. In AES CCM mode with DMA cascade function, the value of CRYPTO_AES_CNT represents the byte count of source text in this cascade function. Thus, the value of CRYPTO_AES_CNT is less than or equal to the total value of {CRYPTO_AES_GCM_ACNT1, CRYPTO_AES_GCM_ACNT0} and {CRYPTO_AES_GCM_PCNT1, CRYPTO_AES_GCM_PCNT0} and must be block alignment, except for the last block of plaintext or ciphertext.

6.36.7.4 SHA/HMAC Register

SHA/HMAC Control Register (CRYPTO_HMAC_CTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_CTL	CRYPTO_BA+0x300	R/W	SHA/HMAC Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
INSWAP	OUTSWAP	FBOUT	FBIN	Reserved			
15	14	13	12	11	10	9	8
Reserved		SM3EN	Reserved		OPMODE		
7	6	5	4	3	2	1	0
DMAEN	DMACSCAD	DMALAST	DMAFIRST	Reserved		STOP	START

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23]	**INSWAP**	**SHA/HMAC Engine Input Data Swap** 0 = Keep the original order. 1 = The order that CPU feeds data to the accelerator will be changed from {byte3, byte2, byte1, byte0} to {byte0, byte1, byte2, byte3}.
[22]	**OUTSWAP**	**SHA/HMAC Engine Output Data Swap** 0 = Keep the original order. 1 = The order that CPU feeds data to the accelerator will be changed from {byte3, byte2, byte1, byte0} to {byte0, byte1, byte2, byte3}.
[21]	**FBOUT**	**Feedback Output From SHA/HMAC Via DMA Automatically** 0 = Disable DMA automatical feedback output function. 1 = Enable DMA automatical feedback output function when DMAEN = 1.
[20]	**FBIN**	**Feedback Input to SHA/HMAC Via DMA Automatically** 0 = Disable DMA automatical feedback input function. 1 = Enable DMA automatical feedback input function when DMAEN = 1.
[19:14]	**Reserved**	Reserved.
[13]	**SM3EN**	**SM3 Engine Enable Bit** 0 = Execute other function. 1 = Execute SM3 function.
[12]	**Reserved**	Reserved.
[11]	**HMACEN**	**HMAC_SHA Engine Operating Mode** 0 = Execute SHA function. 1 = Execute HMAC function.

[10:8]	**OPMODE**	**SHA/HMAC Engine Operation Modes** 0x0xx: SHA1-160 0x100: SHA2-256 0x101: SHA2-224 0x110: SHA2-512 0x111: SHA2-384 **Note:** These bits can be read and written, but writing to them would not take effect as BUSY is 1. **Note:** When SM3EN=1, SHA/HMAC only executes SM3-256.
[7]	**DMAEN**	**SHA/HMAC Engine DMA Enable Bit** 0 = SHA/HMAC DMA engine Disabled. SHA/HMAC engine operates in Non-DMA mode. The data need to be written in CRYPTO_HMAC_DATIN. 1 = SHA/HMAC DMA engine Enabled. SHA/HMAC engine operates in DMA mode, and data movement from/to the engine is done by DMA logic.
[6]	**DMACSCAD**	**SHA/HMAC Engine DMA with Cascade Mode** 0 = DMA cascade function Disabled. 1 = In DMA cascade mode, software can update DMA source address register, destination address register, and byte count register during a cascade operation, without finishing the accelerator operation.
[5]	**DMALAST**	**SHA/HMAC Last Block** This bit must be set as feeding in last byte of data.
[4]	**DMAFIRST**	**SHA/HMAC First Block in Cascadefunction** This bit must be set as feeding in first byte of data.
[3:2]	**Reserved**	Reserved.
[1]	**STOP**	**SHA/HMAC Engine Stop** 0 = No effect. 1 = Stop SHA/HMAC engine. **Note:** This bit is always 0 when it is read back.
[0]	**START**	**SHA/HMAC Engine Start** 0 = No effect. 1 = Start SHA/HMAC engine. BUSY flag will be set. **Note:** This bit is always 0 when it is read back.

SHA/HMAC Status Register (CRYPTO_HMAC_STS)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_STS	CRYPTO_BA+0x304	R	SHA/HMAC Status Flag	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							DATINREQ
15	14	13	12	11	10	9	8
Reserved						KSERR	DMAERR
7	6	5	4	3	2	1	0
Reserved						DMABUSY	BUSY

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**DATINREQ**	**SHA/HMAC Non-dMA Mode Data Input Request** 0 = No effect. 1 = Request SHA/HMAC Non-DMA mode data input.
[15:10]	**Reserved**	Reserved.
[9]	**KSERR**	**HMAC Engine Access Key Store Error Flag** 0 = No error. 1 = Access error will stop HMAC engine.
[8]	**DMAERR**	**SHA/HMAC Engine DMA Error Flag** 0 = Show the SHA/HMAC engine access normal. 1 = Show the SHA/HMAC engine access error.
[7:2]	**Reserved**	Reserved.
[1]	**DMABUSY**	**SHA/HMAC Engine DMA Busy Flag** 0 = SHA/HMAC DMA engine is idle or finished. 1 = SHA/HMAC DMA engine is busy.
[0]	**BUSY**	**SHA/HMAC Engine Busy** 0 = SHA/HMAC engine is idle or finished. 1 = SHA/HMAC engine is busy.

SHA/HMAC Outputs Digest Word Register (CRYPTO_HMAC_DGSTx)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_DGST0	CRYPTO_BA+0x308	R	SHA/HMAC Output Feedback Data 0	0x0000_0000
CRYPTO_HMAC_DGST1	CRYPTO_BA+0x30C	R	SHA/HMAC Output Feedback Data 1	0x0000_0000
CRYPTO_HMAC_DGST2	CRYPTO_BA+0x310	R	SHA/HMAC Output Feedback Data 2	0x0000_0000
CRYPTO_HMAC_DGST3	CRYPTO_BA+0x314	R	SHA/HMAC Output Feedback Data 3	0x0000_0000
CRYPTO_HMAC_DGST4	CRYPTO_BA+0x318	R	SHA/HMAC Output Feedback Data 4	0x0000_0000
CRYPTO_HMAC_DGST5	CRYPTO_BA+0x31C	R	SHA/HMAC Output Feedback Data 5	0x0000_0000
CRYPTO_HMAC_DGST6	CRYPTO_BA+0x320	R	SHA/HMAC Output Feedback Data 6	0x0000_0000
CRYPTO_HMAC_DGST7	CRYPTO_BA+0x324	R	SHA/HMAC Output Feedback Data 7	0x0000_0000
CRYPTO_HMAC_DGST8	CRYPTO_BA+0x328	R	SHA/HMAC Output Feedback Data 8	0x0000_0000
CRYPTO_HMAC_DGST9	CRYPTO_BA+0x32C	R	SHA/HMAC Output Feedback Data 9	0x0000_0000
CRYPTO_HMAC_DGST10	CRYPTO_BA+0x330	R	SHA/HMAC Output Feedback Data 10	0x0000_0000
CRYPTO_HMAC_DGST11	CRYPTO_BA+0x334	R	SHA/HMAC Output Feedback Data 11	0x0000_0000
CRYPTO_HMAC_DGST12	CRYPTO_BA+0x338	R	SHA/HMAC Output Feedback Data 12	0x0000_0000
CRYPTO_HMAC_DGST13	CRYPTO_BA+0x33C	R	SHA/HMAC Output Feedback Data 13	0x0000_0000
CRYPTO_HMAC_DGST14	CRYPTO_BA+0x340	R	SHA/HMAC Output Feedback Data 14	0x0000_0000
CRYPTO_HMAC_DGST15	CRYPTO_BA+0x344	R	SHA/HMAC Output Feedback Data 15	0x0000_0000

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

Bits	Description	
[31:0]	**DGST**	**SHA/HMAC Output Feedback Data Output Register** For SHA-160, the digest is stored in CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST4. For SHA-224, the digest is stored in CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST6. For SHA-256, the digest is stored in CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST7. For SHA-384, the digest is stored in CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST11. For SHA-512, the digest is stored in CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST15.

SHA/HMAC Key Byte Count Register (CRYPTO_HMAC_KEYCNT)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_KEYCNT	CRYPTO_BA+0x348	R/W	SHA/HMAC Key Byte Count Register	0x0000_0000

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

Bits	Description	
[31:0]	KEYCNT	**SHA/HMAC Key Byte Count** The CRYPTO_HMAC_KEYCNT keeps the byte count of key that SHA/HMAC engine operates. The register is 32-bit and the maximum byte count is 4G bytes. It can be read and written. Writing to the register CRYPTO_HMAC_KEYCNT as the SHA/HMAC accelerator operating does not affect the current SHA/HMAC operation. But the value of CRYPTO_HMAC_KEYCNT will be updated later on. Consequently, software can prepare the key count for the next SHA/HMAC operation.

SHA/HMAC DMA Source Address Register (CRYPTO_HMAC_SADDR)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_SADDR	CRYPTO_BA+0x34C	R/W	SHA/HMAC DMA Source Address Register	0x0000_0000

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

Bits	Description	
[31:0]	SADDR	**SHA/HMAC DMA Source Address** The SHA/HMAC accelerator supports DMA function to transfer the plain text between SRAM memory space and embedded FIFO. The CRYPTO_HMAC_SADDR keeps the source address of the data buffer where the source text is stored. Based on the source address, the SHA/HMAC accelerator can read the plain text from SRAM memory space and do SHA/HMAC operation. The start of source address should be located at word boundary. In other words, bit 1 and 0 of CRYPTO_HMAC_SADDR are ignored. CRYPTO_HMAC_SADDR can be read and written. Writing to CRYPTO_HMAC_SADDR while the SHA/HMAC accelerator is operating does not affect the current SHA/HMAC operation. But the value of CRYPTO_HMAC_SADDR will be updated later on. Consequently, software can prepare the DMA source address for the next SHA/HMAC operation. In DMA mode, software can update the next CRYPTO_HMAC_SADDR before triggering START. CRYPTO_HMAC_SADDR and CRYPTO_HMAC_DADDR can be the same in the value.

SHA/HMAC Byte Count Register (CRYPTO_HMAC_DMACNT)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_DMACNT	CRYPTO_BA+0x350	R/W	SHA/HMAC Byte Count Register	0x0000_0000

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

Bits	Description	
[31:0]	DMACNT	**SHA/HMAC Operation Byte Count** The CRYPTO_HMAC_DMACNT keeps the byte count of source text that is for the SHA/HMAC engine operating in DMA mode. The CRYPTO_HMAC_DMACNT is 32-bit and the maximum of byte count is 4G bytes. CRYPTO_HMAC_DMACNT can be read and written. Writing to CRYPTO_HMAC_DMACNT while the SHA/HMAC accelerator is operating does not affect the current SHA/HMAC operation. But the value of CRYPTO_HMAC_DMACNT will be updated later on. Consequently, software can prepare the byte count of data for the next SHA/HMAC operation. In Non-DMA mode, CRYPTO_HMAC_DMACNT must be set as the byte count of the last block before feeding in the last block of data.

SHA/HMAC Data Input Port Register (CRYPTO_HMAC_DATIN)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_DATIN	CRYPTO_BA+0x354	R/W	SHA/HMAC Engine Non-dMA Mode Data Input Port Register	0x0000_0000

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

Bits	Description	
[31:0]	DATIN	**SHA/HMAC Engine Input Port** CPU feeds data to SHA/HMAC engine through this port by checking CRYPTO_HMAC_STS. Feed data as DATINREQ is 1.

SHA/HMAC Feedback Register (CRYPTO_HMAC_FDBCKx)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_FDBCK0	CRYPTO_BA+0x358	R/W	SHA/HMAC Output Feedback Data 0 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK1	CRYPTO_BA+0x35C	R/W	SHA/HMAC Output Feedback Data 1 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK2	CRYPTO_BA+0x360	R/W	SHA/HMAC Output Feedback Data 2 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK3	CRYPTO_BA+0x364	R/W	SHA/HMAC Output Feedback Data 3 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK4	CRYPTO_BA+0x368	R/W	SHA/HMAC Output Feedback Data 4 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK5	CRYPTO_BA+0x36C	R/W	SHA/HMAC Output Feedback Data 5 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK6	CRYPTO_BA+0x370	R/W	SHA/HMAC Output Feedback Data 6 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK7	CRYPTO_BA+0x374	R/W	SHA/HMAC Output Feedback Data 7 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK8	CRYPTO_BA+0x378	R/W	SHA/HMAC Output Feedback Data 8 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK9	CRYPTO_BA+0x37C	R/W	SHA/HMAC Output Feedback Data 9 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK10	CRYPTO_BA+0x380	R/W	SHA/HMAC Output Feedback Data 10 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK11	CRYPTO_BA+0x384	R/W	SHA/HMAC Output Feedback Data 11 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK12	CRYPTO_BA+0x388	R/W	SHA/HMAC Output Feedback Data 12 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK13	CRYPTO_BA+0x38C	R/W	SHA/HMAC Output Feedback Data 13 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK14	CRYPTO_BA+0x390	R/W	SHA/HMAC Output Feedback Data 14 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK15	CRYPTO_BA+0x394	R/W	SHA/HMAC Output Feedback Data 15 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK16	CRYPTO_BA+0x398	R/W	SHA/HMAC Output Feedback Data 16 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK17	CRYPTO_BA+0x39C	R/W	SHA/HMAC Output Feedback Data 17 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK18	CRYPTO_BA+0x3A0	R/W	SHA/HMAC Output Feedback Data 18 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK19	CRYPTO_BA+0x3A4	R/W	SHA/HMAC Output Feedback Data 19 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK20	CRYPTO_BA+0x3A8	R/W	SHA/HMAC Output Feedback Data 20 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK21	CRYPTO_BA+0x3AC	R/W	SHA/HMAC Output Feedback Data 21 After SHA/HMAC Operation	0x0000_0000

CRYPTO_HMAC_FDBCK22	CRYPTO_BA+0x3B0	R/W	SHA/HMAC Output Feedback Data 22 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK23	CRYPTO_BA+0x3B4	R/W	SHA/HMAC Output Feedback Data 23 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK24	CRYPTO_BA+0x3B8	R/W	SHA/HMAC Output Feedback Data 24 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK25	CRYPTO_BA+0x3BC	R/W	SHA/HMAC Output Feedback Data 25 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK26	CRYPTO_BA+0x3C0	R/W	SHA/HMAC Output Feedback Data 26 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK27	CRYPTO_BA+0x3C4	R/W	SHA/HMAC Output Feedback Data 27 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK28	CRYPTO_BA+0x3C8	R/W	SHA/HMAC Output Feedback Data 28 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK29	CRYPTO_BA+0x3CC	R/W	SHA/HMAC Output Feedback Data 29 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK30	CRYPTO_BA+0x3D0	R/W	SHA/HMAC Output Feedback Data 30 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK31	CRYPTO_BA+0x3D4	R/W	SHA/HMAC Output Feedback Data 31 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK32	CRYPTO_BA+0x3D8	R/W	SHA/HMAC Output Feedback Data 32 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK33	CRYPTO_BA+0x3DC	R/W	SHA/HMAC Output Feedback Data 33 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK34	CRYPTO_BA+0x3E0	R/W	SHA/HMAC Output Feedback Data 34 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK35	CRYPTO_BA+0x3E4	R/W	SHA/HMAC Output Feedback Data 35 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK36	CRYPTO_BA+0x3E8	R/W	SHA/HMAC Output Feedback Data 36 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK37	CRYPTO_BA+0x3EC	R/W	SHA/HMAC Output Feedback Data 37 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK38	CRYPTO_BA+0x3F0	R/W	SHA/HMAC Output Feedback Data 38 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK39	CRYPTO_BA+0x3F4	R/W	SHA/HMAC Output Feedback Data 39 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK40	CRYPTO_BA+0x3F8	R/W	SHA/HMAC Output Feedback Data 40 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK41	CRYPTO_BA+0x3FC	R/W	SHA/HMAC Output Feedback Data 41 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK42	CRYPTO_BA+0x400	R/W	SHA/HMAC Output Feedback Data 42 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK43	CRYPTO_BA+0x404	R/W	SHA/HMAC Output Feedback Data 43 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK44	CRYPTO_BA+0x408	R/W	SHA/HMAC Output Feedback Data 44 After SHA/HMAC Operation	0x0000_0000

CRYPTO_HMAC_FDBCK45	CRYPTO_BA+0x40C	R/W	SHA/HMAC Output Feedback Data 45 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK46	CRYPTO_BA+0x410	R/W	SHA/HMAC Output Feedback Data 46 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK47	CRYPTO_BA+0x414	R/W	SHA/HMAC Output Feedback Data 47 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK48	CRYPTO_BA+0x418	R/W	SHA/HMAC Output Feedback Data 48 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK49	CRYPTO_BA+0x41C	R/W	SHA/HMAC Output Feedback Data 49 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK50	CRYPTO_BA+0x420	R/W	SHA/HMAC Output Feedback Data 50 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK51	CRYPTO_BA+0x424	R/W	SHA/HMAC Output Feedback Data 51 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK52	CRYPTO_BA+0x428	R/W	SHA/HMAC Output Feedback Data 52 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK53	CRYPTO_BA+0x42C	R/W	SHA/HMAC Output Feedback Data 53 After SHA/HMAC Operation	0x0000_0000

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

Bits	Description	
[31:0]	**FDBCK**	**SHA/HMAC Feedback Information** The feedback value is 1728 bits in size for SHA1/2. The SHA/HMAC engine uses the data from CRYPTO_HMAC_FDBCKx as the data inputted to CRYPTO_HMAC_FDBCKx for the next block in DMA cascade mode. The SHA/HMAC engine outputs feedback information for initial setting in the next block's operation. Software can store that feedback value temporarily. After switching back, fill the stored feedback value to CRYPTO_HMAC_FDBCKx in the same operation, and then continue the operation with the original setting.

SHA/HMAC DMA Feedback Address Register (CRYPTO_HMAC_FBADDR)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_FBADDR	CRYPTO_BA+0x4FC	R/W	SHA/HMAC DMA Feedback Address Register	0x0000_0000

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

Bits	Description	
[31:0]	FBADDR	**SHA/HMAC DMA Feedback Address** In DMA cascade mode, software can update DMA feedback address register for automatically reading and writing feedback values via DMA.The FBADDR keeps the feedback address of the feedback data for the next cascade operation. Based on the feedback address, the SHA/HMAC accelerator can read thefeedback data of the last cascade opeation from SRAM memory space and write thefeedback data of the current cascade opeation to SRAM memory space. The start of feedback address should be located at word boundary. In other words, bit 1 and 0 of FBADDR are ignored. FBADDR can be read and written. In DMA mode, software can update the next CRYPTO_HMAC_FBADDR before triggering START.

6.36.7.5 ECC Register

ECC Control Register (CRYPTO_ECC_CTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_CTL	CRYPTO_BA+0x800	R/W	ECC Control Register	0x0000_0000

31	30	29	28	27	26	25	24
CURVEM							
23	22	21	20	19	18	17	16
CURVEM		LDK	LDN	LDB	LDA	LDP2	LDP1
15	14	13	12	11	10	9	8
Reserved	SCAP	CSEL	MODOP		ECCOP		FSEL
7	6	5	4	3	2	1	0
DMAEN	Reserved	ECDSAR	ECDSAS	Reserved		STOP	START

Bits	Description	
[31:22]	**CURVEM**	The key length of elliptic curve.
[21]	**LDK**	**The Control Signal of Register for SCALARK** 0 = The register for SCALARK is not modified by DMA or user. 1 = The register for SCALARK is modified by DMA or user.
[20]	**LDN**	**The Control Signal of Register for the Parameter CURVEN of Elliptic Curve** 0 = The register for CURVEN is not modified by DMA or user. 1 = The register for CURVEN is modified by DMA or user.
[19]	**LDB**	**The Control Signal of Register for the Parameter CURVEB of Elliptic Curve** 0 = The register for CURVEB is not modified by DMA or user. 1 = The register for CURVEB is modified by DMA or user.
[18]	**LDA**	**The Control Signal of Register for the Parameter CURVEA of Elliptic Curve** 0 = The register for CURVEA is not modified by DMA or user. 1 = The register for CURVEA is modified by DMA or user.
[17]	**LDP2**	**The Control Signal of Register POINTX2 and POINTY2 for the x and Y Coordinate of the Second Point** 0 = The register for POINTX2 and POINTY2 is not modified by DMA or user. 1 = The register for POINTX2 and POINTY2 is modified by DMA or user.
[16]	**LDP1**	**The Control Signal of Register POINTX1 and POINTY1 for the x and Y Coordinate of the First Point** 0 = The register for POINTX1 and POINTY1 is not modified by DMA or user. 1 = The register for POINTX1 and POINTY1 is modified by DMA or user.
[15]	**Reserved**	Reserved.
[14]	**SCAP**	**Side-channel Attack Protection** 0 = Full speed without side-channel protection. 1 = Less speed with side-channel protection.

[13]	**CSEL**	**Curve Selection** 0 = NIST suggested curve. 1 = Montgomery curve.
[12:11]	**MODOP**	**Modulus Operation for PF** 00 = Division. POINTX1 = (POINTY1 / POINTX1) % CURVEN. 01 = Multiplication :. POINTX1 = (POINTX1 * POINTY1) % CURVEN. 10 = Addition. POINTX1 = (POINTX1 + POINTY1) % CURVEN. 11 = Subtraction :. POINTX1 = (POINTX1 - POINTY1) % CURVEN. MODOP is active only when ECCOP = 01.
[10:9]	**ECCOP**	**Point Operation for BF and PF** 00 = Point multiplication. (POINTX1, POINTY1) = SCALARK * (POINTX1, POINTY1). 01 = Modulus operation : choose by MODOP (CRYPTO_ECC_CTL[12:11]). 10 = Point addition. (POINTX1, POINTY1) = (POINTX1, POINTY1) +. (POINTX2, POINTY2) 11 = Point doubling. (POINTX1, POINTY1) = 2 * (POINTX1, POINTY1). Besides above three input data, point operations still need the parameters of elliptic curve (CURVEA, CURVEB, CURVEN and CURVEM) as shown in Figure 6.27-11.
[8]	**FSEL**	**Field Selection** 0 = Binary Field (GF(2^m)). 1 = Prime Field (GF(p)).
[7]	**DMAEN**	**ECC Accelerator DMA Enable Bit** 0 = ECC DMA engine Disabled. 1 = ECC DMA engine Enabled. **Note:** Only when START and DMAEN are 1, ECC DMA engine will be active.
[6]	**Reserved**	Reserved.
[5]	**ECDSAR**	**Generate R in ECDSA Signature Generation** 0 = No effect. 1 = Formula for generating R. (POINTX1, POINTY1) = SCALARK * (POINTX1, POINTY1).
[4]	**ECDSAS**	**Generate S in ECDSA Signature Generation** 0 = No effect. 1 = Formula for generating S. POINTX1 = ((POINTX2 * POINTY1 + POINTY2) / POINTX1) % CURVEN.
[3:2]	**Reserved**	Reserved.

[1]	**STOP**	**ECC Accelerator Stop** 0 = No effect. 1 = Abort ECC accelerator and make it into idle state. **Note:** This bit is always 0 when it is read back. Remember to clear ECC interrupt flag after stopping ECC accelerator.
[0]	**START**	**ECC Accelerator Start** 0 = No effect. 1 = Start ECC accelerator. BUSY flag will be set. **Note:** This bit is always 0 when it is read back. ECC accelerator will ignore this START signal when BUSY flag is 1.

ECC Status Register (CRYPTO_ECC_STS)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_STS	CRYPTO_BA+0x804	R	ECC Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						KSERR	BUSERR
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						DMABUSY	BUSY

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**KSERR**	**ECC Engine Access Key Store Error Flag** 0 = No error. 1 = Access error will stop ECC engine.
[16]	**BUSERR**	**ECC DMA Access Bus Error Flag** 0 = No error. 1 = Bus error will stop DMA operation and ECC accelerator.
[15:2]	**Reserved**	Reserved.
[1]	**DMABUSY**	**ECC DMA Busy Flag** 0 = ECC DMA is idle or finished. 1 = ECC DMA is busy.
[0]	**BUSY**	**ECC Accelerator Busy Flag** 0 = The ECC accelerator is idle or finished. 1 = The ECC accelerator is under processing and protects all registers. Remember to clear ECC interrupt flag after ECC accelerator finished

ECC the X-coordinate Value of the First Point Register (CRYPTO_ECC_X1)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_X1_00	CRYPTO_BA+0x808	R/W	ECC the X-coordinate Word0 of the First Point	0x0000_0000
CRYPTO_ECC_X1_01	CRYPTO_BA+0x80C	R/W	ECC the X-coordinate Word1 of the First Point	0x0000_0000
CRYPTO_ECC_X1_02	CRYPTO_BA+0x810	R/W	ECC the X-coordinate Word2 of the First Point	0x0000_0000
CRYPTO_ECC_X1_03	CRYPTO_BA+0x814	R/W	ECC the X-coordinate Word3 of the First Point	0x0000_0000
CRYPTO_ECC_X1_04	CRYPTO_BA+0x818	R/W	ECC the X-coordinate Word4 of the First Point	0x0000_0000
CRYPTO_ECC_X1_05	CRYPTO_BA+0x81C	R/W	ECC the X-coordinate Word5 of the First Point	0x0000_0000
CRYPTO_ECC_X1_06	CRYPTO_BA+0x820	R/W	ECC the X-coordinate Word6 of the First Point	0x0000_0000
CRYPTO_ECC_X1_07	CRYPTO_BA+0x824	R/W	ECC the X-coordinate Word7 of the First Point	0x0000_0000
CRYPTO_ECC_X1_08	CRYPTO_BA+0x828	R/W	ECC the X-coordinate Word8 of the First Point	0x0000_0000
CRYPTO_ECC_X1_09	CRYPTO_BA+0x82C	R/W	ECC the X-coordinate Word9 of the First Point	0x0000_0000
CRYPTO_ECC_X1_10	CRYPTO_BA+0x830	R/W	ECC the X-coordinate Word10 of the First Point	0x0000_0000
CRYPTO_ECC_X1_11	CRYPTO_BA+0x834	R/W	ECC the X-coordinate Word11 of the First Point	0x0000_0000
CRYPTO_ECC_X1_12	CRYPTO_BA+0x838	R/W	ECC the X-coordinate Word12 of the First Point	0x0000_0000
CRYPTO_ECC_X1_13	CRYPTO_BA+0x83C	R/W	ECC the X-coordinate Word13 of the First Point	0x0000_0000
CRYPTO_ECC_X1_14	CRYPTO_BA+0x840	R/W	ECC the X-coordinate Word14 of the First Point	0x0000_0000
CRYPTO_ECC_X1_15	CRYPTO_BA+0x844	R/W	ECC the X-coordinate Word15 of the First Point	0x0000_0000
CRYPTO_ECC_X1_16	CRYPTO_BA+0x848	R/W	ECC the X-coordinate Word16 of the First Point	0x0000_0000
CRYPTO_ECC_X1_17	CRYPTO_BA+0x84C	R/W	ECC the X-coordinate Word17 of the First Point	0x0000_0000

31	30	29	28	27	26	25	24
POINTX1							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
POINTX1							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
POINTX1							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
POINTX1							

Bits	Description

[31:0]	**POINTX1**	**ECC the X-coordinate Value of the First Point** For B-163 or K-163, POINTX1 is stored in CRYPTO_ECC_X1_00~CRYPTO_ECC_X1_05 For B-233 or K-233, POINTX1 is stored in CRYPTO_ECC_X1_00~CRYPTO_ECC_X1_07 For B-283 or K-283, POINTX1 is stored in CRYPTO_ECC_X1_00~CRYPTO_ECC_X1_08 For B-409 or K-409, POINTX1 is stored in CRYPTO_ECC_X1_00~CRYPTO_ECC_X1_12 For B-571 or K-571, POINTX1 is stored in CRYPTO_ECC_X1_00~CRYPTO_ECC_X1_17 For P-192, POINTX1 is stored in CRYPTO_ECC_X1_00~CRYPTO_ECC_X1_05 For P-224, POINTX1 is stored in CRYPTO_ECC_X1_00~CRYPTO_ECC_X1_06 For P-256, POINTX1 is stored in CRYPTO_ECC_X1_00~CRYPTO_ECC_X1_07 For P-384, POINTX1 is stored in CRYPTO_ECC_X1_00~CRYPTO_ECC_X1_11 For P-521, POINTX1 is stored in CRYPTO_ECC_X1_00~CRYPTO_ECC_X1_16

ECC the Y-coordinate Value of the First Point Register (CRYPTO_ECC_Y1)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_Y1_00	CRYPTO_BA+0x850	R/W	ECC the Y-coordinate Word0 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_01	CRYPTO_BA+0x854	R/W	ECC the Y-coordinate Word1 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_02	CRYPTO_BA+0x858	R/W	ECC the Y-coordinate Word2 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_03	CRYPTO_BA+0x85C	R/W	ECC the Y-coordinate Word3 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_04	CRYPTO_BA+0x860	R/W	ECC the Y-coordinate Word4 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_05	CRYPTO_BA+0x864	R/W	ECC the Y-coordinate Word5 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_06	CRYPTO_BA+0x868	R/W	ECC the Y-coordinate Word6 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_07	CRYPTO_BA+0x86C	R/W	ECC the Y-coordinate Word7 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_08	CRYPTO_BA+0x870	R/W	ECC the Y-coordinate Word8 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_09	CRYPTO_BA+0x874	R/W	ECC the Y-coordinate Word9 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_10	CRYPTO_BA+0x878	R/W	ECC the Y-coordinate Word10 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_11	CRYPTO_BA+0x87C	R/W	ECC the Y-coordinate Word11 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_12	CRYPTO_BA+0x880	R/W	ECC the Y-coordinate Word12 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_13	CRYPTO_BA+0x884	R/W	ECC the Y-coordinate Word13 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_14	CRYPTO_BA+0x888	R/W	ECC the Y-coordinate Word14 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_15	CRYPTO_BA+0x88C	R/W	ECC the Y-coordinate Word15 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_16	CRYPTO_BA+0x890	R/W	ECC the Y-coordinate Word16 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_17	CRYPTO_BA+0x894	R/W	ECC the Y-coordinate Word17 of the First Point	0x0000_0000

31	30	29	28	27	26	25	24
POINTY1							
23	22	21	20	19	18	17	16
POINTY1							
15	14	13	12	11	10	9	8
POINTY1							
7	6	5	4	3	2	1	0
POINTY1							

Bits	Description

[31:0]	**POINTY1**	**ECC the Y-coordinate Value of the First Point** For B-163 or K-163, POINTY1 is stored in CRYPTO_ECC_Y1_00~CRYPTO_ECC_Y1_05 For B-233 or K-233, POINTY1 is stored in CRYPTO_ECC_Y1_00~CRYPTO_ECC_Y1_07 For B-283 or K-283, POINTY1 is stored in CRYPTO_ECC_Y1_00~CRYPTO_ECC_Y1_08 For B-409 or K-409, POINTY1 is stored in CRYPTO_ECC_Y1_00~CRYPTO_ECC_Y1_12 For B-571 or K-571, POINTY1 is stored in CRYPTO_ECC_Y1_00~CRYPTO_ECC_Y1_17 For P-192, POINTY1 is stored in CRYPTO_ECC_Y1_00~CRYPTO_ECC_Y1_05 For P-224, POINTY1 is stored in CRYPTO_ECC_Y1_00~CRYPTO_ECC_Y1_06 For P-256, POINTY1 is stored in CRYPTO_ECC_Y1_00~CRYPTO_ECC_Y1_07 For P-384, POINTY1 is stored in CRYPTO_ECC_Y1_00~CRYPTO_ECC_Y1_11 For P-521, POINTY1 is stored in CRYPTO_ECC_Y1_00~CRYPTO_ECC_Y1_16

ECC the X-coordinate Value of the Second Point Register (CRYPTO_ECC_X2)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_X2_00	CRYPTO_BA+0x898	R/W	ECC the X-coordinate Word0 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_01	CRYPTO_BA+0x89C	R/W	ECC the X-coordinate Word1 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_02	CRYPTO_BA+0x8A0	R/W	ECC the X-coordinate Word2 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_03	CRYPTO_BA+0x8A4	R/W	ECC the X-coordinate Word3 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_04	CRYPTO_BA+0x8A8	R/W	ECC the X-coordinate Word4 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_05	CRYPTO_BA+0x8AC	R/W	ECC the X-coordinate Word5 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_06	CRYPTO_BA+0x8B0	R/W	ECC the X-coordinate Word6 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_07	CRYPTO_BA+0x8B4	R/W	ECC the X-coordinate Word7 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_08	CRYPTO_BA+0x8B8	R/W	ECC the X-coordinate Word8 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_09	CRYPTO_BA+0x8BC	R/W	ECC the X-coordinate Word9 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_10	CRYPTO_BA+0x8C0	R/W	ECC the X-coordinate Word10 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_11	CRYPTO_BA+0x8C4	R/W	ECC the X-coordinate Word11 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_12	CRYPTO_BA+0x8C8	R/W	ECC the X-coordinate Word12 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_13	CRYPTO_BA+0x8CC	R/W	ECC the X-coordinate Word13 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_14	CRYPTO_BA+0x8D0	R/W	ECC the X-coordinate Word14 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_15	CRYPTO_BA+0x8D4	R/W	ECC the X-coordinate Word15 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_16	CRYPTO_BA+0x8D8	R/W	ECC the X-coordinate Word16 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_17	CRYPTO_BA+0x8DC	R/W	ECC the X-coordinate Word17 of the Second Point	0x0000_0000

31	30	29	28	27	26	25	24
POINTX2							
23	22	21	20	19	18	17	16
POINTX2							
15	14	13	12	11	10	9	8
POINTX2							
7	6	5	4	3	2	1	0
POINTX2							

Bits	Description

[31:0]	**POINTX2**	**ECC the X-coordinate Value of the Second Point** For B-163 or K-163, POINTX2 is stored in CRYPTO_ECC_X2_00~CRYPTO_ECC_X2_05 For B-233 or K-233, POINTX2 is stored in CRYPTO_ECC_X2_00~CRYPTO_ECC_X2_07 For B-283 or K-283, POINTX2 is stored in CRYPTO_ECC_X2_00~CRYPTO_ECC_X2_08 For B-409 or K-409, POINTX2 is stored in CRYPTO_ECC_X2_00~CRYPTO_ECC_X2_12 For B-571 or K-571, POINTX2 is stored in CRYPTO_ECC_X2_00~CRYPTO_ECC_X2_17 For P-192, POINTX2 is stored in CRYPTO_ECC_X2_00~CRYPTO_ECC_X2_05 For P-224, POINTX2 is stored in CRYPTO_ECC_X2_00~CRYPTO_ECC_X2_06 For P-256, POINTX2 is stored in CRYPTO_ECC_X2_00~CRYPTO_ECC_X2_07 For P-384, POINTX2 is stored in CRYPTO_ECC_X2_00~CRYPTO_ECC_X2_11 For P-521, POINTX2 is stored in CRYPTO_ECC_X2_00~CRYPTO_ECC_X2_16

ECC the Y-coordinate Value of the Second Point Register (CRYPTO_ECC_Y2)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_Y2_00	CRYPTO_BA+0x8E0	R/W	ECC the Y-coordinate Word0 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_01	CRYPTO_BA+0x8E4	R/W	ECC the Y-coordinate Word1 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_02	CRYPTO_BA+0x8E8	R/W	ECC the Y-coordinate Word2 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_03	CRYPTO_BA+0x8EC	R/W	ECC the Y-coordinate Word3 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_04	CRYPTO_BA+0x8F0	R/W	ECC the Y-coordinate Word4 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_05	CRYPTO_BA+0x8F4	R/W	ECC the Y-coordinate Word5 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_06	CRYPTO_BA+0x8F8	R/W	ECC the Y-coordinate Word6 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_07	CRYPTO_BA+0x8FC	R/W	ECC the Y-coordinate Word7 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_08	CRYPTO_BA+0x900	R/W	ECC the Y-coordinate Word8 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_09	CRYPTO_BA+0x904	R/W	ECC the Y-coordinate Word9 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_10	CRYPTO_BA+0x908	R/W	ECC the Y-coordinate Word10 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_11	CRYPTO_BA+0x90C	R/W	ECC the Y-coordinate Word11 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_12	CRYPTO_BA+0x910	R/W	ECC the Y-coordinate Word12 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_13	CRYPTO_BA+0x914	R/W	ECC the Y-coordinate Word13 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_14	CRYPTO_BA+0x918	R/W	ECC the Y-coordinate Word14 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_15	CRYPTO_BA+0x91C	R/W	ECC the Y-coordinate Word15 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_16	CRYPTO_BA+0x920	R/W	ECC the Y-coordinate Word16 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_17	CRYPTO_BA+0x924	R/W	ECC the Y-coordinate Word17 of the Second Point	0x0000_0000

31	30	29	28	27	26	25	24
POINTY2							
23	22	21	20	19	18	17	16
POINTY2							
15	14	13	12	11	10	9	8
POINTY2							
7	6	5	4	3	2	1	0
POINTY2							

Bits	Description

[31:0]	**POINTY2**	**ECC the Y-coordinate Value of the Second Point** For B-163 or K-163, POINTY2 is stored in CRYPTO_ECC_Y2_00~CRYPTO_ECC_Y2_05 For B-233 or K-233, POINTY2 is stored in CRYPTO_ECC_Y2_00~CRYPTO_ECC_Y2_07 For B-283 or K-283, POINTY2 is stored in CRYPTO_ECC_Y2_00~CRYPTO_ECC_Y2_08 For B-409 or K-409, POINTY2 is stored in CRYPTO_ECC_Y2_00~CRYPTO_ECC_Y2_12 For B-571 or K-571, POINTY2 is stored in CRYPTO_ECC_Y2_00~CRYPTO_ECC_Y2_17 For P-192, POINTY2 is stored in CRYPTO_ECC_Y2_00~CRYPTO_ECC_Y2_05 For P-224, POINTY2 is stored in CRYPTO_ECC_Y2_00~CRYPTO_ECC_Y2_06 For P-256, POINTY2 is stored in CRYPTO_ECC_Y2_00~CRYPTO_ECC_Y2_07 For P-384, POINTY2 is stored in CRYPTO_ECC_Y2_00~CRYPTO_ECC_Y2_11 For P-521, POINTY2 is stored in CRYPTO_ECC_Y2_00~CRYPTO_ECC_Y2_16

ECC the Parameter CURVEA Value of Elliptic Curve Register (CRYPTO_ECC_A)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_A_00	CRYPTO_BA+0x928	R/W	ECC the Parameter CURVEA Word0 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_01	CRYPTO_BA+0x92C	R/W	ECC the Parameter CURVEA Word1 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_02	CRYPTO_BA+0x930	R/W	ECC the Parameter CURVEA Word2 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_03	CRYPTO_BA+0x934	R/W	ECC the Parameter CURVEA Word3 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_04	CRYPTO_BA+0x938	R/W	ECC the Parameter CURVEA Word4 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_05	CRYPTO_BA+0x93C	R/W	ECC the Parameter CURVEA Word5 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_06	CRYPTO_BA+0x940	R/W	ECC the Parameter CURVEA Word6 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_07	CRYPTO_BA+0x944	R/W	ECC the Parameter CURVEA Word7 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_08	CRYPTO_BA+0x948	R/W	ECC the Parameter CURVEA Word8 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_09	CRYPTO_BA+0x94C	R/W	ECC the Parameter CURVEA Word9 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_10	CRYPTO_BA+0x950	R/W	ECC the Parameter CURVEA Word10 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_11	CRYPTO_BA+0x954	R/W	ECC the Parameter CURVEA Word11 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_12	CRYPTO_BA+0x958	R/W	ECC the Parameter CURVEA Word12 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_13	CRYPTO_BA+0x95C	R/W	ECC the Parameter CURVEA Word13 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_14	CRYPTO_BA+0x960	R/W	ECC the Parameter CURVEA Word14 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_15	CRYPTO_BA+0x964	R/W	ECC the Parameter CURVEA Word15 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_16	CRYPTO_BA+0x968	R/W	ECC the Parameter CURVEA Word16 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_17	CRYPTO_BA+0x96C	R/W	ECC the Parameter CURVEA Word17 of Elliptic Curve	0x0000_0000

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

Bits	Description

[31:0]	**CURVEA**	**ECC the Parameter CURVEA Value of Elliptic Curve** The formula of elliptic curve is $y^2=x^3+CURVEA*x+CURVEB$ in GF(p) and $y^2+x*y=x^3+CURVEA*x^2+CURVEB$ in GF(2^m). For B-163 or K-163, CURVEA is stored in CRYPTO_ECC_A_00~CRYPTO_ECC_A_05 For B-233 or K-233, CURVEA is stored in CRYPTO_ECC_A_00~CRYPTO_ECC_A_07 For B-283 or K-283, CURVEA is stored in CRYPTO_ECC_A_00~CRYPTO_ECC_A_08 For B-409 or K-409, CURVEA is stored in CRYPTO_ECC_A_00~CRYPTO_ECC_A_12 For B-571 or K-571, CURVEA is stored in CRYPTO_ECC_A_00~CRYPTO_ECC_A_17 For P-192, CURVEA is stored in CRYPTO_ECC_A_00~CRYPTO_ECC_A_05 For P-224, CURVEA is stored in CRYPTO_ECC_A_00~CRYPTO_ECC_A_06 For P-256, CURVEA is stored in CRYPTO_ECC_A_00~CRYPTO_ECC_A_07 For P-384, CURVEA is stored in CRYPTO_ECC_A_00~CRYPTO_ECC_A_11 For P-521, CURVEA is stored in CRYPTO_ECC_A_00~CRYPTO_ECC_A_16

ECC the Parameter CURVEB Value of Elliptic Curve Register (CRYPTO_ECC_B)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_B_00	CRYPTO_BA+0x970	R/W	ECC the Parameter CURVEB Word0 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_01	CRYPTO_BA+0x974	R/W	ECC the Parameter CURVEB Word1 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_02	CRYPTO_BA+0x978	R/W	ECC the Parameter CURVEB Word2 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_03	CRYPTO_BA+0x97C	R/W	ECC the Parameter CURVEB Word3 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_04	CRYPTO_BA+0x980	R/W	ECC the Parameter CURVEB Word4 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_05	CRYPTO_BA+0x984	R/W	ECC the Parameter CURVEB Word5 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_06	CRYPTO_BA+0x988	R/W	ECC the Parameter CURVEB Word6 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_07	CRYPTO_BA+0x98C	R/W	ECC the Parameter CURVEB Word7 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_08	CRYPTO_BA+0x990	R/W	ECC the Parameter CURVEB Word8 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_09	CRYPTO_BA+0x994	R/W	ECC the Parameter CURVEB Word9 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_10	CRYPTO_BA+0x998	R/W	ECC the Parameter CURVEB Word10 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_11	CRYPTO_BA+0x99C	R/W	ECC the Parameter CURVEB Word11 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_12	CRYPTO_BA+0x9A0	R/W	ECC the Parameter CURVEB Word12 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_13	CRYPTO_BA+0x9A4	R/W	ECC the Parameter CURVEB Word13 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_14	CRYPTO_BA+0x9A8	R/W	ECC the Parameter CURVEB Word14 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_15	CRYPTO_BA+0x9AC	R/W	ECC the Parameter CURVEB Word15 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_16	CRYPTO_BA+0x9B0	R/W	ECC the Parameter CURVEB Word16 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_17	CRYPTO_BA+0x9B4	R/W	ECC the Parameter CURVEB Word17 of Elliptic Curve	0x0000_0000

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

Bits	Description

[31:0]	**CURVEB**	**ECC the Parameter CURVEB Value of Elliptic Curve** The formula of elliptic curve is $y^2=x^3+CURVEA*x+CURVEB$ in GF(p) and $y^2+x*y=x^3+CURVEA*x^2+CURVEB$ in GF(2^m). For B-163 or K-163, CURVEB is stored in CRYPTO_ECC_B_00~CRYPTO_ECC_B_05 For B-233 or K-233, CURVEB is stored in CRYPTO_ECC_B_00~CRYPTO_ECC_B_07 For B-283 or K-283, CURVEB is stored in CRYPTO_ECC_B_00~CRYPTO_ECC_B_08 For B-409 or K-409, CURVEB is stored in CRYPTO_ECC_B_00~CRYPTO_ECC_B_12 For B-521 or K-521, CURVEB is stored in CRYPTO_ECC_B_00~CRYPTO_ECC_B_17 For P-192, CURVEB is stored in CRYPTO_ECC_B_00~CRYPTO_ECC_B_05 For P-224, CURVEB is stored in CRYPTO_ECC_B_00~CRYPTO_ECC_B_06 For P-256, CURVEB is stored in CRYPTO_ECC_B_00~CRYPTO_ECC_B_07 For P-384, CURVEB is stored in CRYPTO_ECC_B_00~CRYPTO_ECC_B_11 For P-521, CURVEB is stored in CRYPTO_ECC_B_00~CRYPTO_ECC_B_16

ECC the Parameter CURVEN Value of Elliptic Curve Register (CRYPTO_ECC_N)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_N_00	CRYPTO_BA+0x9B8	R/W	ECC the Parameter CURVEN Word0 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_01	CRYPTO_BA+0x9BC	R/W	ECC the Parameter CURVEN Word1 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_02	CRYPTO_BA+0x9C0	R/W	ECC the Parameter CURVEN Word2 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_03	CRYPTO_BA+0x9C4	R/W	ECC the Parameter CURVEN Word3 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_04	CRYPTO_BA+0x9C8	R/W	ECC the Parameter CURVEN Word4 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_05	CRYPTO_BA+0x9CC	R/W	ECC the Parameter CURVEN Word5 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_06	CRYPTO_BA+0x9D0	R/W	ECC the Parameter CURVEN Word6 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_07	CRYPTO_BA+0x9D4	R/W	ECC the Parameter CURVEN Word7 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_08	CRYPTO_BA+0x9D8	R/W	ECC the Parameter CURVEN Word8 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_09	CRYPTO_BA+0x9DC	R/W	ECC the Parameter CURVEN Word9 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_10	CRYPTO_BA+0x9E0	R/W	ECC the Parameter CURVEN Word10 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_11	CRYPTO_BA+0x9E4	R/W	ECC the Parameter CURVEN Word11 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_12	CRYPTO_BA+0x9E8	R/W	ECC the Parameter CURVEN Word12 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_13	CRYPTO_BA+0x9EC	R/W	ECC the Parameter CURVEN Word13 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_14	CRYPTO_BA+0x9F0	R/W	ECC the Parameter CURVEN Word14 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_15	CRYPTO_BA+0x9F4	R/W	ECC the Parameter CURVEN Word15 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_16	CRYPTO_BA+0x9F8	R/W	ECC the Parameter CURVEN Word16 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_17	CRYPTO_BA+0x9FC	R/W	ECC the Parameter CURVEN Word17 of Elliptic Curve	0x0000_0000

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

Bits	Description

[31:0]	**CURVEN**	**ECC the Parameter CURVEN Value of Elliptic Curve** In GF(p), CURVEN is the prime p. In GF(2^m), CURVEN is the irreducible polynomial. For B-163 or K-163, CURVEN is stored in CRYPTO_ECC_N_00~CRYPTO_ECC_N_05 For B-233 or K-233, CURVEN is stored in CRYPTO_ECC_N_00~CRYPTO_ECC_N_07 For B-283 or K-283, CURVEN is stored in CRYPTO_ECC_N_00~CRYPTO_ECC_N_08 For B-409 or K-409, CURVEN is stored in CRYPTO_ECC_N_00~CRYPTO_ECC_N_12 For B-571 or K-571, CURVEN is stored in CRYPTO_ECC_N_00~CRYPTO_ECC_N_17 For P-192, CURVEN is stored in CRYPTO_ECC_N_00~CRYPTO_ECC_N_05 For P-224, CURVEN is stored in CRYPTO_ECC_N_00~CRYPTO_ECC_N_06 For P-256, CURVEN is stored in CRYPTO_ECC_N_00~CRYPTO_ECC_N_07 For P-384, CURVEN is stored in CRYPTO_ECC_N_00~CRYPTO_ECC_N_11 For P-521, CURVEN is stored in CRYPTO_ECC_N_00~CRYPTO_ECC_N_16

ECC the Scalar K Value of Elliptic Curve Register (CRYPTO_ECC_K)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_K_00	CRYPTO_BA+0xA00	W	ECC the Scalar SCALARK Word0 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_01	CRYPTO_BA+0xA04	W	ECC the Scalar SCALARK Word1 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_02	CRYPTO_BA+0xA08	W	ECC the Scalar SCALARK Word2 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_03	CRYPTO_BA+0xA0C	W	ECC the Scalar SCALARK Word3 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_04	CRYPTO_BA+0xA10	W	ECC the Scalar SCALARK Word4 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_05	CRYPTO_BA+0xA14	W	ECC the Scalar SCALARK Word5 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_06	CRYPTO_BA+0xA18	W	ECC the Scalar SCALARK Word6 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_07	CRYPTO_BA+0xA1C	W	ECC the Scalar SCALARK Word7 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_08	CRYPTO_BA+0xA20	W	ECC the Scalar SCALARK Word8 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_09	CRYPTO_BA+0xA24	W	ECC the Scalar SCALARK Word9 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_10	CRYPTO_BA+0xA28	W	ECC the Scalar SCALARK Word10 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_11	CRYPTO_BA+0xA2C	W	ECC the Scalar SCALARK Word11 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_12	CRYPTO_BA+0xA30	W	ECC the Scalar SCALARK Word12 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_13	CRYPTO_BA+0xA34	W	ECC the Scalar SCALARK Word13 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_14	CRYPTO_BA+0xA38	W	ECC the Scalar SCALARK Word14 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_15	CRYPTO_BA+0xA3C	W	ECC the Scalar SCALARK Word15 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_16	CRYPTO_BA+0xA40	W	ECC the Scalar SCALARK Word16 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_17	CRYPTO_BA+0xA44	W	ECC the Scalar SCALARK Word17 of Point Multiplication	0x0000_0000

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

Bits	Description

[31:0]	**SCALARK**	**ECC the Scalar SCALARK Value of Point Multiplication** Because the SCALARK usually stores the private key, ECC accelerator do not allow to read the register SCALARK. For B-163 or K-163, SCALARK is stored in CRYPTO_ECC_K_00~CRYPTO_ECC_K_05 For B-233 or K-233, SCALARK is stored in CRYPTO_ECC_K_00~CRYPTO_ECC_K_07 For B-283 or K-283, SCALARK is stored in CRYPTO_ECC_K_00~CRYPTO_ECC_K_08 For B-409 or K-409, SCALARK is stored in CRYPTO_ECC_K_00~CRYPTO_ECC_K_12 For B-571 or K-571, SCALARK is stored in CRYPTO_ECC_K_00~CRYPTO_ECC_K_17 For P-192, SCALARK is stored in CRYPTO_ECC_K_00~CRYPTO_ECC_K_05 For P-224, SCALARK is stored in CRYPTO_ECC_K_00~CRYPTO_ECC_K_06 For P-256, SCALARK is stored in CRYPTO_ECC_K_00~CRYPTO_ECC_K_07 For P-384, SCALARK is stored in CRYPTO_ECC_K_00~CRYPTO_ECC_K_11 For P-521, SCALARK is stored in CRYPTO_ECC_K_00~CRYPTO_ECC_K_16

ECC DMA Source Address Register (CRYPTO_ECC_SADDR)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_SADDR	CRYPTO_BA+0xA48	R/W	ECC DMA Source Address Register	0x0000_0000

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

Bits	Description	
[31:0]	SADDR	**ECC DMA Source Address** The ECC accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and ECC accelerator. The SADDR keeps the source address of the data buffer where the source text is stored. Based on the source address, the ECC accelerator can read the DATA and PARAMETER from SRAM memory space and do ECC operation. The start of source address should be located at word boundary. That is, bit 1 and 0 of SADDR are ignored. SADDR can be read and written. In DMA mode, software must update the CRYPTO_ECC_SADDR before triggering START.

ECC DMA Destination Address Register (CRYPTO_ECC_DADDR)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_DADDR	CRYPTO_BA+0xA4C	R/W	ECC DMA Destination Address Register	0x0000_0000

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

Bits	Description	
[31:0]	DADDR	**ECC DMA Destination Address** The ECC accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory and ECC accelerator. The DADDR keeps the destination address of the data buffer where output data of ECC engine will be stored. Based on the destination address, the ECC accelerator can write the result data back to SRAM memory space after the ECC operation is finished. The start of destination address should be located at word boundary. That is, bit 1 and 0 of DADDR are ignored. DADDR can be read and written. In DMA mode, software must update the CRYPTO_ECC_DADDR before triggering START.

ECC Starting Address of Updated Registers (CRYPTO_ECC_STARTREG)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_STARTREG	CRYPTO_BA+0xA50	R/W	ECC Starting Address of Updated Registers	0x0000_0000

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

Bits	Description	
[31:0]	STARTREG	**ECC Starting Address of Updated Registers** The address of the updated registers that DMA feeds the first data or parameter to ECC engine. When ECC engine is active, ECC accelerator does not allow users to modify STARTRE.G. for example, to update input data from register CRYPTO_ECC POINTX1. Thus, the value of STARTREG is 0x808.

ECC DMA Word Count (CRYPTO_ECC_WORDCNT)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_WORDCNT	CRYPTO_BA+0xA54	R/W	ECC DMA Word Count	0x0000_0000

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

Bits	Description	
[31:0]	WORDCNT	**ECC DMA Word Count** The CRYPTO_ECC_WORDCNT keeps the word count of source data that is for the required input data of ECC accelerator with various operations in DMA mode. Although CRYPTO_ECC_WORDCNT is 32-bit, the maximum of word count in ECC accelerator is 144 words. CRYPTO_ECC_WORDCNT can be read and written.

6.36.7.6 RSA Register

RSA Control Register (CRYPTO_RSA_CTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_CTL	CRYPTO_BA+0xB00	R/W	RSA Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							SCAP
7	6	5	4	3	2	1	0
Reserved		KEYLENG		CRTBYP	CRT	STOP	START

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**SCAP**	**Side Channel Attack Protection Enable Control** 0 = Side Channel Attack Protection Disabled. 1 = Side Channel Attack Protection Enabled.
[7:6]	**Reserved**	Reserved.
[5:4]	**KEYLENG**	**The Key Length of RSA Operation** 00 = 1024 bits. 01 = 2048 bits. 10 = 3072 bits. 11 = 4096 bits.
[3]	**CRTBYP**	**CRT Bypass Enable Control** 0 = CRT Bypass Disabled. 1 = CRT Bypass Enabled. CRT bypass is only used in CRT decryption with the same key. **Note:** If users want to decrypt repeatedly with the same key, they can execute CRT bypass mode **after the first time CRT decryption (means the second time to the latest time)**, but they cannot set CRTBYP to 1 in non-CRT mode.
[2]	**CRT**	**CRT Enable Control** 0 = CRT Disabled. 1 = CRT Enabled. **Note:** CRT is only used in decryption with key length 2048, 3072,4096 bits.

[1]	**STOP**	**RSA Accelerator Stop** 0 = No effect. 1 = Abort RSA accelerator and make it into initial state. **Note:** This bit is always 0 when it is read back. Remember to clear RSA interrupt flag after stopping RSA accelerator.
[0]	**START**	**RSA Accelerator Start** 0 = No effect. 1 = Start RSA accelerator. BUSY flag will be set. This bit is always 0 when it is read back. RSA accelerator will ignore this START signal when BUSY flag is 1.

RSA Status Register (CRYPTO_RSA_STS)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_STS	CRYPTO_BA+0xB04	R	RSA Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					KSERR	CTLERR	BUSERR
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						DMABUSY	BUSY

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18]	**KSERR**	**RSA Engine Access Key Store Error Flag** 0 = No error. 1 = Access error will stop RSA engine.
[17]	**CTLERR**	**RSA Control Register Error Flag** 0 = No error. 1 = RSA control error. RSA will not start in the unsupported situation. **Note:** If the error combination of control is used even though START(CRYPTO_RSA_CTL[0]) is not set to 1, CTLERR is still set to 1.
[16]	**BUSERR**	**RSA DMA Access Bus Error Flag** 0 = No error. 1 = Bus error will stop DMA operation and RSA accelerator.
[15:2]	**Reserved**	Reserved.
[1]	**DMABUSY**	**RSA DMA Busy Flag** 0 = RSA DMA is idle or finished. 1 = RSA DMA is busy.
[0]	**BUSY**	**RSA Accelerator Busy Flag** 0 = The RSA accelerator is idle or finished. 1 = The RSA accelerator is under processing and protects all registers. Remember to clear RSA interrupt flag after RSA accelerator finished.

RSA DMA Source Address Register0 (CRYPTO_RSA_SADDR0)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_SADDR0	CRYPTO_BA+0xB08	R/W	RSA DMA Source Address Register0	0x0000_0000

31	30	29	28	27	26	25	24
SADDR0							
23	22	21	20	19	18	17	16
SADDR0							
15	14	13	12	11	10	9	8
SADDR0							
7	6	5	4	3	2	1	0
SADDR0							

Bits	Description	
[31:0]	**SADDR0**	**RSA DMA Source Address Register0** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator. This register is stored in the address of RSA the Base of Exponentiation (M).

RSA DMA Source Address Register1 (CRYPTO_RSA_SADDR1)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_SADDR1	CRYPTO_BA+0xB0C	R/W	RSA DMA Source Address Register1	0x0000_0000

31	30	29	28	27	26	25	24
SADDR1							
23	22	21	20	19	18	17	16
SADDR1							
15	14	13	12	11	10	9	8
SADDR1							
7	6	5	4	3	2	1	0
SADDR1							

Bits	Description	
[31:0]	SADDR1	**RSA DMA Source Address Register1** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator. This register is stored in the address of RSA the Base of Modulus Operation (N).

RSA DMA Source Address Register2 (CRYPTO_RSA_SADDR2)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_SADDR2	CRYPTO_BA+0xB10	R/W	RSA DMA Source Address Register2	0x0000_0000

31	30	29	28	27	26	25	24
SADDR2							
23	22	21	20	19	18	17	16
SADDR2							
15	14	13	12	11	10	9	8
SADDR2							
7	6	5	4	3	2	1	0
SADDR2							

Bits	Description	
[31:0]	SADDR2	**RSA DMA Source Address Register2** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator. This register is stored in the address of RSA the Exponent of Exponentiation (E).

RSA DMA Source Address Register3 (CRYPTO_RSA_SADDR3)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_SADDR3	CRYPTO_BA+0xB14	R/W	RSA DMA Source Address Register3	0x0000_0000

31	30	29	28	27	26	25	24
SADDR3							
23	22	21	20	19	18	17	16
SADDR3							
15	14	13	12	11	10	9	8
SADDR3							
7	6	5	4	3	2	1	0
SADDR3							

Bits	Description	
[31:0]	SADDR3	**RSA DMA Source Address Register3** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator. This register is stored in the address of RSA the Factor of Modulus Operation (p).

RSA DMA Source Address Register4 (CRYPTO_RSA_SADDR4)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_SADDR4	CRYPTO_BA+0xB18	R/W	RSA DMA Source Address Register4	0x0000_0000

31	30	29	28	27	26	25	24
SADDR4							
23	22	21	20	19	18	17	16
SADDR4							
15	14	13	12	11	10	9	8
SADDR4							
7	6	5	4	3	2	1	0
SADDR4							

Bits	Description	
[31:0]	SADDR4	**RSA DMA Source Address Register4** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator. This register is stored in the address of RSA the Factor of Modulus Operation (q).

RSA DMA Destination Address Register (CRYPTO_RSA_DADDR)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_DADDR	CRYPTO_BA+0xB1C	R/W	RSA DMA Destination Address Register	0x0000_0000

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

Bits	Description	
[31:0]	DADDR	**RSA DMA Destination Address Register** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator. This register is stored in the address of RSA DMA Destination Address Register (Ans).

RSA DMA Middle Address Register0 (CRYPTO_RSA_MADDR0)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_MADDR0	CRYPTO_BA+0xB20	R/W	RSA DMA Middle Address Register0	0x0000_0000

31	30	29	28	27	26	25	24
MADDR0							
23	22	21	20	19	18	17	16
MADDR0							
15	14	13	12	11	10	9	8
MADDR0							
7	6	5	4	3	2	1	0
MADDR0							

Bits	Description	
[31:0]	MADDR0	**RSA DMA Middle Address Register0** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator.

RSA DMA Middle Address Register1 (CRYPTO_RSA_MADDR1)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_MADDR1	CRYPTO_BA+0xB24	R/W	RSA DMA Middle Address Register1	0x0000_0000

31	30	29	28	27	26	25	24
MADDR1							
23	22	21	20	19	18	17	16
MADDR1							
15	14	13	12	11	10	9	8
MADDR1							
7	6	5	4	3	2	1	0
MADDR1							

Bits	Description	
[31:0]	MADDR1	**RSA DMA Middle Address Register1** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator.

RSA DMA Middle Address Register2 (CRYPTO_RSA_MADDR2)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_MADDR2	CRYPTO_BA+0xB28	R/W	RSA DMA Middle Address Register2	0x0000_0000

31	30	29	28	27	26	25	24
MADDR2							
23	22	21	20	19	18	17	16
MADDR2							
15	14	13	12	11	10	9	8
MADDR2							
7	6	5	4	3	2	1	0
MADDR2							

Bits	Description	
[31:0]	MADDR2	**RSA DMA Middle Address Register2** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator.

RSA DMA Middle Address Register3 (CRYPTO_RSA_MADDR3)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_MADDR3	CRYPTO_BA+0xB2C	R/W	RSA DMA Middle Address Register3	0x0000_0000

31	30	29	28	27	26	25	24
MADDR3							
23	22	21	20	19	18	17	16
MADDR3							
15	14	13	12	11	10	9	8
MADDR3							
7	6	5	4	3	2	1	0
MADDR3							

Bits	Description	
[31:0]	MADDR3	**RSA DMA Middle Address Register3** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator.

RSA DMA Middle Address Register4 (CRYPTO_RSA_MADDR4)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_MADDR4	CRYPTO_BA+0xB30	R/W	RSA DMA Middle Address Register4	0x0000_0000

31	30	29	28	27	26	25	24
MADDR4							
23	22	21	20	19	18	17	16
MADDR4							
15	14	13	12	11	10	9	8
MADDR4							
7	6	5	4	3	2	1	0
MADDR4							

Bits	Description	
[31:0]	MADDR4	**RSA DMA Middle Address Register4** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator.

RSA DMA Middle Address Register5 (CRYPTO_RSA_MADDR5)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_MADDR5	CRYPTO_BA+0xB34	R/W	RSA DMA Middle Address Register5	0x0000_0000

31	30	29	28	27	26	25	24
MADDR5							
23	22	21	20	19	18	17	16
MADDR5							
15	14	13	12	11	10	9	8
MADDR5							
7	6	5	4	3	2	1	0
MADDR5							

Bits	Description	
[31:0]	MADDR5	**RSA DMA Middle Address Register5** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator.

RSA DMA Middle Address Register6 (CRYPTO_RSA_MADDR6)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_MADDR6	CRYPTO_BA+0xB38	R/W	RSA DMA Middle Address Register6	0x0000_0000

31	30	29	28	27	26	25	24
MADDR6							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
MADDR6							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
MADDR6							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MADDR6							

Bits	Description	
[31:0]	**MADDR6**	**RSA DMA Middle Address Register6** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator.

6.36.7.7 Key Control Register

PRNG Key Control Register (CRYPTO_PRNG_KSCTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_PRNG_KSCTL	CRYPTO_BA+0xF00	W	PRNG Key Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved					OWNER		
23	22	21	20	19	18	17	16
WSDST		WDST	ECDSA	ECDH	PRIV	Reserved	TRUST
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			NUM				

Bits	Description	
[31:27]	**Reserved**	Reserved.
[26:24]	**OWNER**	**Write Key Owner Selection Bits** 000 = Only for AES use. 001 = Only for HMAC engine use. 100 = Only for ECC engine use. 101 = Only for CPU engine use. Others = reserved.
[23:22]	**WSDST**	**Write Key Store Destination** 00 = Key is written to the SRAM of key store. 10 = Key is written to the OTP of key store. Others = Reserved.
[21]	**WDST**	**Write Key Destination** 0 = Key is written to registers CRYPTO_PRNG_KEYx. 1 = Key is written to key store.

[20]	**ECDSA**	**ECDSA Control Bit** 0 = Reserved. 1 = Key is written to key store and used in ECDSA. **Note:** When ECDSA is set to '1', **1.** PRNG seed must come from TRNG and key must be written to the SRAM of key store (WSDST, CRYPTO_PRNG_KSCTL[23:22] must set to '00'). Otherwise, KCTLERR will become '1'(CRYPTO_PRNG_KSSTS[16]). **2.** Key must be in the interval [1, n-1] (the parameter n is from ECC). The value of n cannot be 0 or 1; otherwise, PRNG will always keep busy.
[19]	**ECDH**	**ECDH Control Bit** 0 = Reserved. 1 = Key is written to key store and used in ECDH. **Note:** When ECDH is set to '1', **1.** PRNG seed must come from TRNG and key must be written to the SRAM of key store (WSDST, CRYPTO_PRNG_KSCTL[23:22] must set to '00'). Otherwise, KCTLERR will become '1'(CRYPTO_PRNG_KSSTS[16]). **2.** Key must be in the interval [1, n-1] (the parameter n is from ECC). The value of n cannot be 0 or 1; otherwise, PRNG will always keep busy.
[18]	**PRIV**	**Privilege Key Selection Bit** 0 = Set key as the non-privilege key. 1 = Set key as the privilege key.
[17]	**Reserved**	Reserved.
[16]	**TRUST**	**Write Key Trust Selection Bit** 0 = Set written key as the non-secure key. 1 = Set written key as the secure key.
[15:5]	**Reserved**	Reserved.
[4:0]	**NUM**	**Write Key Number** The key number is sent to key store **Note:** Only for destination is OTP of Key Store.

PRNG Key Status Register (CRYPTO_PRNG_KSSTS)

Register	Offset	R/W	Description	Reset Value
CRYPTO_PRNG_KSSTS	CRYPTO_BA+0xF04	R	PRNG Key Status Register	0x0000_0000

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

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4:0]	**NUM**	**Key Number** The key number is generated by key store

AES Key Control Register (CRYPTO_AES_KSCTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_KSCTL	CRYPTO_BA+0xF10	W	AES Key Control Register	0x0000_0000

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:6]	**RSSRC**	**Read Key Store Source** 00 = Key is read from the SRAM of key store. 10 = Key is read from the OTP of key store. Others = Reserved.
[5]	**RSRC**	**Read Key Source** 0 = Key is read from registers CRYPTO_AESx_KEYx. 1 = Key is read from key store.
[4:0]	**NUM**	**Read Key Number** The key number is sent to key store

HMAC Key Control Register (CRYPTO_HMAC_KSCTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_KSCTL	CRYPTO_BA+0xF30	W	HMAC Key Control Register	0x0000_0000

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:6]	**RSSRC**	**Read Key Store Source** 00 = Key is read from the SRAM of key store. 10 = Key is read from the OTP of key store. Others = Reserved.
[5]	**RSRC**	**Read Key Source** 0 = Key is read from HMAC registers. 1 = Key is read from key store.
[4:0]	**NUM**	**Read Key Number** The key number is sent to key store

ECC Key Control Register (CRYPTO_ECC_KSCTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_KSCTL	CRYPTO_BA+0xF40	W	ECC Key Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved					OWNER		
23	22	21	20	19	18	17	16
WSDST		WDST	XY	Reserved	PRIV	Reserved	TRUST
15	14	13	12	11	10	9	8
Reserved	ECDH	Reserved					
7	6	5	4	3	2	1	0
RSSRCK		RSRCK	NUMK				

Bits	Description	
[31:27]	**Reserved**	Reserved.
[26:24]	**OWNER**	**Write Key Owner Selection Bits** 000 = ECDH written key is only for AES use. 001 = ECDH written key is only for HMAC engine use. 100 = ECDH written key is only for ECC engine use. 101 = ECDH written key is only for CPU engine use. Others = Reserved.
[23:22]	**WSDST**	**Write Key Store Destination** 00 = ECDH written key is written to the SRAM of key store. 10 = ECDH written key is written to the OTP of key store. Others = Reserved.
[21]	**WDST**	**Write Key Destination** 0 = ECDH written key is in registers CRYPTO_ECC_X1 and CRYPTO_ECC_Y. 1 = ECDH written key is written to key store.
[20]	**XY**	**ECDH Output Select Bit** 0 = ECDH written key is from X-coordinate value. 1 = ECDH written key is from Y-coordinate value.
[19]	**Reserved**	Reserved.
[18]	**PRIV**	**Write Key Privilege Selection Bit** 0 = Set ECDH written key as the non-privilege key. 1 = Set ECDH written key as the privilege key.
[17]	**Reserved**	Reserved.
[16]	**TRUST**	**Write Key Trust Selection Bit** 0 = Set ECDH written key as the non-secure key. 1 = Set ECDH written key as the secure key.

[15]	**Reserved**	Reserved.
[14]	**ECDH**	**ECDH Control Bit** 0 = Reserved. 1 = Set ECC opereration is in ECDH. When this bit and RSRCK are equal to 0x1, ECC will read ECDH private key to CRYPTO_ECC_K from key store.
[13:8]	**Reserved**	Reserved.
[7:6]	**RSSRCK**	**Read Key Store Source for Key Number K** 00 = Key is read from the SRAM of key store. 10 = Key is read from the OTP of key store. Others = Reserved.
[5]	**RSRCK**	**Read Key Source for Key Number K** 0 = Key is read from ECC registers. 1 = Key is read from key store.
[4:0]	**NUMK**	**Read Key Number K** The key number of CRYPTO_ECC_K is sent to key store when RSRCK =1.

ECC Key Status Register (CRYPTO_ECC_KSSTS)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_KSSTS	CRYPTO_BA+0xF44	R	ECC Key Status Register	0x0000_0000

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

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4:0]	**NUM**	**Key Number** The key number is generated by key store after ECDH.

ECC XY Number Register (CRYPTO_ECC_KSXY)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_KSXY	CRYPTO_BA+0xF48	W	ECC XY Number Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
RSSRCY		Reserved	NUMY				
7	6	5	4	3	2	1	0
RSSRCX		RSRCXY	NUMX				

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:14]	**RSSRCY**	**Read Key Store Source for Key Number Y** 00 = Key is read from the SRAM of key store. 10 = Key is read from the OTP of key store. Others = Reserved.
[13]	**Reserved**	Reserved.
[12:8]	**NUMY**	**Read Key Number Y** The key number of CRYPTO_ECC_Y1 is sent to key store when RSRCXY =1.
[7:6]	**RSSRCX**	**Read Key Store Source for Key Number X** 00 = Key is read from the SRAM of key store. 10 = Key is read from the OTP of key store. Others = Reserved.
[5]	**RSRCXY**	**Read Key Source for Key Number x and Y** 0 = key is read from ECC registers. 1 = key is read from key store.
[4:0]	**NUMX**	**Read Key Number X** The key number of CRYPTO_ECC_X1 is sent to key store when RSRCXY =1.

RSA Key Control Register (CRYPTO_RSA_KSCTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_KSCTL	CRYPTO_BA+0xF50	W	RSA Key Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved			BKNUM				
7	6	5	4	3	2	1	0
RSSRC		RSRC	NUM				

Bits	Description	
[31:13]	**Reserved**	Reserved.
[12:8]	**BKNUM**	**Read Exponent Blind Key Number** The key number is sent to key store, and its destination always be the SRAM of key store. CPU can't read the exponent blind key. **Note:** Use this key number only when executing SCA protection but no-CRT mode.
[7:6]	**RSSRC**	**Read Key Store Source** 00 = Key is read from the SRAM of key store. Others = Reserved.
[5]	**RSRC**	**Read Key Source** 0 = key is read from RSA engine. 1 = key is read from key store.
[4:0]	**NUM**	**Read Key Number** The key number is sent to key store

RSA Key Status 0 Register (CRYPTO_RSA_KSSTS0)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_KSSTS0	CRYPTO_BA+0xF54	R/W	RSA Key Status Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			NUM3				
23	22	21	20	19	18	17	16
Reserved			NUM2				
15	14	13	12	11	10	9	8
Reserved			NUM1				
7	6	5	4	3	2	1	0
Reserved			NUM0				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**NUM3**	**Key Number3** The key number is generated by key store, RSA can get or store the intermediate temporary value by key number in Key Store while operating. **Note:** The size of this key as key length.
[23:21]	**Reserved**	Reserved.
[20:16]	**NUM2**	**Key Number2** The key number is generated by key store, RSA can get or store the intermediate temporary value by key number in Key Store while operating. **Note:** The size of this key as key length.
[15:13]	**Reserved**	Reserved.
[12:8]	**NUM1**	**Key Number1** The key number is generated by key store, RSA can get complete q by key number in Key Store while operating. **Note:** The size of this key as half key length.
[7:5]	**Reserved**	Reserved.
[4:0]	**NUM0**	**Key Number0** The key number is generated by key store, RSA can get complete p by key number in Key Store while operating. **Note:** The size of this key as half key length.

RSA Key Status 1 Register (CRYPTO_RSA_KSSTS1)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_KSSTS1	CRYPTO_BA+0xF58	R/W	RSA Key Status Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			NUM7				
23	22	21	20	19	18	17	16
Reserved			NUM6				
15	14	13	12	11	10	9	8
Reserved			NUM5				
7	6	5	4	3	2	1	0
Reserved			NUM4				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**NUM7**	**Key Number7** The key number is generated by key store, RSA can get or store the intermediate temporary value by key number in Key Store while operating. **Note:** The size of this key as key length.
[23:21]	**Reserved**	Reserved.
[20:16]	**NUM6**	**Key Number6** The key number is generated by key store, RSA can get or store the intermediate temporary value by key number in Key Store while operating. **Note:** The size of this key as key length.
[15:13]	**Reserved**	Reserved.
[12:8]	**NUM5**	**Key Number5** The key number is generated by key store, RSA can get or store the intermediate temporary value by key number in Key Store while operating. **Note:** The size of this key as half key length.
[7:5]	**Reserved**	Reserved.
[4:0]	**NUM4**	**Key Number4** The key number is generated by key store, RSA can get or store the intermediate temporary value by key number in Key Store while operating. **Note:** The size of this key as half key length.

6.37 Enhanced 12-bit Analog-to-Digital Converter (EADC)

6.37.1 Overview

The chip contains one 12-bit successive approximation analog-to-digital converter (SAR ADC converter) with 16 external input channels and 3 internal channels. The ADC converter can be started by software trigger, EPWM0/1 triggers, BPWM0/1 triggers, Timer0~5 overflow pulse triggers, ADINT0, ADINT1 interrupt EOC (End of conversion) pulse trigger and external pin (EADC0_ST) input signal.

6.37.2 Features

- Analog input voltage range: 0~V_{REF} (Max to 3.6V)
- Reference voltage from V_{REF} pin
- 12-bit resolution and 10-bit accuracy is guaranteed
- Up to 16 single-end analog external input channels or 8 pair differential analog input channels
- Up to 3 internal channels, they are band-gap voltage (V_{BG}), temperature sensor (V_{TEMP}), and Battery power (V_{BAT})
- Four ADC interrupts (ADINT0~3) with individual interrupt vector addresses
- Maximum ADC clock frequency is 80 MHz
- Up to 5.71 MSPS conversion rate
- Configurable ADC internal sampling time.
- 12-bit, 10-bit, 8-bit, 6-bit configurable resolution.
- Supports calibration and load calibration words capability.
- Supports internal reference voltage V_{REF}: 1.6V, 2.0V, 2.5V, and 3.0V.
- Supports three power saving modes:
  - Deep Power-down mode
  - Power-down mode
  - Standby mode
- Up to 19 sample modules:
  - Each of sample modules which is configurable for ADC converter channel EADC_CH0~15 and trigger source
  - Sample module 16~18 is fixed for ADC channel 16, 17, 18 input sources as band-gap voltage, temperature sensor, and battery power (V_{BAT})
  - Double buffer for sample control logic module 0~3
  - Configurable sampling time for each sample module
  - Conversion results are held in 19 data registers with valid and overrun indicators
- An ADC conversion can be started by:
  - Write 1 to SWTRG (EADC_SWTRG[n], n = 0~18)
  - External pin EADC0_ST
  - Timer0~5 overflow pulse triggers
  - ADINT0 and ADINT1 interrupt EOC (End of conversion) pulse triggers

- EPWM/BPWM triggers
- Supports PDMA transfer
- Conversion Result Monitor by Compare Mode

6.37.3 Block Diagram

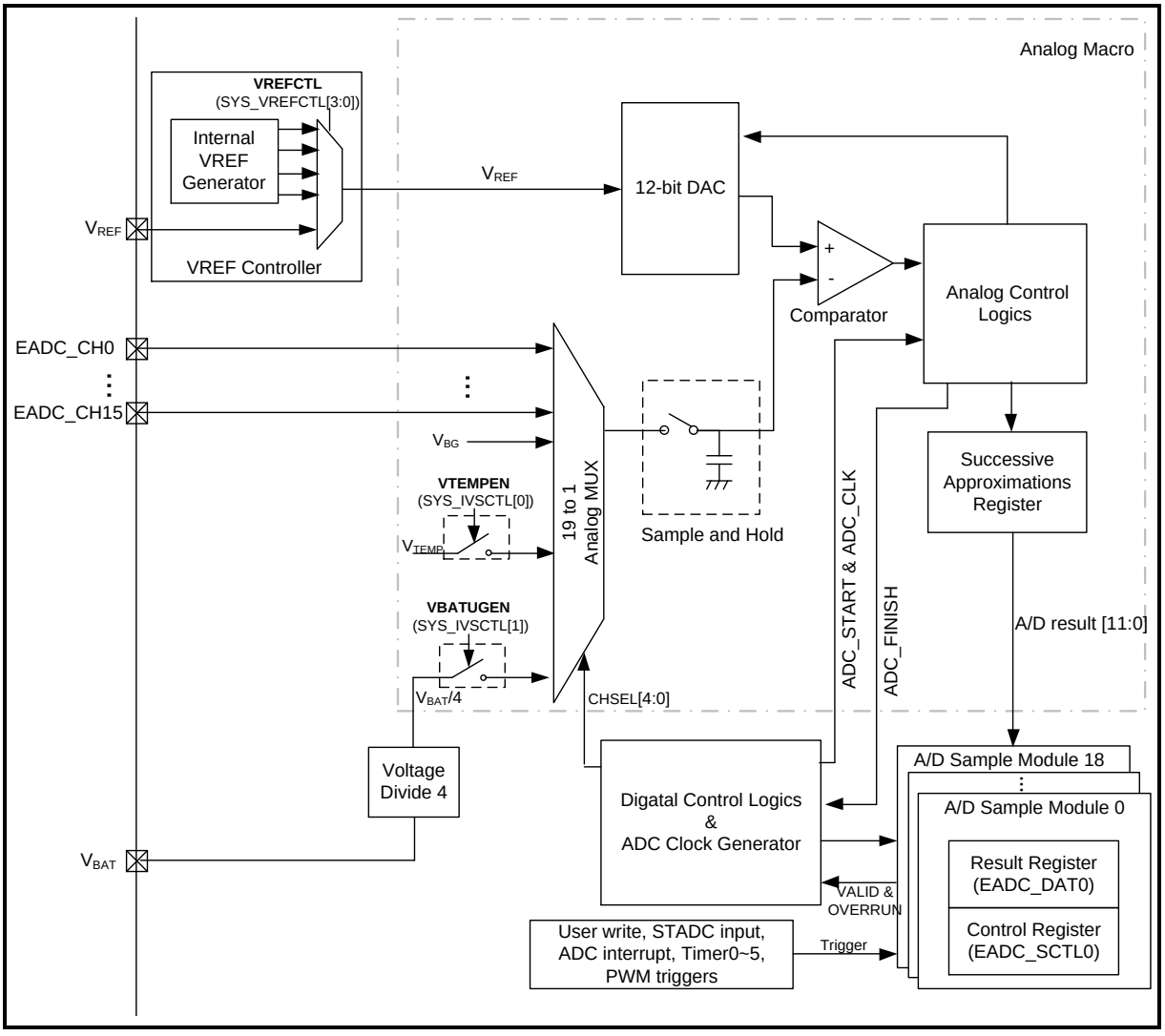


Figure 6.37-1 ADC Converter Block Diagram

6.37.4 Basic Configuration

- Clock Source Configuration
  - Select the clock divider number on EADCDIV (CLK_CLKDIV0[23:16])
  - Enable EADC peripheral clock in EADCCKEN (CLK_APBCLK0[28]).
- Reset Configuration
  - Reset EADC controller in ADCRST (EADC_CTL [1]).

- Reset EADC controller in EADCRST(SYS_IPRST1[28]).

- Pin Configuration

Group	Pin Name	GPIO	MFP
EADC0	EADC0_CH0	PB.0	MFP1
	EADC0_CH1	PB.1	MFP1
	EADC0_CH2	PB.2	MFP1
	EADC0_CH3	PB.3	MFP1
	EADC0_CH4	PB.4	MFP1
	EADC0_CH5	PB.5	MFP1
	EADC0_CH6	PB.6	MFP1
	EADC0_CH7	PB.7	MFP1
	EADC0_CH8	PB.8	MFP1
	EADC0_CH9	PB.9	MFP1
	EADC0_CH10	PB.10	MFP1
	EADC0_CH11	PB.11	MFP1
	EADC0_CH12	PB.12	MFP1
	EADC0_CH13	PB.13	MFP1
	EADC0_CH14	PB.14	MFP1
	EADC0_CH15	PB.15	MFP1
	EADC0_ST	PF.5	MFP11
		PC.13, PD.12	MFP14
		PG.15	MFP15

6.37.5 Functional Description

The EADC controller consists of a 19 channel analog switch, 19 sample modules and a 12-bit successive approximation analog-to-digital converter. The EADC operation is based on sample module 0~18, and each of them has its configuration to decide which trigger source to start the conversion, which channel to convert. Sample module 0~15 can be configured to EADC_CH0~15 channel, and different trigger source. It provides user a flexible means to get the over-sampling results. The sample module 0~3 and sample module 4~15 are shown as follows.

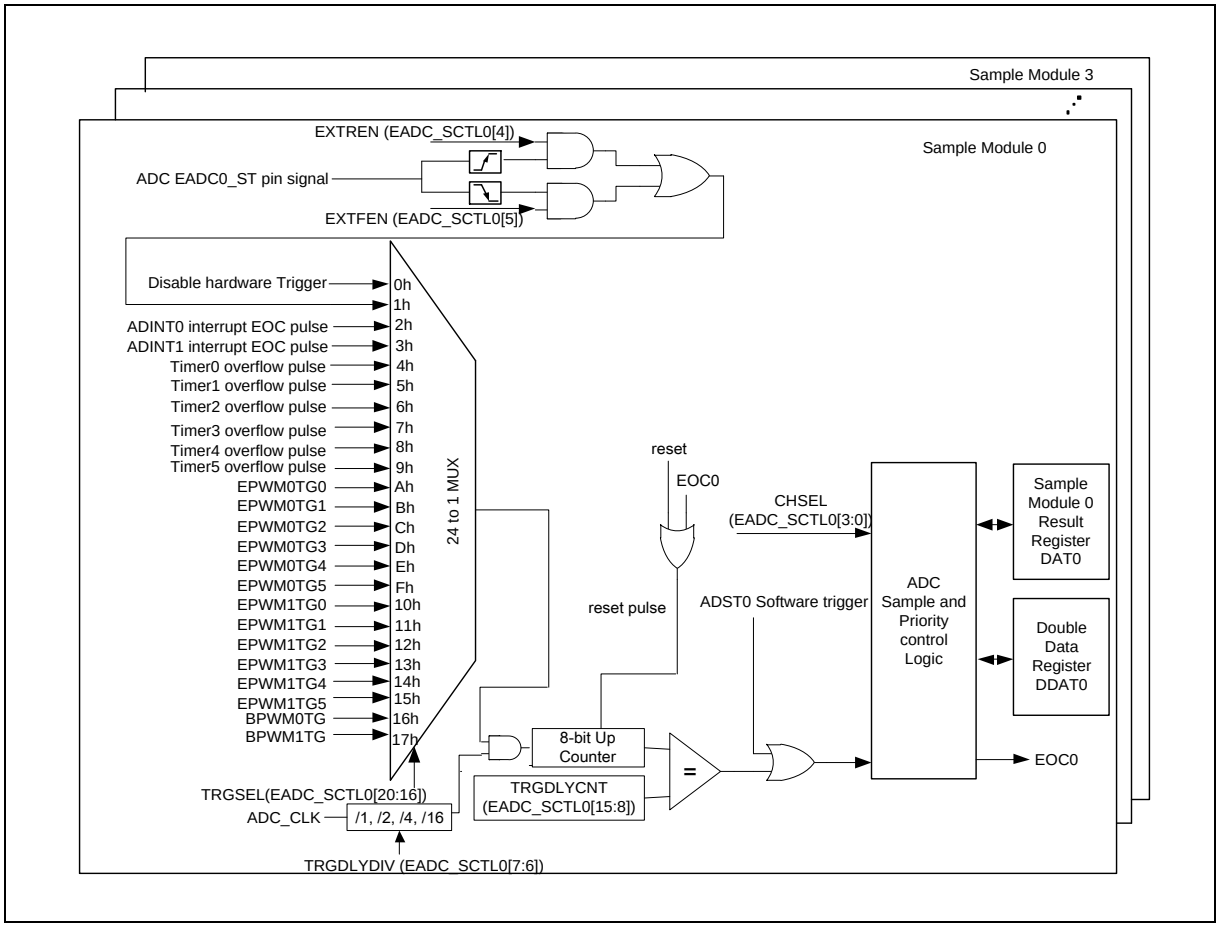


Figure 6.37-2 Sample Module 0~3 Block Diagram

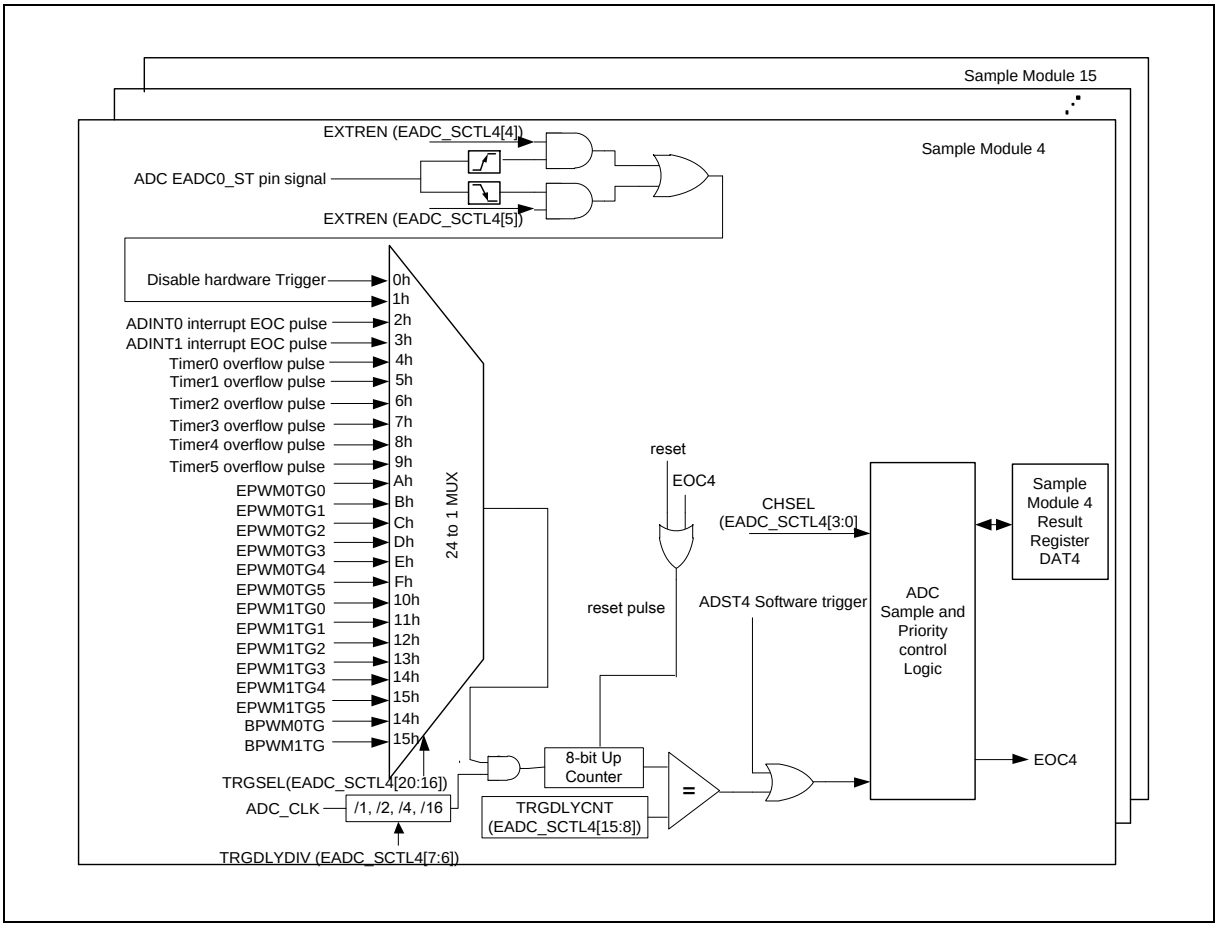


Figure 6.37-3 Sample Module 4~15 Block Diagram

Sample module 16~18 can convert internal channel (V_{BG}, V_{TEMP}, V_{BAT}) and can be triggered by user write SWTRG (EADC_SWTRG[n], n = 16~18). Figure 6.37-4 shows the sample module 16~18.

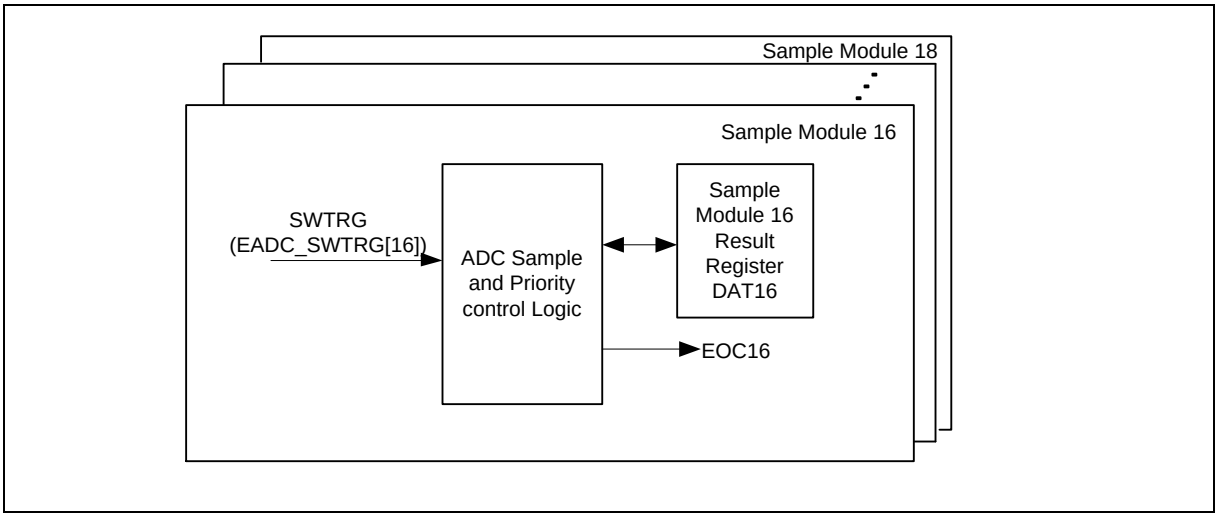


Figure 6.37-4 Sample Module 16~18 Block Diagram

The ADC conversion trigger sources in sample module 0~15 are listed below:

- Write 1 to SWTRG (EADC_SWTRG[n], n = 0~15)
- External pin EADC0_ST
- Timer0~5 overflow pulse triggers
- ADINT0, ADINT1 ADC interrupt EOC (End of conversion) pulse triggers
- EPWM/BPWM triggers

The ADINT0 or ADINT1 interrupt pulses are generated whenever the specific sample module ADC EOC (End of conversion) pulse is generated. ADINT0 or ADINT1 interrupt pulse triggers can be fed back to trigger another ADC conversion, and is useful if a continuous scan conversion is needed.

6.37.5.1 *ADC Clock Generator*

The maximum ADC clock frequency is up to 80 MHz and the maximum sampling rate is up to 5.71 MSPS.

The clock control of EADC is shown as Figure 6.37-5. The EADC peripheral clock source is from PCLK1 clock, the EADC clock frequency is divided by an 8-bit pre-scalar with the following formula:

EADC clock frequency = (PCLK1) / (EADCDIV (CLK_CLKDIV0[23:16])+1)

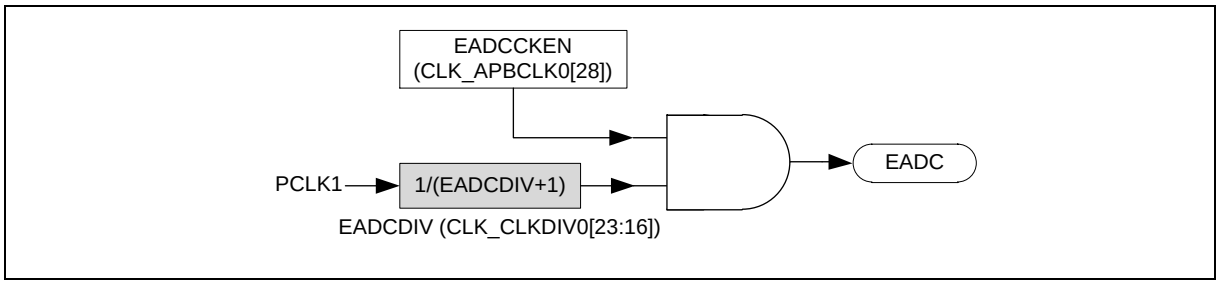


Figure 6.37-5 EADC Clock Control

6.37.5.2 *ADC Software Trigger Mode*

When an ADC conversion is performed on the sample module specified single channel, the operations are as follows:

1. ADC conversion is started when the SWTRG (EADC_SWTRG[n], n=0~18) is set to 1 by user or other trigger inputs.
2. When ADC conversion is finished, the 12-bit result is stored in the ADC data register EADC_DATn (n=0~18) corresponding to the sample module.
3. On completion of conversion, the ADIFn (EADC_STATUS2[3:0], n=0~3) is set to 1 and ADC interrupt (ADINTn, n=0~3) is requested if the ADCIENn (EADC_CTL[5:2], n=0~3) bit is set to 1.
4. The SWTRG (EADC_SWTRG[n], n=0~18) bit remains 1 during ADC conversion. When ADC conversion ends, the SWTRG (EADC_SWTRG[n], n=0~18) bit is automatically cleared to 0 and the ADC converter will do another pending conversion.

The timing diagram of a conversion cycle is shown in Figure 6.37-6.

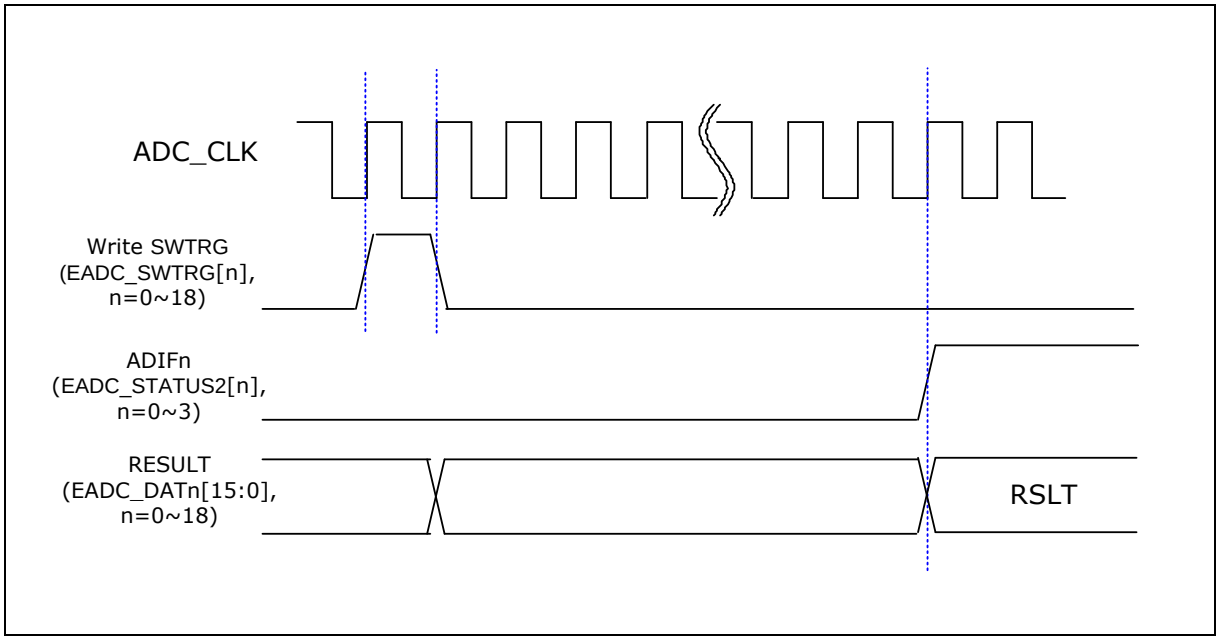


Figure 6.37-6 Example ADC Conversion Timing Diagram, n=0~18

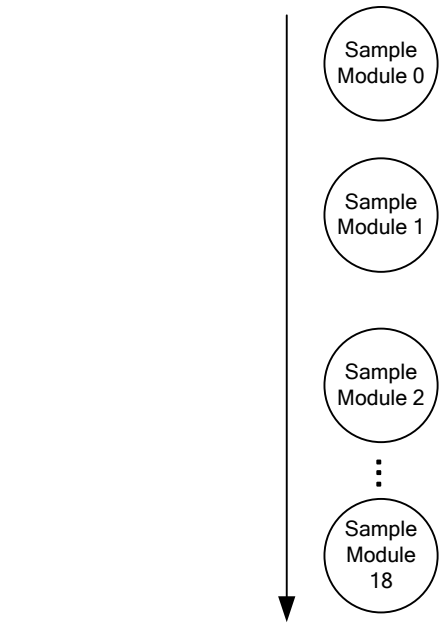
If more than one sample module is enabled to convert analog signal, the sample module specified channel with highest priority is firstly converted and other enabled sample module will be pended. The lower number sample module has higher priority. The sample module 0 is highest priority and the sample module 18 is lowest priority.

Note: If the interval between next conversion is more than 100 us, ADC would enter idle state automatically. User needs to execute a dummy conversion before normal operation. In other words, the first conversion result is incorrect when ADC is in idle state.

6.37.5.3 *ADC Conversion Priority*

There is a priority group converter for determining the conversion order when multiple sample module trigger flags are set at the same time. Sample module with lower number has higher priority than the higher number sample module. The priority of sample module is shown as Figure 6.37-7. When more than one Sample Module are triggered at the same time, the Sample Module with lower number will start to convert first. The other Sample Module will be in the queue and the corresponding pending flag STPF(EADC_PENDSTS[n], n=0~18) are set to 1 by HW. After the Sample Module finish the conversion, STPF(EADC_PENDSTS[n], n=0~18) will be set to 0 automatically. If the Sample Module which is in the queue is triggered once more, the corresponding Overrun Flag SPOVF(EADC_OVSTS[n], n=0~18) will be set to 1 by HW.

For example, the Sample Module 0, 2, 3, 5 are triggered simultaneously. The input channel of Sample Module 0 will be converted first. Sample Module 2, 3, 5 will be suspended and STPF (EADC_PENDSTS[2], EADC_PENDSTS [3], EADC_PENDSTS [5]) will be set to 1. If Sample Module 5 is trigger once more in the same time, SPOVF(EADC_OVSTS[5]) will be set to 1.



Sample module 0 has highest priority, sample module 18 has lowest priority for ADC converter

Figure 6.37-7 Sample Module Conversion Priority Arbitrator Diagram

6.37.5.4 Conversion Cycles and Sampling Rate Frequency

There are four kinds of resolutions which could be configured by RESSEL (EADC_CTL[7:6]). Each resolution corresponds to different conversion cycles. The relation is as Table 6.37-1.

Resolution	Minimum Conversion Cycles
6 bit	8 ADC_CLK
8 bit	10 ADC_CLK
10 bit	12 ADC_CLK
12 bit	14 ADC_CLK

Table 6.37-1 Relation between Resolution and Conversion Cycles

There are two kinds of analog input channels which are fast and slow channel. EADC_CH10~15 are fast channel and EADC_CH0~9 are slow channel. The maximum sampling rate of fast channel is 5.71 MSPS and slow channel is 2.14 MSPS. Exceed the limitation of sampling frequency will cause wrong conversion results. The sampling rate frequency can be computed with the following formula:

Sampling rate frequency = (EADC clock frequency) / (conversion cycles+ ADC sampling time extend)

Note: ADC sampling time extend is the value of EXTSMPT (EADC_SCTLn[31:24], n=0~15)

6.37.5.5 *Maximum Sampling Frequency Conversion by Software Trigger*

If user needs to scan the fast channel at maximum sampling frequency, the conversion needs to be executed by the condition as: multiple sample modules, triggered by software, and triggered repeatedly during the last conversion. An example of continuous scan is as follows:

- Using Module 0~15 to carry out successive conversion. Set CHSEL (EADC_SCTL0~15[3:0]) as one of fast channel (EADC_CH10~ EADC_CH15). Set EXTSMPT (EADC_SCTL0~15[31:24]) and TRGDLYCNT (EADC_SCTL0~15[15:8]) as 0x00 to minimize the sampling time.
- Set SWTRG (EADC_SWTRG[18:0]) as 0xffff to trigger Module 0~15.
- Wait CURSPL (EADC_STATUS3[4:0]) changes to 0xf which means Module 0~14 have been executed and Module 15 is in the process. Set SWTRG (EADC_SWTRG[18:0]) as 0x7fff to trigger Module 0~14 again for next round.
- Wait CURSPL (EADC_STATUS3[4:0]) changes to 0x1, set SWTRG (EADC_SWTRG[18:0]) as 0x8000 to trigger Module 15.
- Repeat Step 3~4 to continue the conversion.

6.37.5.6 *ADC Sample Module End of Conversion Interrupt Operation*

There are 4 ADC interrupts ADINT0~3, and each of these interrupts has its own interrupt vector address and can be configured to set multiple sample module EOC pulse (sample module 0~18 End of conversion pulses) as its interrupt trigger source. Figure 6.37-8 shows the control logic of interrupts. Take ADINT0 as an example, when ADCIEN0 (EADC_CTL[2]) = 1 and SPLIEn (EADC_INTSRC0[n]) = 1 (n=0~18), the specific module EOC (End of conversion) pulses will set flag ADIF0 (EADC_STATUS2[0]) as 1 and interrupt (ADINT0) will be asserted either.

The interrupt pulses (ADINT0/1) are generated whenever the specific sample module ADC EOC pulse is generated. It also can be the sample module conversion trigger sources, and user can use it to do the ADC continuous scan conversion.

The example of continuous scan triggered by interrupt is as follows:

1. If ADC sample module 2 EOC2 pulse is selected as ADINT0 interrupt trigger SPLIE2 (EADC_INTSRC0[2]) = 1 and ADINT0 is selected as sample module 0, 1, 2 hardware conversion trigger.
2. Set software trigger SWTRG (EADC_SWTRG[2]) to 0x4 to start a sample module 2 ADC conversion, after the conversion completes, it generates an EOC2 pulse signal and ADINT0 interrupt pulse at end of sample module 2 ADC conversion, ADINT0 interrupt pulse will trigger the sample module 0, 1, 2 to start the ADC conversions.
3. ADINT0 interrupt pulse repeats to trigger sample module 0, 1, 2 ADC conversions automatically.
4. Clear TRGSEL (EADC_SCTL2[20:16]) to 0 to disable sample module 2 ADINT0 interrupt pulse hardware trigger, if needs to stop the continuous scan.

Note: Because the system takes 3 ADC_CLK to trigger the next module by interrupt pulse, the average conversion cycles of continuous scan triggered by interrupt is 17 ADC_CLK.

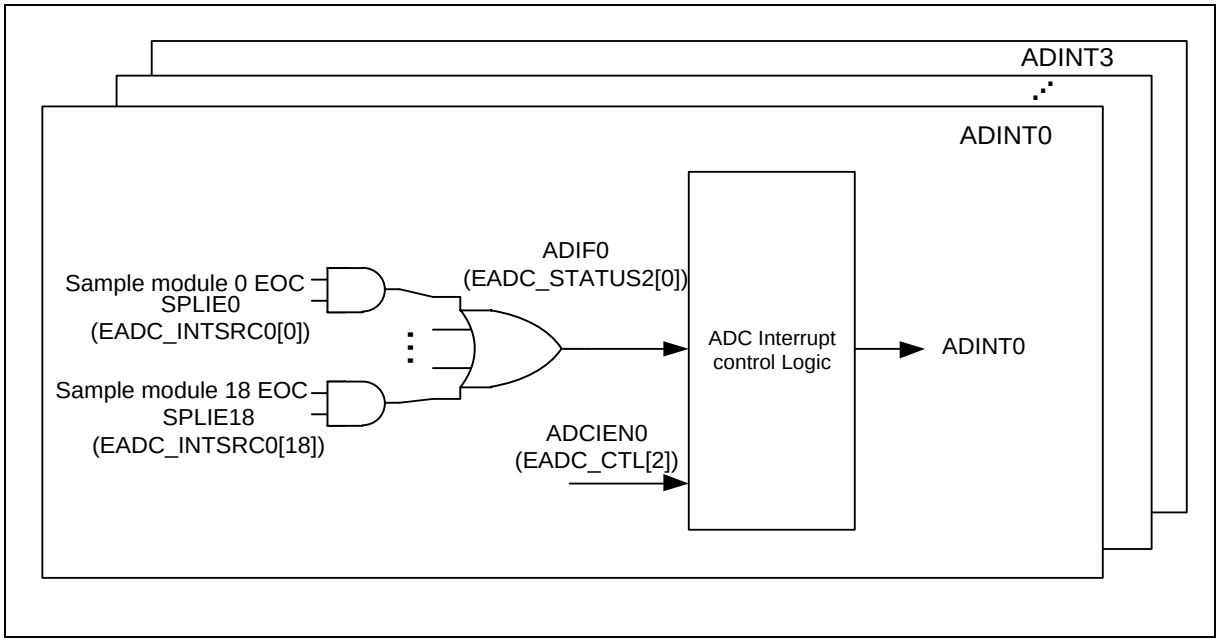


Figure 6.37-8 Specific Sample Module ADC EOC Signal for ADINT0~3 Interrupt

6.37.5.7 *ADC Trigger by Timer Trigger and External Pin EADC0_ST*

There are 6 Timer trigger sources and an external pin EADC0_ST that can configure sample module 0~15 to trigger ADC start when timer overflow occurs.

ADC conversion can be triggered by external pin EADC0_ST request. Setting the TRGSEL (EADC_SCTLn[20:16], n=0~15) to 0x01 is to select external trigger input from the EADC0_ST pin. User can set EXTFEN (EADC_SCTLn[5], n=0~15) and EXTREN (EADC_SCTLn[4], n=0~15) to enable pin EADC0_ST trigger condition is falling or rising edge. There is a de-bounce circuit to detect falling or rising edge. If rising edge trigger condition is selected, the low state must be kept at least 2 PCLK cycles and the following high state must be kept at least 3 PCLK cycles. If falling edge trigger condition is selected, the high state must be kept at least 2 PCLK cycles and the following low state must be kept at least 3 PCLK cycles. Pulse that is shorter than this specification will be ignored. The external trigger timing is shown in Figure 6.37-9.

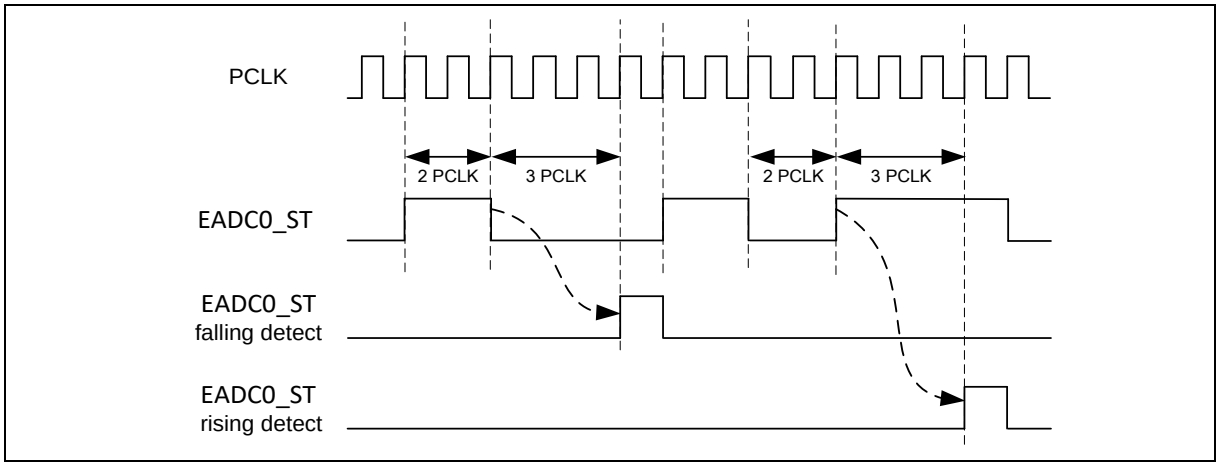


Figure 6.37-9 EADC0_ST De-bounce Timing Diagram

6.37.5.8 *ADC Start Synchronous with EPWM/BPWM Trigger*

Besides user start, ADINT0/1 interrupt pulse, external pin EADC0_ST and Timer0~5 overflow pulse to start ADC conversion, this device has new feature to allow EPWM/BPWM channels to trigger the ADC start. User may configure EPWM/BPWM trigger types: rising, falling EPWM/BPWM edge or center point of EPWM/BPWM (center-aligned mode only) to trigger ADC start. The device also allows user to configure the amount of delay period to ADC start after hardware detected the external trigger. User can configure the trigger delay time by setting TRGDLYCNT (EADC_SCTLn[15:8], n=0~15) and TRGDLYDIV (EADC_SCTLn[7:6], n=0~15). Figure 6.37-10 shows the programmable delay time for EPWM/BPWM-triggered ADC start conversion.

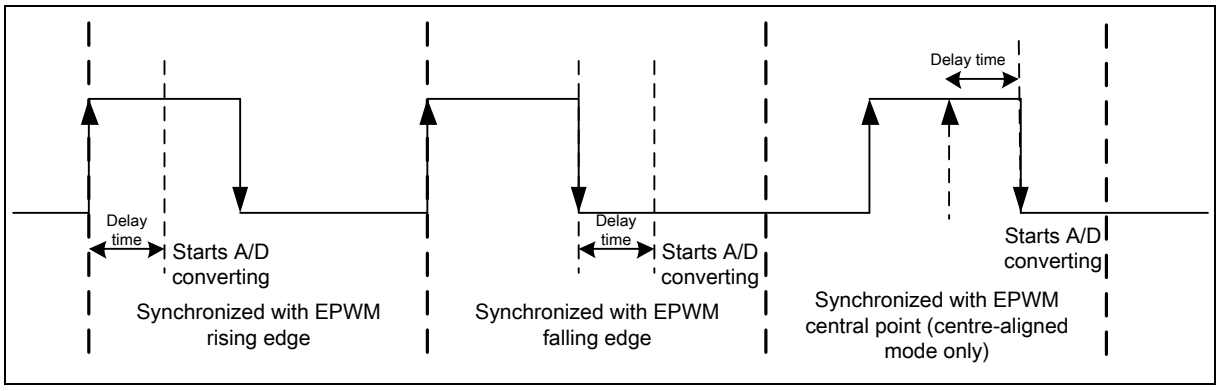


Figure 6.37-10 EPWM-triggered ADC Start Conversion

Figure 6.37-11 shows the programmable delay time for other trigger source.

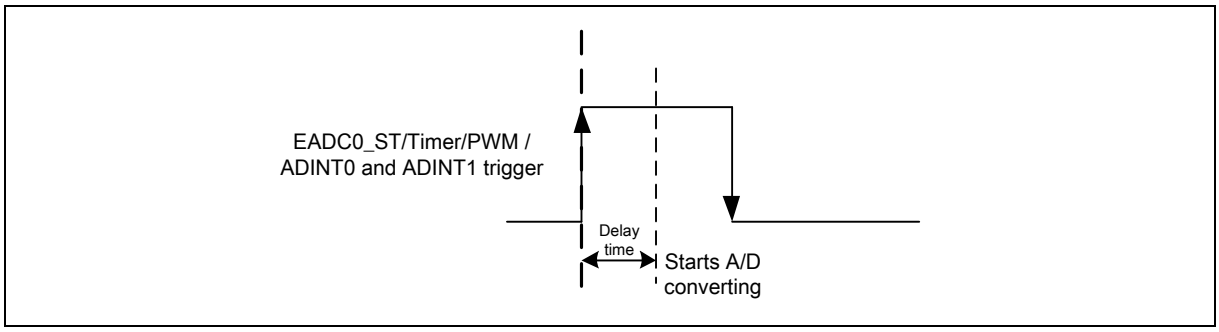


Figure 6.37-11 External Triggered ADC Start Conversion

6.37.5.9 *Input Sampling and ADC Conversion Time*

The ADC converter sample the analog input when ADC conversion start delay time (Td) has passed after SWTRG (EADC_SWTRG[n], n=0~18) is set to 1, then start conversion. Due to ADC clock is generated by PCLK divided by (EADCDIV(CLK_CLKDIV0 [23:16])+1), the maximum delay time from user write SWTRG to ADC start sampling analog input time is two ADC clock cycles. The start delay time is shown in Figure 6.37-12.

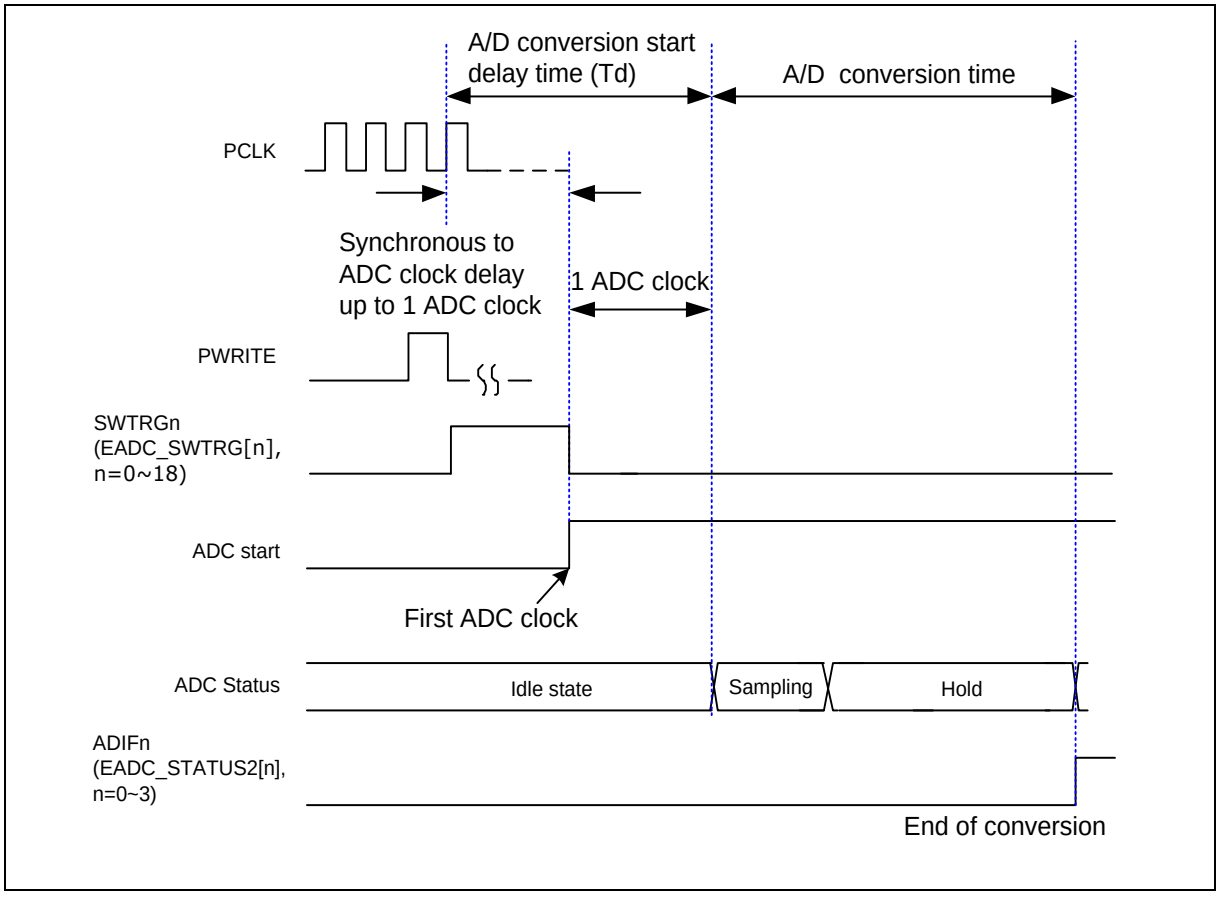
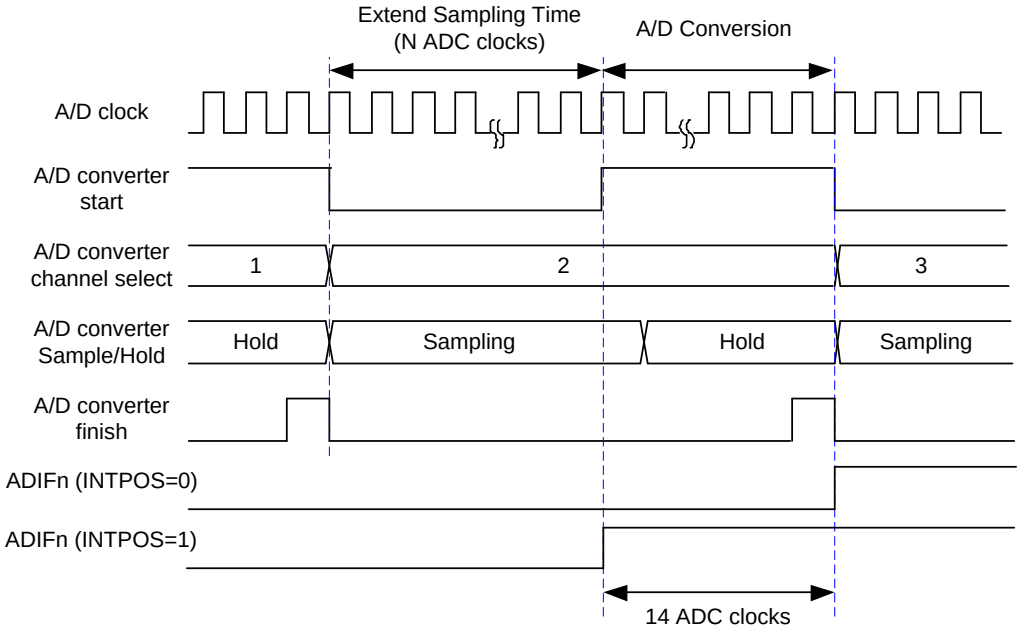


Figure 6.37-12 Conversion Start Delay Timing Diagram

6.37.5.10 *ADC Extend Sampling Time*

When ADC operates at high ADC clock rate, the sampling time of analog input voltage may not be enough if the analog channel has heavy loading to cause fully charge time is longer. User can set extend sampling time by writing EXTSMPT (EADC_SCTLn[31:24], n=0~15) for each sample module. The ADC extend sampling time is present between ADC controller judging which channel to be converted and ADC starting conversion. The range of extend sampling time is from 0 ~255 ADC clock. The extended sampling time is shown in Figure 6.37-13.



Note: N = EXTSMPT (EADC_SCTLn[31:24], n=0~18) (Extend sampling cycle for each sample module)
ADIFn (EADC_STATUS2[3:0], n=0~3)
INTPOS (EADC_SCTLn[22], n=0~15)

Figure 6.37-13 ADC Extend Sampling Timing Diagram

6.37.5.11 *Conversion Result Monitor by Compare Mode*

The ADC controller provides four sets of compare registers EADC_CMP0 ~ EADC_CMP3 to monitor a maximum of four specified sample module 0~18 conversion results from ADC conversion module, as shown in Figure 6.37-14. User can select which sample module result to be monitored by set CMPSPL (EADC_CMPn[7:3], n =0~3) and CMPCOND (EADC_CMPn[2], where n =0~3) is used to check conversion result is less than specify value or greater than (equal to) value specified in CMPDAT (EADC_CMPn[27:16], where n =0~3). When the conversion of the sample module specified by CMPSPL is completed, the comparing action will be triggered one time automatically. When the compare result meets the compare condition, the internal compare match counter will increase 1. If the compare result does not meet the condition, the compare match counter will reset to 0. When counter value reach the setting of (CMPMCNT (EADC_CMPn[11:8])+1, where n =0~3) then ADCMPFn (EADC_STATUS2[7:4], where n =0~3) bit will be set to 1, if ADCMPIE (EADC_CMPn[1] , n =0~3) is set then an ADINT3 interrupt request is generated. User can use it to monitor the external analog input pin voltage transition. Detailed logics diagram is shown in Figure 6.37-14.

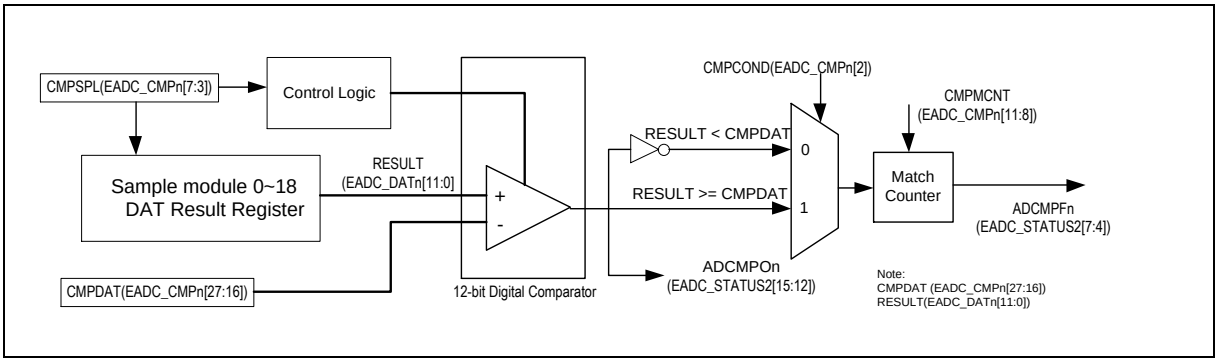


Figure 6.37-14 ADC Conversion Result Monitor Logics Diagram

The ADC controller supports a window compare mode. User can set CMPWEN (EADC_CMP0[15]/ EADC_CMP2[15]) to enable this function. If user enables this function, ADCMPF0 (EADC_STATUS2[4]) will be set when both EADC_CMP0 and EADC_CMP1 compared condition matched. ADCMPF2 (EADC_STATUS2[6]) will be set when both EADC_CMP2 and EADC_CMP3 compared condition matched.

6.37.5.12 *Differential Mode*

The ADC controller supports analog differential mode. If user enables DIFFEN (EADC_CTL[8]), the differential mode will enable. The pair of analog input channel is as Table 6.37-2.

Differential analog input voltage (Vdiff) = Vplus - Vminus, where Vplus is the analog input; Vminus is the inverted analog input.

Differential Analog Input Paired Channel	ADC Analog Input	
	V_{plus}	V_{minus}
0	EADC_CH0	EADC_CH1
1	EADC_CH2	EADC_CH3
2	EADC_CH4	EADC_CH5
3	EADC_CH6	EADC_CH7
4	EADC_CH8	EADC_CH9
5	EADC_CH10	EADC_CH11
6	EADC_CH12	EADC_CH13
7	EADC_CH14	EADC_CH15

Table 6.37-2 EADC Differential Model Channel Selection

In differential analog input mode, only the even number of the two corresponding channels needs to be enabled in CHSEL (EADC_SCTLn[3:0]). The conversion result will be placed to the corresponding data register of the enabled channel. The conversion result will store with 2'complement format when DMOF (EADC_CTL[9]) = 1.

6.37.5.13 *Double Buffer Mode*

The ADC controller supports a double buffer mode in sample module 0~3. If user enable DBMEN (EADC_SCTLn[23], n=0~3), the double buffer mode will enable. In double buffer mode, after first time ADC convert finish, the VALID (EADC_DATn[17], n=0~3) will set to high, but VALID

(EADC_DDATn[17], n=0~3) will keep low. And the second time ADC converts finish, VALID (EADC_DDATn[17], n=0~3) will set to high either. Then, user can get the ADC results from EADC_DATn and EADC_DDATn register.

6.37.5.14 *PDMA Request*

The ADC controller supports PDMA. PDMA could service each channel when corresponding channel PDMA transfer enable bit, PDMATEN (EADC_PDMACTL[18:0]), is activated. For example, user can enable PDMATEN for specific channels and configure PDMA channel's source address as EADC_CURDAT (EADC_BA+0x4C). After enabling PDMATEN and PDMA channel, if any VALID (EADC_DATn[17],n=0~18) is high, EADC controller will send request to PDMA and PDMA will read EADC_CURDAT to get result. The EADC_CURDAT register is a shadow register of highest priority EADC_DAT register. The lower number sample module is higher priority. After PDMA read EADC_CURDAT register, the VAILD of the EADC_DAT register will be automatically cleared.

6.37.5.15 *Interrupt Sources*

The ADC converter generates ADIFn (EADC_STATUS2[3:0], n=0~3) at the start of conversion or the end of conversion decide by INTPOS (EADC_SCTLn[22], n=0~15). If ADCIENn (EADC_CTL[5:2], n=0~3) is set then conversion end interrupt request ADINTn (n=0~3) is generated. The controller of interrupts is shown as Figure 6.37-15.

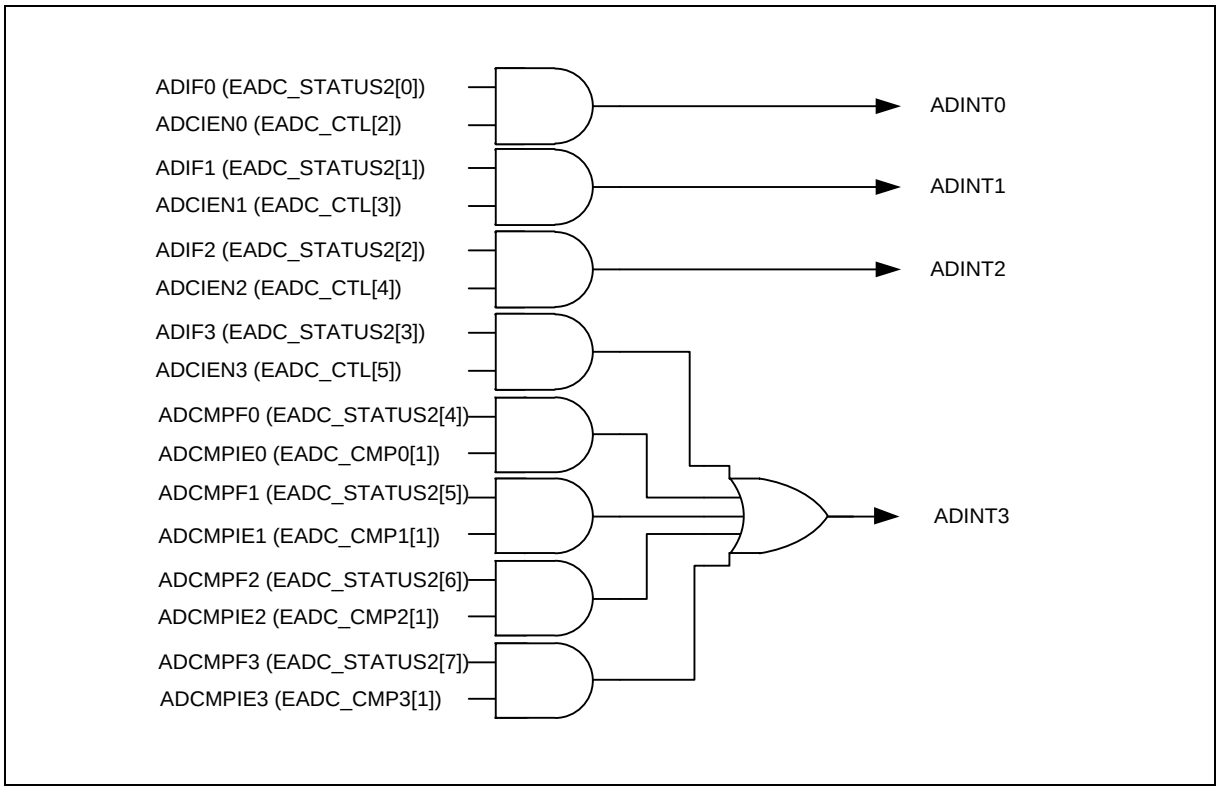


Figure 6.37-15 ADC Controller Interrupts

6.37.5.16 *Power Management and Calibration*

There are three kinds of power saving modes including Deep Power-down, Power-down, and Standby. User may set PWDMOD (EADC_PWRM[3:2]) to select which power saving mode EADC would enter when ADCEN (EADC_CTL[0]) is set as 0. The difference of these Power-down mode is shown as Table 6.37-3. Because the internal LDO will be shut down in Deep Power-down and Power-down mode, EADC needs to take extra time to resume. The interval of time to resume is set by LDOSUT (EADC_PWRM[19:8]) which must be longer than 20 us. As for the Standby mode, LDO will keep

enable and start-up time is unnecessary.

Power Supplies	Deep Power-Down	Power-Down	Standby
Internal LDO	Disable	Disable	Enable
Internal power switch	Disable	Enable	Enable

Table 6.37-3 EADC Power Saving Mode

When EADC is activated by setting ADCEN(EADC_CTL[0]) to 1, the start up sequence will execute automatically. After start up sequence finished, PWUPRDY (EADC_PWRM[0]) will be set to 1 by HW which means ready to convert. ADCEN (EADC_CTL[0]) must be kept at 1 until PWUPRDY (EADC_PWRM[0]) is set to 1 during the start up sequence. Changing ADCEN (EADC_CTL[0]) arbitrarily at start up sequence will cause EADC function failure.

The conversion results of ADC will be more accurate with calibration. User may set PWUCALEN (EADC_PWRM[1]) as 1 to carry out calibration at start up. This bit needs to cooperate with CALSEL (EADC_CALCTL [3]), the configuration of {PWUCALEN, CALSEL} is shown as Table 6.37-4. An example about start up with calibration is shown as Figure 6.37-16.

PWUCALEN	CALSEL	Configuration
0	0	Start up without calibration
0	1	Start up without calibration
1	0	Load calibration word at start up
1	1	Start up with calibration

Table 6.37-4 EADC Start up with Calibration

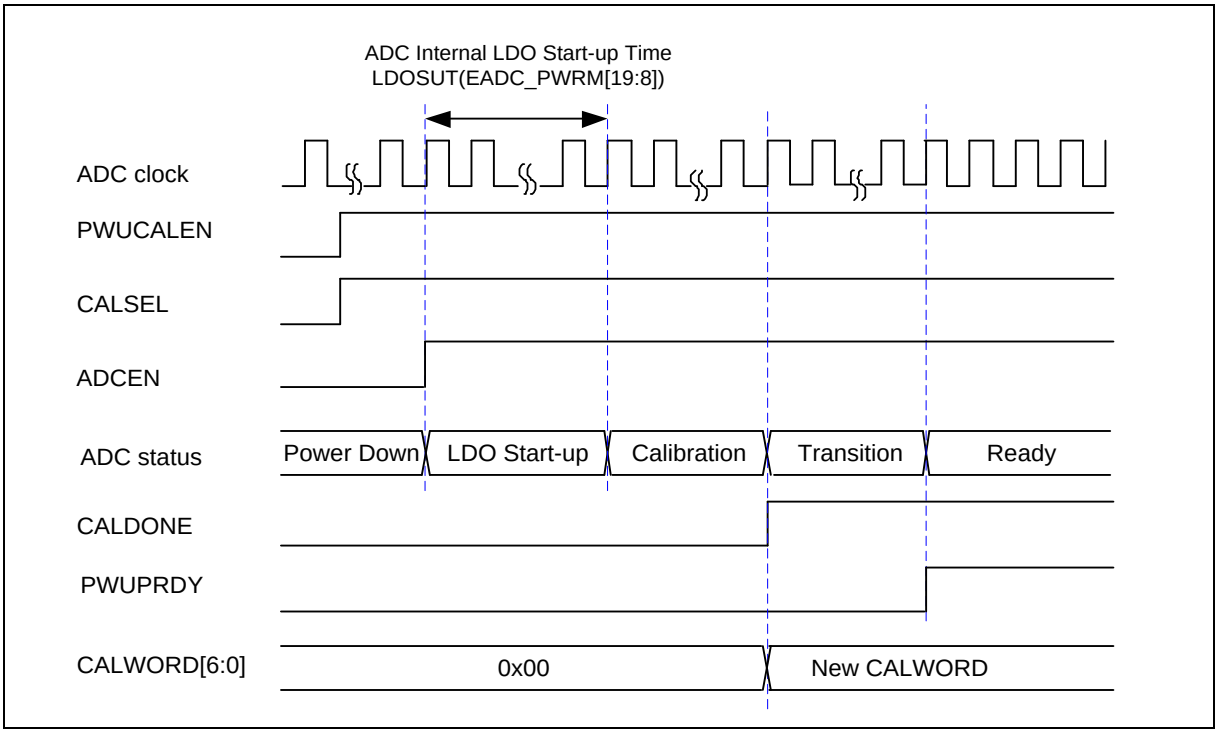


Figure 6.37-16 ADC Start up Sequence with Calibration

To get precise result, user may calibrate again after a few conversion. Setting CALSTART (EADC_CALCTL[1]) as 1 could enable calibration again, but this bit needs to work with CALSEL (EADC_CALCTL[3]). Setting CALSTART (EADC_CALCTL[1]) as 1 and CALSEL (EADC_CALCTL[3]) as 1 will execute calibration again then update CALWORD (EADC_CALDWRD[6:0]). Setting CALSTART (EADC_CALCTL[1]) as 1 and CALSEL (EADC_CALCTL[3]) as 0 will load CALWORD (EADC_CALDWRD[6:0]) which was defined by user. The re-calibration sequence should be as follows:

1. Set CALSEL (EADC_CALCTL[3]) as 1 or 0 to select calibration function
2. Set CALSTART (EADC_CALCTL[1]) as 1 to active calibration
3. CALDONE (EADC_CALCTL[2]) will be set as 0 by HW during calibration
4. Wait for CALDONE (EADC_CALCTL[2]) is set as 1 by HW. If CALSEL (EADC_CALCTL[3]) is set as 1, the new calibration word will be updated to CALWORD (EADC_CALDWRD[6:0]). If CALSEL (EADC_CALCTL[3]) is set as 0, the specific CALWORD (EADC_CALDWRD[6:0]) was loaded rather than executing calibration.

6.37.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
EADC Base Address: **EADC_BA = 0x4004_3000** **EADC non-secure base address is EADC_BA + 0x1000_0000**				
EADC_DAT0	EADC_BA+0x00	R	ADC Data Register 0 for Sample Module 0	0x0000_0000
EADC_DAT1	EADC_BA+0x04	R	ADC Data Register 1 for Sample Module 1	0x0000_0000
EADC_DAT2	EADC_BA+0x08	R	ADC Data Register 2 for Sample Module 2	0x0000_0000
EADC_DAT3	EADC_BA+0x0C	R	ADC Data Register 3 for Sample Module 3	0x0000_0000
EADC_DAT4	EADC_BA+0x10	R	ADC Data Register 4 for Sample Module 4	0x0000_0000
EADC_DAT5	EADC_BA+0x14	R	ADC Data Register 5 for Sample Module 5	0x0000_0000
EADC_DAT6	EADC_BA+0x18	R	ADC Data Register 6 for Sample Module 6	0x0000_0000
EADC_DAT7	EADC_BA+0x1C	R	ADC Data Register 7 for Sample Module 7	0x0000_0000
EADC_DAT8	EADC_BA+0x20	R	ADC Data Register 8 for Sample Module 8	0x0000_0000
EADC_DAT9	EADC_BA+0x24	R	ADC Data Register 9 for Sample Module 9	0x0000_0000
EADC_DAT10	EADC_BA+0x28	R	ADC Data Register 10 for Sample Module 10	0x0000_0000
EADC_DAT11	EADC_BA+0x2C	R	ADC Data Register 11 for Sample Module 11	0x0000_0000
EADC_DAT12	EADC_BA+0x30	R	ADC Data Register 12 for Sample Module 12	0x0000_0000
EADC_DAT13	EADC_BA+0x34	R	ADC Data Register 13 for Sample Module 13	0x0000_0000
EADC_DAT14	EADC_BA+0x38	R	ADC Data Register 14 for Sample Module 14	0x0000_0000
EADC_DAT15	EADC_BA+0x3C	R	ADC Data Register 15 for Sample Module 15	0x0000_0000
EADC_DAT16	EADC_BA+0x40	R	ADC Data Register 16 for Sample Module 16	0x0000_0000
EADC_DAT17	EADC_BA+0x44	R	ADC Data Register 17 for Sample Module 17	0x0000_0000
EADC_DAT18	EADC_BA+0x48	R	ADC Data Register 18 for Sample Module 18	0x0000_0000
EADC_CURDAT	EADC_BA+0x4C	R	ADC PDMA Current Transfer Data Register	0x0000_0000
EADC_CTL	EADC_BA+0x50	R/W	ADC Control Register	0x0000_00C0
EADC_SWTRG	EADC_BA+0x54	W	ADC Sample Module Software Start Register	0x0000_0000
EADC_PENDSTS	EADC_BA+0x58	R/W	ADC Start of Conversion Pending Flag Register	0x0000_0000
EADC_OVSTS	EADC_BA+0x5C	R/W	ADC Sample Module Start of Conversion Overrun Flag Register	0x0000_0000
EADC_SCTL0	EADC_BA+0x80	R/W	ADC Sample Module 0 Control Register	0x0000_0000

EADC_SCTL1	EADC_BA+0x84	R/W	ADC Sample Module 1 Control Register	0x0000_0000
EADC_SCTL2	EADC_BA+0x88	R/W	ADC Sample Module 2 Control Register	0x0000_0000
EADC_SCTL3	EADC_BA+0x8C	R/W	ADC Sample Module 3 Control Register	0x0000_0000
EADC_SCTL4	EADC_BA+0x90	R/W	ADC Sample Module 4 Control Register	0x0000_0000
EADC_SCTL5	EADC_BA+0x94	R/W	ADC Sample Module 5 Control Register	0x0000_0000
EADC_SCTL6	EADC_BA+0x98	R/W	ADC Sample Module 6 Control Register	0x0000_0000
EADC_SCTL7	EADC_BA+0x9C	R/W	ADC Sample Module 7 Control Register	0x0000_0000
EADC_SCTL8	EADC_BA+0xA0	R/W	ADC Sample Module 8 Control Register	0x0000_0000
EADC_SCTL9	EADC_BA+0xA4	R/W	ADC Sample Module 9 Control Register	0x0000_0000
EADC_SCTL10	EADC_BA+0xA8	R/W	ADC Sample Module 10 Control Register	0x0000_0000
EADC_SCTL11	EADC_BA+0xAC	R/W	ADC Sample Module 11 Control Register	0x0000_0000
EADC_SCTL12	EADC_BA+0xB0	R/W	ADC Sample Module 12 Control Register	0x0000_0000
EADC_SCTL13	EADC_BA+0xB4	R/W	ADC Sample Module 13 Control Register	0x0000_0000
EADC_SCTL14	EADC_BA+0xB8	R/W	ADC Sample Module 14 Control Register	0x0000_0000
EADC_SCTL15	EADC_BA+0xBC	R/W	ADC Sample Module 15 Control Register	0x0000_0000
EADC_SCTL16	EADC_BA+0xC0	R/W	ADC Sample Module 16 Control Register	0x0000_0000
EADC_SCTL17	EADC_BA+0xC4	R/W	ADC Sample Module 17 Control Register	0x0000_0000
EADC_SCTL18	EADC_BA+0xC8	R/W	ADC Sample Module 18 Control Register	0x0000_0000
EADC_INTSRC0	EADC_BA+0xD0	R/W	ADC Interrupt 0 Source Enable Control Register.	0x0000_0000
EADC_INTSRC1	EADC_BA+0xD4	R/W	ADC Interrupt 1 Source Enable Control Register.	0x0000_0000
EADC_INTSRC2	EADC_BA+0xD8	R/W	ADC Interrupt 2 Source Enable Control Register.	0x0000_0000
EADC_INTSRC3	EADC_BA+0xDC	R/W	ADC Interrupt 3 Source Enable Control Register.	0x0000_0000
EADC_CMP0	EADC_BA+0xE0	R/W	ADC Result Compare Register 0	0x0000_0000
EADC_CMP1	EADC_BA+0xE4	R/W	ADC Result Compare Register 1	0x0000_0000
EADC_CMP2	EADC_BA+0xE8	R/W	ADC Result Compare Register 2	0x0000_0000
EADC_CMP3	EADC_BA+0xEC	R/W	ADC Result Compare Register 3	0x0000_0000
EADC_STATUS0	EADC_BA+0xF0	R	ADC Status Register 0	0x0000_0000
EADC_STATUS1	EADC_BA+0xF4	R	ADC Status Register 1	0x0000_0000
EADC_STATUS2	EADC_BA+0xF8	R/W	ADC Status Register 2	0x0011_0000
EADC_STATUS3	EADC_BA+0xFC	R	ADC Status Register 3	0x0000_001F

EADC_DDAT0	EADC_BA+0x100	R	ADC Double Data Register 0 for Sample Module 0	0x0000_0000
EADC_DDAT1	EADC_BA+0x104	R	ADC Double Data Register 1 for Sample Module 1	0x0000_0000
EADC_DDAT2	EADC_BA+0x108	R	ADC Double Data Register 2 for Sample Module 2	0x0000_0000
EADC_DDAT3	EADC_BA+0x10C	R	ADC Double Data Register 3 for Sample Module 3	0x0000_0000
EADC_PWRM	EADC_BA+0x110	R/W	ADC Power Management Register	0x0006_E012
EADC_CALCTL	EADC_BA+0x114	R/W	ADC Calibration Control Register	0x0000_0008
EADC_CALDWRD	EADC_BA+0x118	R/W	ADC Calibration Load Word Register	0x0000_00XX
EADC_PDMACTL	EADC_BA+0x130	R/W	ADC PDMA Control Register	0x0000_0000

6.37.7 Register Description

ADC Data Registers (EADC_DAT0~ EADC_DAT18)

Register	Offset	R/W	Description	Reset Value
EADC_DAT0	EADC_BA+0x00	R	ADC Data Register 0 for Sample Module 0	0x0000_0000
EADC_DAT1	EADC_BA+0x04	R	ADC Data Register 1 for Sample Module 1	0x0000_0000
EADC_DAT2	EADC_BA+0x08	R	ADC Data Register 2 for Sample Module 2	0x0000_0000
EADC_DAT3	EADC_BA+0x0C	R	ADC Data Register 3 for Sample Module 3	0x0000_0000
EADC_DAT4	EADC_BA+0x10	R	ADC Data Register 4 for Sample Module 4	0x0000_0000
EADC_DAT5	EADC_BA+0x14	R	ADC Data Register 5 for Sample Module 5	0x0000_0000
EADC_DAT6	EADC_BA+0x18	R	ADC Data Register 6 for Sample Module 6	0x0000_0000
EADC_DAT7	EADC_BA+0x1C	R	ADC Data Register 7 for Sample Module 7	0x0000_0000
EADC_DAT8	EADC_BA+0x20	R	ADC Data Register 8 for Sample Module 8	0x0000_0000
EADC_DAT9	EADC_BA+0x24	R	ADC Data Register 9 for Sample Module 9	0x0000_0000
EADC_DAT10	EADC_BA+0x28	R	ADC Data Register 10 for Sample Module 10	0x0000_0000
EADC_DAT11	EADC_BA+0x2C	R	ADC Data Register 11 for Sample Module 11	0x0000_0000
EADC_DAT12	EADC_BA+0x30	R	ADC Data Register 12 for Sample Module 12	0x0000_0000
EADC_DAT13	EADC_BA+0x34	R	ADC Data Register 13 for Sample Module 13	0x0000_0000
EADC_DAT14	EADC_BA+0x38	R	ADC Data Register 14 for Sample Module 14	0x0000_0000
EADC_DAT15	EADC_BA+0x3C	R	ADC Data Register 15 for Sample Module 15	0x0000_0000
EADC_DAT16	EADC_BA+0x40	R	ADC Data Register 16 for Sample Module 16	0x0000_0000
EADC_DAT17	EADC_BA+0x44	R	ADC Data Register 17 for Sample Module 17	0x0000_0000
EADC_DAT18	EADC_BA+0x48	R	ADC Data Register 18 for Sample Module 18	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved						**VALID**	**OV**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RESULT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESULT							

Bits		Description
[31:18]	**Reserved**	Reserved.
[17]	**VALID**	**Valid Flag** This bit is set to 1 when corresponding sample module channel analog input conversion is completed and cleared by hardware after EADC_DAT register is read. 0 = Data in RESULT[11:0] bits is not valid. 1 = Data in RESULT[11:0] bits is valid.
[16]	**OV**	**Overrun Flag** If converted data in RESULT[11:0] has not been read before new conversion result is loaded to this register, OV is set to 1. 0 = Data in RESULT[11:0] is recent conversion result. 1 = Data in RESULT[11:0] is overwrite. **Note:** It is cleared by hardware after EADC_DAT register is read.
[15:0]	**RESULT**	**ADC Conversion Result** This field contains 12 bits conversion result. When DMOF (EADC_CTL[9]) is set to 0, 12-bit ADC conversion result with unsigned format will be filled in RESULT[11:0] and zero will be filled in RESULT[15:12]. When DMOF (EADC_CTL[9]) set to 1, 12-bit ADC conversion result with 2'complement format will be filled in RESULT[11:0] and signed bits to will be filled in RESULT[15:12].

ADC PDMA Current Transfer Data Register (EADC_CURDAT)

Register	Offset	R/W	Description	Reset Value
EADC_CURDAT	EADC_BA+0x4C	R	ADC PDMA Current Transfer Data Register	0x0000_0000

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

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17:0]	**CURDAT**	**ADC PDMA Current Transfer Data (Read Only)** This register is a shadow register of EADC_DATn (n=0~18) for PDMA support.

ADC Control Register (EADC_CTL)

Register	Offset	R/W	Description	Reset Value
EADC_CTL	EADC_BA+0x50	R/W	ADC Control Register	0x0000_00C0

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						DMOF	DIFFEN
7	6	5	4	3	2	1	0
RESSEL		ADCIEN3	ADCIEN2	ADCIEN1	ADCIEN0	ADCRST	ADCEN

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**DMOF**	**ADC Differential Input Mode Output Format** 0 = ADC conversion result will be filled in RESULT (EADC_DATn[15:0], where n= 0 ~18) with unsigned format. 1 = ADC conversion result will be filled in RESULT (EADC_DATn[15:0], where n= 0 ~18) with 2'complement format.
[8]	**DIFFEN**	**Differential Analog Input Mode Enable Bit** 0 = Single-end analog input mode. 1 = Differential analog input mode.
[7:6]	**RESSEL**	**Resolution Selection** 00 = 6-bit ADC result will be put at RESULT (EADC_DATn[5:0]). 01 = 8-bit ADC result will be put at RESULT (EADC_DATn[7:0]). 10 = 10-bit ADC result will be put at RESULT (EADC_DATn[9:0]). 11 = 12-bit ADC result will be put at RESULT (EADC_DATn[11:0]).
[5]	**ADCIEN3**	**Specific Sample Module ADC ADINT3 Interrupt Enable Bit** The ADC converter generates a conversion end ADIF3 (EADC_STATUS2[3]) upon the end of specific sample module ADC conversion. If ADCIEN3 bit is set then conversion end interrupt request ADINT3 is generated. 0 = Specific sample module ADC ADINT3 interrupt function Disabled. 1 = Specific sample module ADC ADINT3 interrupt function Enabled.
[4]	**ADCIEN2**	**Specific Sample Module ADC ADINT2 Interrupt Enable Bit** The ADC converter generates a conversion end ADIF2 (EADC_STATUS2[2]) upon the end of specific sample module ADC conversion. If ADCIEN2 bit is set then conversion end interrupt request ADINT2 is generated. 0 = Specific sample module ADC ADINT2 interrupt function Disabled. 1 = Specific sample module ADC ADINT2 interrupt function Enabled.

Bits	Description	
[3]	**ADCIEN1**	**Specific Sample Module ADC ADINT1 Interrupt Enable Bit** The ADC converter generates a conversion end ADIF1 (EADC_STATUS2[1]) upon the end of specific sample module ADC conversion. If ADCIEN1 bit is set then conversion end interrupt request ADINT1 is generated. 0 = Specific sample module ADC ADINT1 interrupt function Disabled. 1 = Specific sample module ADC ADINT1 interrupt function Enabled.
[2]	**ADCIEN0**	**Specific Sample Module ADC ADINT0 Interrupt Enable Bit** The ADC converter generates a conversion end ADIF0 (EADC_STATUS2[0]) upon the end of specific sample module ADC conversion. If ADCIEN0 bit is set then conversion end interrupt request ADINT0 is generated. 0 = Specific sample module ADC ADINT0 interrupt function Disabled. 1 = Specific sample module ADC ADINT0 interrupt function Enabled.
[1]	**ADCRST**	**ADC Converter Control Circuits Reset** 0 = No effect. 1 = Cause ADC control circuits reset to initial state, but not change the ADC registers value. **Note:** ADCRST bit remains 1 during ADC reset, when ADC reset end, the ADCRST bit is automatically cleared to 0.
[0]	**ADCEN**	**ADC Converter Enable Bit** 0 = Disabled EADC. 1 = Enabled EADC. **Note:** Before starting ADC conversion function, this bit should be set to 1. Clear it to 0 to disable ADC converter analog circuit power consumption.

ADC Sample Module Software Start Register (EADC_SWTRG)

Register	Offset	R/W	Description	Reset Value
EADC_SWTRG	EADC_BA+0x54	W	ADC Sample Module Software Start Register	0x0000_0000

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

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18:0]	**SWTRG**	**ADC Sample Module 0~18 Software Force to Start ADC Conversion** 0 = No effect. 1 = Cause an ADC conversion when the priority is given to sample module. **Note:** After writing this register to start ADC conversion, the EADC_PENDSTS register will show which sample module will conversion. If user wants to disable the conversion of the sample module, user can write EADC_PENDSTS register to clear it.

ADC Sample Module Start of Conversion Pending Flag Register (EADC_PENDSTS)

Register	Offset	R/W	Description	Reset Value
EADC_PENDSTS	EADC_BA+0x58	R/W	ADC Start of Conversion Pending Flag Register	0x0000_0000

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

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18:0]	**STPF**	**ADC Sample Module 0~18 Start of Conversion Pending Flag** **Read Operation:** 0 = There is no pending conversion for sample module. 1 = Sample module ADC start of conversion is pending. **Write Operation:** 1 = Clear pending flag & cancel the conversion for sample module. **Note:** This bit remains 1 during pending state. When the respective ADC conversion is end, the STPFn (n=0~18) bit is automatically cleared to 0.

ADC Sample Module Overrun Flag Register (EADC_OVSTS)

Register	Offset	R/W	Description	Reset Value
EADC_OVSTS	EADC_BA+0x5C	R/W	ADC Sample Module Start of Conversion Overrun Flag Register	0x0000_0000

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

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18:0]	**SPOVF**	**ADC SAMPLE0~18 Overrun Flag** 0 = No sample module event overrun. 1 = Indicates a new sample module event is generated while an old one event is pending. **Note:** This bit is cleared by writing 1 to it.

ADC Sample Module 0~3 Control Registers (EADC_SCTL0~EADC_SCTL3)

Register	Offset	R/W	Description	Reset Value
EADC_SCTL0	EADC_BA+0x80	R/W	ADC Sample Module 0 Control Register	0x0000_0000
EADC_SCTL1	EADC_BA+0x84	R/W	ADC Sample Module 1 Control Register	0x0000_0000
EADC_SCTL2	EADC_BA+0x88	R/W	ADC Sample Module 2 Control Register	0x0000_0000
EADC_SCTL3	EADC_BA+0x8C	R/W	ADC Sample Module 3 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
EXTSMPT							
23	22	21	20	19	18	17	16
DBMEN	INTPOS	Reserved	TRGSEL				
15	14	13	12	11	10	9	8
TRGDLYCNT							
7	6	5	4	3	2	1	0
TRGDLYDIV		EXTFEN	EXTREN	CHSEL			

Bits	Description	
[31:24]	**EXTSMPT**	**ADC Sampling Time Extend** When ADC converting at high conversion rate, the sampling time of analog input voltage may not enough if input channel loading is heavy, user can extend ADC sampling time after trigger source is coming to get enough sampling time. The range of start delay time is from 0~255 ADC clock.
[23]	**DBMEN**	**Double Buffer Mode Enable Bit** 0 = Sample has one sample result register (default). 1 = Sample has two sample result registers.
[22]	**INTPOS**	**Interrupt Flag Position Select** 0 = Set ADIFn (EADC_STATUS2[n], n=0~3) at ADC end of conversion. 1 = Set ADIFn (EADC_STATUS2[n], n=0~3) at ADC start of conversion.
[21]	**Reserved**	Reserved.

Bits	Description	
[20:16]	**TRGSEL**	**ADC Sample Module Start of Conversion Trigger Source Selection** 0H = Disable trigger. 1H = External trigger from EADC0_ST pin input. 2H = ADC ADINT0 interrupt EOC (End of conversion) pulse trigger. 3H = ADC ADINT1 interrupt EOC (End of conversion) pulse trigger. 4H = Timer0 overflow pulse trigger. 5H = Timer1 overflow pulse trigger. 6H = Timer2 overflow pulse trigger. 7H = Timer3 overflow pulse trigger. 8H = Timer4 overflow pulse trigger. 9H = Timer5 overflow pulse trigger. AH = EPWM0TG0. BH = EPWM0TG1. CH = EPWM0TG2. DH = EPWM0TG3. EH = EPWM0TG4. FH = EPWM0TG5. 10H = EPWM1TG0. 11H = EPWM1TG1. 12H = EPWM1TG2. 13H = EPWM1TG3. 14H = EPWM1TG4. 15H = EPWM1TG5. 16H =BPWM0TG. 17H =BPWM1TG. other = Reserved.
[15:8]	**TRGDLYCNT**	**ADC Sample Module Start of Conversion Trigger Delay Time** Trigger delay time = TRGDLYCNT x ADC_CLK period x n (n=1,2,4,16 from TRGDLYDIV setting).
[7:6]	**TRGDLYDIV**	**ADC Sample Module Start of Conversion Trigger Delay Clock Divider Selection** Trigger delay clock frequency: 00 = ADC_CLK/1. 01 = ADC_CLK/2. 10 = ADC_CLK/4. 11 = ADC_CLK/16.
[5]	**EXTFEN**	**ADC External Trigger Falling Edge Enable Bit** 0 = Falling edge Disabled when ADC selects EADC0_ST as trigger source. 1 = Falling edge Enabled when ADC selects EADC0_ST as trigger source.
[4]	**EXTREN**	**ADC External Trigger Rising Edge Enable Bit** 0 = Rising edge Disabled when ADC selects EADC0_ST as trigger source. 1 = Rising edge Enabled when ADC selects EADC0_ST as trigger source.

Bits	Description	
[3:0]	**CHSEL**	**ADC Sample Module Channel Selection** 00H = EADC_CH0 (slow channel). 01H = EADC_CH1 (slow channel). 02H = EADC_CH2 (slow channel). 03H = EADC_CH3 (slow channel). 04H = EADC_CH4 (slow channel). 05H = EADC_CH5 (slow channel). 06H = EADC_CH6 (slow channel). 07H = EADC_CH7 (slow channel). 08H = EADC_CH8 (slow channel). 09H = EADC_CH9 (slow channel). 0AH = EADC_CH10 (fast channel). 0BH = EADC_CH11 (fast channel). 0CH = EADC_CH12 (fast channel). 0DH = EADC_CH13 (fast channel). 0EH = EADC_CH14 (fast channel). 0FH = EADC_CH15 (fast channel).

ADC Sample Module 4~15 Control Registers (EADC_SCTL4~EADC_SCTL15)

Register	Offset	R/W	Description	Reset Value
EADC_SCTL4	EADC_BA+0x90	R/W	ADC Sample Module 4 Control Register	0x0000_0000
EADC_SCTL5	EADC_BA+0x94	R/W	ADC Sample Module 5 Control Register	0x0000_0000
EADC_SCTL6	EADC_BA+0x98	R/W	ADC Sample Module 6 Control Register	0x0000_0000
EADC_SCTL7	EADC_BA+0x9C	R/W	ADC Sample Module 7 Control Register	0x0000_0000
EADC_SCTL8	EADC_BA+0xA0	R/W	ADC Sample Module 8 Control Register	0x0000_0000
EADC_SCTL9	EADC_BA+0xA4	R/W	ADC Sample Module 9 Control Register	0x0000_0000
EADC_SCTL10	EADC_BA+0xA8	R/W	ADC Sample Module 10 Control Register	0x0000_0000
EADC_SCTL11	EADC_BA+0xAC	R/W	ADC Sample Module 11 Control Register	0x0000_0000
EADC_SCTL12	EADC_BA+0xB0	R/W	ADC Sample Module 12 Control Register	0x0000_0000
EADC_SCTL13	EADC_BA+0xB4	R/W	ADC Sample Module 13 Control Register	0x0000_0000
EADC_SCTL14	EADC_BA+0xB8	R/W	ADC Sample Module 14 Control Register	0x0000_0000
EADC_SCTL15	EADC_BA+0xBC	R/W	ADC Sample Module 15 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
EXTSMPT							
23	22	21	20	19	18	17	16
Reserved	INTPOS	Reserved	TRGSEL				
15	14	13	12	11	10	9	8
TRGDLYCNT							
7	6	5	4	3	2	1	0
TRGDLYDIV		EXTFEN	EXTREN	CHSEL			

Bits	Description	
[31:24]	**EXTSMPT**	**ADC Sampling Time Extend** When ADC converting at high conversion rate, the sampling time of analog input voltage may not enough if input channel loading is heavy, SW can extend ADC sampling time after trigger source is coming to get enough sampling time. The range of start delay time is from 0~255 ADC clock.
[23]	**Reserved**	Reserved.
[22]	**INTPOS**	**Interrupt Flag Position Select** 0 = Set ADIFn (EADC_STATUS2[n], n=0~3) at ADC end of conversion. 1 = Set ADIFn (EADC_STATUS2[n], n=0~3) at ADC start of conversion.
[21]	**Reserved**	Reserved.

Bits	Description	
[20:16]	**TRGSEL**	**ADC Sample Module Start of Conversion Trigger Source Selection** 0H = Disable trigger. 1H = External trigger from EADC0_ST pin input. 2H = ADC ADINT0 interrupt EOC pulse trigger. 3H = ADC ADINT1 interrupt EOC pulse trigger. 4H = Timer0 overflow pulse trigger. 5H = Timer1 overflow pulse trigger. 6H = Timer2 overflow pulse trigger. 7H = Timer3 overflow pulse trigger. 8H = Timer4 overflow pulse trigger. 9H = Timer5 overflow pulse trigger. AH = EPWM0TG0. BH = EPWM0TG1. CH = EPWM0TG2. DH = EPWM0TG3. EH = EPWM0TG4. FH = EPWM0TG5. 10H = EPWM1TG0. 11H = EPWM1TG1. 12H = EPWM1TG2. 13H = EPWM1TG3. 14H = EPWM1TG4. 15H = EPWM1TG5. 16H =BPWM0TG. 17H =BPWM1TG. other = Reserved.
[15:8]	**TRGDLYCNT**	**ADC Sample Module Start of Conversion Trigger Delay Time** Trigger delay time = TRGDLYCNT x ADC_CLK period x n (n=1,2,4,16 from TRGDLYDIV setting).
[7:6]	**TRGDLYDIV**	**ADC Sample Module Start of Conversion Trigger Delay Clock Divider Selection** Trigger delay clock frequency: 00 = ADC_CLK/1. 01 = ADC_CLK/2. 10 = ADC_CLK/4. 11 = ADC_CLK/16.
[5]	**EXTFEN**	**ADC External Trigger Falling Edge Enable Bit** 0 = Falling edge Disabled when ADC selects EADC0_ST as trigger source. 1 = Falling edge Enabled when ADC selects EADC0_ST as trigger source.
[4]	**EXTREN**	**ADC External Trigger Rising Edge Enable Bit** 0 = Rising edge Disabled when ADC selects EADC0_ST as trigger source. 1 = Rising edge Enabled when ADC selects EADC0_ST as trigger source.

Bits	Description	
[3:0]	**CHSEL**	**ADC Sample Module Channel Selection** 00H = EADC_CH0 (slow channel). 01H = EADC_CH1 (slow channel). 02H = EADC_CH2 (slow channel). 03H = EADC_CH3 (slow channel). 04H = EADC_CH4 (slow channel). 05H = EADC_CH5 (slow channel). 06H = EADC_CH6 (slow channel). 07H = EADC_CH7 (slow channel). 08H = EADC_CH8 (slow channel). 09H = EADC_CH9 (slow channel). 0AH = EADC_CH10 (fast channel). 0BH = EADC_CH11 (fast channel). 0CH = EADC_CH12 (fast channel). 0DH = EADC_CH13 (fast channel). 0EH = EADC_CH14 (fast channel). 0FH = EADC_CH15 (fast channel).

ADC Sample Module 16~18 Control Registers (EADC_SCTL16~EADC_SCTL18)

Register	Offset	R/W	Description	Reset Value
EADC_SCTL16	EADC_BA+0xC0	R/W	ADC Sample Module 16 Control Register	0x0000_0000
EADC_SCTL17	EADC_BA+0xC4	R/W	ADC Sample Module 17 Control Register	0x0000_0000
EADC_SCTL18	EADC_BA+0xC8	R/W	ADC Sample Module 18 Control Register	0x0000_0000

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

Bits	Description	
[31:24]	**EXTSMPT**	**ADC Sampling Time Extend** When ADC converting at high conversion rate, the sampling time of analog input voltage may not enough if input channel loading is heavy, SW can extend ADC sampling time after trigger source is coming to get enough sampling time. The range of start delay time is from 0~255 ADC clock.
[23:0]	**Reserved**	Reserved.

ADC Interrupt Source Enable Control Registers (EADC_INTSRC0~EADC_INTSRC3)

Register	Offset	R/W	Description	Reset Value
EADC_INTSRC0	EADC_BA+0xD0	R/W	ADC Interrupt 0 Source Enable Control Register.	0x0000_0000
EADC_INTSRC1	EADC_BA+0xD4	R/W	ADC Interrupt 1 Source Enable Control Register.	0x0000_0000
EADC_INTSRC2	EADC_BA+0xD8	R/W	ADC Interrupt 2 Source Enable Control Register.	0x0000_0000
EADC_INTSRC3	EADC_BA+0xDC	R/W	ADC Interrupt 3 Source Enable Control Register.	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved					**SPLIE18**	**SPLIE17**	**SPLIE16**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SPLIE15	**SPLIE14**	**SPLIE13**	**SPLIE12**	**SPLIE11**	**SPLIE10**	**SPLIE9**	**SPLIE8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SPLIE7	**SPLIE6**	**SPLIE5**	**SPLIE4**	**SPLIE3**	**SPLIE2**	**SPLIE1**	**SPLIE0**

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18]	**SPLIE18**	**Sample Module 18 Interrupt Enable Bit** 0 = Sample Module 18 interrupt Disabled. 1 = Sample Module 18 interrupt Enabled.
[17]	**SPLIE17**	**Sample Module 17 Interrupt Enable Bit** 0 = Sample Module 17 interrupt Disabled. 1 = Sample Module 17 interrupt Enabled.
[16]	**SPLIE16**	**Sample Module 16 Interrupt Enable Bit** 0 = Sample Module 16 interrupt Disabled. 1 = Sample Module 16 interrupt Enabled.
[15]	**SPLIE15**	**Sample Module 15 Interrupt Enable Bit** 0 = Sample Module 15 interrupt Disabled. 1 = Sample Module 15 interrupt Enabled.
[14]	**SPLIE14**	**Sample Module 14 Interrupt Enable Bit** 0 = Sample Module 14 interrupt Disabled. 1 = Sample Module 14 interrupt Enabled.
[13]	**SPLIE13**	**Sample Module 13 Interrupt Enable Bit** 0 = Sample Module 13 interrupt Disabled. 1 = Sample Module 13 interrupt Enabled.

Bits	Description	
[12]	**SPLIE12**	**Sample Module 12 Interrupt Enable Bit** 0 = Sample Module 12 interrupt Disabled. 1 = Sample Module 12 interrupt Enabled.
[11]	**SPLIE11**	**Sample Module 11 Interrupt Enable Bit** 0 = Sample Module 11 interrupt Disabled. 1 = Sample Module 11 interrupt Enabled.
[10]	**SPLIE10**	**Sample Module 10 Interrupt Enable Bit** 0 = Sample Module 10 interrupt Disabled. 1 = Sample Module 10 interrupt Enabled.
[9]	**SPLIE9**	**Sample Module 9 Interrupt Enable Bit** 0 = Sample Module 9 interrupt Disabled. 1 = Sample Module 9 interrupt Enabled.
[8]	**SPLIE8**	**Sample Module 8 Interrupt Enable Bit** 0 = Sample Module 8 interrupt Disabled. 1 = Sample Module 8 interrupt Enabled.
[7]	**SPLIE7**	**Sample Module 7 Interrupt Enable Bit** 0 = Sample Module 7 interrupt Disabled. 1 = Sample Module 7 interrupt Enabled.
[6]	**SPLIE6**	**Sample Module 6 Interrupt Enable Bit** 0 = Sample Module 6 interrupt Disabled. 1 = Sample Module 6 interrupt Enabled.
[5]	**SPLIE5**	**Sample Module 5 Interrupt Enable Bit** 0 = Sample Module 5 interrupt Disabled. 1 = Sample Module 5 interrupt Enabled.
[4]	**SPLIE4**	**Sample Module 4 Interrupt Enable Bit** 0 = Sample Module 4 interrupt Disabled. 1 = Sample Module 4 interrupt Enabled.
[3]	**SPLIE3**	**Sample Module 3 Interrupt Enable Bit** 0 = Sample Module 3 interrupt Disabled. 1 = Sample Module 3 interrupt Enabled.
[2]	**SPLIE2**	**Sample Module 2 Interrupt Enable Bit** 0 = Sample Module 2 interrupt Disabled. 1 = Sample Module 2 interrupt Enabled.
[1]	**SPLIE1**	**Sample Module 1 Interrupt Enable Bit** 0 = Sample Module 1 interrupt Disabled. 1 = Sample Module 1 interrupt Enabled.
[0]	**SPLIE0**	**Sample Module 0 Interrupt Enable Bit** 0 = Sample Module 0 interrupt Disabled. 1 = Sample Module 0 interrupt Enabled.

ADC Result Compare Register 0/1/2/3 (EADC_CMP0/1/2/3)

Register	Offset	R/W	Description	Reset Value
EADC_CMP0	EADC_BA+0xE0	R/W	ADC Result Compare Register 0	0x0000_0000
EADC_CMP1	EADC_BA+0xE4	R/W	ADC Result Compare Register 1	0x0000_0000
EADC_CMP2	EADC_BA+0xE8	R/W	ADC Result Compare Register 2	0x0000_0000
EADC_CMP3	EADC_BA+0xEC	R/W	ADC Result Compare Register 3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				CMPDAT			
23	22	21	20	19	18	17	16
CMPDAT							
15	14	13	12	11	10	9	8
CMPWEN	Reserved			CMPMCNT			
7	6	5	4	3	2	1	0
CMPSPL					CMPCOND	ADCMPIE	ADCMPEN

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:16]	**CMPDAT**	**Comparison Data** The 12 bits data is used to compare with conversion result of specified sample module. User can use it to monitor the external analog input pin voltage transition without imposing a load on software.
[15]	**CMPWEN**	**Compare Window Mode Enable Bit** 0 = ADCMPF0 (EADC_STATUS2[4]) will be set when EADC_CMP0 compared condition matched. ADCMPF2 (EADC_STATUS2[6]) will be set when EADC_CMP2 compared condition matched 1 = ADCMPF0 (EADC_STATUS2[4]) will be set when both EADC_CMP0 and EADC_CMP1 compared condition matched. ADCMPF2 (EADC_STATUS2[6]) will be set when both EADC_CMP2 and EADC_CMP3 compared condition matched. **Note:** This bit is only present in EADC_CMP0 and EADC_CMP2 register.
[14:12]	**Reserved**	Reserved.
[11:8]	**CMPMCNT**	**Compare Match Count** When the specified ADC sample module analog conversion result matches the compare condition defined by CMPCOND (EADC_CMPn[2], n=0~3), the internal match counter will increase 1. If the compare result does not meet the compare condition, the internal compare match counter will reset to 0. When the internal counter reaches the value to (CMPMCNT +1), the ADCMPFn (EADC_STATUS2[7:4], n=0~3) will be set.

Bits	Description	
[7:3]	**CMPSPL**	**Compare Sample Module Selection** 00000 = Sample Module 0 conversion result EADC_DAT0 is selected to be compared. 00001 = Sample Module 1 conversion result EADC_DAT1 is selected to be compared. 00010 = Sample Module 2 conversion result EADC_DAT2 is selected to be compared. 00011 = Sample Module 3 conversion result EADC_DAT3 is selected to be compared. 00100 = Sample Module 4 conversion result EADC_DAT4 is selected to be compared. 00101 = Sample Module 5 conversion result EADC_DAT5 is selected to be compared. 00110 = Sample Module 6 conversion result EADC_DAT6 is selected to be compared. 00111 = Sample Module 7 conversion result EADC_DAT7 is selected to be compared. 01000 = Sample Module 8 conversion result EADC_DAT8 is selected to be compared. 01001 = Sample Module 9 conversion result EADC_DAT9 is selected to be compared. 01010 = Sample Module 10 conversion result EADC_DAT10 is selected to be compared. 01011 = Sample Module 11 conversion result EADC_DAT11 is selected to be compared. 01100 = Sample Module 12 conversion result EADC_DAT12 is selected to be compared. 01101 = Sample Module 13 conversion result EADC_DAT13 is selected to be compared. 01110 = Sample Module 14 conversion result EADC_DAT14 is selected to be compared. 01111 = Sample Module 15 conversion result EADC_DAT15 is selected to be compared. 10000 = Sample Module 16 conversion result EADC_DAT16 is selected to be compared. 10001 = Sample Module 17 conversion result EADC_DAT17 is selected to be compared. 10010 = Sample Module 18 conversion result EADC_DAT18 is selected to be compared.
[2]	**CMPCOND**	**Compare Condition** 0= Set the compare condition as that when a 12-bit ADC conversion result is less than the 12-bit CMPDAT (EADC_CMPn [27:16]), the internal match counter will increase one. 1= Set the compare condition as that when a 12-bit ADC conversion result is greater or equal to the 12-bit CMPDAT (EADC_CMPn [27:16]), the internal match counter will increase one. **Note:** When the internal counter reaches the value to (CMPMCNT (EADC_CMPn[11:8], n=0~3) +1), the CMPF bit will be set.
[1]	**ADCMPIE**	**ADC Result Compare Interrupt Enable Bit** 0 = Compare function interrupt Disabled. 1 = Compare function interrupt Enabled. If the compare function is enabled and the compare condition matches the setting of CMPCOND (EADC_CMPn[2], n=0~3) and CMPMCNT (EADC_CMPn[11:8], n=0~3), ADCMPFn (EADC_STATUS2[7:4], n=0~3) will be asserted, in the meanwhile, if ADCMPIE is set to 1, a compare interrupt request is generated.
[0]	**ADCMPEN**	**ADC Result Compare Enable Bit** 0 = Compare Disabled. 1 = Compare Enabled. Set this bit to 1 to enable compare CMPDAT (EADC_CMPn[27:16], n=0~3) with specified sample module conversion result when converted data is loaded into EADC_DAT register.

ADC Status Register 0 (EADC_STATUS0)

Register	Offset	R/W	Description	Reset Value
EADC_STATUS0	EADC_BA+0xF0	R	ADC Status Register 0	0x0000_0000

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

Bits	Description	
[31:16]	OV	**EADC_DAT0~15 Overrun Flag** It is a mirror to OV bit in sample module ADC result data register EADC_DATn. (n=0~15).
[15:0]	VALID	**EADC_DAT0~15 Data Valid Flag** It is a mirror of VALID bit in sample module ADC result data register EADC_DATn. (n=0~15).

ADC Status Register 1 (EADC_STATUS1)

Register	Offset	R/W	Description	Reset Value
EADC_STATUS1	EADC_BA+0xF4	R	ADC Status Register 1	0x0000_0000

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

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18:16]	**OV**	**EADC_DAT16~18 Overrun Flag** It is a mirror to OV bit in sample module ADC result data register EADC_DATn. (n=16~18).
[15:3]	**Reserved**	Reserved.
[2:0]	**VALID**	**EADC_DAT16~18 Data Valid Flag** It is a mirror of VALID bit in sample module ADC result data register EADC_DATn. (n=16~18).

ADC Status Register 2 (EADC_STATUS2)

Register	Offset	R/W	Description	Reset Value
EADC_STATUS2	EADC_BA+0xF8	R/W	ADC Status Register 2	0x0011_0000

31	30	29	28	27	26	25	24
Reserved				AOV	AVALID	STOVF	ADOVIF
23	22	21	20	19	18	17	16
BUSY	Reserved	Reserved	CHANNEL				
15	14	13	12	11	10	9	8
ADCMPO3	ADCMPO2	ADCMPO1	ADCMPO0	ADOVIF3	ADOVIF2	ADOVIF1	ADOVIF0
7	6	5	4	3	2	1	0
ADCMPF3	ADCMPF2	ADCMPF1	ADCMPF0	ADIF3	ADIF2	ADIF1	ADIF0

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27]	**AOV**	**All Sample Module ADC Result Data Register Overrun Flags Check (Read Only)** n=0~18. 0 = None of sample module data register overrun flag OV (EADC_DATn[16]) is set to 1. 1 = Any one of sample module data register overrun flag OV (EADC_DATn[16]) is set to 1. **Note:** This bit will keep 1 when any OV Flag is equal to 1.
[26]	**AVALID**	**All Sample Module ADC Result Data Register EADC_DAT Data Valid Flag Check (Read Only)** n=0~18. 0 = None of sample module data register valid flag VALID (EADC_DATn[17]) is set to 1. 1 = Any one of sample module data register valid flag VALID (EADC_DATn[17]) is set to 1. **Note:** This bit will keep 1 when any VALID Flag is equal to 1.
[25]	**STOVF**	**All ADC Sample Module Start of Conversion Overrun Flags Check (Read Only)** n=0~18. 0 = None of sample module event overrun flag SPOVF (EADC_OVSTS[n]) is set to 1. 1 = Any one of sample module event overrun flag SPOVF (EADC_OVSTS[n]) is set to 1. **Note:** This bit will keep 1 when any SPOVF Flag is equal to 1.
[24]	**ADOVIF**	**All ADC Interrupt Flag Overrun Bits Check (Read Only)** n=0~3. 0 = None of ADINT interrupt flag ADOVIFn (EADC_STATUS2[11:8]) is overwritten to 1. 1 = Any one of ADINT interrupt flag ADOVIFn (EADC_STATUS2[11:8]) is overwritten to 1. **Note:** This bit will keep 1 when any ADOVIFn Flag is equal to 1.
[23]	**BUSY**	**Busy/Idle (Read Only)** 0 = EADC is in idle state. 1 = EADC is busy at conversion.
[22:21]	**Reserved**	Reserved.

Bits	Description	
[20:16]	**CHANNEL**	**Current Conversion Channel (Read Only)** This filed reflects ADC current conversion channel when BUSY=1. It is read only. 00H = EADC_CH0. 01H = EADC_CH1. 02H = EADC_CH2. 03H = EADC_CH3. 04H = EADC_CH4. 05H = EADC_CH5. 06H = EADC_CH6. 07H = EADC_CH7. 08H = EADC_CH8. 09H = EADC_CH9. 0AH = EADC_CH10. 0BH = EADC_CH11. 0CH = EADC_CH12. 0DH = EADC_CH13. 0EH = EADC_CH14. 0FH = EADC_CH15. 10H = VBG. 11H = VTEMP. 12H = V_{BAT}/4.
[15]	**ADCMPO3**	**ADC Compare 3 Output Status (Read Only)** The 12 bits compare3 data CMPDAT (EADC_CMP3[27:16]) is used to compare with conversion result of specified sample module. User can use it to monitor the external analog input pin voltage status. 0 = Conversion result in EADC_DAT is less than CMPDAT setting. 1 = Conversion result in EADC_DAT is greater than or equal to CMPDAT setting.
[14]	**ADCMPO2**	**ADC Compare 2 Output Status (Read Only)** The 12 bits compare2 data CMPDAT (EADC_CMP2[27:16]) is used to compare with conversion result of specified sample module. User can use it to monitor the external analog input pin voltage status. 0 = Conversion result in EADC_DAT is less than CMPDAT setting. 1 = Conversion result in EADC_DAT is greater than or equal to CMPDAT setting.
[13]	**ADCMPO1**	**ADC Compare 1 Output Status (Read Only)** The 12 bits compare1 data CMPDAT (EADC_CMP1[27:16]) is used to compare with conversion result of specified sample module. User can use it to monitor the external analog input pin voltage status. 0 = Conversion result in EADC_DAT is less than CMPDAT setting. 1 = Conversion result in EADC_DAT is greater than or equal to CMPDAT setting.
[12]	**ADCMPO0**	**ADC Compare 0 Output Status (Read Only)** The 12 bits compare0 data CMPDAT (EADC_CMP0[27:16]) is used to compare with conversion result of specified sample module. User can use it to monitor the external analog input pin voltage status. 0 = Conversion result in EADC_DAT is less than CMPDAT setting. 1 = Conversion result in EADC_DAT is greater than or equal to CMPDAT setting.

Bits	Description	
[11]	**ADOVIF3**	**ADC ADINT3 Interrupt Flag Overrun** 0 = ADINT3 interrupt flag is not overwritten to 1. 1 = ADINT3 interrupt flag is overwritten to 1. **Note:** This bit is cleared by writing 1 to it.
[10]	**ADOVIF2**	**ADC ADINT2 Interrupt Flag Overrun** 0 = ADINT2 interrupt flag is not overwritten to 1. 1 = ADINT2 interrupt flag is overwritten to 1. **Note:** This bit is cleared by writing 1 to it.
[9]	**ADOVIF1**	**ADC ADINT1 Interrupt Flag Overrun** 0 = ADINT1 interrupt flag is not overwritten to 1. 1 = ADINT1 interrupt flag is overwritten to 1. **Note:** This bit is cleared by writing 1 to it.
[8]	**ADOVIF0**	**ADC ADINT0 Interrupt Flag Overrun** 0 = ADINT0 interrupt flag is not overwritten to 1. 1 = ADINT0 interrupt flag is overwritten to 1. **Note:** This bit is cleared by writing 1 to it.
[7]	**ADCMPF3**	**ADC Compare 3 Flag** When the specific sample module ADC conversion result meets setting condition in EADC_CMP3 then this bit is set to 1. 0 = Conversion result in EADC_DAT does not meet EADC_CMP3 register setting. 1 = Conversion result in EADC_DAT meets EADC_CMP3 register setting. **Note:** This bit is cleared by writing 1 to it.
[6]	**ADCMPF2**	**ADC Compare 2 Flag** When the specific sample module ADC conversion result meets setting condition in EADC_CMP2 then this bit is set to 1. 0 = Conversion result in EADC_DAT does not meet EADC_CMP2 register setting. 1 = Conversion result in EADC_DAT meets EADC_CMP2 register setting. **Note:** This bit is cleared by writing 1 to it.
[5]	**ADCMPF1**	**ADC Compare 1 Flag** When the specific sample module ADC conversion result meets setting condition in EADC_CMP1 then this bit is set to 1. 0 = Conversion result in EADC_DAT does not meet EADC_CMP1 register setting. 1 = Conversion result in EADC_DAT meets EADC_CMP1 register setting. **Note:** This bit is cleared by writing 1 to it.
[4]	**ADCMPF0**	**ADC Compare 0 Flag** When the specific sample module ADC conversion result meets setting condition in EADC_CMP0 then this bit is set to 1. 0 = Conversion result in EADC_DAT does not meet EADC_CMP0 register setting. 1 = Conversion result in EADC_DAT meets EADC_CMP0 register setting. **Note:** This bit is cleared by writing 1 to it.

Bits	Description	
[3]	**ADIF3**	**ADC ADINT3 Interrupt Flag** 0 = No ADINT3 interrupt pulse received. 1 = ADINT3 interrupt pulse has been received. **Note 1:** This bit is cleared by writing 1 to it. **Note 2:**This bit indicates whether an ADC conversion of specific sample module has been completed
[2]	**ADIF2**	**ADC ADINT2 Interrupt Flag** 0 = No ADINT2 interrupt pulse received. 1 = ADINT2 interrupt pulse has been received. **Note 1:** This bit is cleared by writing 1 to it. **Note 2:**This bit indicates whether an ADC conversion of specific sample module has been completed
[1]	**ADIF1**	**ADC ADINT1 Interrupt Flag** 0 = No ADINT1 interrupt pulse received. 1 = ADINT1 interrupt pulse has been received. **Note 1:** This bit is cleared by writing 1 to it. **Note 2:**This bit indicates whether an ADC conversion of specific sample module has been completed
[0]	**ADIF0**	**ADC ADINT0 Interrupt Flag** 0 = No ADINT0 interrupt pulse received. 1 = ADINT0 interrupt pulse has been received. **Note 1:** This bit is cleared by writing 1 to it. **Note 2:**This bit indicates whether an ADC conversion of specific sample module has been completed

ADC Status Register 3 (EADC_STATUS3)

Register	Offset	R/W	Description	Reset Value
EADC_STATUS3	EADC_BA+0xFC	R	ADC Status Register 3	0x0000_001F

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

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4:0]	**CURSPL**	**ADC Current Sample Module (Read Only)** This register shows the current ADC is controlled by which sample module control logic modules. If the ADC is Idle, the bit filed will be set to 0x1F.

ADC Double Data Register n for Sample Module n (EADC_DDAT0~3)

Register	Offset	R/W	Description	Reset Value
EADC_DDAT0	EADC_BA+0x100	R	ADC Double Data Register 0 for Sample Module 0	0x0000_0000
EADC_DDAT1	EADC_BA+0x104	R	ADC Double Data Register 1 for Sample Module 1	0x0000_0000
EADC_DDAT2	EADC_BA+0x108	R	ADC Double Data Register 2 for Sample Module 2	0x0000_0000
EADC_DDAT3	EADC_BA+0x10C	R	ADC Double Data Register 3 for Sample Module 3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved						**VALID**	**OV**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RESULT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESULT							

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**VALID**	**Valid Flag** 0 = Double data in RESULT (EADC_DDATn[15:0]) is not valid. 1 = Double data in RESULT (EADC_DDATn[15:0]) is valid. This bit is set to 1 when corresponding sample module channel analog input conversion is completed and cleared by hardware after EADC_DDATn register is read. (n=0~3).
[16]	**OV**	**Overrun Flag** 0 = Data in RESULT (EADC_DATn[15:0], n=0~3) is recent conversion result. 1 = Data in RESULT (EADC_DATn[15:0], n=0~3) is overwrite. If converted data in RESULT[15:0] has not been read before new conversion result is loaded to this register, OV is set to 1. It is cleared by hardware after EADC_DDAT register is read.
[15:0]	**RESULT**	**ADC Conversion Results** This field contains 12 bits conversion results. When the DMOF (EADC_CTL[9]) is set to 0, 12-bit ADC conversion result with unsigned format will be filled in RESULT [11:0] and zero will be filled in RESULT [15:12]. When DMOF (EADC_CTL[9]) is set to 1, 12-bit ADC conversion result with 2'complement format will be filled in RESULT [11:0] and signed bits to will be filled in RESULT [15:12].

ADC Power Management Register (EADC_PWRM)

Register	Offset	R/W	Description	Reset Value
EADC_PWRM	EADC_BA+0x110	R/W	ADC Power Management Register	0x0006_E012

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved				**LDOSUT**			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
LDOSUT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**PWDMOD**		**PWUCALEN**	**PWUPRDY**

Bits	Description	
[31:20]	**Reserved**	Reserved.
[19:8]	**LDOSUT**	**ADC Internal LDO Start-up Time** Set this bit field to control LDO start-up time. The minimum required LDO start-up time is 20us. LDO start-up time = (1/ADC_CLK) x LDOSUT.
[7:4]	**Reserved**	Reserved.
[3:2]	**PWDMOD**	**ADC Power-down Mode** Set this bit field to select ADC Power-down mode when system power-down. 00 = ADC Deep Power-down mode. 01 = ADC Power down. 10 = ADC Standby mode. 11 = ADC Deep Power-down mode. **Note:** Different PWDMOD has different power down/up sequence, in order to avoid ADC powering up with wrong sequence; user must keep PWDMOD consistent each time in power down and start up.
[1]	**PWUCALEN**	**Power Up Calibration Function Enable Bit** 0 = Calibration function Disabled at power up. 1 = Calibration function Enabled at power up. **Note:** This bit work together with CALSEL (EADC_CALCTL [3]), see the following {PWUCALEN, CALSEL } Description: PWUCALEN is 0 and CALSEL is 0: No need to calibrate. PWUCALEN is 0 and CALSEL is 1: No need to calibrate. PWUCALEN is 1 and CALSEL is 0: Load calibration word when power up. PWUCALEN is 1 and CALSEL is 1: Calibrate when power up.
[0]	**PWUPRDY**	**ADC Power-up Sequence Completed and Ready for Conversion (Read Only)** 0 = ADC is not ready for conversion may be in power down state or in the progress of start up. 1 = ADC is ready for conversion.

ADC Calibration Control Register (EADC_CALCTL)

Register	Offset	R/W	Description	Reset Value
EADC_CALCTL	EADC_BA+0x114	R/W	ADC Calibration Control Register	0x0000_0008

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

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**CALSEL**	**Select Calibration Functional Block** 0 = Load calibration word when calibration functional block is active. 1 = Execute calibration when calibration functional block is active.
[2]	**CALDONE**	**Calibration Functional Block Complete (Read Only)** 0 = During a calibration. 1 = Calibration is completed.
[1]	**CALSTART**	**Calibration Functional Block Start** 0 = Stop calibration functional block. 1 = Start calibration functional block. **Note:** This bit is set by SW and clear by HW after re-calibration finish.
[0]	**Reserved**	Reserved.

ADC Calibration Load Word Register (EADC_CALDWRD)

Register	Offset	R/W	Description	Reset Value
EADC_CALDWRD	EADC_BA+0x118	R/W	ADC Calibration Load Word Register	0x0000_00XX

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

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6:0]	**CALWORD**	**Calibration Word Bits** Write to this register with the previous calibration word before load calibration action. Read this register after calibration done. **Note:** The calibration block contains two parts “CALIBRATION“ and “LOAD CALIBRATION”; if the calibration block configure as “CALIBRATION“; then this register represent the result of calibration when calibration is completed; if configure as “LOAD CALIBRATION” ; configure this register before loading calibration action, after loading calibration complete, the loaded calibration word will apply to the ADC; while in loading calibration function the loaded value will not be equal to the original CALWORD until calibration is done.

ADC PDMA Control Register (EADC_PDMACTL)

Register	Offset	R/W	Description	Reset Value
EADC_PDMACTL	EADC_BA+0x130	R/W	ADC PDMA Control Register	0x0000_0000

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

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18:0]	**PDMATEN**	**PDMA Transfer Enable Bit** When EADC conversion is completed, the converted data is loaded into EADC_DATn (n: 0 ~ 18) register, user can enable this bit to generate a PDMA data transfer request. 0 = PDMA data transfer Disabled. 1 = PDMA data transfer Enabled. **Note:** When set this bit field to 1, user must set ADCIENn (ADC_CTL[5:2], n=0~3) = 0 to disable interrupt.

6.38 True Random Number Generator (TRNG)

6.38.1 Overview

The purpose of True Random Number Generator (TRNG) is to generate the randomness by extracting from physical phenomena.

6.38.2 Features

- 800 random bits per second
- Provides the true random number seed for PRNG

6.38.3 Block Diagram

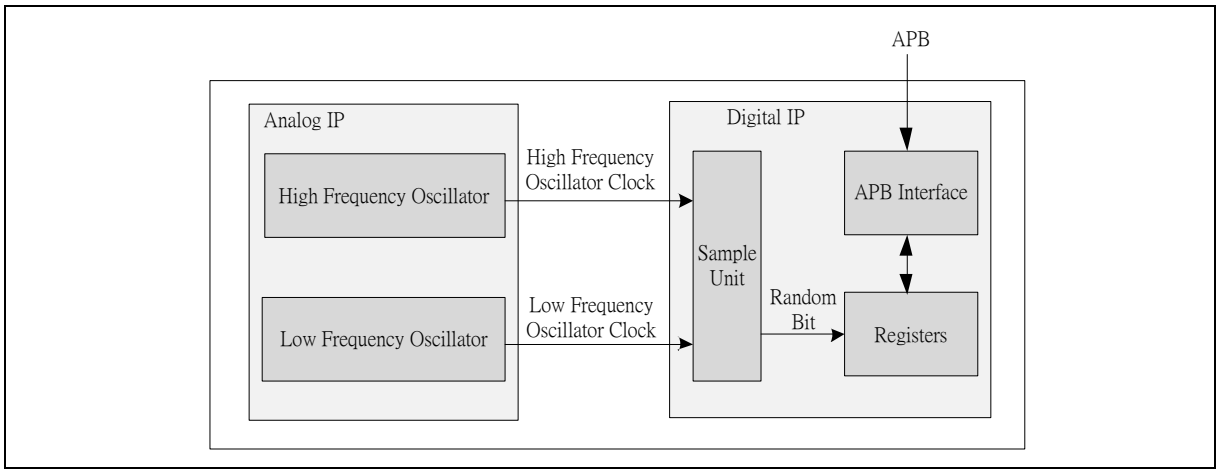


Figure 6.38-11.1 True Random Number Generator Block Diagram

6.38.4 Basic Configuration

6.38.4.1 Clock Source Configuration

- Enable 32 kHz clock in LXTEN (CLK_PWRCTL[1]), source from LXT or
- Enable 32 kHz clock in LIRC32KEN (RTC_LXTCTL[0]) and C32KSEL (RTC_LXTCTL[6]), source from LIRC
- Enable TRNG peripheral clock in TRNGCKEN (CLK_APBCLK1[25]).

6.38.5 Functional Description

The TRNG can provide the random number when activated and enabled. When DVIF (TRNG_CTL[1]) bit is set, the random bits may be read one byte at a time form TRNG_DATA register. If DVIF is 0, the TRNG_DATA register returns the value 0x0 when read. After read the TRNG_DATA register, the DVIF will be set to 0. If DVIEN (TRNG_CTL[6]) is enabled, the TRNG will generate interrupt to CPU when DVIF is set.

Before changing CLKP (TRNG_CTL[5:2]), TRNGEN (TRNG_CTL[0]) bit should be set to 0.

The programming steps to get the true random number are depicted below.

1. Select TRNG peripheral clock frequency in CLKP (TRNG_CTL[5:2]).
2. Enable ACT (TRNG_ACT[7]) bit.
3. Wait for READY (TRNG_CTL[7]) bit to be set to 1.

4. Enable TRNGEN (TRNG_CTL[0]) bit.
5. Wait DVIF (TRNG_CTL[1]) to bit be set to 1.
6. Read TRNG_DATA register to get random number.

Repeat Step 5 and 6 to get more random number.

TRNG generates the true random number seed for PRNG

The programming steps to get the random number with the true random number seed from TRNG (including TRNG generate the true random number seed steps) are depicted below. When SEEDGEN (TRNG_CTL[8]) is set to 1, users cannot read the data from TRNG Data Register.

1. Select TRNG peripheral clock frequency in CLKP (TRNG_CTL[5:2]).
2. Enable ACT (TRNG_ACT[7]) bit.
3. Wait READY (TRNG_CTL[7]) bit to be set to 1.
4. Enable SEEDGEN (TRNG_CTL[8]) bit.
5. Wait SEEDRDY (TRNG_CTL[9]) bit to be set to 1 to get 32 bits true random number seed.
6. Check the BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0.
7. Set SEEDSEL(CRYPTO_PRNG_CTL[6]) to 1.
8. Wait for SEEDSRC(CRYPTO_PRNG_CTL[7]) to be set to 1.
9. Configure PRNG control register CRYPTO_PRNG_CTL for key size(KEYSZ), seed reload(SEEDRLD), and PRNG start(START).
10. Software checks BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0, or waits for the PRNGIF (CRYPTO_INTSTS[16]) (must enable PRNGIEN (CRYPTO_INTEN[16])). Then software can read the output random numbers (KEY) from CRYPTO_PRNG_KEY0 ~ CRYPTO_PRNG_KEY7.

If users want to get the next (not first time) true random number seed from TRNG, they should set SEEDSEL(CRYPTO_PRNG_CTL[6]) bit to 0 before Step 4.

Some Notices of TRNG:

1. If users want to get the next (not the first time) true random number seed for PRNG, they should set SEEDSEL(CRYPTO_PRNG_CTL[6]) bit to 0 before step 4 (enable SEEDGEN (TRNG_CTL[8]) bit).
2. It is not suggested that enabling TRNGEN (TRNG_CTL[0]) and SEEDGEN (TRNG_CTL[8]) at the same time. If users still want to execute the situation, SEEDGEN will have a higher priority than TRNGEN. After SEEDRDY (TRNG_CTL[9]) bit is set to 1, TRNG starts to execute TRNGEN part.
3. After TRNG provides the random number (8 bits), it means that DVIF (TRNG_CTL[1]) bit is set to 1. If users execute SEEDGEN (TRNG_CTL[8]) and do not read TRNG_DATA register, the DVIF (TRNG_CTL[1]) bit is still set to 0 because the random number (8-bits) has been taken away for generating TRNG seed.

6.38.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
TRNG Base Address: **TRNG_BA = 0x400B_9000** **TRNG non secure base address is TRNG_BA + 0x1000_0000**				
TRNG_CTL	TRNG_BA+0x00	R/W	TRNG Control Register and Status	0x0000_0000
TRNG_DATA	TRNG_BA+0x04	R	TRNG Data Register	0x0000_0000
TRNG_ACT	TRNG_BA+0x0C	R/W	TRNG Activation Register	0x0000_0002

6.38.7 Register Description

TRNG Control Register and Status (TRNG_CTL)

Register	Offset	R/W	Description	Reset Value
TRNG_CTL	TRNG_BA+0x00	R/W	TRNG Control Register and Status	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						SEEDRDY	SEEDGEN
7	6	5	4	3	2	1	0
READY	DVIEN	CLKPSC				DVIF	TRNGEN

Bits	Description	
[31:10]	**Reversed**	Reversed
[9]	**SEEDRDY**	**Random Number Seed Ready (Read Only) [for TRNG+PRNG]** 0 = Seed is not ready or not activated. 1 = Seed is ready for PRNG. **Note 1:** This bit is cleared to '0' when SEEDGEN is 1. **Note 2:** If SEEDRDY becomes 1, then SEEDGEN will be cleared to 0.
[8]	**SEEDGEN**	**Random Number Seed Generator Enable Bit [for TRNG+PRNG]** This bit can be set to 1 only after ACT (TRNG_ACT[7]) bit is set to 1 and READY (TRNG_CTL[7]) bit becomes 1. 0 = Seed generator Disabled. 1 = Seed generator Enabled. **Note:** If users want to execute TRNG+PRNG mode, they should set SEEDGEN to 1. When SEEDGEN is set to 1, users cannot read the data from TRNG Data Register.
[7]	**READY**	**Random Number Generator Ready (Read Only)** After ACT (TRNG_ACT[7]) bit is set, the READY bit becomes 1 after a delay of 90us~120us. 0 = RNG is not ready or not activated. 1 = RNG is ready to be enabled.
[6]	**DVIEN**	**Data Valid Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[5:2]	**CLKPSC**	**Clock Prescaler** The CLKP is the peripheral clock frequency range for the selected value. The CLKP must be higher than or equal to the actual peripheral clock frequency (for correct random bit generation). To change the CLKP contents, first set TRNGEN bit to 0 and then change CLKP; finally, set TRNGEN bit to 1 to re-enable the TRNG module.

		0000 = 80 ~ 100 MHz. 0001 = 60 ~ 80 MHz. 0010 = 50 ~60 MHz. 0011 = 40 ~50 MHz. 0100 = 30 ~40 MHz. 0101 = 25 ~30 MHz. 0110 = 20 ~25 MHz. 0111 = 15 ~20 MHz. 1000 = 12 ~15 MHz. 1001 = 9 ~12 MHz. 1010 = 7 ~9 MHz. 1011 = 6 ~7 MHz. 1100 = 5 ~6 MHz. 1101 = 4 ~5 MHz. 1111 = Reserved.
[1]	**DVIF**	**Data Valid (Read Only)** 0 = Data is not valid. Reading from RNGD returns 0x00000000. 1 = Data is valid. A valid random number can be read form RNGD. **Note:** This bit is cleared to '0' by reading TRNG_DATA.
[0]	**TRNGEN**	**Random Number Generator Enable Bit** This bit can be set to 1 only after ACT (TRNG_ACT[7]) bit is set to 1 and READY (TRNG_CTL[7]) bit becames 1. 0 = TRNG Disabled. 1 = TRNG Enabled. **Note:** TRNGEN is an enable bit of digital part. When TRNG is not required to generate random number, TRNGEN bit and ACT (TRNG_ACT[7]) bit should be set to 0 to reduce power consumption.

TRNG Data Register (TRNG_DATA)

Register	Offset	R/W	Description	Reset Value
TRNG_DATA	TRNG_BA+0x04	R	TRNG Data Register	0x0000_0000

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**DATA**	**Random Number Generator Data (Read Only)** The DATA stores the random number generated by TRNG and can be read only once.

TRNG Activation Register (TRNG_ACT)

Register	Offset	R/W	Description	Reset Value
TRNG_ACT	TRNG_BA+0x0C	R/W	TRNG Activation Register	0x0000_0002

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**ACT**	**Random Number Generator Activation** After enabling the ACT bit, it will activate the TRNG module and wait the READY (TRNG_CTL[7]) bit to become 1. 0 = TRNG inactive. 1 = TRNG active. **Note:** ACT is an enable bit of analog part. When TRNG is not required to generate random number, TRNGEN (TRNG_CTL[0]) bit and ACT bit should be set to 0 to reduce power consumption.
[6:0]	**Reserved**	Reserved.

6.39 Key Store (KS)

6.39.1 Overview

The Key Store (KS) is the key management device and has a 4 Kbytes SRAM, 2 Kbytes Flash and OTP for key storage. The Key Store is capable of providing a crypto engine to access or storing the key while encryption, decryption and generation. The Key Store supports revoke key operation if the key is unused. The Key Store is able to protect the key by data scrambling, data remanence prevention and silent access.

6.39.2 Features

- Supports programming interface for key management
- Supports multiple key size
- Supports 4 Kbytes SRAM, 2 Kbytes Flash and 544bytes OTP for key storage
- Supports 32 keys for SRAM, 32 keys for Flash and 8 keys for OTP at most
- Supports crypto engine access or store key in key store directly
- Supports ECDH operation with ECC and PRNG engine
- Supports to store middle data for RSA CRT and SCAP mode
- Supports revoke operation for each key
- Supports erase key in SRAM/Flash and revoke key in OTP while tamper detected
- Supports integrity checking
- Supports data scrambling at SRAM, Flash and OTP
- Supports data remanence prevention at SRAM
- Supports silent access for side-channel protection at SRAM, Flash and OTP

6.39.3 Block Diagram

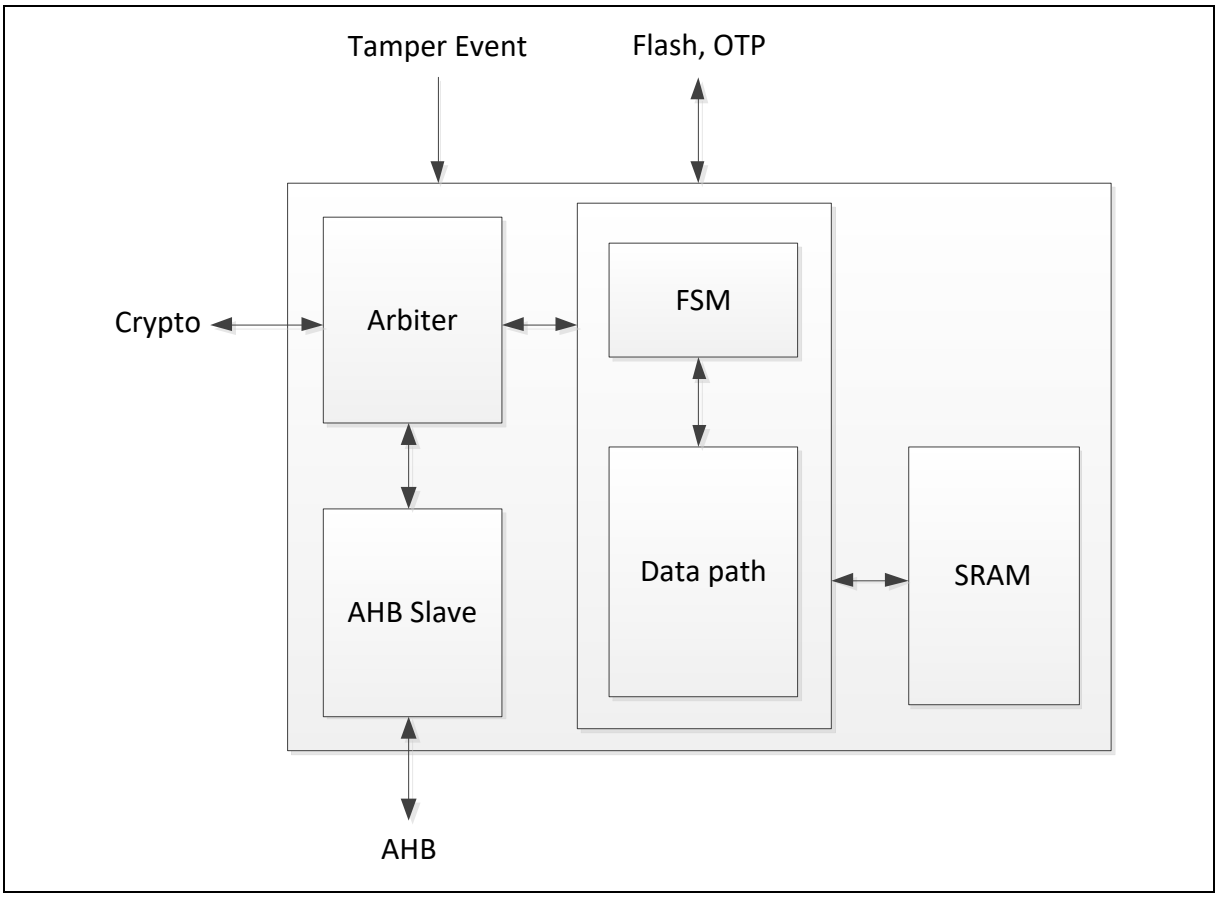


Figure 6.39-1 Key Store Block Diagram

6.39.4 Basic Configuration

- Clock Source Configuration
  - Enable KS peripheral clock in KSCKEN (CLK_AHBCLK[13]).
- SRAM Power Mode Configuration
  - Set KS SRAM power mode to normal mode in KS(SYS_SRAMPC1[29:28])

6.39.5 Functional Description

6.39.5.1 Initialization

When chip is powered up, the Key Store has no information about key at Flash and OTP. The Key Store needs to read the metadata of keys from Flash and OTP, so Key Store should be initialized by setting INIT(KS_CTL[8]) and START(KS_CTL[0]). When the Key Store is initialized, INITDONE(KS_STS[7]) will be set. The Key Store will check INITDONE before executing the operation except the initialization. If INITDONE is 0, the current operation will fail and EIF(KS_STS[1]) will be set. User should clear EIF by writing it to 0 and re-initialize the Key Store.

Before initialization, user must check whether KS(SYS_SRAMPC1[29:28]) returns to normal mode. KS_REMAIN and KS_OTPSTS registers keep the reset value. If initialization is finished, KS_REMAIN and KS_OTPSTS registers will update current status. If initialization is executed when KS SRAM is not

at normal mode, KS will have an unexpected situation.

The Key Store needs to be initialized when chip powered up, woken up from SPD/DPD mode, and reset. If INITDONE is set, and the Key Store does not need to be initialized again.

If the chip is changed to RMA stage, the existing OTP keys will be revoked and SRAM keys and Flash keys will be erased after initialization. If the OTP key is created in the RMA stage, this key can be normally used in the RMA stage.

Initial flow:

1. Check BUSY(KS_STS[2]) is 0 and INITDONE(KS_STS[7]) is 0.
2. Check KS(SYS_SRAMPC1[29:28]) is 00
3. Set INIT(KS_CTL[8]) to 1 and START(KS_CTL[0]) to 1
4. Wait BUSY(KS_STS[2]) to 0
5. Check INITDONE(KS_STS[7]) is 1

6.39.5.2 Basic Operation

The Key Store has different placement mechanism at SRAM, Flash and OTP. The Key Store puts a key in SRAM and Flash by hardware placement, and puts the key in OTP by software placement. User does not need to set the key number as long as user can read the key according key number returned by Key Store after key is placed in SRAM or Flash. If a key is placed in OTP, user needs to give the key number before creating key operation.

There is a basic operation about create key, read key and erase key in Key Store. User can select the location, fill in the metadata, select the operation and enable start bit, and then the Key Store will start operating. When the operation is finished, the Key Store will set the interrupt flag. If key creation is finished, the Key Store will return the key number for access this key in the future.

Each key has its metadata and metadata contains the information shown as follows.

1. Owner – OWNER(KS_METADATA[18:16])
   - Key can be set to which owner can use. Only the owner can access the key.
2. Size – SIZE(KS_METADATA[12:8])
   - Set key size.
3. Secure – SEC(KS_METADATA[0])
   - The secure/non-secure attribution of the key.
   - Secure owner can access secured key and non-secured key. Non-secure owner can only access non-secured key.
4. Privilege – PRIV(KS_ METADATA [1])
   - The privilege/non-privilege attribution of the key.
   - Privilege owner can access privilege d key and non-privilege d key. Non-privilege owner can only access non-privilege key.
5. Readable – READABLE(KS_METADATA[2])
   - Key can be read by CPU or not.
   - If OWNER is set to AES and READABLE is set to 1, CPU can read out key and AES can access.
   - If OWNER is set to CPU and READABLE is set to 0, this key is unused. CPU cannot read

out and Crypto engine cannot access.

- If OWNER is set to RSA middle data, CPU cannot read out no matter what READABLE is set to.

6. Revoke – RVK(KS_METADATA[3])
   - Set key to revoke or not.
7. Booting State – BS(KS_METADATA[4])
   - Key can be used at booting state(BL1 state) or not. If BS is set, key can be used at booting state only.

Among the metadata mentioned above, only revoke attribute can be changed to 1 by revoking key operation and other metadata is not allowed to be changed if the key is stored in Flash or OTP . When a key is placed in SRAM, this key can be erased by erase one key operation and Key Store will release the storage space.

User can create keys to Flash and OTP ,and then Key Store programs the key and its metadata to Flash and OTP. It means the key storage at Flash is not totally 2 Kbytes space. When EIF(KS_STS[1]) is set, the EIF should be cleared by writing 1 before the next operation starts.

If the create key operation is not finished at Flash, the next create key operation will write the wrong key. Table 6.39-1 shows the detailed information for SRAM, Flash and OTP.

	SRAM	Flash	OTP
Storage Size	4 Kbytes	2 Kbytes	544 bytes
Store maximum key size	4096 bits	4096 bits	256 bits
Store maximum key count	32 keys	32 keys	8 keys
Store RSA key	V	V	-
Store RSA middle data	V	-	-
Store ECDH or ECDSA random key generated from PRNG	V	-	-
Read operation	V	V	V
Create operation	V	V	V
Erase one key operation	V	-	-
Erase all key operation	V	V	-
Revoke key operation	V	V	V
Data Remanence	V	-	-
Support Integrity Checking	V	V	V

Data Scrambling	Control by SCMB(KS_CTL[11])	Control by SCMB(KS_CTL[11])	Control by SCMB(KS_CTL[11])
Silent Access	Control by SILENT(KS_CTL[10])	Control by SILENT(KS_CTL[10])	Always Enabled

Table 6.39-1 Detailed Information for SRAM, Flash and OTP

The read, create, erase and revoke operation is described as follows.

Read Operation Programming Flow

Use read operation to read out key to see which key is readable. If the key is unreadable, EIF(KS_STS[1]) will set to 1.

1. Check BUSY(KS_STS[2]) is 0 and EIF(KS_STS[1]) is 0.
2. Configure DST(KS_METADATA[31:30]), NUMBER(KS_METADATA[25:20])
3. Set OPMODE(KS_CTL[3:1]) to 000
4. Set START(KS_CTL[0]) to 1
5. Wait BUSY(KS_STS[2]) to 0.
6. Read key from KS_KEY0~KS_KEY7 registers if EIF(KS_STS[1]) is 0.
7. Repeat step 4 to step 6 and set CONT(KS_CTL[7]) to 1 when key size is greater than 256 bits

Create Operation Programming Flow at SRAM/Flash

Use create operation can create key to SRAM or Flash. After the create operation is finished, the Key Store will return key number. If the number of create operation is greater than the number needed, EIF(KS_STS[1]) will be set to 1.

1. Check BUSY(KS_STS[2]) is 0 and EIF(KS_STS[1]) is 0.
2. Check FLASHFULL(KS_STS[4]) or SRAMFULL(KS_STS[3]) is 0.
3. Configure DST(KS_METADATA[31:30]), OWNER(KS_METADATA[18:16]), SIZE(KS_METADATA[12:8]),BS(KS_METADATA[4]), READABLE(KS_METADATA[2]), PRIV(KS_METADATA[1]), SEC(KS_METADATA[0]) according key's metadata.
4. Set OPMODE(KS_CTL[3:1]) to 001
5. Write key to KS_KEY0~KEY7 registers
6. Set START(KS_CTL[0]) to 1
7. Wait BUSY(KS_STS[2]) to 0.
8. Check EIF(KS_STS[1]) is 0.
9. Repeat step 4 to step 7 and set CONT(KS_CTL[7]) to 1 when key size is greater than 256 bits
10. Read key number from NUMBER(KS_METADATA [25:20])

Note: Step 4 to Step 7 do not need to be repeated if key size is equal to or less than 256 bits. Step 4 to Step 7 need to be repeated once if the key size is greater than 256 bits and equal to or less than 512 bits. Step 4 to Step 7 need to be repeated twice if the key size is greater than 512 bits and equal to or less than 384 bits, and so on.

Create Operation Programming Flow at OTP

Use create operation to create key to OTP with specified key number. After create operation is finished, KS will return key number according written the specified number. If the number of create operation is greater than the number needed, EIF(KS_STS[1]) will set to 1.

1. Check BUSY(KS_STS[2]) is 0 and EIF(KS_STS[1]) is 0.
2. Configure DST(KS_METADATA[31:30]), NUMBER(KS_METADATA[25:20]), OWNER(KS_METADATA[18:16]), SIZE(KS_METADATA[12:8]),BS(KS_METADATA[4]), READABLE(KS_METADATA[2]), PRIV(KS_METADATA[1]), SEC(KS_METADATA[0]) according key's metadata.
3. Set OPMODE(KS_CTL[3:1]) to 001
4. Write key to KS_KEY0~KEY7 registers
5. Set START(KS_CTL[0]) to 1
6. Wait BUSY(KS_STS[2]) to 0.
7. Check EIF(KS_STS[1]) is 0.
8. Read key number from NUMBER(KS_METADATA [25:20]) and KS_OTPSTS register for double check.

Erase One Key Operation Programming Flow

Use erase one key operation to erase the specified key if key is unused. After the operation is finished, the space of erased key used will be released. This operation can only be used when the key is stored in SRAM.

1. Check BUSY(KS_STS[2]) is 0 and EIF(KS_STS[1]) is 0.
2. Configure DST(KS_METADATA[31:30]), NUMBER(KS_METADATA[25:20])
3. Set OPMODE(KS_CTL[3:1]) to 010 and START(KS_CTL[0]) to 1
4. Wait BUSY(KS_STS[2]) to 0.
5. Check EIF(KS_STS[1]) is 0.

Erase All Keys Operation Programming Flow

Use all one key operation can erase SRAM or Flash. After the operation is finished, the space of key used will be released. This operation can only be used to SRAM or Flash.

1. Check BUSY(KS_STS[2]) is 0 and EIF(KS_STS[1]) is 0.
2. Configure DST(KS_METADATA[31:30]).
3. Set OPMODE(KS_CTL[3:1]) to 011 and START(KS_CTL[0]) to 1
4. Wait BUSY(KS_STS[2]) to 0.
5. Check EIF(KS_STS[1]) is 0 and KS_REMAINING register to reset value

Revoke Key Operation Programming Flow

Use revoke key operation to revoke the specified key if key is unused. After the operation is finished, the key is unused but the space which key is used will not be released.

1. Check BUSY(KS_STS[2]) is 0 and EIF(KS_STS[1]) is 0.
2. Configure DST(KS_METADATA[31:30]), NUMBER(KS_METADATA[25:20])
3. Set OPMODE(KS_CTL[3:1]) to 100 and START(KS_CTL[0]) to 1
4. Wait BUSY(KS_STS[2]) to 0.

5. Check EIF(KS_STS[1]) is 0.

6.39.5.3 Access/Store keys by Crypto Engine

The key can be generated by crypto engine and located in Key Store. User can control PRNG to generate random key or operate the ECDH/ECDSA flow with PRNG and ECC without the need to control the programming interface of Key Store. The Key Store supports RSA to store the middle data for CRT and SCAP mode to improve decryption time.

There are policy restrictions for the crypto engine:

- Secure crypto engine creates the secure key and non-secure key; Non-secure crypto engine create the non-secure key only. The secure policy is shown in. Table 6.39-2
- Secure crypto engine can access the secure key and non-secure key; Non-secure crypto engine can access non-secure key only. The secure policy is shown in Table 6.39-2
- Privileged crypto engine creates the privileged key and non-privileged key; Non-privileged crypto engine create the non-privileged key only. The privilege policy is shown in Table 6.39-3
- Privileged crypto engine can access the privileged key and non- privileged key; Non-privileged crypto engine can access non- privileged key only. The privilege policy is shown in Table 6.39-3
- If a key is created by crypto engine and owner is set to CPU, the key is readable; If a key is created by crypto engine and owner is not set to CPU, the key is un-readable. User can erase key (in SRAM) or revoke key (in SRAM/Flash/OTP) when the key is no longer used. The condition that CPU and Crypto access key is shown in Table 6.39-4.
- The RSA key can be stored in SRAM and Flash, and the RSA middle data can be stored in SRAM only. CPU cannot read out RSA middle data.
- The size of HMAC key in Key Store is block size (i.e., HMAC-SHA512 and HMAC-SHA384 are 1024 bits; others are 512 bits).
- The random key generated from PRNG for ECDH/ECDSA used only can be stored in SRAM.

	Create/Access Secure Key	Create/Access Non-Secure Key
Secured Crypto Engine / CPU	V	V
Non-secured Crypto Engine / CPU	-	V

Table 6.39-2 Security Policy for Accessing Key

	Create/Access Privilege Key	Create/Access Non-Privilege Key
Privileged Crypto Engine / CPU	V	V
Non-privileged Crypto Engine / CPU	-	V

Table 6.39-3 Privilege Policy for Accessing Key

	CPU Read Key	Crypto Engine Access Key
Key created by CPU	Controlled by READABLE (KS_METADATA[2]) while create key operation	Controlled by OWNER (KS_METADATA[2]) while create key operation
Key created by Crypto Engine	OWNER in Key Control Register of Crypto Engine is set to CPU while Crypto	Controlled by OWNER in Key Control Register of Crypto Engine while Crypto

	Engine is operating.	Engine is operating.

Table 6.39-4 CPU and Crypto Engine Access Key for All Situations

6.39.5.4 Integrity Checking

The integrity checking function is always enabled to check key integrity. The integrity checking can detect if the key has been damaged.

While create key operation, the Key Store calculates the number of '0' and '1' and record the results. While read key operation, the Key Store will read out key, re-calculate the number of '0' and '1', and compare the result. If comparison is correct, the key can be read out from KS_KEY0~KS_KEY7. If comparison is failed, the key will not be read out and EIF(KS_STS[1]) will rising.

6.39.5.5 Data Scrambling

When data scrambling function is enabled, stored key is scrambled by scramble key. If data scrambling function is disabled, stored key is original key same as written in KS_KEY0~KS_KEY7 registers. The data scrambling function can protect key to avoid probing attack.

The data scrambling function can be enabled by SCMB(KS_CTL[11]) and scramble key is be written in KS_SCMBKEY registers(KS_SCMBKEY0~3) before creating key or reading key operation.

After OPMODE(KS_CTL[3:1]) is set to 001 and START(KS_CTL[0]) is set to 1, the key will be scrambled by scramble key and stored in Key Store. After OPMODE(KS_CTL[3:1]) is set to 000 and START(KS_CTL[0]) is set to 1, Key Store will read out the key and de-scrambled by scramble key.

6.39.5.6 Data Remanence Prevention

When data remanence prevention function is enabled, the Key Store will overwrite inverted data into same location.

Data remanence is a physical phenomenon in RAM and is the residual data that remains after delete or erase data. When RAM is power off, the data can remains a period of time. The hacker can use this property to get data. There are various countermeasure, such as clearing, encryption, overwriting and so on. The Key Store uses overwriting as the data remanence prevention method.

The data remanence prevention function is operated after OPMODE(KS_CTL[3:1]) is set to 101 and START(KS_CTL[0]) is set to 1. The Key Store will read out all key and invert key, and then write the inverted key in same location. After data remanence prevention function is finished, RAMINV(KS_STS[8]) will be inverted and IF(KS_STS[0]) will be set. User should to check that BUSY(KS_STS[2]) is set to 0 before starting the data remanence function.

If RAMINV(KS_STS[8]) is set to 1, it means key in SRAM is inverted. User can use the Key Store to operate any function normally no matter what states RAMINV is in.

6.39.5.7 Silent Access

When silent access function is enabled, the power of access key are all equal by equal number of '0' and '1'. If silent access function is disabled, the power of access key is sum of number of '0' and '1'.

The silent access function can be enabled by SILENT(KS_CTL[10]) before creating key or reading key. After SILENT (KS_CTL[10]) is enabled, the key stored in Flash will occupied double space to store key and inverted key, and the key stored in SRAM will occupied same space. The key stored in OTP is always enabled silent access function.

When CPU or Crypto engine creates key to Key Store, the key and its bitwise invert key are stored in Flash or OTP. When CPU or Crypto engine reads key from Key Store, both the key and its bitwise invert key will be read out from Flash or OTP. If Key Store verify the key and its bitwise invert key is failed, EIF(KS_STS[1]) will be set and cannot read the key from KS_KEY0~7 registers.

When CPU or Crypto engine creates a key and store it in SRAM, the Key Store calculates the number of '0' and '1', record the result, store key to SRAM, and store invert key to invalid entry. When CPU or

Crypto engine reads key from SRAM, the Key Store read out the record first, and then generate the complement data that opposite number of ‘0’ and ‘1’ as the record and store in invalid entry, and read out key with complement data finally.

6.39.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
KS Base Address: **KS_BA = 0x4003_5000**				
KS_CTL	KS_BA+0x000	R/W	Key Store Control Register	0x0000_0000
KS_METADATA	KS_BA+0x004	R/W	Key Store Metadata Register	0x0000_0000
KS_STS	KS_BA+0x008	R/W	Key Store Status Register	0x0000_0000
KS_REMAIN	KS_BA+0x00C	R	Key Store Remaining Space Register	0x0800_1000
KS_SCMBKEY0	KS_BA+0x010	R/W	Key Store Scramble Key Word 0 Register	0x0000_0000
KS_SCMBKEY1	KS_BA+0x014	R/W	Key Store Scramble Key Word 1 Register	0x0000_0000
KS_SCMBKEY2	KS_BA+0x018	R/W	Key Store Scramble Key Word 2 Register	0x0000_0000
KS_SCMBKEY3	KS_BA+0x01C	R/W	Key Store Scramble Key Word 3 Register	0x0000_0000
KS_KEY0	KS_BA+0x020	R/W	Key Store Entry Key Word 0 Register	0x0000_0000
KS_KEY1	KS_BA+0x024	R/W	Key Store Entry Key Word 1 Register	0x0000_0000
KS_KEY2	KS_BA+0x028	R/W	Key Store Entry Key Word 2 Register	0x0000_0000
KS_KEY3	KS_BA+0x02C	R/W	Key Store Entry Key Word 3 Register	0x0000_0000
KS_KEY4	KS_BA+0x030	R/W	Key Store Entry Key Word 4 Register	0x0000_0000
KS_KEY5	KS_BA+0x034	R/W	Key Store Entry Key Word 5 Register	0x0000_0000
KS_KEY6	KS_BA+0x038	R/W	Key Store Entry Key Word 6 Register	0x0000_0000
KS_KEY7	KS_BA+0x03C	R/W	Key Store Entry Key Word 7 Register	0x0000_0000
KS_OTPSTS	KS_BA+0x040	R	Key Store OTP Keys Status Register	0x0000_0000
KS_REMKCNT	KS_BA+0x044	R	Key Store Remaining Key Count Register	0x0020_0020

6.39.7 Register Description

Key Store Control Register (KS_CTL)

Register	Offset	R/W	Description	Reset Value
KS_CTL	KS_BA+0x000	R/W	Key Store Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
IEN	Reserved			SCMB	SILENT	Reserved	INIT
7	6	5	4	3	2	1	0
CONT	Reserved			OPMODE			START

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**IEN**	**Key Store Interrupt Enable Bit** 0 = Key Store Interrupt Disabled. 1 = Key Store Interrupt Enabled.
[14:12]	**Reserved**	Reserved.
[11]	**SCMB**	**Data Scramble Enable Bit** 0 = Data Scramble Disabled. 1 = Data Scramble Enabled.
[10]	**SILENT**	**Silent Access Enable Bit** 0 = Silent Access Disabled. 1 = Silent Access Enabled.
[9]	**Reserved**	Reserved.
[8]	**INIT**	**Key Store Initialization** User should to check BUSY(KS_STS[2]) is 0, and then write 1 to this bit and START(KS_CTL[0[), the Key Store will start to be initialized. After Key Store is initialized, INIT will be cleared. **Note:** Before executing INIT, user must check KS(SYS_SRAMPC1) is 00.
[7]	**CONT**	**Read/Write Key Continue Bit** 0 = Read/Write key operation is not continuous to previous operation. 1 = Read/Write key operation is continuous to previous operation.
[6:4]	**Reserved**	Reserved.
[3:1]	**OPMODE**	**Key Store Operation Mode** 000 = Read operation.

		001 = Create operation. 010 = Erase one key operation (only for key is in SRAM). 011 = Erase all keys operation (only for SRAM and Flash). 100 = Revoke key operation. 101 = Data Remanence prevention operation (only for SRAM). Others = reserved.
[0]	**START**	**Key Store Start Control Bit** 0 = No operation. 1 = Start the operation.

Key Store Metadata Register (KS_METADATA)

Register	Offset	R/W	Description	Reset Value
KS_METADATA	KS_BA+0x004	R/W	Key Store Metadata Register	0x0000_0000

31	30	29	28	27	26	25	24
DST		Reserved				NUMBER	
23	22	21	20	19	18	17	16
NUMBER				Reserved	OWNER		
15	14	13	12	11	10	9	8
Reserved			SIZE				
7	6	5	4	3	2	1	0
Reserved			BS	RVK	READABLE	PRIV	SEC

Bits	Description	
[31:30]	**DST**	**Key Location Selection Bits** 00 = Key is in SRAM. 01 = Key is in Flash. 10 = Key is in OTP. Others = reserved.
[29:26]	**Reserved**	Reserved.
[25:20]	**NUMBER**	**Key Number** Before read or erase one key operation starts, user should write the key number to be operated. When create operation is finished, user can read these bits to get its key number.
[19]	**Reserved**	Reserved.
[18:16]	**OWNER**	**Key Owner Selection Bits** 000 = Only for AES used. 001 = Only for HMAC engine used. 010 = Only for RSA engine exponential used (private key). 011 = Only for RSA engine middle data used. 100 = Only for ECC engine used. 101 = Only for CPU engine use. Others = reserved.
[15:13]	**Reserved**	Reserved.
[12:8]	**SIZE**	**Key Size Selection Bits** 00000 = 128 bits. 00001 = 163 bits. 00010 = 192 bits. 00011 = 224 bits. 00100 = 233 bits.

		00101 = 255 bits. 00110 = 256 bits. 00111 = 283 bits. 01000 = 384 bits. 01001 = 409 bits. 01010 = 512 bits. 01011 = 521 bits. 01100 = 571 bits. 10000 = 1024 bits. 10001 = 1536 bits. 10010 = 2048 bits. 10011 = 3072 bits. 10100 = 4096 bits. Others = reserved.
[7:5]	**Reserved**	Reserved.
[4]	**BS**	**Booting State Selection Bit** 0 = Set key used at all state. 1 = Set key used at boot loader state 1 (BL1 state).
[3]	**RVK**	**Key Revoke Control Bit** 0 = Key current selected will not be changed. 1 = key current selected will be change to revoked state.
[2]	**READABLE**	**Key Readable Control Bit** 0 = key is un-readable. 1 = key is readable.
[1]	**PRIV**	**Privilege Key Selection Bit** 0 = Set key as the non-privilege key. 1 = Set key as the privilege key.
[0]	**SEC**	**Secure Key Selection Bit** 0 = Set key as the non-secure key. 1 = Set key as the secure key.

Key Store Status Register (KS_STS)

Register	Offset	R/W	Description	Reset Value
KS_STS	KS_BA+0x008	R/W	Key Store Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							RAMINV
7	6	5	4	3	2	1	0
INITDONE	Reserved		FLASHFULL	SRAMFULL	BUSY	EIF	IF

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**RAMINV**	**Key Store SRAM Invert Status (Read Only)** 0 = Key Store key in SRAM is normal. 1 = Key Store key in SRAM is inverted.
[7]	**INITDONE**	**Key Store Initialization Done Status (Read Only)** 0 = Key Store is un-initialized. 1 = Key Store is initialized.
[6:5]	**Reserved**	Reserved.
[4]	**FLASHFULL**	**Key Storage at Flash Full Status Bit (Read Only)** 0 = Key Storage at Flash is not full. 1 = Key Storage at Flash is full.
[3]	**SRAMFULL**	**Key Storage at SRAM Full Status Bit (read only)** 0 = Key Storage at SRAM is not full. 1 = Key Storage at SRAM is full.
[2]	**BUSY**	**Key Store Busy Flag (read only)** 0 = Key Store is idle or finished. 1 = Key Store is busy.
[1]	**EIF**	**Key Store Error Flag** This bit is cleared by writing 1 and it has no effect by writing 0. 0 = No Key Store error. 1 = Key Store error interrupt.
[0]	**IF**	**Key Store Finish Interrupt Flag** This bit is cleared by writing 1 and it has no effect by writing 0. 0 = No Key Store interrupt.

		1 = Key Store operation done interrupt.

Key Store Remaining Space Register (KS_REMAIN)

Register	Offset	R/W	Description	Reset Value
KS_REMAIN	KS_BA+0x00C	R	Key Store Remaining Space Register	0x0800_1000

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

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:16]	**FRMNG**	**Key Store Flash Remaining Space** The FRMNG shows the remaining byte count space for Flash.
[15:13]	**Reserved**	Reserved.
[12:0]	**RRMNG**	**Key Store SRAM Remaining Space** The RRMNG shows the remaining byte count space for SRAM.

Key Store Scramble Key Register (KS_SCMBKEY)

Register	Offset	R/W	Description	Reset Value
KS_SCMBKEY0	KS_BA+0x010	R/W	Key Store Scramble Key Word 0 Register	0x0000_0000
KS_SCMBKEY1	KS_BA+0x014	R/W	Key Store Scramble Key Word 1 Register	0x0000_0000
KS_SCMBKEY2	KS_BA+0x018	R/W	Key Store Scramble Key Word 2 Register	0x0000_0000
KS_SCMBKEY3	KS_BA+0x01C	R/W	Key Store Scramble Key Word 3 Register	0x0000_0000

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

Bits	Description	
[31:0]	**SCMBKEY**	**Key Store Scramble Key** When SCMB(KS_CTL[]) is set to 1, user should write the scramble key in this register before new key stores in Key Store. If user does not write the scramble key in this register, the Key Store will use previous scramble key to execute data scramble function.

Key Store Entry Key Register (KS_KEY)

Register	Offset	R/W	Description	Reset Value
KS_KEY0	KS_BA+0x020	R/W	Key Store Entry Key Word 0 Register	0x0000_0000
KS_KEY1	KS_BA+0x024	R/W	Key Store Entry Key Word 1 Register	0x0000_0000
KS_KEY2	KS_BA+0x028	R/W	Key Store Entry Key Word 2 Register	0x0000_0000
KS_KEY3	KS_BA+0x02C	R/W	Key Store Entry Key Word 3 Register	0x0000_0000
KS_KEY4	KS_BA+0x030	R/W	Key Store Entry Key Word 4 Register	0x0000_0000
KS_KEY5	KS_BA+0x034	R/W	Key Store Entry Key Word 5 Register	0x0000_0000
KS_KEY6	KS_BA+0x038	R/W	Key Store Entry Key Word 6 Register	0x0000_0000
KS_KEY7	KS_BA+0x03C	R/W	Key Store Entry Key Word 7 Register	0x0000_0000

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

Bits	Description	
[31:0]	**KEY**	**Key Data** The register will be cleared if the Key Store executes the write operation or CPU completes the reading key.

Key Store OTP Keys Status Register (KS_OTPSTS)

Register	Offset	R/W	Description	Reset Value
KS_OTPSTS	KS_BA+0x040	R	Key Store OTP Keys Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
KEY7	KEY6	KEY5	KEY4	KEY3	KEY2	KEY1	KEY0

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**KEY7**	**OTP Key 7 Used Status** 0 = OTP key 7 is unused. 1 = OTP key 7 is used. **Note:** If chip is changed to RMA stage, the existing key will be revoked after initialization.
[6]	**KEY6**	**OTP Key 6 Used Status** 0 = OTP key 6 is unused. 1 = OTP key 6 is used. **Note:** If chip is changed to RMA stage, the existing key will be revoked after initialization.
[5]	**KEY5**	**OTP Key 5 Used Status** 0 = OTP key 5 is unused. 1 = OTP key 5 is used. **Note:** If chip is changed to RMA stage, the existing key will be revoked after initialization.
[4]	**KEY4**	**OTP Key 4 Used Status** 0 = OTP key 4 is unused. 1 = OTP key 4 is used. **Note:** If chip is changed to RMA stage, existing key will be revoked after initialization.
[3]	**KEY3**	**OTP Key 3 Used Status** 0 = OTP key 3 is unused. 1 = OTP key 3 is used. **Note:** If chip is changed to RMA stage, the existing key will be revoked after initialization.
[2]	**KEY2**	**OTP Key 2 Used Status** 0 = OTP key 2 is unused. 1 = OTP key 2 is used. **Note:** If chip is changed to RMA stage, the existing key will be revoked after initialization.

[1]	**KEY1**	**OTP Key 1 Used Status** 0 = OTP key 1 is unused. 1 = OTP key 1 is used. **Note:** If chip is changed to RMA stage, the existing key will be revoked after initialization.
[0]	**KEY0**	**OTP Key 0 Used Status** 0 = OTP key 0 is unused. 1 = OTP key 0 is used. **Note:** If chip is changed to RMA stage, the existing key will be revoked after initialization.

Key Store Remaining Key Count Register (KS_REMKCNT)

Register	Offset	R/W	Description	Reset Value
KS_REMKCNT	KS_BA+0x044	R	Key Store Remaining Key Count Register	0x0020_0020

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

Bits	Description	
[31:22]	**Reserved**	Reserved.
[21:16]	**FRMKCNT**	**Key Store Flash Remaining Key Count** The FRMKCNT shows the remaining key count for Flash.
[15:6]	**Reserved**	Reserved.
[5:0]	**RRMKCNT**	**Key Store SRAM Remaining Key Count** The RRMKCNT shows the remaining key count for SRAM.

6.40 LCD Controller

6.40.1 Overview

The LCD controller controls the device's built-in voltage/current drivers, which can drive externally connected LCD panels with up to 8 common planes (or called common electrodes, COMs) and 44 segments (SEGs). Every COM or SEG output pin of the device can supply the necessary voltage waveform to the connected LCD panels.

The LCD controller provides several configuration registers, by which users can effectively control a variety of LCD panels with specific considerations for display modes, driving capability, and power consumption.

6.40.2 Features

- Supports the following maximum COM/SEG combinations:
  - 320 pixels (8-COM x 40-SEG)
  - 252 pixels (6-COM x 42-SEG)
  - 176 pixels (4-COM x 44-SEG)
  - 104 pixels (8-COM x 13-SEG) for 64-pin package
- Supports up to 8 COM output pins, multiplexed with GPIO pins
- Supports up to 44 SEG output pins, multiplexed with GPIO pins
- Supports 3 bias levels: 1/2, 1/3, and 1/4
- Supports 8 duty ratios: 1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, and 1/8
- Supports both types A and B waveforms
- Supports a clock frequency divider, programmable from 0 to 1023, to generate the LCD operating frequency (F_{LCD})
- Supports LCD operating voltage (V_{LCD}) from 2.6 V to 3.6 V
- Selectable LCD operating voltage sources:
  - V_{LCD} (External dedicated V_{DD} pin for LCD) power
  - AV_{DD} (Analog V_{DD}) power
  - Built-in charge pump
- A built-in resistive network to generate required bias voltages
  - supports 2 drive modes: low-drive and high-drive modes
  - supports voltage buffers which are active only in the low-drive mode
- Supports a configurable power-saving mode. During this mode,
  - the resistive network temporarily changes to the low-drive mode, or
  - the voltage buffers are temporarily turned off.
- At the end of every frame, a dedicated flag is set and an interrupt can be programmed to occur.
- Supports a frame counter. At the end of frame counting, a dedicated flag is set and an interrupt can be programmed to occur.
- Supports LCD blinking capability. By using the frame counter, users have more flexibility

to adjust the blinking frequency.

- The LCD clock source is LIRC or LXT. LCD display or blinking can keep working even when the chip is in the power-down modes, only if at least one of LIRC and LXT is active.
- Supports a charging timer for the charge pump. By using this timer, users can estimate the loading of the charge pump, and adjust, if necessary, its charging power.

6.40.3 Block Diagram

The LCD controller block diagram is shown as follows.

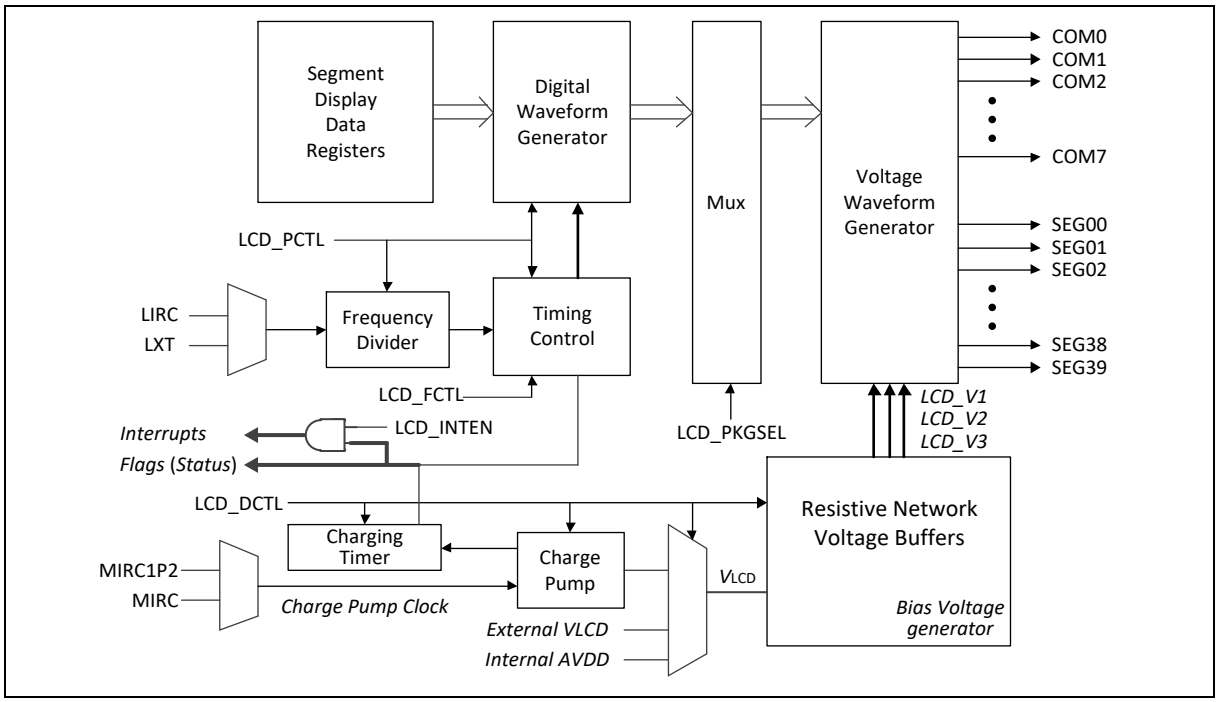


Figure 6.40-1 LCD Controller Block Diagram

6.40.4 Basic Configuration

- Reset Configuration
  - Reset the LCD controller by writing LCDRST (SYS_IPRST1[30]).
- LCD Clock (CLK_{LCD}) Configuration
  - Enable the LCD clock by setting LCDCKEN (CLK_APBCLK1[24]).
  - Select the source of the LCD clock, i.e., LIRC or LXT, by setting LCDSEL (CLK_CLKSEL1[2]).
- LCD Charge Pump Clock Configuration
  - Enable the LCD charge pump clock by setting LCDCPCKEN (CLK_APBCLK1[28]).
  - Select the LCD charge pump clock source, 1.2 MHz or 4 MHz, by setting LCDCPSEL (CLK_CLKSEL1[3]).
- COM/SEG Output Pin Configuration

Pin Name	128-Pin Package, LCD_PKGSEL[0] = 0	64-Pin Package, LCD_PKGSEL[0] = 1

	GPIO	MFP	GPIO	MFP
COM0	PC.0	MFP13	PC.0	MFP13
COM1	PC.1	MFP13	PC.1	MFP13
COM2	PC.2	MFP15	PC.2	MFP15
COM3	PC.3	MFP15	PC.3	MFP15
COM4	PC.4	MFP15	PC.4	MFP14
COM5	PC.5	MFP15	PC.5	MFP14
COM6	PD.8	MFP15	PA.0	MFP5
COM7	PD.9	MFP15	PA.1	MFP5
SEG00	PD.14	MFP15	PD.1	MFP15
SEG01	PH.11	MFP15	PD.2	MFP15
SEG02	PH.10	MFP15	PD.3	MFP15
SEG03	PH.9	MFP15	PA.2	MFP5
SEG04	PH.8	MFP15	PA.3	MFP5
SEG05	PE.0	MFP15	PA.4	MFP5
SEG06	PE.1	MFP15	PA.5	MFP5
SEG07	PE.2	MFP15	PA.6	MFP9
SEG08	PE.3	MFP15	PA.7	MFP9
SEG09	PE.4	MFP15	PC.6	MFP13
SEG10	PE.5	MFP15	PC.7	MFP13
SEG11	PE.6	MFP15	PA.8	MFP14
SEG12	PE.7	MFP15	PA.9	MFP14
SEG13	PD.6	MFP15	(Same as COM7 pin when Duty > 1/8)	
SEG14	PD.7	MFP15	(Same as COM6 pin when Duty > 1/7)	
SEG15	PG.15	MFP13	(Same as COM5 pin when Duty > 1/6)	
SEG16	PG.14	MFP15	(Same as COM4 pin when Duty > 1/5)	
SEG17	PG.13	MFP15	N/A	
SEG18	PG.12	MFP15	N/A	
SEG19	PG.11	MFP15	N/A	
SEG20	PG.10	MFP15	N/A	
SEG21	PG.9	MFP15	N/A	
SEG22	PE.15	MFP15	N/A	
SEG23	PE.14	MFP15	N/A	
SEG24	PA.0	MFP11	N/A	

SEG25	PA.1	MFP11	N/A
SEG26	PA.2	MFP11	N/A
SEG27	PA.3	MFP11	N/A
SEG28	PA.4	MFP15	N/A
SEG29	PA.5	MFP15	N/A
SEG30	PE.10	MFP15	N/A
SEG31	PE.9	MFP15	N/A
SEG32	PE.8	MFP15	N/A
SEG33	PH.7	MFP15	N/A
SEG34	PH.6	MFP15	N/A
SEG35	PH.5	MFP15	N/A
SEG36	PH.4	MFP15	N/A
SEG37	PG.4	MFP15	N/A
SEG38	PG.3	MFP15	N/A
SEG39	PG.2	MFP15	N/A
SEG40	(Same as COM7 pin when Duty > 1/8)		N/A
SEG41	(Same as COM6 pin when Duty > 1/7)		N/A
SEG42	(Same as COM5 pin when Duty > 1/6)		N/A
SEG43	(Same as COM4 pin when Duty > 1/5)		N/A

Table 6.40-1 COM/SEG Output Configuration

6.40.5 Functional Description

6.40.5.1 LCD Controller Enable/Disable

To enable the LCD controller, write 1 to EN (LCD_CTL[0]). The connected LCD panel starts to display.

To disable the LCD controller, write 0 to EN (LCD_CTL[0]). The connected LCD panel stops display after the last frame is finished. (**Note:** a frame, which is explained in more details later, is a complete period of waveform repeatedly applied to every COM and SEG) Voltages of all COM and SEG output pins will become V_{SS}.

When the LCD controller is disabled, LCD-related analog circuits (the resistive network, the voltage buffers, the charge pump, etc.) are turned off to save power consumption.

6.40.5.2 LCD Configurations

The LCD controller should be configured before enabled. There are 4 aspects of configurations:

- about LCD panel specification
- about frame setting and control
- about driving capability and power consumption
- about output pin selection (multiplexing)

To obtain optimized display effects and power consumption, the user should fully understand the

characteristics of the connected LCD panel, and do proper configurations supported by the LCD controller.

6.40.5.3 LCD Panel Specification

Most of LCD panels have several general specifications. Those supported by the LCD controller are:

- Bias Level. BIAS (LCD_PCTL[1:0]) support 3 bias levels: 1/2, 1/3, and 1/4.
- Duty Ratio. DUTY (LCD_PCTL[4:2]) support 8 duty ratios: 1, 1/2, 1/3, …, and 1/8.
- Waveform Type. TYPE (LCD_PCTL[5]) supports types A and B both.
- LCD Operating Frequency (F_{LCD}). FREQDIV (LCD_PCTL[17:8]) provide a divider to generate a clock with frequencies about from 32 Hz to 32 kHz.
- LCD Operating Voltage (V_{LCD}). For the built-in charge pump, CPVSEL (LCD_PCTL[20:18]) and CPVTUNE (LCD_PCTL[27:24]) can configure the output voltage from 2.6 V to 3.6 V.

LCD Clock (CLK_{LCD})

The clock is the fundamental clock on which the timing of all LCD waveforms is based. Its source is LIRC or LXT, and the frequency is about 32 kHz.

Frame

A frame is a period of waveform repeatedly applied to each COM/SEG while the LCD controller is enabled.

Duty Ratio

Duty ratio is defined as 1 / (number of COMs used by the LCD panel).

Assume the number is *n*. A frame can be divided into *n* time slots. In each time slot, only one COM is active and others are inactive. COM[0], COM[1], …, and COM[*n*-1] will become active for each time slot in turn.

LCD Operating Frequency (F_{LCD})

F_{LCD} is programmable by using the formula: (CLK_{LCD} Frequency) / (FREQDIV + 1), where FREQDIV is the value of the register bits LCD_PCTL[17:8] and ranges from 0 to 1023.

Waveform Type A

A frame is divided into *n* time slots. The length of each time slot is ($1/F_{LCD}$) x 2.

Waveform Type B

A frame is divided into an even frame and an odd frame. Each even or odd frame is divided into *n* time slots. The length of a time slot is ($1/F_{LCD}$).

Frame Time

Frame time is the duration of a frame.

The frame time for type A is (1/(Duty Ratio)) x ($1/F_{LCD}$) x 2.

The frame time for type B is (1/(Duty Ratio)) x ($1/F_{LCD}$).

Frame Rate

Frame rate is 1 / (Frame Time), i.e., the number of frames applied per second

The frame rate for type A is (Duty Ratio) x F_{LCD} x 1/2.

The frame rate for type B is (Duty Ratio) x F_{LCD}

Setting a proper frame rate is important to achieve a high LCD display quality. If the frame rate is too low, flickering may occur. If the frame rate is too high, ghosting may occur and unnecessary power is wasted.

[Example] Assume Duty Ratio = 1/6, and F_{LCD} = 1 kHz,

The frame rate for type A is 1/6 x 1 kHz x 1/2 ≈ 85.3 Hz.

The frame rate for type B is 1/6 x 1 kHz ≈ 170.7 Hz.

Note: In some LCD technical document, the frame rate for type B should be (Duty Ratio) x F_{LCD} x 1/2, since an even or odd frame is considered as a half-frame. However, In this document, we consider an even or odd frame as a full frame.

LCD Operating Voltage (V_{LCD})

V_{LCD} is the maximum voltage level required by an LCD panel. The device can support LCD panels operating with V_{LCD} from 2.6 V to 3.6 V.

There are 3 possible V_{LCD} voltage sources for the device: V_{LCD} power, AV_{DD} power, and the built-in charge pump. V_{LCD} power is supplied by the external dedicated V_{DD} pin for LCD. AV_{DD} power is chip's internal dedicated power supply for analog circuits.

If V_{LCD} or AV_{DD} power is selected, users must make sure that the V_{LCD} or AV_{DD} power pin is connected to a power supply with the same voltage as V_{LCD}.

If the charge pump is selected, users can set its output voltage by programming CPVSEL (LCD_PCTL[20:18]). Furthermore, users can fine tune this voltage by programming CPVTUNE (LCD_PCTL[27:24]) to slightly increase or decrease V_{LCD}.

Bias Voltage

Bias voltages are intermediate voltages evenly between 0 V and V_{LCD}.

Bias Level

Bias level is defined as 1/ (number of bias voltages used by the LCD panel + 1). Any voltage waveform applied to a COM/SEG is composed of these bias voltages.

For bias 1/2, one bias voltage is used: 1/2 V_{LCD}.

For bias 1/3, two bias voltages are used: 1/3 V_{LCD} and 2/3 V_{LCD}

For bias 1/4, three bias voltages are used: 1/4 V_{LCD}, 2/4 V_{LCD}, and 3/4 V_{LCD}

Segment Display Data

The LCD controller supports up to 8 COMs, therefore, every SEG can correspond to up to 8 pixels, i.e., SEG-COM0, SEG-COM1, SEG-COM2, …, and SEG-COM7.

The display data accompanying with a SEG is consist of 8 bits. Each bit corresponds to a pixel. Bit 0 corresponds to pixel SEG-COM0, bit 1 corresponds to pixel SEG-COM1, bit 2 corresponds to pixel SEG-COM2, and so on.

A pixel appears as "Light" if its corresponding bit is 0, and "Dark" if 1. (**Note:** for some LCD panels, the light/dark status may be reversed).

All display data is stored in the LCD Segment Display Data registers from LCD_DATA00 to LCD_DATA10. Users can update the registers at any time, however, the LCD display cannot change until the next frame.

Figure 6.40-2 to Figure 6.40-7 show the examples of LCD output waveform. The first 3 waveforms are of type A, with bias 1/2, 1/3, and 1/4, respectively. The next 3 waveforms are of type B, with bias 1/2, 1/3, and 1/4, respectively.

All waveforms are duty 1/4. That is, only COM0, COM1, COM2, and COM3 are involved in the display. The output voltage of COM4 ~ COM7 is 0 V. According to the regularity shown in the figures, the COM waveforms for other duty ratios can be easily obtained.

Two SEG outputs are also depicted in the figures: SEG00 with 8-bit display data = (xxxx_1101) and SEG01 with 8-bit display data = (xxxx_0100). Only 4 COMs are involved, therefore, the 4 MSBs of segment display data are don't care.

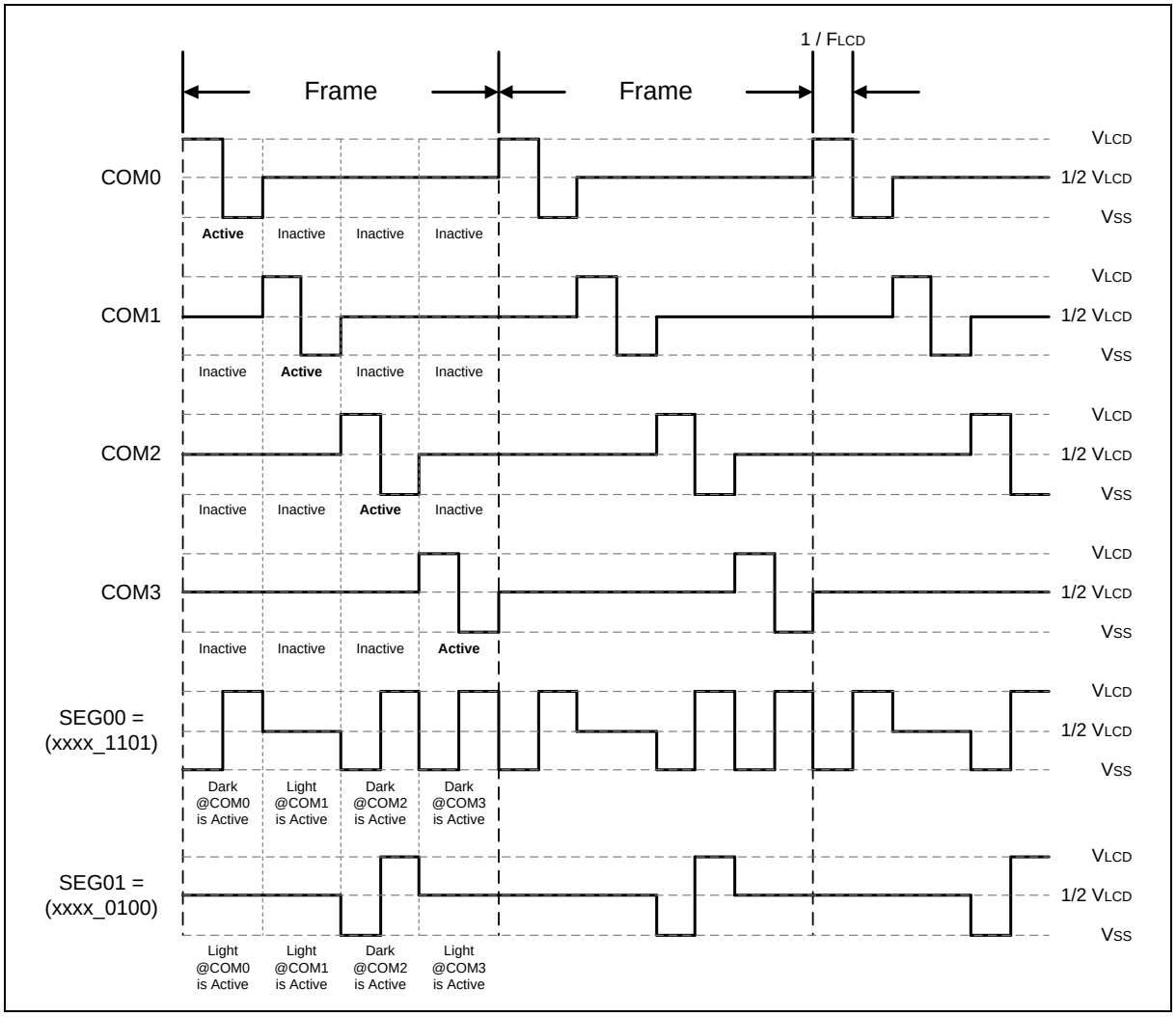


Figure 6.40-2 LCD Output Waveform of Type A, Duty 1/4, Bias 1/2

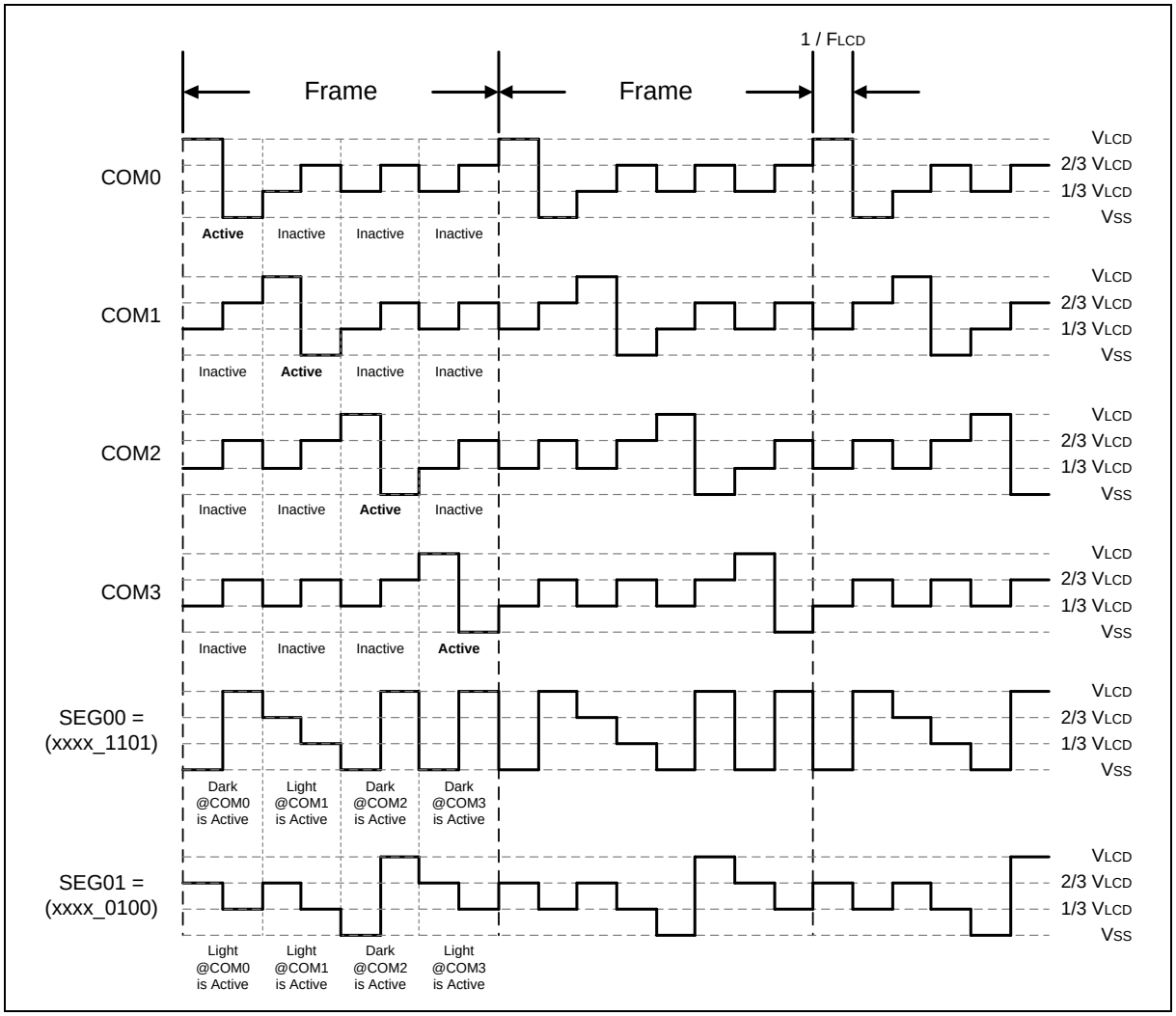


Figure 6.40-3 LCD Output Waveform of Type A, Duty 1/4 Bias 1/3

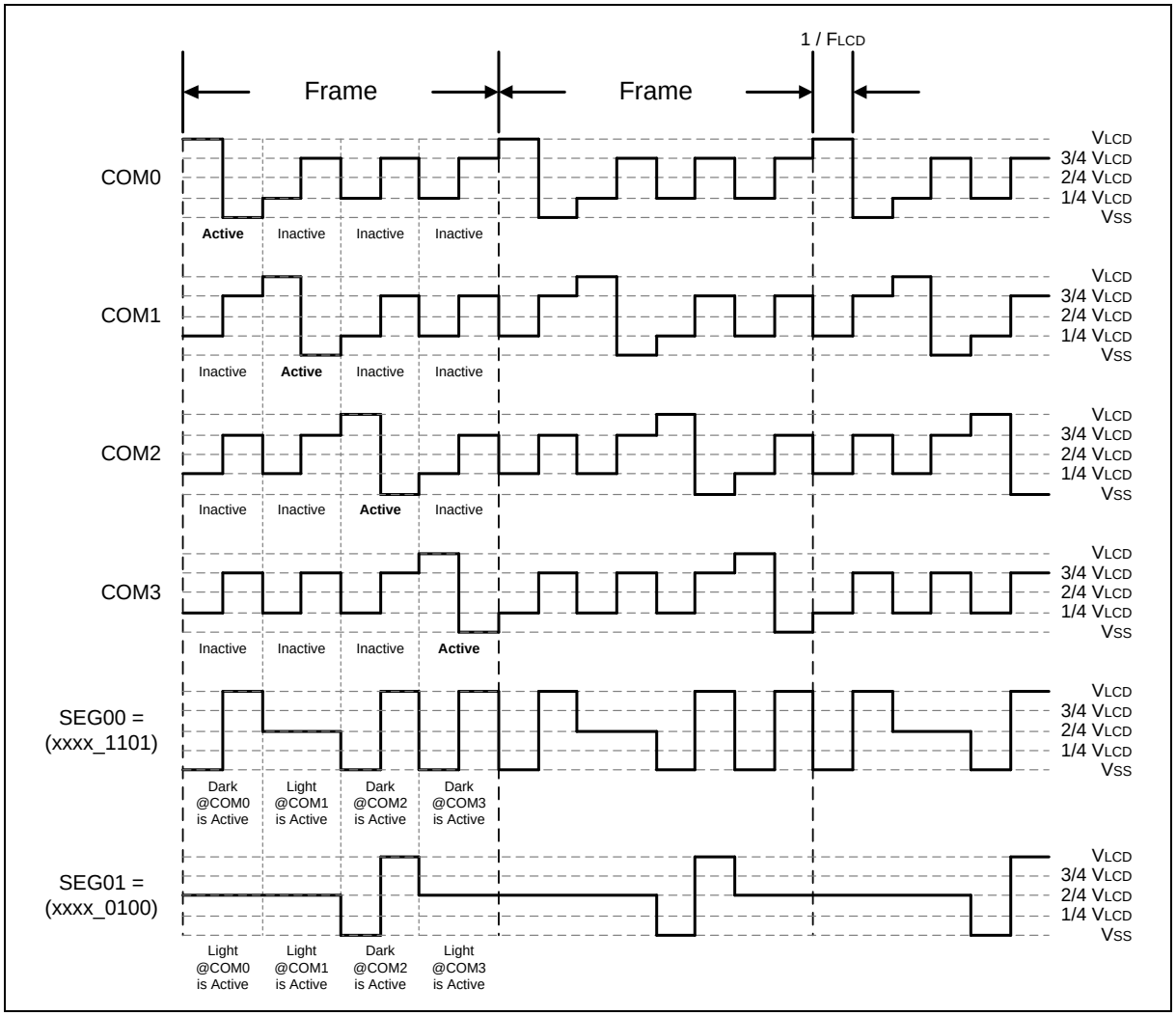


Figure 6.40-4 LCD Output Waveform of Type A, Duty 1/4, Bias 1/4

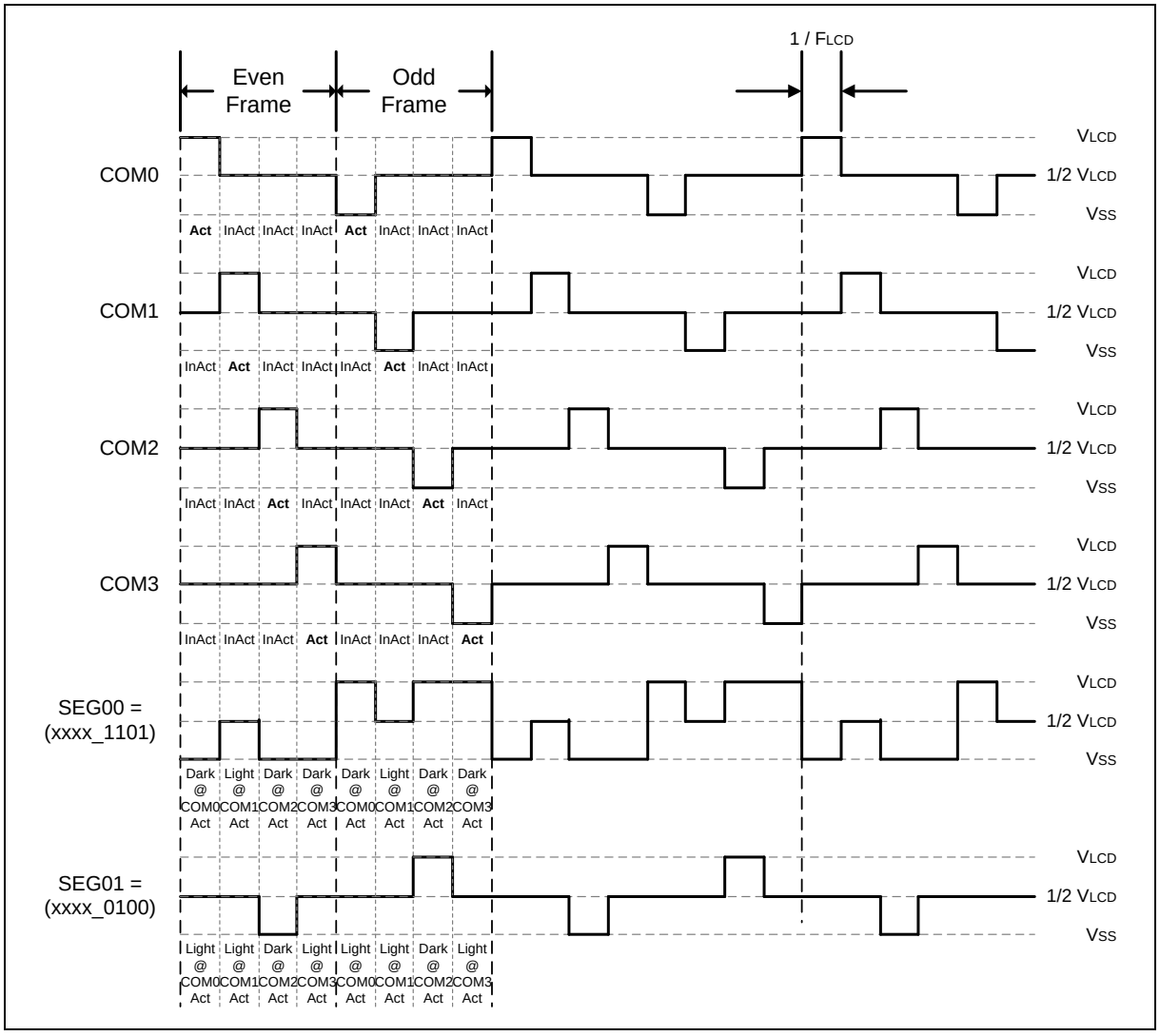


Figure 6.40-5 LCD Output Waveform of Type B, Duty 1/4, Bias 1/2

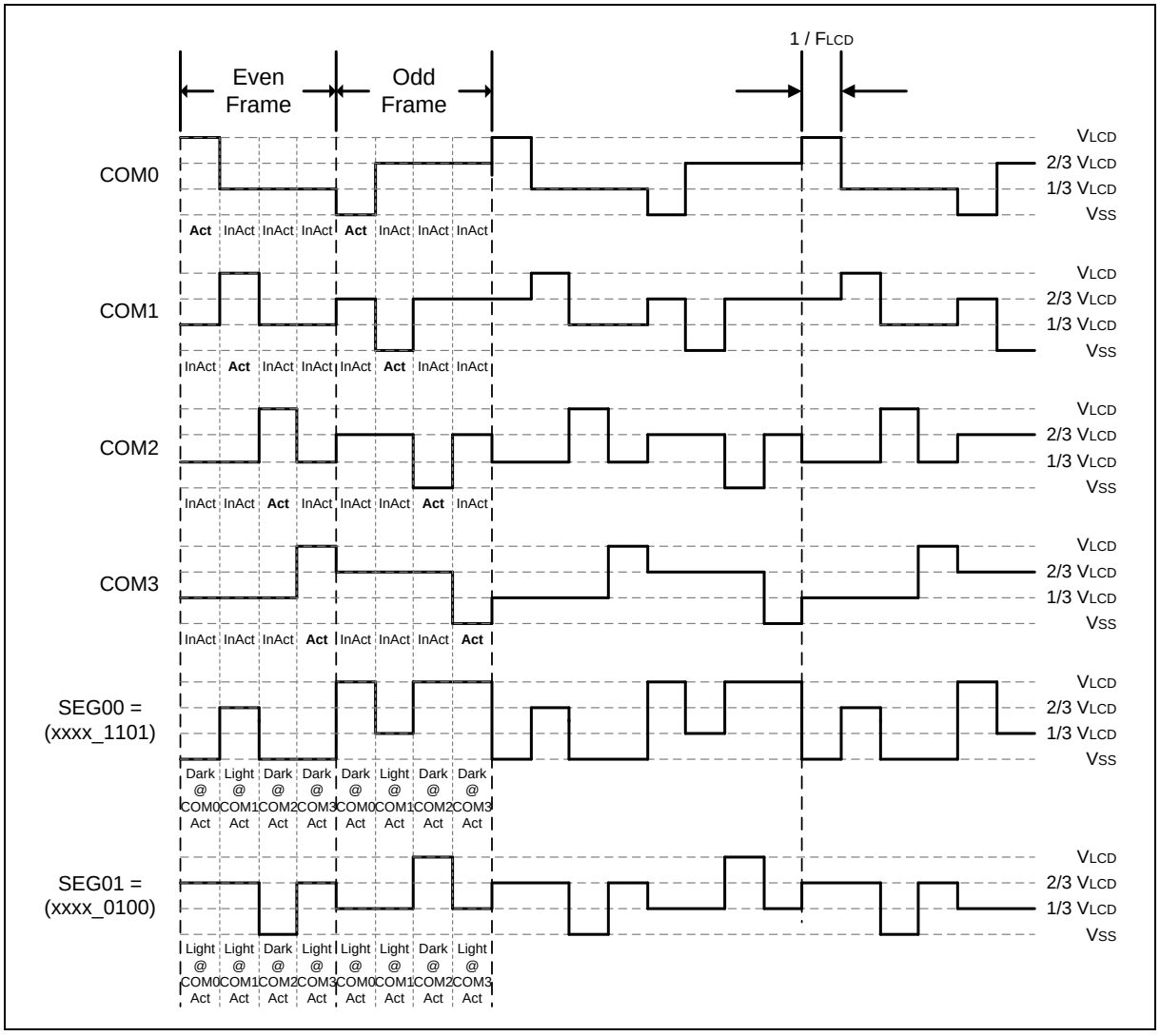


Figure 6.40-6 LCD Output Waveform of Type B, Duty 1/4, Bias 1/3

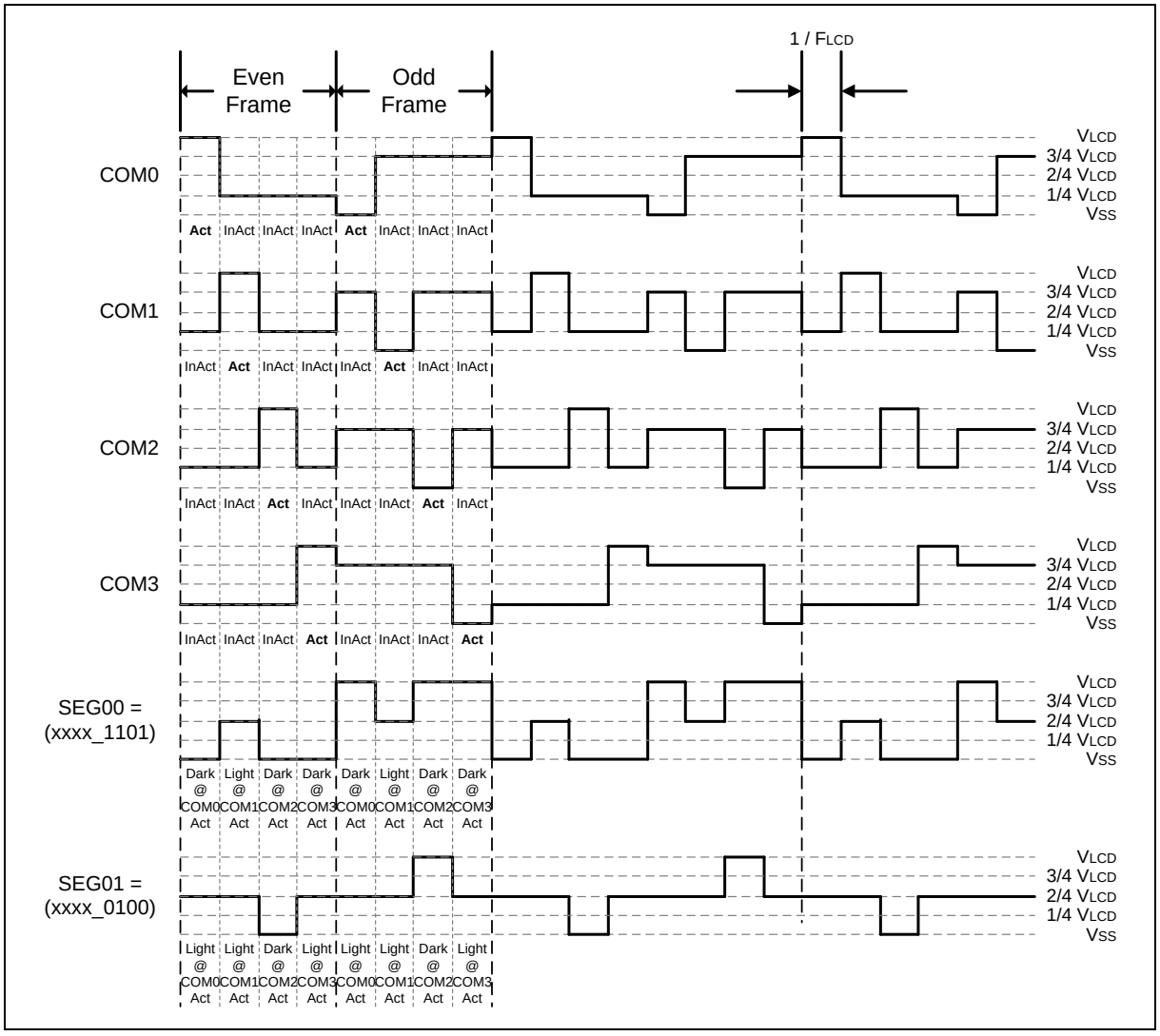


Figure 6.40-7 LCD Output Waveform of Type B, Duty 1/4, Bias 1/4

Waveform Inverse

For some LCD panels, the display effects (brightness or contrast) or power consumption will be better if the applied waveforms are inverted. Any voltage level *V* in the original waveform is converted to (V_{LCD} – *V*). An example of inverted waveform is depicted in the Figure 6.40-8.

To toggle the waveforms inversely, write 1 to INV (LCD_PCTL[6]).

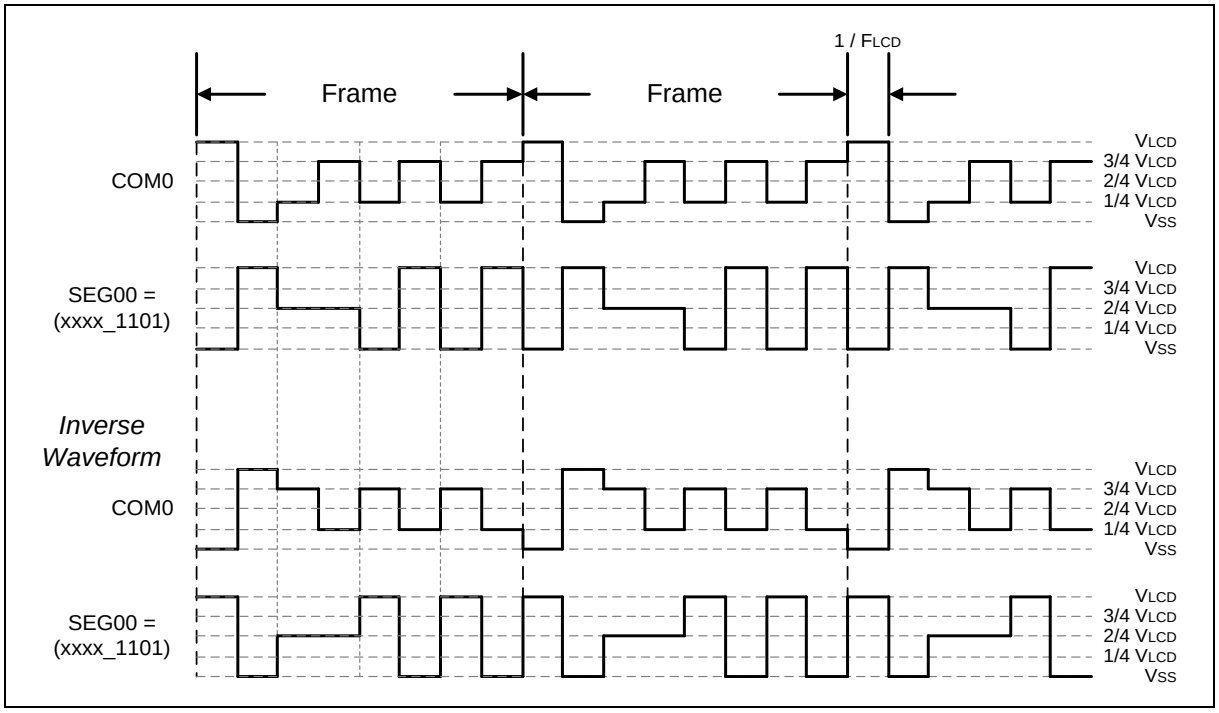


Figure 6.40-8 Waveform Inverse

6.40.5.4 Frame Setting and Control

Frame Counter

The LCD controller provides a frame counter, which automatically increases by one at the end of every frame. When the counter reaches the frame counting value FCV (LCD_FCTL[17:8]), it recounts from 0 at the end of the next frame.

Users can set the value of FCV in the register bits LCD_FCTL[17:8].

Flags and Interrupts

At the end of every frame, the hardware automatically set a dedicated flag to 1. Users can, if necessary, trigger an interrupt when this event occurs.

At the end of frame counting, which is also an end of a frame, the hardware set another dedicated flag to 1. Similarly, users can trigger an interrupt on this event.

Users can read these two flags FCEF (LCD_STS[0] End of Frame Counting Flag) and FEF (LCD_STS[1] End of Frame Flag) from LCD_STS[1:0]. Each individual flag can be cleared by writing 1 to this flag.

Interrupts can be enabled by setting FCEIEN (End of Frame Counting Interrupt Enable Bit) and FEIEN (End of Frame Interrupt Enable Bit). Clearing an interrupt uses the same way as clearing a flag.

Figure 6.40-9 shows the timing of frame counting in more details.

For type B waveform, these events only occur at the end of odd frames, not even frames.

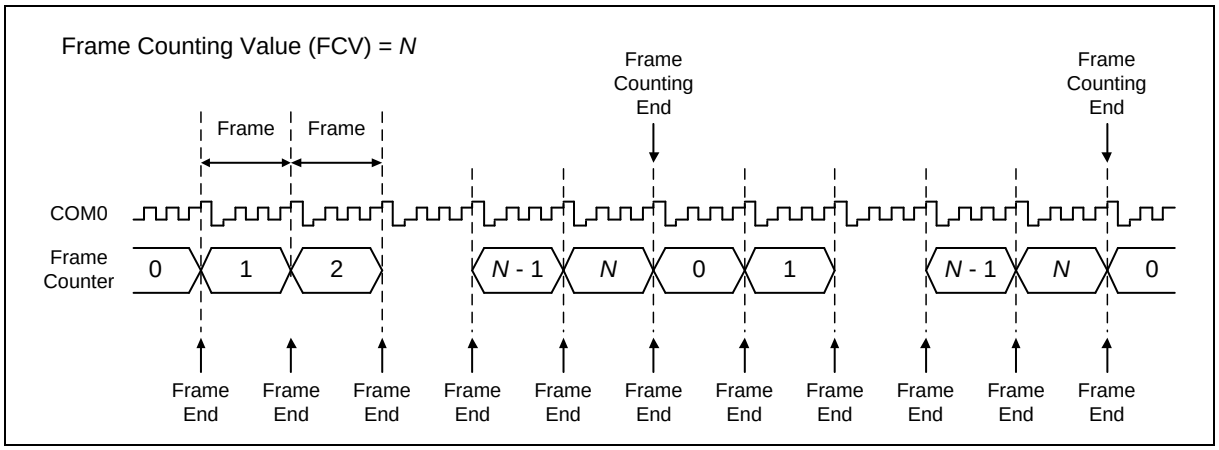


Figure 6.40-9 Frame Counting

Blinking

The LCD controller supports the blinking feature. The LCD display switches on/off continuously at a given frequency.

The frequency is determined by the value of FCV. Figure 6.40-10 demonstrates the timing of blinking. The blinking frequency can be obtained by calculating the following formula:

Blinking Frequency = (F_{LCD}/2) x (Duty Ratio) x (1/(FCV + 1)) x 1/2

The blinking feature can be enabled by writing 1 to BLINK (LCD_FCTL[0]).

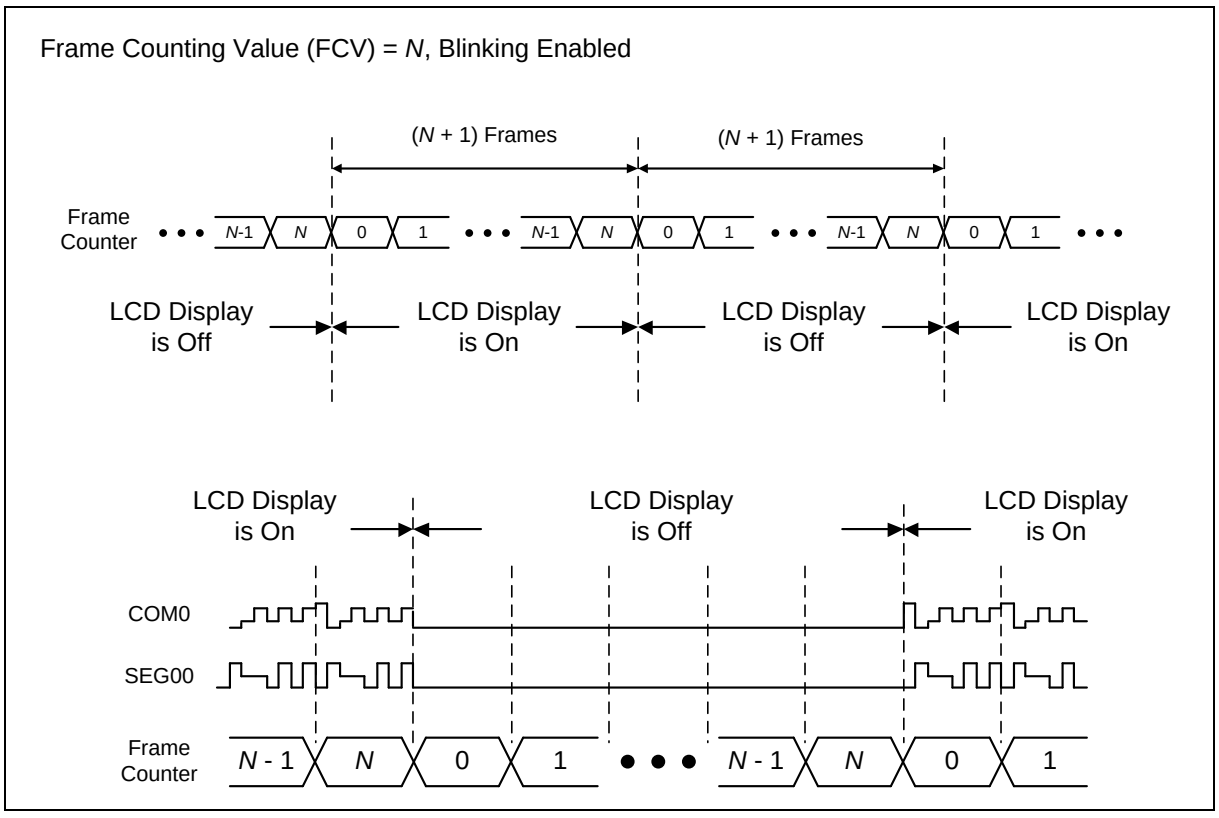


Figure 6.40-10 Blinking

6.40.5.5 Driving Capability and Power Consumption

LCD Operating Voltage Source

There are 3 possible sources of V_{LCD}:

- V_{LCD} Power, external power supply through the V_{LCD} power pin
- AV_{DD} Power, external power supply through the AV_{DD} power pin and dedicated to the analog circuits
- Built-In Charge Pump

Users can select the source by programming VSRC (LCD_DCTL[1:0]).

Whenever the V_{LCD} or AV_{DD} power is selected, the charge pump is always turned off to save power consumption.

If the V_{LCD} source is the internal charge pump, users can set its output voltage by programming CPVSEL (LCD_PCTL[20:18]).

Due to process variations, the actual V_{LCD} generated by the charge pump may have small errors. Users can fine tune this voltage by writing proper values to CPVTUNE (LCD_PCTL[27:24]) to slightly increase or decrease V_{LCD}.

Resistive Network

All the intermediate bias voltages are generated by a built-in resistive network, as shown in Figure 6.40-11.

There are two drive modes for the resistive network:

- Low-Drive Mode. Only high-resistance resistors are utilized. Smaller driving current is supplied.
- High-Drive Mode. Both low- and high-resistance resistors are utilized. Larger driving current is supplied.

According to the driving-current requirement for the connected LCD panel, users can select a proper mode by setting RESMODE (LCD_DCTL[2]).

Voltage Buffer

To cope with large capacitive loading on some large-scale LCD panels, voltage buffers associated with each intermediate bias voltage output are provided. With these buffers turned on, a more stable waveform can be obtained.

The voltage buffers can be turned on only when the resistive network is in the low-drive mode.

User can turn on the voltage buffers by writing 1 to BUFEN (LCD_DCTL[3]). However, when the resistive network is in the high-drive mode, the voltage buffers will be automatically turned off, and the setting of BUFEN will be ignored in this situation.

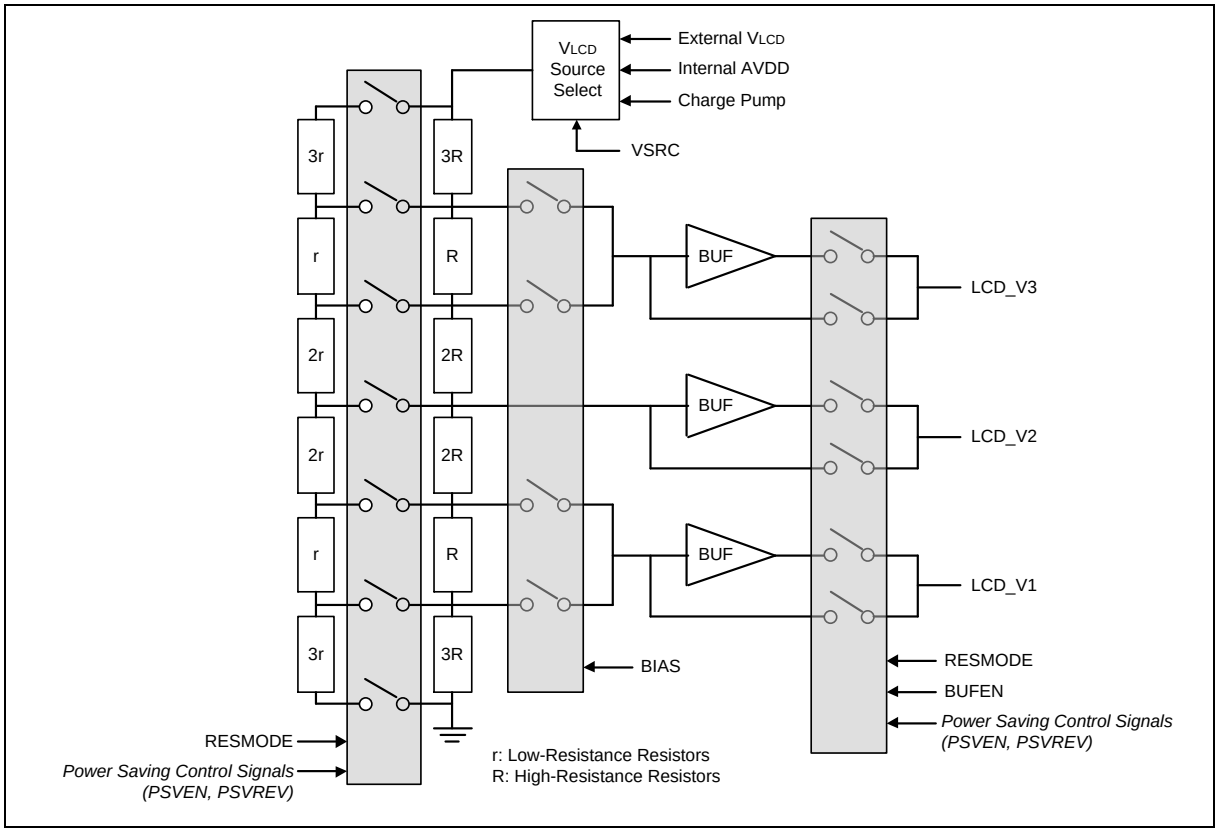


Figure 6.40-11 Resistive Network and Voltage Buffers

Charging Timer

If the charge pump is selected as the source of V_{LCD}, the following 3 steps are executed repeatedly once the LCD controller is enabled:

1. The charge pump is turned on and continues to charge V_{LCD} until V_{LCD} reaches the voltage specified in CPVSEL (LCD_PCTL[20:18]).
2. The charge pump is turned off.
3. When V_{LCD} drops, due to driving the LCD panel, by a preset voltage, the charge pump will go to step 1 to recharge V_{LCD} again.

If the duration of step 1 is short, it means that the charge pump has sufficient charging power to drive the LCD panel.

If the duration of step 1 is very long or, even worse, endless, it means V_{LCD} is very hard to or never reaches the specified voltage. That is, the charging power is seriously insufficient to drive the LCD panel.

The LCD controller provides a charging timer, which keep counting during the charge pump being in step 1. When the charge pump goes to step 2, the timer stops. When the charge pump goes back to step 1 from step 3, the timer will restart all over again.

The LCD controller also provides a programmable timeout value for the charging timer. Once charging timer reaches the timeout value, the hardware automatically set a dedicated flag to 1. Users can, if necessary, trigger an interrupt when this event occurs.

The charging timer stops counting when the charge pump stops charging or a timeout occurs. At this point, the value of the charging timer is recorded in CTIME (LCD_STS[28:16]).

The charging timer restarts counting when the charge pump restarts charging, or the flag or interrupt is cleared.

Users can write a reasonable timeout value to CTOTIME (LCD_DCTL[28:16]). By monitoring the occurrence of this interrupt, users can evaluate the necessity of adjusting the charging power owned by the charge pump.

Brightness (Contrast) Enhancement

For some large-scale LCD panels, larger diving current may be required. To improve the brightness or contrast, users can adopt the following configurations:

- Set the resistive network in the high-drive mode.
- Set the resistive network in the low-drive mode, with voltage buffers turned on.
- If the source of V_{LCD} is the charge pump, the charging power can be improved by switching the charge pump clock from 1.2 MHz to 4 MHz. The driving current supplied by the charge pump will be raised.

Low Power Operation --- Null Frame

To meet the low-power operating requirements, the LCD controller can generate null frames inserted into the normal frames. A null frame actually is a time period of all COM/SEG outputs being 0 V. So no power consumed during this period.

Users can decide the length of a null frame by setting NFTIME (LCD_FCTL[27:24]), and the number of continuous normal frames by setting NFNUM (LCD_FCTL[31:28]) that will be inserted by one null frame. One full frame time is (1 / F_{LCD}) x NFTIME. Figure 6.40-12 shows the insertion of null frames.

Even though the null frame can achieve low power effects, it may make the LCD display fade. Users should carefully evaluate the trade-off.

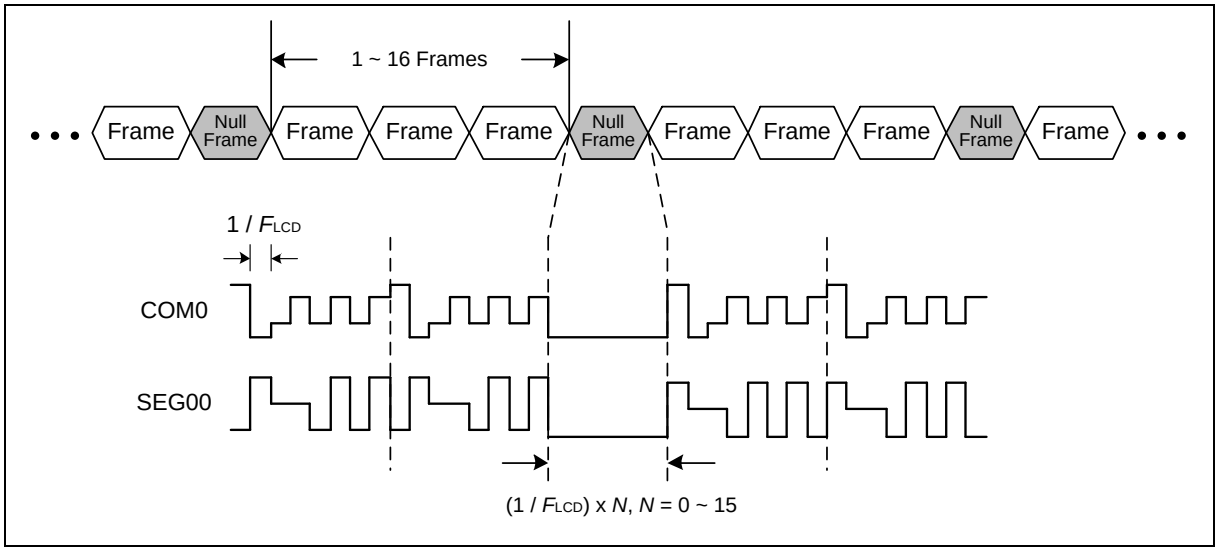


Figure 6.40-12 Null Frame

Low Power Operation --- Power Saving Mode

The LCD controller also provide a power saving mode to cope with low-power operating environments. Besides enabling the power saving mode, users should also define a power saving period. During this period,

- if the resistive network is in the high-drive mode, it is temporarily switched to the low-drive

mode.

- if the resistive network is in the low-drive mode with the voltage buffers turned on, the voltage buffers are temporarily turned off.
- If the resistive network is in the low-drive mode without the voltage buffers turned on, nothing happens. The power saving mode takes no effect.

Figure 6.40-13 shows the timing of the power saving mode.

Users can enable the power saving mode by writing 1 to PSVEN (LCD_DCTL[4]). The periods of T1 and T2, depicted in Figure 6.40-13, are set in PSVT1 (LCD_DCTL[11:8]) and PSVT2 (LCD_DCTL[15:12]), respectively.

Sometimes reversing the timing of power saving can result in better low-power effect under some operating conditions. Users can try it by writing 1 to PSVREV (LCD_DCTL[5]).

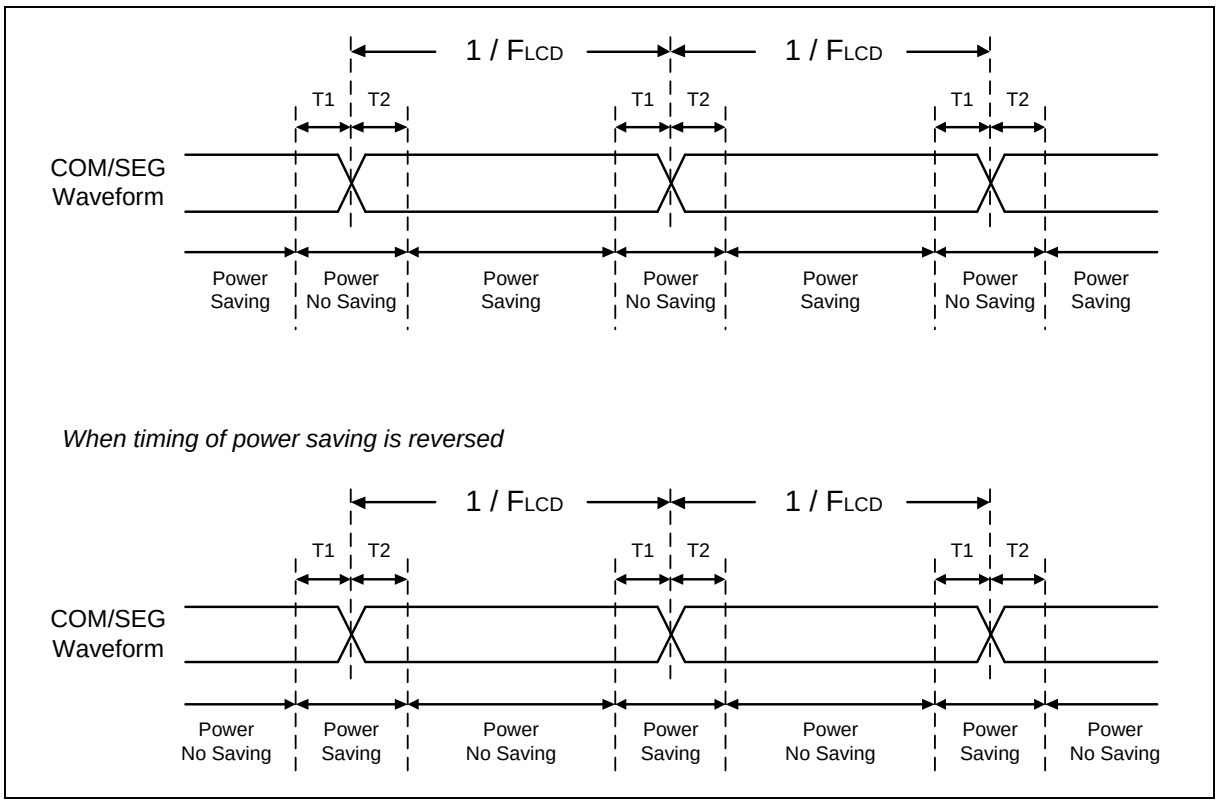


Figure 6.40-13 Power Saving Mode

Power Down Prerequisites

Voltage waveform generated by the LCD controller and applied to LCD panels relies on F_{LCD} and V_{LCD}. The LCD controller can continue to drive the connected LCD panel even when the chip is in the Power-down modes, if F_{LCD} and V_{LCD} are available.

To make the LCD panel keep display or blinking, users must make sure that the following requirements must be met before the chip enters a Power-down mode:

- At least one of LIRC and LXT is available.
- At least one of three voltage sources, V_{LCD} power, AV_{DD} power, and the charge pump, can supply the voltage.
  - If the source of V_{LCD} is the charge pump, at least one of 1.2 MHz or 4 MHz clock

is available.

6.40.5.6 Output Pin Selection (Multiplexing)

The LCD Controller supports up to 48 output pins. Some COMs or SEGs are connected to different output pins for devices with different package type. Users can set up the outputs by programming PKG (LCD_PGKSEL[0]). If the PKG is 0, the maximum COM/SEG combinations are 320 pixels (8-COM x 40-SEG) at 128-pin package device. And if the PKG is 1, the maximum COM/SEG combinations are 104 pixels (8-COM x 13-SEG) at 64-pin package device.

6.40.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
LCD Base Address: **LCD_BA = 0x400B_B000** **LCD non-secure base address is LCD_BA + 0x1000_0000.**				
LCD_CTL	LCD_BA+0x00	R/W	LCD Control Register	0x0000_0000
LCD_PCTL	LCD_BA+0x04	R/W	LCD Panel Control Register	0x0000_0000
LCD_FCTL	LCD_BA+0x08	R/W	LCD Frame Control Register	0x0000_0000
LCD_DCTL	LCD_BA+0x0C	R/W	LCD Driving Control Register	0x0000_0000
LCD_PKGSEL	LCD_BA+0x10	R/W	LCD Package Selection Register	0x0000_0000
LCD_STS	LCD_BA+0x14	R/W	LCD Status Register	0x0000_0000
LCD_INTEN	LCD_BA+0x18	R/W	LCD Interrupt Enable Register	0x0000_0000
LCD_DATA00	LCD_BA+0x20	R/W	LCD Segment Display Data Register 0	0x0000_0000
LCD_DATA01	LCD_BA+0x24	R/W	LCD Segment Display Data Register 1	0x0000_0000
LCD_DATA02	LCD_BA+0x28	R/W	LCD Segment Display Data Register 2	0x0000_0000
LCD_DATA03	LCD_BA+0x2C	R/W	LCD Segment Display Data Register 3	0x0000_0000
LCD_DATA04	LCD_BA+0x30	R/W	LCD Segment Display Data Register 4	0x0000_0000
LCD_DATA05	LCD_BA+0x34	R/W	LCD Segment Display Data Register 5	0x0000_0000
LCD_DATA06	LCD_BA+0x38	R/W	LCD Segment Display Data Register 6	0x0000_0000
LCD_DATA07	LCD_BA+0x3C	R/W	LCD Segment Display Data Register 7	0x0000_0000
LCD_DATA08	LCD_BA+0x40	R/W	LCD Segment Display Data Register 8	0x0000_0000
LCD_DATA09	LCD_BA+0x44	R/W	LCD Segment Display Data Register 9	0x0000_0000
LCD_DATA10	LCD_BA+0x48	R/W	LCD Segment Display Data Register 10	0x0000_0000

6.40.7 Register Description

LCD Control Register (LCD_CTL)

Register	Offset	R/W	Description	Reset Value
LCD_CTL	LCD_BA+0x00	R/W	LCD Control Register	0x0000_0000

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

Bits	Description	
[31]	SYNC	**LCD Enable/Disable Synchronizing Indicator (Read Only)** When user writes 0/1 to EN (LCD_CTL[0]), the LCD Controller needs some synchronizing time to completely disable/enable the LCD display function. During this time, this bit keeps at 1. 0 = LCD display function is completely Disabled/Enabled. 1 = LCD display function is not yet completely Disabled/Enabled. **Note 1:** The synchronizing time to enable LCD display function is not constant. It is between one and two cycles of CLK_{LCD}. **Note 2:** The LCD display function cannot be disabled until the end of a frame. Thus, the maximum synchronizing time to disable LCD display function could be as long as one frame time.
[30:1]	Reserved	Reserved.
[0]	EN	**LCD Display Enable Bit** 0 = LCD display function Disabled. 1 = LCD display function Enabled. **Note 1:** When user writes 1 to this bit, the LCD Controller needs some synchronizing time to completely enable the LCD display function. Before that, the read value of this bit is still 0. **Note 2**: When user writes 0 to this bit, the LCD Controller needs some synchronizing time to completely disable the LCD display function. Before that, the read value of this bit is still 1.

LCD Panel Control Register (LCD_PCTL)

Register	Offset	R/W	Description	Reset Value
LCD_PCTL	LCD_BA+0x04	R/W	LCD Panel Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				CPVTUNE			
23	22	21	20	19	18	17	16
Reserved			CPVSEL			FREQDIV	
15	14	13	12	11	10	9	8
FREQDIV							
7	6	5	4	3	2	1	0
Reserved	INV	TYPE	DUTY			BIAS	

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:24]	**CPVTUNE**	**LCD Operating Voltage Fine Tuning** This field is used to fine-tune the LCD operating voltage. 0 = No tuning. 1 = decrease by 1 unit of voltage. 2 = decrease by 2 units of voltage. 3 = decrease by 3 units of voltage. 4 = decrease by 4 units of voltage. 5 = decrease by 5 units of voltage. 6 = decrease by 6 units of voltage. 7 = decrease by 7 units of voltage. 8 = increase by 8 units of voltage. 9 = increase by 7 units of voltage. 10 = increase by 6 units of voltage. 11 = increase by 5 units of voltage. 12 = increase by 4 units of voltage. 13 = increase by 3 units of voltage. 14 = increase by 2 units of voltage. 15 = increase by 1 unit of voltage. **Note 1:** A unit of voltage is about 0.03 V. **Note 2:** This field is meaningful only if the V_{LCD} source is the charge pump. Otherwise, this field is ignored.
[23:21]	**Reserved**	Reserved.
[20:18]	**CPVSEL**	**LCD Operating Voltage Select** This field is used to select the LCD operating voltage. 0 = 2.6 V.

		1 = 2.8 V. 2 = 3.0 V. 3 = 3.2 V. 4 = 3.4 V. 5 = 3.6 V. Others = Reserved. **Note:** This field is meaningful only if the V_{LCD} source is the charge pump. Otherwise, this field is ignored.
[17:8]	**FREQDIV**	**LCD Operating Frequency Divider** The field is used to divide CLK_{LCD} to generate the LCD operating frequency. LCD Operating Frequency, F_{LCD} = (CLK_{LCD} Frequency) / (FREQDIV + 1). **Note 1:** FREQDIV can be set from 0 to 1023, therefore, the fastest F_{LCD} is equal to CLK_{LCD} frequency, and the lowest F_{LCD} is equal to CLK_{LCD} frequency divided by 1024. **Note 2:** LCD frame rate is (F_{LCD}) x (Duty Ratio) x 1/2 for type A waveform, and (F_{LCD}) x (Duty Ratio) for type B waveform. **Example:** Assume F_{LCD} is 1 kHz, duty ratio is 1/4, then the LCD frame rate is 1 kHz x (1/4) x (1/2) = 128 Hz for type A waveform, and. 1 kHz x (1/4) = 256 Hz for type B waveform.
[7]	**Reserved**	Reserved.
[6]	**INV**	**LCD Waveform Inverse** This bit is used to set the inverse LCD waveform. 0 = COM/SEG waveform is normal. 1 = COM/SEG waveform is inversed.
[5]	**TYPE**	**LCD Waveform Type Selection** This bit is used to select the waveform type. 0 = Type A. 1 = Type B.
[4:2]	**DUTY**	**LCD Duty Ratio Selection** This field is used to select the duty ratio. 0 = 1/1 Duty. 1 = 1/2 Duty. 2 = 1/3 Duty. 3 = 1/4 Duty. 4 = 1/5 Duty. 5 = 1/6 Duty. 6 = 1/7 Duty. 7 = 1/8 Duty.
[1:0]	**BIAS**	**LCD Bias Level Selection** This field is used to select the bias level. 0 = Reserved. 1 = 1/2 Bias. 2 = 1/3 Bias. 3 = 1/4 Bias.

LCD Frame Control Register (LCD_FCTL)

Register	Offset	R/W	Description	Reset Value
LCD_FCTL	LCD_BA+0x08	R/W	LCD Frame Control Register	0x0000_0000

31	30	29	28	27	26	25	24
NFNUM				NFTIME			
23	22	21	20	19	18	17	16
Reserved						FCV	
15	14	13	12	11	10	9	8
FCV							
7	6	5	4	3	2	1	0
Reserved							BLINK

Bits	Description	
[31:28]	**NFNUM**	**Number of Frames Inserted By One Null Frame** This field is used to specify the number of continuous normal frames inserted by one null frame. The number of continuous normal frames is (NFNUM + 1) frames.
[27:24]	**NFTIME**	**Null Frame Time** This field is used to configure the length of a null frame. One null frame time is (1 / F_{LCD}) x NFTIME. **Note:** All COM and SEG output voltages are 0 V during a null frame.
[23:18]	**Reserved**	Reserved.
[17:8]	**FCV**	**Frame Counting Value** This field indicates the maximum value that the frame counter can reach. **Note 1:** The frame counter automatically increases by 1 at the end of every frame. When the counter reaches FCV, it will recounts from 0 at the end of the next frame. At this moment, the hardware sets a dedicated flag to 1, and triggers a dedicated interrupt if it is enabled. **Note 2:** For type B waveform, the frame counter increases at the end of odd frames, not even frames.
[7:1]	**Reserved**	Reserved.
[0]	**BLINK**	**LCD Blinking Enable Bit** 0 = LCD blinking function Disabled. 1 = LCD blinking function Enabled.

LCD Driving Control Register (LCD_DCTL)

Register	Offset	R/W	Description	Reset Value
LCD_DCTL	LCD_BA+0x0C	R/W	LCD Driving Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			CTOTIME				
23	22	21	20	19	18	17	16
CTOTIME							
15	14	13	12	11	10	9	8
PSVT2				PSVT1			
7	6	5	4	3	2	1	0
Reserved		PSVREV	PSVEN	BUFEN	RESMODE	VSRC	

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:16]	**CTOTIME**	**Charging Timer Timeout Time** This field is used to specify the timeout value for the charging timer. When the charging timer reaches this timeout value, a status bit or an interrupt will occur. The timeout is calculated by the following formula: Timeout = 31.25 us x (CTOTIME + 1.), where 31.25 us is the cycle time of CLK_{LCD}, whose frequency is assumed to be 32 kHz. CTOTIME can be set as 0, 1, 2,…, 8191, so the minimum timeout is 31.25 us, and the maximum timeout is 31.25 x 8192 = 256 ms.
[15:12]	**PSVT2**	**Power Saving "On Time" Setting** The "On Time" of the power saving mode is calculated as "On Time" = 15.625 us x (PSVT2 + 1.), where 15.625 us is the half-cycle time of CLK_{LCD}, whose frequency is assumed to be 32 kHz. PSVT2 can be set as 0, 1, 2, …, 15, so the minimum "On Time" is about 15.625 us, and the maximum "On Time" is about 15.625 x 16 = 250 us. **Note:** In the following two cases, the power saving mode is disabled. The setting of PSVT2 bits is ignored. 1. PSVEN = 0. 2. RESMODE = 0 and BUFEN = 0 (In this case, PSVEN is ignored).
[11:8]	**PSVT1**	**Power Saving "Enable Time" Setting** The "Enable Time" of the power saving mode is calculated as "Enable Time" = 15.625 us x (PSVT1 + 1),. where 15.625 us is the half-cycle time of CLK_{LCD}, whose frequency is assumed to be 32 kHz. PSVT1 can be set as 0, 1, 2, …, 15, so the minimum "Enable Time" is about 15.625 us, and

		the maximum “Enable Time” is about 15.625 x 16 = 250 us. **Note:** In the following two cases, the power saving mode is disabled. The setting of PSVT1 bits is ignored. 1. PSVEN = 0. 2. RESMODE = 0 and BUFEN = 0.
[7:6]	**Reserved**	Reserved.
[5]	**PSVREV**	**Power Saving Timing Reverse** When the timing is reversed, the original power saving period becomes no power saving, and the original no power saving period becomes power saving. 0 = Timing of power saving is normal. 1 = Timing of power saving is reversed.
[4]	**PSVEN**	**Power Saving Mode Enable Bit** 0 = Power Saving Mode Disabled. 1 = Power Saving Mode Enabled. **Note:** when RESMODE = 0 and BUFEN = 0, the output drivers consumes the least driving current. In this case, the power saving mode is automatically disabled. The setting of PSVEN bit is ignored.
[3]	**BUFEN**	**Voltage Buffer Enable Bit** 0 = Voltage Buffer Disabled. 1 = Voltage Buffer Enabled. **Note:** When RESMODE = 1, the voltage buffers are automatically disabled. The setting of BUFEN bit is ignored. **Note:** When RESMODE = 0 and BUFEN = 0, the output driving capability will be too weak to drive the external LCD panel. Users should avoid this setting.
[2]	**RESMODE**	**Resistive Network Driving Mode** 0 = Low-Drive Mode. 1 = High-Drive Mode. **Note:** When RESMODE = 1, the voltage buffers are automatically disabled. The setting of BUFEN bit is ignored. **Note:** When RESMODE = 0 and BUFEN = 0, the output driving capability will be too weak to drive the external LCD panel. Users should avoid this setting.
[1:0]	**VSRC**	**LCD Operating Voltage Source** 0 = V_{LCD} Power. 1 = AV_{DD} Power. 2 = Built-In Charge Pump. 3 = None. **Note:** Whenever the LCD controller is disabled, all V_{LCD} sources are automatically cut off.

LCD Package Selection Register (LCD_PKGSEL)

Register	Offset	R/W	Description	Reset Value
LCD_PKGSEL	LCD_BA+0x10	R/W	LCD Package Selection Register	0x0000_0000

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

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	PKG	Device Package Type Selection 0 = 128-Pin Package. 1 = 64-Pin Package.

LCD Status Register (LCD_STS)

Register	Offset	R/W	Description	Reset Value
LCD_STS	LCD_BA+0x14	R/W	LCD Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			CTIME				
23	22	21	20	19	18	17	16
CTIME							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					CTOF	FEF	FCEF

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:16]	**CTIME**	**Charging Timer Value (Read Only)** The field contains the value of the charging timer. It records the charging time of the charge pump. The charging timer stops counting when the charge pump stops charging or a timeout occurs. At this moment, the hardware dumps the current charging timer value into this field. Charging Time = 31.25 us x (CTIME + 1), where 31.25 us is the cycle time of CLK_{LCD}, whose frequency is assumed to be 32 kHz.
[15:3]	**Reserved**	Reserved.
[2]	**CTOF**	**Charging Timeout Flag** This flag is automatically set by hardware when the charging timer reaches the timeout value. 0 = Charging Timeout did not occur. 1 = Charging Timeout occurred. **Note:** User can clear this bit by writing 1 to it.
[1]	**FEF**	**End of Frame Flag** This flag is automatically set by hardware at the end of a frame. 0 = End of Frame did not occur. 1 = End of Frame occurred. **Note 1:** User can clear this bit by writing 1 to it. **Note 2:** For type B waveform, this flag is set only at the end of an odd frame.
[0]	**FCEF**	**End of Frame Counting Flag** This flag is automatically set by hardware at the end of a frame, and the frame counter value must be equal to FCV (LCD_FCTL[17:8], Frame Counting Value). 0 = End of Frame Counting did not occur. 1 = End of Frame Counting occurred. **Note 1:** User can clear this bit by writing 1 to it.

		Note 2: For type B waveform, this flag is set only at the end of an odd frame.

LCD Interrupt Enable Register (LCD_INTEN)

Register	Offset	R/W	Description	Reset Value
LCD_INTEN	LCD_BA+0x18	R/W	LCD Interrupt Enable Register	0x0000_0000

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

Bits	Description	
[31:3]	Reserved	Reserved.
[2]	CTOIEN	**Charging Timeout Interrupt Enable Bit** An interrupt occurs when the charging timer reaches the timeout value. 0 = Charging Timeout Interrupt Disabled. 1 = Charging Timeout Interrupt Enabled.
[1]	FEIEN	**End of Frame Interrupt Enable Bit** An interrupt occurs at the end of a frame. 0 = End of Frame Interrupt Disabled. 1 = End of Frame Interrupt Enabled. **Note:** For type B waveform, the interrupt occurs only at the end of an odd frame.
[0]	FCEIEN	**End of Frame Counting Interrupt Enable Bit** An interrupt occurs at the end of a frame, and the frame counter value must be equal to FCV (LCD_FCTL[17:8], Frame Counting Value). 0 = End of Frame Counting Interrupt Disabled. 1 = End of Frame Counting Interrupt Enabled. **Note:** For type B waveform, the interrupt occurs only at the end of an odd frame.

LCD Segment Display Data Register (LCD_DATAxx)

Register	Offset	R/W	Description	Reset Value
LCD_DATA00	LCD_BA+0x20	R/W	LCD Segment Display Data Register 0	0x0000_0000
LCD_DATA01	LCD_BA+0x24	R/W	LCD Segment Display Data Register 1	0x0000_0000
LCD_DATA02	LCD_BA+0x28	R/W	LCD Segment Display Data Register 2	0x0000_0000
LCD_DATA03	LCD_BA+0x2C	R/W	LCD Segment Display Data Register 3	0x0000_0000
LCD_DATA04	LCD_BA+0x30	R/W	LCD Segment Display Data Register 4	0x0000_0000
LCD_DATA05	LCD_BA+0x34	R/W	LCD Segment Display Data Register 5	0x0000_0000
LCD_DATA06	LCD_BA+0x38	R/W	LCD Segment Display Data Register 6	0x0000_0000
LCD_DATA07	LCD_BA+0x3C	R/W	LCD Segment Display Data Register 7	0x0000_0000
LCD_DATA08	LCD_BA+0x40	R/W	LCD Segment Display Data Register 8	0x0000_0000
LCD_DATA09	LCD_BA+0x44	R/W	LCD Segment Display Data Register 9	0x0000_0000
LCD_DATA10	LCD_BA+0x48	R/W	LCD Segment Display Data Register 10	0x0000_0000

31	30	29	28	27	26	25	24
DD3							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DD2							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DD1							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DD0							

Bits	Description	
[31:24]	**DD3**	**Display Data of Segments *S*, where *S* is 4 x *N* + 3, and *N* is 0, 1, 2, …, 10** Each bit specifies the brightness of each pixel in a segment. 0 = The pixel is light. 1 = The pixel is dark. **Note 1:** DD3 corresponds to SEG03, SEG07, SEG11, SEG15, SEG19, SEG23, SEG27, SEG31, SEG35, SEG39, and SEG43. **Note 2:** Each bit, DD3[n], corresponds to COMn, n= 0 ~ 7. **[Example]** Assume 1/4 Duty, and DD3 = 1001_0110 at LCD_DATA07,. LCD_DATA07[31:24] corresponds to SEG31 (4 x 7 + 3 = 31.), the pixel SEG31-COM0 is light (LCD_DATA07[24] = 0.), the pixel SEG31-COM1 is dark (LCD_DATA07[25] = 1.),

		the pixel SEG31-COM2 is dark (LCD_DATA07[26] = 1.), the pixel SEG31-COM3 is light (LCD_DATA07[27] = 0.), LCD_DATA07[31:28] are ignored, since COMs from 4 to 7 are not used.
[23:16]	**DD2**	**Display Data of Segments *S*, where *S* is 4 x *N* + 2, and *N* is 0, 1, 2, …, 10** Each bit specifies the brightness of each pixel in a segment. 0 = The pixel is light. 1 = The pixel is dark. **Note 1:** DD2 corresponds to SEG02, SEG06, SEG10, SEG14, SEG18, SEG22, SEG26, SEG30, SEG34, SEG38, and SEG42. **Note 2:** Each bit, DD2[n], corresponds to COMn, n= 0 ~ 7. **[Example]** Assume 1/4 Duty, and DD2 = 1001_0110 at LCD_DATA07,. LCD_DATA07[23:16] corresponds to SEG30 (4 x 7 + 2 = 30.), the pixel SEG30-COM0 is light (LCD_DATA07[16] = 0.), the pixel SEG30-COM1 is dark (LCD_DATA07[17] = 1.), the pixel SEG30-COM2 is dark (LCD_DATA07[18] = 1.), the pixel SEG30-COM3 is light (LCD_DATA07[19] = 0.), LCD_DATA07[23:20] are ignored, since COMs from 4 to 7 are not used.
[15:8]	**DD1**	**Display Data of Segments *S*, where *S* is 4 x *N* + 1, and *N* is 0, 1, 2, …, 10** Each bit specifies the brightness of each pixel in a segment. 0 = The pixel is light. 1 = The pixel is dark. **Note 1:** DD1 corresponds to SEG01, SEG05, SEG09, SEG13, SEG17, SEG21, SEG25, SEG29, SEG33, SEG37, and SEG41. **Note 2:** Each bit, DD1[n], corresponds to COMn, n= 0 ~ 7. **[Example]** Assume 1/4 Duty, and DD1 = 1001_0110 at LCD_DATA07,. LCD_DATA07[15:8] corresponds to SEG29 (4 x 7 + 1 = 29.), the pixel SEG29-COM0 is light (LCD_DATA07[8] = 0.), the pixel SEG29-COM1 is dark (LCD_DATA07[9] = 1.), the pixel SEG29-COM2 is dark (LCD_DATA07[10] = 1.), the pixel SEG29-COM3 is light (LCD_DATA07[11] = 0.), LCD_DATA07[15:12] are ignored, since COMs from 4 to 7 are not used.
[7:0]	**DD0**	**Display Data of Segments *S*, where *S* is 4 x *N* + 0, and *N* is 0, 1, 2, …, 10** Each bit specifies the brightness of each pixel in a segment. 0 = The pixel is light. 1 = The pixel is dark. **Note 1:** DD0 corresponds to SEG00, SEG04, SEG08, SEG12, SEG16, SEG20, SEG24, SEG28, SEG32, SEG36, and SEG40. **Note 2:** Each bit, DD0[n], corresponds to COMn, n= 0 ~ 7. **[Example]** Assume 1/4 Duty, and DD0 = 1001_0110 at LCD_DATA07,. LCD_DATA07[7:0] corresponds to SEG28 (4 x 7 + 0 = 28.), the pixel SEG28-COM0 is light (LCD_DATA07[0] = 0.), the pixel SEG28-COM1 is dark (LCD_DATA07[1] = 1.), the pixel SEG28-COM2 is dark (LCD_DATA07[2] = 1.), the pixel SEG28-COM3 is light (LCD_DATA07[3] = 0.), LCD_DATA07[7:4] are ignored, since COMs from 4 to 7 are not used.

6.41 Tamper Controller (TC)

6.41.1 Overview

To protect the content of the Internal secrets from being attacked by hackers, the Tamper controller provides various attack detection and attack event response. The attack detection includes pins, clock and system voltage. When an attack is detected, the sensitive data like crypto session keys can be cleared by the attack event response.

6.41.2 Features

- Includes voltage, clock and I/O tamper detectors:
  - Voltage detector: detects voltage glitch including low voltage domain (LV) and high voltage domain (HV).
    - HV detector detects if V_{DD} > 4.0V
    - LV detector detects if LDO_CAP > ± 20%
    - Power loss detector indicates power status of VBAT < 1.4 V
  - Clock detector: detects if external clock (LXT) is failed or stopped
  - I/O tamper detector: detects GPF6~11 pins
- Active shield in SRAM with power/GND and tamper I/O.
- Provides event response after an attack detected:
  - Clear key or data content in SRAM and Flash of Key Store, and revoke the OTP in Key Store
  - Clear RTC spare register
  - Reset Crypto
  - Chip reset
  - Interrupt
  - Wake up the system
- Not supported in Deep Power-down mode.

6.41.3 Block Diagram

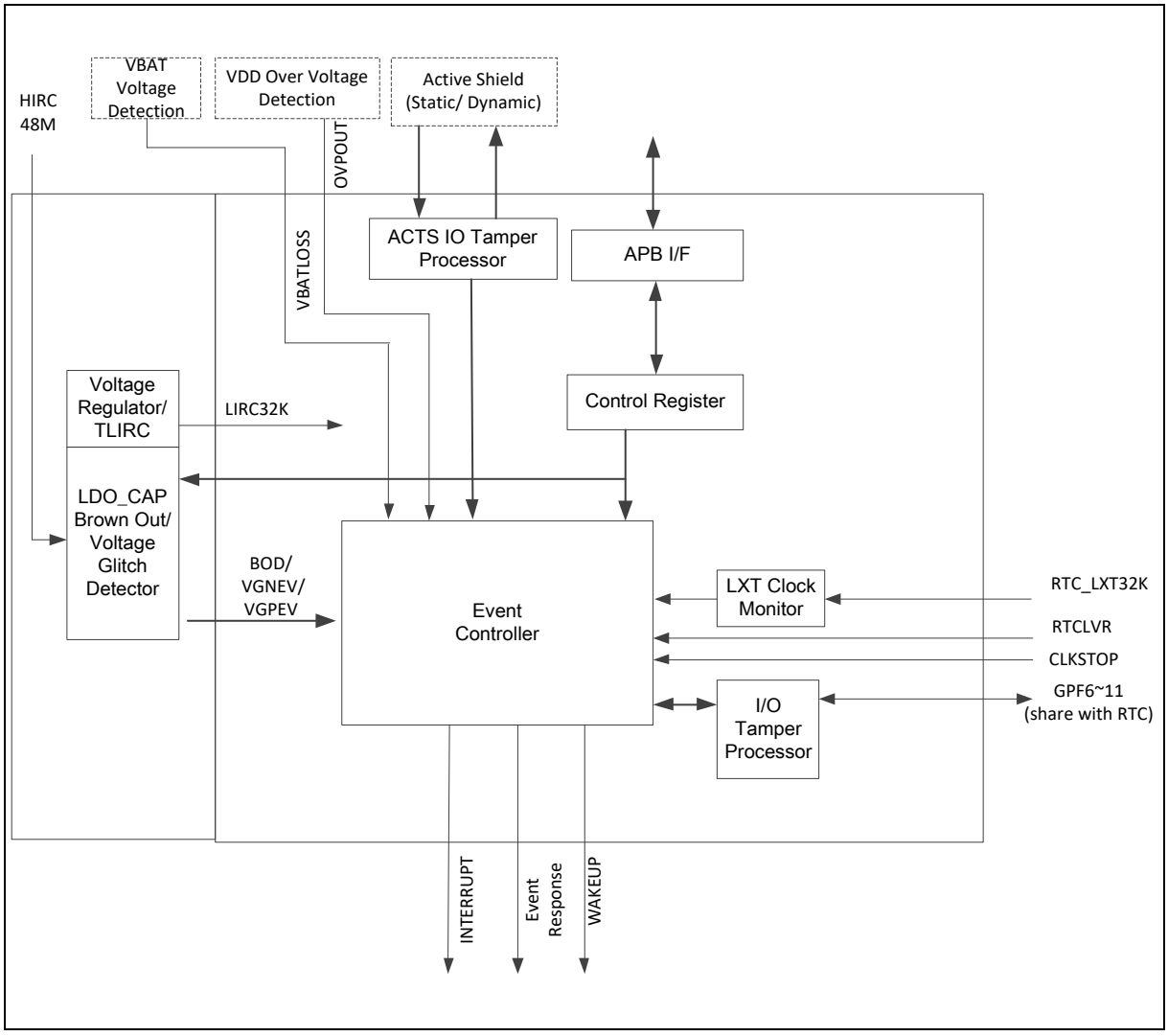


Figure 6.41-1 Tamper Block Diagram (Normal Mode)

6.41.4 Basic Configuration

- Clock Source Configuration
  - The Tamper controller clock source is enabled by TAMPERCKEN (APBCLK0[22]).
- Pin Configuration
  - The Tamper pin is reused with RTC's tamper pins.

6.41.5 Functional Description

The Tamper is powered by voltage regulator, which is enabled by setting CONFIG3[31:16] as no 0x5AA5.

The Tamper controller includes a LXT external clock range detector whose detect range can be configured by user.

To prevent voltage attack, a low voltage glitch detector is powered in LDO_CAP. This detector can detect more than 10ns glitch under typical conditions and an over voltage detector is powered in V_{DD}

The Tamper also supports active shielding function, including power/GND and dynamic pattern detection.

When an attack event is detected, the following actions will be performed if the related enable bits are set (refer to TAMPER_TRIEN for details).

- Key Store's SRAM and Flash content will be cleared, and the OTP in Key Store will be revoked
- The spare register in RTC will be cleared.
- Crypto reset.
- Chip reset.
- System wake-up.

6.41.5.1 Trust RTC Stable Indicator

To detect the voltage loss of RTC V_{BAT}, the Tamper uses VBATLOSS to judge the power loss of RTC and uses RTCLVR to indicate there is no trusted voltage.

6.41.5.2 Watchdog Condition

To avoid the system crash under special circumstances, but the Tamper attack detection is not truly reflected, the WDT occurrence condition of the system is also put into the attack condition of the Tamper to ensure the system is abnormal. When the attack event of the Tamper did not occur, the WDT can still be used to prevent the risk from being attacked and stolen.

6.41.5.3 External Clock Monitor

The external clock monitor indicates when the external clock source (LXT) is stopped or clock frequency accuracy is out of acceptable range.

Low speed crystal oscillator (LXT) clock detector

The Tamper LXT clock source filters out glitch by low pass filter (using STOPBD(TAMPER_CDBR[7:0]) and FAILBD(TAMPER_CDBR[23:16])), and if the clock is failed or stopped, it will respond the attack event to the Event Controller.

6.41.5.4 Tamper Detection for Pins

A tamper I/O event will be detected if the input state of a tamper pin or received data of a tamper loop is different from original setting.

There are 6 individual tamper pins (GPF6~11), which share with the RTC tamper function to detect input state, or can be configured into 3 tamper loops to detect a special data pattern.

When the circuit of this module is enabled (any bit of bits 28, 24, 20, 16, 12, 8 in TAMPER_TIOCTL), the relative RTC_TAMPCTL of the RTC module must be disabled. Similarly, when the RTC tamper I/O is started, the function of the Tamper controller should not be enabled.

6.41.5.5 Voltage Tamper Sensor

To prevent side-channel attack from external power pin, the chip provides voltage glitch detector on both high and low voltage domains.

The high voltage detector is used to detect the V_{DD} voltage glitch and the low voltage glitch detector is used to detect LDO_CAP voltage glitch.

High Voltage Detector

When it is detected that the voltage source is being attacked, the VBATLOSS (notifying V_{BAT} of the possibility of being attacked including falling or exceeding V_{BAT} band-gap) and OVPOUT (notifying system V_{DD} of the possibility of being attacked to exceed 4.0V) signals are sent to the Tamper. The VBATLOSSIF (TAMPER_INTSTS[20]) and OVPOUTIF (TAMPER_INTSTS[8]) indicate the voltage

information when it had been attacked.

To detect whether the V_{DD} has been attacked, the voltage is pulled up to exceed 4.0V. User must set the bit OVDEN (SYS_OVDCTL[0]) to 1 and wait for OVDSTB (SYS_OVDCTL[31]) to be set to 1 to start the detection of V_{DD} voltage glitch.

For the voltage power loss detection of V_{BAT}, the voltage drops to a certain level (less than 1.4V), the VBATLOSS signal will be output and sent to the Tamper, and VBATLOSSIF (TAMPER_INTSTS[20) will immediately reflect the changed state of its voltage. In order to detect the attack of V_{BAT}, both of the BATLDEN (SYS_BATLDCTL[0]) and BATDETEN (RTC_TEST[4]) must be enabled.

Note: BATDETEN (RTC_TEST[4]) is reserved in RTC section. If user wants to detect the attacked event of V_{BAT}, it should be released.

Low Voltage Glitch Detector

The low voltage glitch detector is started after the voltage regulator is enabled and the HIRC48M must be enabled.

The counting action of the Low Voltage Glitch Detector alarm event

Since low voltage glitch detection is directly related to the input clock frequency, the alarm event is also counted by the input clock. After that, compare the value of this count with the 8-bit tolerance value set in the TAMPER_VGEV register. When the alarm event of low voltage glitch detector is greater than the set tolerance value, a real attack message will be sent. Then, the event controller decides whether the message is to be used.

There are four groups of circuit that can handle voltage glitch detection at the same time, and they can be started individually. These four groups of circuit increase the signal detection capability by judging the positive edge clock, the negative edge clock, the positive edge clock delay, and the negative edge clock delay. For example, using 48 MHz as input clock when the negative edge clock is activated (TAMPER_FUNEN[25] = 1), the voltage glitch detection ability can be increased to 96 MHz. When the positive edge clock delay and the negative edge clock delay are turned on (TAMPER_FUNEN[27:26] = 2'b11) simultaneously, the voltage glitch detection capability can be improved to reach 192 MHz. Before the voltage glitch detection function is enabled, the user should confirm if the clock source of 48 MHz is stable.

In the low voltage glitch detector, there are 2 sets of clock delay chains, and there are 4 select bits PCLKSEL/PDATSEL or NCLKSEL/NDATSEL for each delay chain to get one delay cell's output. The cell delay will vary when its supplied voltage is changed. Therefore, the voltage glitch detector utilizes this phenomenon to examine whether monitored supplied voltage is over/under certain voltage threshold, which can be adjusted by related clock/toggled data selecting bits TAMPER_VG and TAMPER_VG2.

There are four different power levels in system that are defined in Power Level Control Register SYS_PLCTL. Each power level has each select value for PCLKSEL/PDATSEL or NCLKSEL/NDATSEL. The TAMPER_VG and TAMPER_VG2 are used to define the setting value for each power level. When the power level of the system is switched by user, the function of voltage glitch detector must be turned off in advance and it can be turned on after the power level switch busy flag PLCBUSY (SYS_PLSTS[0]) is inactive.

The VGECNTN (TAMPER_VGEV[15:8]) and VGECNTP (TAMPER_VGEV[7:0]) are used to indicate the tolerance count for negative and positive voltage glitch event respectively.

The clock source is HIRC48M. It should be turned on first before the voltage glitch detection function is enabled and it is always turned on when the HIRC48MEN is set.

After four voltage glitch detection circuits are turned on, users must confirm whether the VGNEVIF and VGPEVIF (TAMPER_EVSTS[10:9]) have been set or not. If any of them is set to 1, it should be cleared first and VGNIEN or VGPIEN (TAMPER_INTEN[10:9]) should be set to 1 to clear the internal counter of negative voltage glitch counter or positive voltage glitch counter to 0.

The following table is a reference trim value that provides voltage glitch detection error tolerance within 15%. It contains the reference values for different power levels. Each chip may have different detection sensitivity due to the manufacturing process, and the final result is still based on the actual setting of the product. The power level information is described in the section of “Power Mode and Wake-up Source” (sub-section of System Manager).

Power Level(PL)	PCLKSEL[3:0]	PDATSEL[3:0]	NCLKSEL[3:0]	NDATSEL[3:0]
PL0 (1.26V)	0x8	0xC	0xC	0x8
PL1 (1.2V)	0x9	0xC	0xC	0x7
PL2 (1.1V)	0xB	0xC	0xC	0x5
PL3 (0.9V)	0xD	0xC	0xC	0x2

Table 6.41-1 Voltage Glitch Reference Value

Note: If the detected voltage level is adjusted to other voltage values (such as 1.1V or 1.0V), then the precision of each trim value is not applicable to the preset state under 1.2V conditions. It is necessary to re-adjust and set various trim worthy specifications under this voltage condition.

6.41.5.6 Active Shield Tamper Sensor

The chip provides a chip level probe resistance technology for SRAM storage. The main purpose of this technology is to prevent the system chip from being attacked in this area, which can be detected correctly because the original ground or the signal originally connected to power plane is destroyed.

There are four pairs of tamper I/O, such as pin detection (the operation is the same as RTC tamper I/O function). The first two pairs use the seed value that is the same as the Tamper I/O seed pattern (TAMPER_SEED). The second two pairs dynamic tamper I/O use the 2nd seed value pattern (TAMPER_SEED2). It will be reloaded when the bit of SEEDRLD2 (TAMPER_ACTSTIOCTL2[4]) is written to 1. All the control register of the 2nd dynamic tamper I/O pairs is located in TAMPER_ACTSTIOCTL2.

6.41.5.7 Power Operation Condition

Power Mode	Clock For Tamper	Attack Event In Tamper									Voltage Regulator
		RTC			Tamper I/O	Active Shield	OVPOUT	BATLOSS	Low Voltage Glitch	BOD	
		LVR	CLK Monitor	Tamper I/O							
Normal Run	HIRC48M, TLIRC32K	V	V	V	V	V	V	V	V	V	Enable
PD	HIRC48M, TLIRC32K	V			V	V	V	V	V	V	Enable
SPD	HIRC48M, TLIRC32K	V			V	V	V	V	V	V	Enable
DPD	No	X			X	X	X	X	X	X	Disable

Table 6.41-2 shows the behavior of each clock source in different power modes.

Power Mode	Clock For Tamper	Attack Event In Tamper									Voltage Regulator
		RTC			Tamper I/O	Active Shield	OVPOUT	BATLOSS	Low Voltage Glitch	BOD	
		LVR	CLK Monitor	Tamper I/O							
Normal Run	HIRC48M, TLIRC32K	V	V	V	V	V	V	V	V	V	Enable
PD	HIRC48M, TLIRC32K	V			V	V	V	V	V	V	Enable
SPD	HIRC48M, TLIRC32K	V			V	V	V	V	V	V	Enable
DPD	No	X			X	X	X	X	X	X	Disable

Table 6.41-2 Tamper Power Condition in Each Mode

Note: HIRC48M is used for the low voltage glitch detector only.

6.41.5.8 Brown-out Detection

The BOD is used to detect the output status of LDO_CAP voltage. TLVDSEL is used to detect low voltage reflection, and TOVDSEL is used to detect over voltage reflection. Their detection level is shown in Table 6.41-3 and Table 6.41-4.

TLVDSEL[2:0]	Detect Level
000	1.25V
001	1.2V
010	1.15V
011	1.1V
100	1.05V
101(default)	1V
110	0.95V
111	0.9V

Table 6.41-3 TLVD Detect Level

TOVDSEL	Detect Level
0(default)	1.35V
1	1.4V

Table 6.41-4 TOVD Detect Level

6.41.5.9 Chip Reset

When the response of chip reset is enabled (TAMPER_TRIEN[4] is set to 1), the content of control registers will be cleared and the function of Tamper controller will be turned off. The user has to turn it on and reset the control register again.

6.41.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
Tamper Base Address: **TAMPER_BA = 0x400B_D000**				
TAMPER_INIT	TAMPER_BA+0x00	R/W	Tamper Function Initiation Register	0x0000_0000
TAMPER_FUNEN	TAMPER_BA+0x04	R/W	Tamper Block Function Enable Register	0x005A_0000
TAMPER_TRIEN	TAMPER_BA+0x08	R/W	Tamper Trigger Enable Register	0x0000_0000
TAMPER_INTEN	TAMPER_BA+0x0C	R/W	Tamper Event Interrupt Enable Register	0x0000_0000
TAMPER_INTSTS	TAMPER_BA+0x10	R/W	Tamper Interrupt Status Register	0x0000_0000
TAMPER_TIOCTL	TAMPER_BA+0x18	R/W	Tamper I/O Function Control Register	0x0000_0000
TAMPER_SEED	TAMPER_BA+0x1C	R/W	Tamper Seed Value Control Register	0x0000_0000
TAMPER_SEED2	TAMPER_BA+0x20	R/W	Tamper 2nd Seed Value Control Register	0x0000_0000
TAMPER_ACTSTIOCTL1	TAMPER_BA+0x24	R/W	Tamper Active Shield Tamper I/O Function Control Register 1	0x8080_8000
TAMPER_ACTSTIOCTL2	TAMPER_BA+0x28	R/W	Tamper Active Shield Tamper I/O Function Control Register 2	0x8080_8000
TAMPER_CDBR	TAMPER_BA+0x2C	R/W	Tamper Clock Frequency Detector Boundary Register	0x00F0_000F
TAMPER_VG	TAMPER_BA+0x30	R/W	Tamper Voltage Glitch Control Register	0x5876_5876
TAMPER_VGEV	TAMPER_BA+0x34	R/W	Tamper Voltage Glitch Event Tolerance Control Register	0x0000_0303
TAMPER_LDOTRIM	TAMPER_BA+0x38	R/W	Tamper LDO Trim Value Control Register	0x0000_0108
TAMPER_LBSTRIM	TAMPER_BA+0x3C	R/W	Tamper LDO BIAS Trim Value Control Register	0x0000_5505
TAMPER_VG2	TAMPER_BA+0x40	R/W	Tamper Voltage Glitch Control Register 2	0x5876_5876

6.41.7 Register Description

Tamper Function Initiation Register (TAMPER_INIT)

Register	Offset	R/W	Description	Reset Value
TAMPER_INIT	TAMPER_BA+0x00	R/W	Tamper Function Initiation Register	0x0000_0000

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

Bits	Description	
[31]	**TLDORDY**	**Voltage Regulator Power Ready (Read Only)** 0 = The power status of voltage regulator is not ready. 1 = The power status of voltage regulator is ready.
[30:1]	**Reserved**	Reserved.
[0]	**TCORERST**	**Tamper Core Reset** 0 = Write 0x5500; the Tamper core block reset will be released. 1 = Write 0x55AA; the Tamper core block will be reset.

Tamper Block Function Enable Register (TAMPER_FUNEN)

Register	Offset	R/W	Description	Reset Value
TAMPER_FUNEN	TAMPER_BA+0x04	R/W	Tamper Block Function Enable Register	0x005A_0000

31	30	29	28	27	26	25	24
Reserved				VGCHEN3	VGCHEN2	VGCHEN1	VGCHEN0
23	22	21	20	19	18	17	16
HIRC48MEN							
15	14	13	12	11	10	9	8
Reserved		TMPIOSEL					
7	6	5	4	3	2	1	0
Reserved							LXTDETEN

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27]	**VGCHEN3**	**Voltage Glitch Channel 3 Enable Bit** 0 = Voltage glitch channel 3 Disabled. 1 = Voltage glitch channel 3 Enabled. **Note:** To avoid the voltage glitch when the voltage channel is enabled, it is better to detect the voltage glitch about 150us after the channel is enabled.
[26]	**VGCHEN2**	**Voltage Glitch Channel 2 Enable Bit** 0 = Voltage glitch channel 2 Disabled. 1 = Voltage glitch channel 2 Enabled. **Note:** To avoid the voltage glitch when the voltage channel is enabled, it is better to detect the voltage glitch about 150us after the channel is enabled.
[25]	**VGCHEN1**	**Voltage Glitch Channel 1 Enable Bit** 0 = Voltage glitch channel 1 Disabled. 1 = Voltage glitch channel 1 Enabled. **Note:** To avoid the voltage glitch when the voltage channel is enabled, it is better to detect the voltage glitch about 150us after the channel is enabled.
[24]	**VGCHEN0**	**Voltage Glitch Channel 0 Enable Bit** 0 = Voltage glitch channel 0 Disabled. 1 = Voltage glitch channel 0 Enabled. **Note:** To avoid the voltage glitch when the voltage channel is enabled, it is better to detect the voltage glitch about 150us after the channel is enabled.
[23:16]	**HIRC48MEN**	**HIRC48M Enable Bit** The HIRC48M is disabled when these bits equal 0x5A, otherwise it will be enabled with any other values.
[15:14]	**Reserved**	Reserved.
[13:8]	**TMPIOSEL**	**Tamper I/O Detection Selection Bit** 0 = Write 0x90/0xA0/0xB0/0xC0/0xD0/0xE0 for tamper I/O 0~5; the I/O tamper function is detected

		through RTC block. 1 = Write 0x94/0xA4/0xB4/0xC4/0xD4/0xE4 for tamper I/O 0~5; the I/O tamper function is detected through Tamper block.
[7:1]	**Reserved**	Reserved.
[0]	**LXTDETEN**	**LXT Clock Detection Enable Bit** 0 = Write pattern 0x40 into this register; the LXT clock detection Disabled. 1 = Write pattern 0x44 into this register; the LXT clock detection Enabled.

Tamper Trigger Enable Register (TAMPER_TRIEN)

Register	Offset	R/W	Description	Reset Value
TAMPER_TRIEN	TAMPER_BA+0x08	R/W	Tamper Trigger Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		RTCSPCLREN	CHIPRSTEN	CRYPTOEN	WAKEUPEN	KSTRIGEN	Reserved

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**RTCSPCLREN**	**RTC Spare Register Clear Enable Bit** 0 = Tamper event trigger RTC spare register reset Disabled. 1 = Tamper event trigger RTC spare register reset Enabled.
[4]	**CHIPRSTEN**	**Chip Reset Enable Bit** 0 = Tamper event trigger chip reset Disabled. 1 = Tamper event trigger chip reset Enabled.
[3]	**CRYPTOEN**	**Crypto Enable Bit** 0 = Tamper event clear Crypto Disabled. 1 = Tamper event clear Crypto Enabled.
[2]	**WAKEUPEN**	**Wakeup Enable Bit** 0 = Tamper wakeup event Disabled. 1 = Tamper wakeup event Enabled.
[1]	**KSTRIGEN**	**Key Store Trigger Enable Bit** 0 = Tamper event is detected and to trigger Key Store Disabled. 1 = Tamper event is detected and to trigger Key Store Enabled.
[0]	**Reserved**	Reserved.

Tamper Interrupt Event Enable Register (TAMPER_INTEN)

Register	Offset	R/W	Description	Reset Value
TAMPER_INTEN	TAMPER_BA+0x0C	R/W	Tamper Event Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved	BODIEN	WDTIEN	VLOSSIEN	Reserved	RTCLKIEN	RTCIOIEN	RTCLVRIEN
15	14	13	12	11	10	9	8
Reserved				ACTSIEN	VGNIEN	VGPIEN	OVPIEN
7	6	5	4	3	2	1	0
CLKSTOPIEN	CLKFIEN	TAMP5IEN	TAMP4IEN	TAMP3IEN	TAMP2IEN	TAMP1IEN	TAMP0IEN

Bits	Description	
[31:23]	**Reserved**	Reserved.
[22]	**BODIEN**	**BOD Event Interrupt Enable Bit** 0 = Brown-out event interrupt Disabled. 1 = Brown-out event interrupt Enabled.
[21]	**WDTIEN**	**Watchdog Event Interrupt Enable Bit** 0 = Watchdog event interrupt Disabled. 1 = Watchdog event interrupt Enabled. **Note:** If there is WDT event, it can be used to enable to tamper trigger function when the enable bit is set to 1.
[20]	**VLOSSIEN**	**V_{BAT} Power Loss Event Interrupt Enable Bit** 0 = V_{BAT} power loss event interrupt Disabled. 1 = V_{BAT} power loss event interrupt Enabled.
[23]	**Reserved**	Reserved.
[18]	**RTCLKIEN**	**RTC Clock Monitor Detection Event Interrupt Enable Bit** 0 = RTC clock monitor event interrupt Disabled. 1 = RTC clock monitor event interrupt Enabled.
[17]	**RTCIOIEN**	**RTC Tamper I/O Event Interrupt Enable Bit** 0 = RTC tamper I/O detection event interrupt Disabled. 1 = RTC tamper I/O detection event interrupt Enabled.
[16]	**RTCLVRIEN**	**RTC Low Voltage Detection Event Interrupt Enable Bit** 0 = V_{BAT} low voltage detection event interrupt Disabled. 1 = V_{BAT} low voltage detection event interrupt Enabled.
[15:12]	**Reserved**	Reserved.
[11]	**ACTSIEN**	**Active Shield Event Interrupt Enable Bit**

		0 = Active shield event interrupt Disabled. 1 = Active shield event interrupt Enabled.
[10]	**VGNIEN**	**Voltage Glitch Negative Detection Event Interrupt Enable Bit** 0 = LDO_CAP negative glitch event interrupt Disabled. 1 = LDO_CAP negative glitch event interrupt Enabled.
[9]	**VGPIEN**	**Voltage Glitch Positive Detection Event Interrupt Enable Bit** 0 = LDO_CAP positive glitch event interrupt Disabled. 1 = LDO_CAP positive glitch event interrupt Enabled.
[8]	**OVPIEN**	**V_{DD} Over Voltage Protect Detection Interrupt Enable Bit** 0 = Detect V_{DD} over voltage protect detection interrupt Disabled. 1 = Detect V_{DD} over voltage protect detection interrupt Enabled. **Note:** The function enable of the over voltage detection is defined in system manager.
[7]	**CLKSTOPIEN**	**LXT Clock Frequency Monitor Stop Event Interrupt Enable Bit** 0 = LXT frequency stop event interrupt Disabled. 1 = LXT frequency stop event interrupt Enabled.
[6]	**CLKFIEN**	**LXT Clock Frequency Monitor Fail Event Interrupt Enable Bit** 0 = LXT frequency fail event interrupt Disabled. 1 = LXT frequency fail event interrupt Enabled.
[5]	**TAMP5IEN**	**Tamper 5 or Pair 2 Event Interrupt Enable Bit** 0 = Tamper 5 or Pair 2 event interrupt Disabled. 1 = Tamper 5 or Pair 2 event interrupt Enabled.
[4]	**TAMP4IEN**	**Tamper 4 Event Interrupt Enable Bit** 0 = Tamper 4 event interrupt Disabled. 1 = Tamper 4 event interrupt Enabled.
[3]	**TAMP3IEN**	**Tamper 3 or Pair 1 Event Interrupt Enable Bit** 0 = Tamper 3 or Pair 1 event interrupt Disabled. 1 = Tamper 3 or Pair 1 event interrupt Enabled.
[2]	**TAMP2IEN**	**Tamper 2 Event Interrupt Enable Bit** 0 = Tamper 2 event interrupt Disabled. 1 = Tamper 2 event interrupt Enabled.
[1]	**TAMP1IEN**	**Tamper 1 or Pair 0 Event Interrupt Enable Bit** 0 = Tamper 1 or Pair 0 event interrupt Disabled. 1 = Tamper 1 or Pair 0 event interrupt Enabled.
[0]	**TAMP0IEN**	**Tamper 0 Event Interrupt Enable Bit** 0 = Tamper 0 event interrupt Disabled. 1 = Tamper 0 event interrupt Enabled.

Tamper Interrupt Status Register (TAMPER_INTSTS)

Register	Offset	R/W	Description	Reset Value
TAMPER_INTSTS	TAMPER_BA+0x10	R/W	Tamper Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
ACTST23IF	Reserved	ACTST21IF	Reserved	ACTST3IF	Reserved	ACTST1IF	Reserved
23	22	21	20	19	18	17	16
Reserved	BODIF	SECWDTIF	VBATLOSSIF	Reserved	RCLKTRIGIF	RIOTRIGIF	RTCLVRIF
15	14	13	12	11	10	9	8
ACTST25IF	Reserved	ACTST5IF	Reserved	ACTSEIF	VGNEVIF	VGPEVIF	OVPOUTIF
7	6	5	4	3	2	1	0
CLKSTOPIF	CLKFAILIF	TAMP5IF	TAMP4IF	TAMP3IF	TAMP2IF	TAMP1IF	TAMP0IF

Bits	Description	
[31]	**ACTST23IF**	**2th Active Shield Tamper 3 Event Interrupt Flag** 0 = No Active shield Tamper 3 event interrupt flag is generated. 1 = 2th Active shield Tamper 3 event interrupt flag is generated. **Note:** Write 1 to clear this bit.
[30]	**Reserved**	Reserved.
[29]	**ACTST21IF**	**2th Active Shield Tamper 1 Event Interrupt Flag** 0 = No Active shield Tamper 1 event interrupt flag is generated. 1 = 2th Active shield Tamper 1 event interrupt flag is generated. **Note:** Write 1 to clear this bit.
[28]	**Reserved**	Reserved.
[27]	**ACTST3IF**	**Active Shield Tamper 3 Event Interrupt Flag** 0 = No Active shield Tamper 3 event interrupt flag is generated. 1 = Active shield Tamper 3 event interrupt flag is generated. **Note:** Write 1 to clear this bit.
[26]	**Reserved**	Reserved.
[25]	**ACTST1IF**	**Active Shield Tamper 1 Event Interrupt Flag** 0 = No Active shield Tamper 1 event interrupt flag is generated. 1 = Active shield Tamper 1 event interrupt flag is generated. **Note:** Write 1 to clear this bit.
[24:23]	**Reserved**	Reserved.
[22]	**BODIF**	**BOD Event Interrupt Flag** 0 = Brown-out event interrupt flag is no detected. 1 = Brown-out interrupt flag is detected. **Note:** It is used to detect the LDO_CAP. Write 1 to clear this bit.

[21]	**SECWDTIF**	**Security Watchdog Event Interrupt Flag** 0 = No security WDT event interrupt flag is detected. 1 = Security WDT event interrupt flag is detected. **Note:** Write 1 to clear this bit.
[20]	**VBATLOSSIF**	**V_{BAT} LOSS Detection Event Interrupt Flag** 0 = V_{BAT} Power loss detection event interrupt flag is not detected. 1 = V_{BAT} Power loss detection event interrupt flag is detected. **Note:** Write 1 to clear this bit.
[19]	**Reserved**	Reserved.
[18]	**RCLKTRIGIF**	**RTC Clock Monitor Detection Event Interrupt Flag** 0 = There is no RTC clock monitor detection event interrupt flag. 1 = There is RTC clock monitor detection event interrupt flag.
[17]	**RIOTRIGIF**	**RTC Tamper I/O Event Interrupt Flag** 0 = There is no RTC tamper I/O detection event interrupt flag. 1 = There is RTC tamper I/O detection event interrupt flag.
[16]	**RTCLVRIF**	**RTC Low Voltage Detection Event Interrupt Flag** 0 = V_{BAT} low voltage detection event interrupt flag is not detected. 1 = V_{BAT} low voltage detection event interrupt flag is detected.
[15]	**ACTST25IF**	**Active Shield Tamper 5 Event Interrupt Flag** 0 = No Active shield Tamper 5 event interrupt flag is generated. 1 = 2th Active shield Tamper 5 event interrupt flag is generated. **Note:** Write 1 to clear this bit.
[14]	**Reserved**	Reserved.
[13]	**ACTST5IF**	**Active Shield Tamper 5 Event Interrupt Flag** 0 = No Active shield Tamper 5 event interrupt flag is generated. 1 = Active shield Tamper 5 event interrupt flag is generated. **Note:** Write 1 to clear this bit.
[12]	**Reserved**	Reserved.
[11]	**ACTSEIF**	**Active Shield Event Detection Interrupt Flag** 0 = Active shield event interrupt flag is not detected. 1 = Active shield event interrupt flag is detected including the voltage of voltage regulator and GND attack. **Note:** Write 1 to clear this bit after all of ACTSTxIF bits have been cleaned.
[10]	**VGNEVIF**	**Voltage Glitch Negative Detection Interrupt Flag** 0 = LDO_CAP negative glitch is not detected. 1 = LDO_CAP negative glitch is detected. **Note:** It can be written 1 to clear only (No clear by **TCORERST**)
[9]	**VGPEVIF**	**Voltage Glitch Positive Detection Interrupt Flag** 0 = LDO_CAP positive glitch is not detected. 1 = LDO_CAP positive glitch is detected. **Note:** It can be written 1 to clear only (No clear by **TCORERST**)
[8]	**OVPOUTIF**	**V_{DD} Over Voltage Event Interrupt Flag** 0 = V_{DD} no over voltage is detected.

		1 = V_{DD} over voltage is detected. **Note:** Write 1 to clear this bit.
[7]	**CLKSTOPIF**	**LXT Clock Frequency Monitor Stop Event Interrupt Flag** 0 = LXT frequency is normal. 1 = LXT frequency is almost stopped. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** LXT detector will be automatically disabled when Fail/Stop flag rises, and resumes after Fail/Stop flag is cleared.
[6]	**CLKFAILIF**	**LXT Clock Frequency Monitor Fail Event Interrupt Flag** 0 = LXT frequency is normal. 1 = LXT frequency is abnormal. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** LXT detector will be automatically disabled when Fail/Stop flag rises, and resumes after Fail/Stop flag is cleared.
[5]	**TAMP5IF**	**Tamper 5 Event Interrupt Flag** 0 = No Tamper 5 event interrupt flag is generated. 1 = Tamper 5 event interrupt flag is generated. **Note:** Write 1 to clear this bit.
[4]	**TAMP4IF**	**Tamper 4 Event Interrupt Flag** 0 = No Tamper 4 event interrupt flag is generated. 1 = Tamper 4 event interrupt flag is generated. **Note:** Write 1 to clear this bit.
[3]	**TAMP3IF**	**Tamper 3 Event Interrupt Flag** 0 = No Tamper 3 event interrupt flag is generated. 1 = Tamper 3 event interrupt flag is generated. **Note:** Write 1 to clear this bit.
[2]	**TAMP2IF**	**Tamper 2 Event Interrupt Flag** 0 = No Tamper 2 event interrupt flag is generated. 1 = Tamper 2 event interrupt flag is generated. **Note:** Write 1 to clear this bit.
[1]	**TAMP1IF**	**Tamper 1 Event Interrupt Flag** 0 = No Tamper 1 event interrupt flag is generated. 1 = Tamper 1 event interrupt flag is generated. **Note:** Write 1 to clear this bit.
[0]	**TAMP0IF**	**Tamper 0 Event Interrupt Flag** 0 = No Tamper 0 event interrupt flag is generated. 1 = Tamper 0 event interrupt flag is generated. **Note:** Write 1 to clear this bit.

Tamper I/O Function Control Register (TAMPER_TIOCTL)

Register	Offset	R/W	Description	Reset Value
TAMPER_TIOCTL	TAMPER_BA+0x18	R/W	Tamper I/O Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
DYNPR2EN	TAMP5DBEN	TAMP5LV	TAMP5EN	Reserved	TAMP4DBEN	TAMP4LV	TAMP4EN
23	22	21	20	19	18	17	16
DYNPR1EN	TAMP3DBEN	TAMP3LV	TAMP3EN	Reserved	TAMP2DBEN	TAMP2LV	TAMP2EN
15	14	13	12	11	10	9	8
DYNPR0EN	TAMP1DBEN	TAMP1LV	TAMP1EN	Reserved	TAMP0DBEN	TAMP0LV	TAMP0EN
7	6	5	4	3	2	1	0
DYNRATE			SEEDRLD	DYNSRC	Reserved	DYN2ISS	DYN1ISS

Bits	Description	
[31]	**DYNPR2EN**	**Dynamic Pair 2 Enable Bit** 0 = Static detect. 1 = Dynamic detect.
[30]	**TAMP5DBEN**	**Tamper 5 De-bounce Enable Bit** 0 = Tamper 5 de-bounce Disabled. 1 = Tamper 5 de-bounce Enabled. Tamper detection pin will sync 1 RTC clock.
[29]	**TAMP5LV**	**Tamper 5 Level** This bit depends on level attribute of tamper pin for static tamper detection. 0 = Detect voltage level is low. 1 = Detect voltage level is high.
[28]	**TAMP5EN**	**Tamper 5 Detect Enable Bit** 0 = Tamper 5 detect Disabled. 1 = Tamper 5 detect Enabled. **Note:** The reference is RTC-clock. Tamper detector needs sync 2 ~ 3 RTC-clock.
[27]	**Reserved**	Reserved.
[26]	**TAMP4DBEN**	**Tamper 4 De-bounce Enable Bit** 0 = Tamper 4 de-bounce Disabled. 1 = Tamper 4 de-bounce Enabled. Tamper detection pin will sync 1 RTC clock.
[25]	**TAMP4LV**	**Tamper 4 Level** This bit depends on level attribute of tamper pin for static tamper detection. 0 = Detect voltage level is low. 1 = Detect voltage level is high.
[24]	**TAMP4EN**	**Tamper4 Detect Enable Bit** 0 = Tamper 4 detect Disabled. 1 = Tamper 4 detect Enabled.

		Note: The reference is RTC-clock. Tamper detector needs sync 2 ~ 3 RTC-clock.
[23]	**DYNPR1EN**	**Dynamic Pair 1 Enable Bit** 0 = Static detect. 1 = Dynamic detect.
[22]	**TAMP3DBEN**	**Tamper 3 De-bounce Enable Bit** 0 = Tamper 3 de-bounce Disabled. 1 = Tamper 3 de-bounce Enabled. Tamper detection pin will sync 1 RTC clock.
[21]	**TAMP3LV**	**Tamper 3 Level** This bit depends on level attribute of tamper pin for static tamper detection. 0 = Detect voltage level is low. 1 = Detect voltage level is high.
[20]	**TAMP3EN**	**Tamper 3 Detect Enable Bit** 0 = Tamper 3 detect Disabled. 1 = Tamper 3 detect Enabled. **Note:** The reference is RTC-clock. Tamper detector needs sync 2 ~ 3 RTC-clock.
[19]	**Reserved**	Reserved.
[18]	**TAMP2DBEN**	**Tamper 2 De-bounce Enable Bit** 0 = Tamper 2 de-bounce Disabled. 1 = Tamper 2 de-bounce Enabled. Tamper detection pin will sync 1 RTC clock.
[17]	**TAMP2LV**	**Tamper 2 Level** This bit depends on level attribute of tamper pin for static tamper detection. 0 = Detect voltage level is low. 1 = Detect voltage level is high.
[16]	**TAMP2EN**	**Tamper 2 Detect Enable Bit** 0 = Tamper 2 detect Disabled. 1 = Tamper 2 detect Enabled. **Note:** The reference is RTC-clock. Tamper detector need sync 2 ~ 3 RTC-clock.
[15]	**DYNPR0EN**	**Dynamic Pair 0 Enable Bit** 0 = Static detect. 1 = Dynamic detect.
[14]	**TAMP1DBEN**	**Tamper 1 De-bounce Enable Bit** 0 = Tamper 1 de-bounce Disabled. 1 = Tamper 1 de-bounce Enabled, tamper detection pin will sync 1 RTC clock.
[13]	**TAMP1LV**	**Tamper 1 Level** This bit depends on level attribute of tamper pin for static tamper detection. 0 = Detect voltage level is low. 1 = Detect voltage level is high.
[12]	**TAMP1EN**	**Tamper 1 Detect Enable Bit** 0 = Tamper 1 detect Disabled. 1 = Tamper 1 detect Enabled. **Note:** The reference is RTC-clock. Tamper detector needs sync 2 ~ 3 RTC-clock.
[11]	**Reserved**	Reserved.

[10]	**TAMP0DBEN**	**Tamper 0 De-bounce Enable Bit** 0 = Tamper 0 de-bounce Disabled. 1 = Tamper 0 de-bounce Enabled. Tamper detection pin will sync 1 RTC clock.
[9]	**TAMP0LV**	**Tamper 0 Level** This bit depends on level attribute of tamper pin for static tamper detection. 0 = Detect voltage level is low. 1 = Detect voltage level is high.
[8]	**TAMP0EN**	**Tamper0 Detect Enable Bit** 0 = Tamper 0 detect Disabled. 1 = Tamper 0 detect Enabled. **Note:** The reference is RTC-clock. Tamper detector needs sync 2 ~ 3 RTC-clock.
[7:5]	**DYNRATE**	**Dynamic Change Rate** This item is choice the dynamic tamper output change rate. 000 = 2^6 * RTC_CLK. 001 = 2^7 * RTC_CLK. 010 = 2^8 * RTC_CLK. 011 = 2^9 * RTC_CLK. 100 = 2^{10} * RTC_CLK. 101 = 2^{11} * RTC_CLK. 110 = 2^{12} * RTC_CLK. 111 = 2^{13} * RTC_CLK. **Note:** After revising this field, setting SEEDRLD (TAMPER_TIOCTL[4]) can reload change rate immediately.
[4]	**SEEDRLD**	**Reload New Seed for PRNG Engine** Setting this bit, the tamper configuration will be reloaded. 0 = Generating key based on the current seed. 1 = Reload new seed. **Note 1:** Before this bit is set, the tamper configuration should be set to complete and this bit will be auto clear to 0 after reload new seed completed. **Note 2:** The reference is RTC-clock. Tamper detector needs sync 2 ~ 3 RTC-clock.
[3]	**DYNSRC**	**Dynamic Reference Pattern** This field determines the new reference pattern when current pattern run out in dynamic pair mode. 0 = The new reference pattern is generated by random number generator when the reference pattern run out. 1 = The new reference pattern is repeated from SEED (TAMPER_SEED[31:0]) when the reference pattern run out. **Note:** After this bit is modified, the SEEDRLD (TAMPER_TIOCTL[4]) should be set.
[2]	**Reserved**	Reserved.
[1]	**DYN2ISS**	**Dynamic Pair 2 Input Source Select** This bit determines Tamper 5 input is from Tamper 4 or Tamper 0 in dynamic mode. 0 = Tamper input is from Tamper 4. 1 = Tamper input is from Tamper 0. **Note:** This bit has effect only when DYNPR2EN (TAMPER_TIOCTL[31]) and DYNPR0EN (TAMPER_TIOCTL[15]) are set.
[0]	**DYN1ISS**	**Dynamic Pair 1 Input Source Select** This bit determines Tamper 3 input is from Tamper 2 or Tamper 0 in dynamic mode.

		0 = Tamper input is from Tamper 2. 1 = Tamper input is from Tamper 0. **Note:** This bit is effective only when DYNPR1EN (TAMPER_TIOCTL[23]) and DYNPR0EN (TAMPER_TIOCTL[15]) are set.

Tamper Seed Value Control Register (TAMPER_SEED)

Register	*Offset*	R/W	Description	Reset Value
TAMPER_SEED	TAMPER_BA+0x1C	R/W	Tamper Seed Value Control Register	0x0000_0000

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

Bits	Description	
[31:0]	**SEED**	Seed value.

Tamper 2nd Seed Value Control Register (TAMPER_SEED2)

Register	*Offset*	R/W	Description	Reset Value
TAMPER_SEED2	TAMPER_BA+0x20	R/W	Tamper 2nd Seed Value Control Register	0x0000_0000

31	30	29	28	27	26	25	24
SEED2							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SEED2							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SEED2							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SEED2							

Bits	Description	
[31:0]	**SEED2**	Seed value**.** These seed value are used for 2nd active shield I/O.

Tamper Active Shield Tamper I/O Function Control Register 1 (TAMPER_ACTSTIOCTL1)

Register	Offset	R/W	Description	Reset Value
TAMPER_ACTSTIOCTL1	TAMPER_BA+0x24	R/W	Tamper Active Shield Tamper I/O Function Control Register 1	0x8080_8000

31	30	29	28	27	26	25	24
ADYNPR2EN	Reserved		ATAMP5EN	Reserved			ATAMP4EN
23	22	21	20	19	18	17	16
ADYNPR1EN	Reserved		ATAMP3EN	Reserved			ATAMP2EN
15	14	13	12	11	10	9	8
ADYNPR0EN	Reserved		ATAMP1EN	Reserved			ATAMP0EN
7	6	5	4	3	2	1	0
ADYNRATE			Reserved	ADYNSRC	Reserved		ADYN1ISS

Bits	Description	
[31]	**ADYNPR2EN**	**Active Shied Dynamic Pair 2 Enable Bit** 0 = Static detect (Not supported). 1 = Dynamic detect.
[30:29]	**Reserved**	Reserved.
[28]	**ATAMP5EN**	**Active Tamper 5 Detect Enable Bit** 0 = Tamper 5 detect Disabled. 1 = Tamper 5 detect Enabled. **Note:** The reference is TLIRC 32K-clock. Tamper detector need sync 2 ~ 3 TLIRC 32K-clock.
[27:25]	**Reserved**	Reserved.
[24]	**ATAMP4EN**	**Active Tamper4 Detect Enable Bit** 0 = Tamper 4 detect Disabled. 1 = Tamper 4 detect Enabled. **Note:** The reference is TLIRC 32K-clock. Tamper detector need sync 2 ~ 3 TLIRC 32K-clock.
[23]	**ADYNPR1EN**	**Active Shied Dynamic Pair 1 Enable Bit** 0 = Static detect (Not supported). 1 = Dynamic detect.
[22:21]	**Reserved**	Reserved.
[20]	**ATAMP3EN**	**Active Shied Tamper 3 Detect Enable Bit** 0 = Tamper 3 detect Disabled. 1 = Tamper 3 detect Enabled. **Note:** The reference is TLIRC 32K-clock. Tamper detector needs sync 2 ~ 3 TLIRC 32K-clock.
[19:17]	**Reserved**	Reserved.
[16]	**ATAMP2EN**	**Active Shied Tamper 2 Detect Enable Bit**

		0 = Tamper 2 detect Disabled. 1 = Tamper 2 detect Enabled. **Note:** The reference is TLIRC 32K-clock. Tamper detector needs sync 2 ~ 3 TLIRC 32K-clock.
[15]	**ADYNPR0EN**	**Active Shied Dynamic Pair 0 Enable Bit** 0 = Static detect (Not supported). 1 = Dynamic detect.
[14:13]	**Reserved**	Reserved.
[12]	**ATAMP1EN**	**Active Shied Tamper 1 Detect Enable Bit** 0 = Tamper 1 detect Disabled. 1 = Tamper 1 detect Enabled. **Note:** The reference is TLIRC 32K-clock. Tamper detector needs sync 2 ~ 3 TLIRC 32K-clock.
[11:9]	**Reserved**	Reserved.
[8]	**ATAMP0EN**	**Active Shied Tamper0 Detect Enable Bit** 0 = Tamper 0 detect Disabled. 1 = Tamper 0 detect Enabled. **Note:** The reference is TLIRC 32K-clock. Tamper detector need sync 2 ~ 3 TLIRC 32K-clock.
[7:5]	**ADYNRATE**	**Active Shied Dynamic Change Rate** Use the bits to choose the dynamic tamper output change rate. 000 = 2^{10} * TLIRC32K. 001 = 2^{11} * TLIRC32K. 010 = 2^{12} * TLIRC32K. 011 = 2^{13} * TLIRC32K. 100 = 2^{14} * TLIRC32K. 101 = 2^{15} * TLIRC32K. 110 = 2^{16} * TLIRC32K. 111 = 2^{17} * TLIRC32K. **Note:** After this field is modified, setting SEEDRLD (TAMPER_TIOCTL[4]) can reload the change rate immediately.
[4]	**Reserved**	Reserved.
[3]	**ADYNSRC**	**Active Shied Dynamic Reference Pattern** This field determines the new reference pattern when current pattern run out in dynamic pair mode. 0 = The new reference pattern is generated by random number generator when the reference pattern run out. 1 = The new reference pattern is repeated from SEED (TAMPER_SEED[31:0]) when the reference pattern run out. **Note:** After this bit is modified, the SEEDRLD (TAMPER_TIOCTL[4]) should be set.
[2:1]	**Reserved**	Reserved.
[0]	**ADYN1ISS**	**Active Shied Dynamic Pair 1 Input Source Select** This bit determines Tamper 3 input is from Tamper 2 or Tamper 0 in dynamic mode. 0 = Tamper input is from Tamper 2. 1 = Tamper input is from Tamper 0. **Note:** This bit is effective only when ADYNPR1EN (TAMPER_ACTSTIOCTL1[23]) and ADYNPR0EN (TAMPER_ACTSTIOCTL1[15]) are set.

Tamper Active Shield Tamper I/O Function Control Register 2 (TAMPER_ACTSTIOCTL2)

Register	Offset	R/W	Description	Reset Value
TAMPER_ACTSTIOCTL2	TAMPER_BA+0x28	R/W	Tamper Active Shield Tamper I/O Function Control Register 2	0x8080_8000

31	30	29	28	27	26	25	24
ADYNPR2EN2	Reserved		ATAMP5EN2	Reserved			ATAMP4EN2
23	22	21	20	19	18	17	16
ADYNPR1EN2	Reserved		ATAMP3EN2	Reserved			ATAMP2EN2
15	14	13	12	11	10	9	8
ADYNPR0EN2	Reserved		ATAMP1EN2	Reserved			ATAMP0EN2
7	6	5	4	3	2	1	0
ADYNRATE2			SEEDRLD2	ADYNSRC2	Reserved		ADYN1ISS2

Bits	Description	
[31]	**ADYNPR2EN2**	**Active Shied Dynamic Pair 2 Enable Bit 2** 0 = Static detect (Not supported). 1 = Dynamic detect.
[30:29]	**Reserved**	Reserved.
[28]	**ATAMP5EN2**	**Active Tamper 5 Detect Enable Bit 2** 0 = Tamper 5 detect Disabled. 1 = Tamper 5 detect Enabled. **Note:** The reference is TLIRC 32K-clock. Tamper detector need sync 2 ~ 3 TLIRC 32K-clock.
[27:25]	**Reserved**	Reserved.
[24]	**ATAMP4EN2**	**Active Shied Tamper4 Detect Enable Bit 2** 0 = Tamper 4 detect Disabled. 1 = Tamper 4 detect Enabled. **Note:** The reference is TLIRC 32K-clock. Tamper detector need sync 2 ~ 3 TLIRC 32K-clock.
[23]	**ADYNPR1EN2**	**Active Shied Dynamic Pair 1 Enable Bit 2** 0 = Static detect (Not supported). 1 = Dynamic detect.
[22:21]	**Reserved**	Reserved.
[20]	**ATAMP3EN2**	**Active Shied Tamper 3 Detect Enable Bit 2** 0 = Tamper 3 detect Disabled. 1 = Tamper 3 detect Enabled. **Note:** The reference is TLIRC 32K-clock. Tamper detector needs sync 2 ~ 3 TLIRC 32K-clock.
[19:17]	**Reserved**	Reserved.
[16]	**ATAMP2EN2**	**Active Shied Tamper 2 Detect Enable Bit 2**

		0 = Tamper 2 detect Disabled. 1 = Tamper 2 detect Enabled. **Note:** The reference is TLIRC 32K-clock. Tamper detector needs sync 2 ~ 3 TLIRC 32K-clock.
[15]	**ADYNPR0EN2**	**Active Shied Dynamic Pair 0 Enable Bit 2** 0 = Static detect (Not supported). 1 = Dynamic detect.
[14:13]	**Reserved**	Reserved.
[12]	**ATAMP1EN2**	**Active Shied Tamper 1 Detect Enable Bit 2** 0 = Tamper 1 detect Disabled. 1 = Tamper 1 detect Enabled. **Note:** The reference is TLIRC 32K-clock. Tamper detector needs sync 2 ~ 3 TLIRC 32K-clock.
[11:9]	**Reserved**	Reserved.
[8]	**ATAMP0EN2**	**Active Shied Tamper0 Detect Enable Bit 2** 0 = Tamper 0 detect Disabled. 1 = Tamper 0 detect Enabled. **Note:** The reference is TLIRC 32K-clock. Tamper detector needs sync 2 ~ 3 TLIRC 32K-clock.
[7:5]	**ADYNRATE2**	**Active Shied Dynamic Change Rate 2** Use the bits to choose the dynamic tamper output change rate. 000 = 2^{10} * TLIRC32K. 001 = 2^{11} * TLIRC32K. 010 = 2^{12} * TLIRC32K. 011 = 2^{13} * TLIRC32K. 100 = 2^{14} * TLIRC32K. 101 = 2^{15} * TLIRC32K. 110 = 2^{16} * TLIRC32K. 111 = 2^{17} * TLIRC32K. **Note:** After this field is modified, setting SEEDRLD2 (TAMPER_ACTSTIOCTL2[4]) can reload change rate immediately.
[4]	**SEEDRLD2**	**Reload New Seed for PRNG Engine 2** Setting this bit, the tamper configuration will be reloaded. 0 = Generating key based on the current seed. 1 = Reload new seed. **Note 1:** Before this bit is set, the tamper configuration should be set to complete and this bit will be auto clear to 0 after reload new seed completed. **Note 2:** The reference is TLIRC 32K-clock. Tamper detector needs sync 2 ~ 3 TLIRC 32K-clock.
[3]	**ADYNSRC2**	**Active Shied Dynamic Reference Pattern 2** This field determines the new reference pattern when current pattern run out in dynamic pair mode. 0 = The new reference pattern is generated by random number generator when the reference pattern run out. 1 = The new reference pattern is repeated from SEED2 (TAMPER_SEED2[31:0]) when the reference pattern run out. **Note:** After this bit is modified, the SEEDRLD2 (TAMPER_ACTSTIOCTL2[4]) should be set.
[2:1]	**Reserved**	Reserved.
[0]	**ADYN1ISS2**	**Active Shied Dynamic Pair 1 Input Source Select 2** This bit determines if Tamper 3 input is from Tamper 2 or Tamper 0 in dynamic mode.

		0 = Tamper input is from Tamper 2. 1 = Tamper input is from Tamper 0. **Note:** This bit is effective only when ADYNPR1EN2 (TAMPER_ACTSTIOCTL2[23]) and ADYNPR0EN2 (TAMPER_ACTSTIOCTL2[15]) are set.

Tamper Clock Frequency Detector Boundary Register (TAMPER_CDBR)

Register	Offset	R/W	Description	Reset Value
TAMPER_CDBR	TAMPER_BA+0x2C	*6.41.7.1 R/W*	Tamper Clock Frequency Detector Boundary Register	0x00F0_000F

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

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:16]	**FAILBD**	**LXT Clock Frequency Detector Fail Boundary** The bits define the fail value of frequency monitor window. When LXT frequency monitor counter lower than Clock Frequency Detector Fail Boundary, the LXT frequency detect fail interrupt flag will set to 1. **Note:** The boundary is defined as the minimum value of LXT among 256 Tamper clock time.
[15:8]	**Reserved**	Reserved.
[7:0]	**STOPBD**	**LXT Clock Frequency Detector Stop Boundary** The bits define the stop value of frequency monitor window. When LXT frequency monitor counter lower than Clock Frequency Detector Stop Boundary, the LXT frequency detect stop interrupt flag will set to 1. **Note:** The boundary is defined as the maximum value of LXT among 256 Tamper clock time.

Tamper Voltage Glitch Control Register (TAMPER_VG)

Register	Offset	R/W	Description	Reset Value
TAMPER_VG	TAMPER_BA+0x30	R/W	Tamper Voltage Glitch Control Register	0x5876_5876

31	30	29	28	27	26	25	24
NDATSEL1				PDATSEL1			
23	22	21	20	19	18	17	16
NCLKSEL1				PCLKSEL1			
15	14	13	12	11	10	9	8
NDATSEL0				PDATSEL0			
7	6	5	4	3	2	1	0
NCLKSEL0				PCLKSEL0			

Bits	Description	
[31:28]	**NDATSEL1**	**PL1 Negative Data Trim Range** The setting value of the negative data tolerance. One step is about 2.5% tolerance. The maximum tolerance is 20%.
[27:24]	**PDATSEL1**	**PL1 Positive Data Trim Range** The setting value of the positive data tolerance. One step is about 2.5% tolerance. The maximum tolerance is 20%.
[23:20]	**NCLKSEL1**	**PL1 Negative Clock Trim Range** The setting value of the negative clock tolerance. One step is about 2.5% tolerance. The maximum tolerance is 20%.
[19:16]	**PCLKSEL1**	**PL1 Positive Clock Trim Range** The setting value of the positive clock tolerance. One step is about 2.5% tolerance. The maximum tolerance is 20%. **Note:** PL1 means the power level is 1.2V
[15:12]	**NDATSEL0**	**PL0 Negative Data Trim Range** The setting value of the negative data tolerance. One step is about 2.5% tolerance. The maximum tolerance is 20%.
[11:8]	**PDATSEL0**	**PL0 Positive Data Trim Range** The setting value of the positive data tolerance. One step is about 2.5% tolerance. The maximum tolerance is 20%.
[7:4]	**NCLKSEL0**	**PL0 Negative Clock Trim Range** The setting value of the negative clock tolerance. One step is about 2.5% tolerance. The maximum tolerance is 20%.
[3:0]	**PCLKSEL0**	**PL0 Positive Clock Trim Range** The setting value of the positive clock tolerance. One step is about 2.5% tolerance. The maximum tolerance is 20%.

		Note: PL0 means the power level is 1.26V The power level is controlled in system manager

Tamper Voltage Glitch Event Tolerance Control Register (TAMPER_VGEV)

Register	Offset	R/W	Description	Reset Value
TAMPER_VGEV	TAMPER_BA+0x34	R/W	Tamper Voltage Glitch Event Tolerance Control Register	0x0000_0303

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**VGECNTN**	**Negative Voltage Glitch Error Tolerance** The value indicates the tolerance count for negative voltage glitch event.
[7:0]	**VGECNTP**	**Positive Voltage Glitch Error Tolerance** The value indicates the tolerance count for positive voltage glitch event.

Tamper LDO Trim Control Register (TAMPER_LDOTRIM)

Register	Offset	R/W	Description	Reset Value
TAMPER_LDOTRIM	TAMPER_BA+0x38	R/W	Tamper LDO Trim Value Control Register	0x0000_0108

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

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9:8]	**TLDOIQSEL**	**Voltage Regulator Qu Current Selection** Indicates the Qu current selection of voltage regulator.
[7:4]	**Reserved**	Reserved.
[3:0]	**TLDOTRIM**	**Voltage Regulator Output Voltage Trim** The value indicates the trim value of the voltage regulator output voltage.

Tamper LDO BIAS Trim Control Register (TAMPER_LBSTRIM)

Register	Offset	R/W	Description	Reset Value
TAMPER_LBSTRIM	TAMPER_BA+0x3C	R/W	Tamper LDO BIAS Trim Value Control Register	0x0000_5505

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**Reserved**	Reserved
[7:5]	**Reserved**	Reserved.
[4]	**TOVDSEL**	**Over-shoot Detect Level Trim Bits** The value indicates the trim value of the over-shoot detection level
[3]	**Reserved**	Reserved.
[2:0]	**TLVDSEL**	**Under-shoot Detect Level Trim Bits** The value indicates the trim value of the under-shoot detection level

Tamper Voltage Glitch Control Register2 (TAMPER_VG2)

Register	Offset	R/W	Description	Reset Value
TAMPER_VG2	TAMPER_BA+0x40	R/W	Tamper Voltage Glitch Control Register 2	0x5876_5876

31	30	29	28	27	26	25	24
NDATSEL3				PDATSEL3			
23	22	21	20	19	18	17	16
NCLKSEL3				PCLKSEL3			
15	14	13	12	11	10	9	8
NDATSEL2				PDATSEL2			
7	6	5	4	3	2	1	0
NCLKSEL2				PCLKSEL2			

Bits	Description	
[31:28]	**NDATSEL3**	**PL3 Negative Data Trim Range** The setting value of the negative data tolerance. One step is about 2.5% tolerance. The maximum tolerance is 20%.
[27:24]	**PDATSEL3**	**PL3 Positive Data Trim Range** The setting value of the positive data tolerance. One step is about 2.5% tolerance. The maximum tolerance is 20%.
[23:20]	**NCLKSEL3**	**PL3 Negative Clock Trim Range** The setting value of the negative clock tolerance. One step is about 2.5% tolerance. The maximum tolerance is 20%.
[19:16]	**PCLKSEL3**	**PL3 Positive Clock Trim Range** The setting value of the positive clock tolerance. One step is about 2.5% tolerance. The maximum tolerance is 20%. **Note:** PL3 means the power level is 0.9V
[15:12]	**NDATSEL2**	**PL2 Negative Data Trim Range** The setting value of the negative data tolerance. One step is about 2.5% tolerance. The maximum tolerance is 20%.
[11:8]	**PDATSEL2**	**PL2 Positive Data Trim Range** The setting value of the positive data tolerance. One step is about 2.5% tolerance. The maximum tolerance is 20%.
[7:4]	**NCLKSEL2**	**PL2 Negative Clock Trim Range** The setting value of the negative clock tolerance. One step is about 2.5% tolerance. The maximum tolerance is 20%.
[3:0]	**PCLKSEL2**	**PL2 Positive Clock Trim Range** The setting value of the positive clock tolerance.

		One step is about 2.5% tolerance. The maximum tolerance is 20%. **Note:** PL2 means the power level is 1.1V The power level is controlled in system manager.

6.42 Digital to Analog Converter (DAC)

6.42.1 Overview

The DAC module is a 12-bit, voltage output digital-to-analog converter. It can be configured to 12-or 8-bit output mode and can be used in conjunction with the PDMA controller. The DAC integrates a voltage output buffer that can be used to reduce output impendence and drive external loads directly without having to add an external operational amplifier.

6.42.2 Features

- Analog output voltage range: 0~AV_{DD}.
- Supports 12-or 8-bit output mode.
- Rail to rail settle time 8us.
- Supports up to two 12-bit 1 MSPS voltage type DAC.
- Reference voltage from internal reference voltage (INT_VREF), V_{REF} pin.
- DAC maximum conversion updating rate 1 MSPS.
- Supports voltage output buffer mode and bypass voltage output buffer mode.
- Supports software and hardware trigger, including Timer0~3, EPWM0, EPWM1, and external trigger pin to start DAC conversion.
- Supports PDMA mode.
- Supports group mode of synchronized update capability for two DACs.

6.42.3 Block Diagram

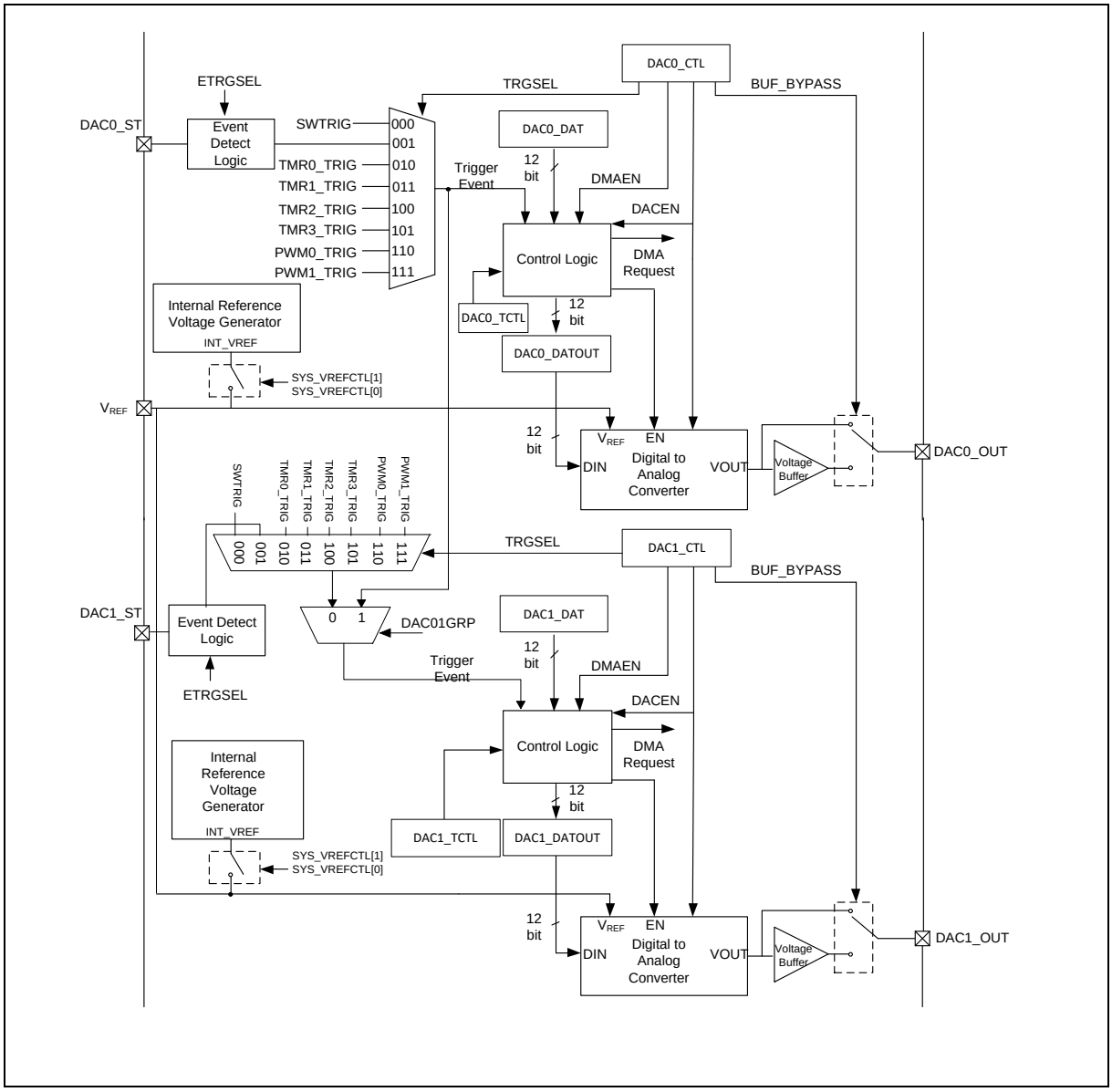


Figure 6.42-1 Digital-to-Analog Converter Block Diagram

6.42.4 Basic Configuration

6.42.4.1 DAC0 Basic Configuration

- Clock source Configuration
  - Enable DAC0 peripheral clock in DACCKEN (CLK_APBCLK1[12]).
- Reset Configuration
  - Reset DAC0 controller in DACRST (SYS_IPRST2[12]).
- Pin configuration

Group	Pin Name	GPIO	MFP
DAC0	DAC0_OUT	PB.12	MFP1

	DAC0_ST	PA.10	MFP14
		PA.0	MFP15

6.42.4.2 DAC1 Basic Configuration

- Clock source Configuration
  - Enable DAC1 peripheral clock in DACCKEN (CLK_APBCLK1[12]).
- Reset Configuration
  - Reset DAC1 controller in DACRST (SYS_IPRST2[12]).
- Pin configuration

Group	Pin Name	GPIO	MFP
DAC1	DAC1_OUT	PB.13	MFP1
	DAC1_ST	PA.11	MFP14
		PA.1	MFP15

6.42.5 Functional Description

6.42.5.1 DAC Output

The DAC is a 12-bit voltage output digital-to-analog converter and can be configured as 12-or 8-bit operation mode. The DAC integrates a voltage output buffer that can be used to reduce output impendence and drive external loads directly without having to add an external operational amplifier. The DAC channel output buffer can be enabled and disabled by BYPASS (DACn_CTL[8]), n=0, 1. The maximum DAC output voltage is limited to the selected reference voltage source.

6.42.5.2 DAC Reference Voltage

The DAC reference voltage is shared with EADC reference voltage and it is configured by VREFCTL (SYS_VREFCTL[4:0]) in system manager control registers. The reference voltage for the DAC can be configured from external reference voltage pin (V_{REF}) or internal reference voltage generator (INT_VREF).

6.42.5.3 DAC Data Format

The DAC supports conversion data left alignment or right alignment mode. Depending on the selected configuration mode, the data needs to be written into the specified register as follows:

- 12-bit left alignment: user has to load data into DACn_DAT[15:4] bits. DACn_DAT[31:16] and DACn_DAT[3:0] are ignored in DAC conversion.
- 12-bit right alignment: user has to load data into DACn_DAT[11:0] bits, DACn_DAT[31:12] are ignored in DAC conversion.

While DAC is working in 8-bit mode, alignment setting has no effect. To enable 8-bit mode, set BWSEL(DACn_CTL[15:14]) to 01. Otherwise, keep BWSEL as 00.

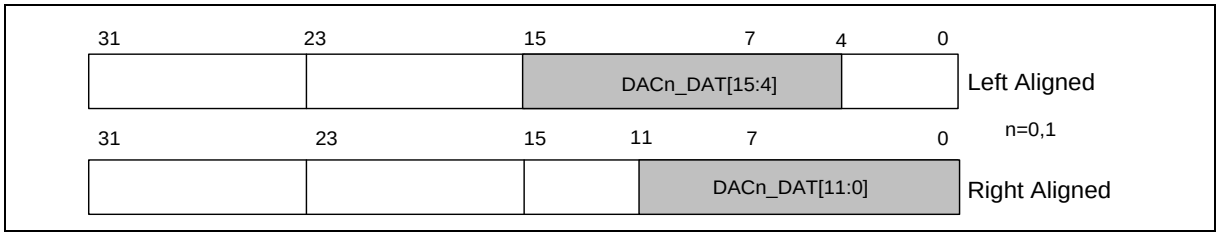


Figure 6.42-2 Data Holding Register Format

6.42.5.4 DAC Conversion

Any data transfer to the DAC channel is performed by loading the data into DACn_DAT register. Figure 6.42-3 shows the DAC conversion started by software write operation. When user writes the conversion data to data holding register DACn_DAT, the data is loaded into data output register DACn_DATOUT by hardware and DAC starts data conversion after one PCLK (APB clock) clock cycle. Figure 6.42-4 shows the DAC conversion started by hardware trigger (external pin DACn_ST, timer trigger event or EPWM timer trigger event). The data stored in the DACn_DAT register is automatically transferred to the data output buffer DACn_DATOUT after occurring one PCLK (APB clock) the event.

When DAC data output register DACn_DATOUT is loaded with the DACn_DAT contents, the analog output voltage becomes available after specified conversion settling time. The conversion settling time is 8us when 12- bit input code transition from lowest code (0x000) to highest code (0xFFF). Two adjacent codes conversion settling time is 1us. The DAC controller provides a 10-bit time counter for user to count the conversion time period. In continuous conversion operation, user needs to write appropriate value to SETTLET (DACn_TCTL[9:0]) to define DAC conversion time period. The value must be longer than DAC conversion settling time which is specified in DAC electric characteristic table. For example, when DAC controller APB clock speed is 80 MHz and DAC conversion settling time is 8us, the selected SETTLET value must be greater than 0x280. When the conversion is started, the conversion finish flag FINISH (DACn_STATUS[0]) is cleared to 0 by hardware and set to 1 after the time counter counts to SETTLET. Note that n=0,1.

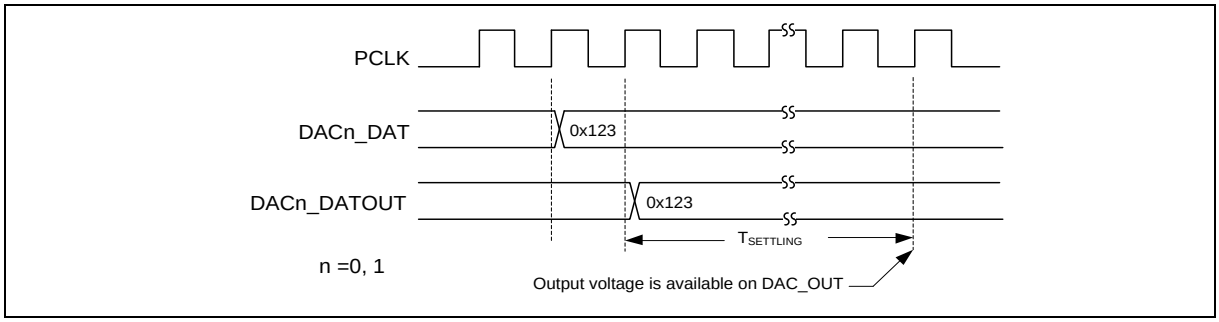


Figure 6.42-3 DAC Conversion Started by Software Write Trigger

6.42.5.5 DAC Output Voltage

Digital inputs are converted to output voltage on a linear conversion between 0 and reference voltage V_{REF}. The analog output voltage on DAC pin is determined by the following equation:

$$DACOUT = V_{REF} * \frac{DATnOUT[11:0]}{4096}, n=0,1$$

6.42.5.6 DAC Trigger Selection

The DAC conversion can be started by writing DACn_DAT, software trigger or hardware trigger. When TRGEN (DACn_CTL[4]) is 0, the data conversion is started by writing DACn_DAT register. When

TRGEN (DACn_CTL[4]) is 1, the data conversion is started by external DACn_ST pin, timer event, or EPWM timer event. If the software trigger is selected, the conversion starts once the SWTRG (DACn_SWTRG[0]) is set to 1. The SWTRG is cleared to 0 by hardware automatically when DACn_DATOUT has been loaded with DACDAT content. The TRGSEL (DACn_CTL[7:5]) determines which one of eight events is selected to start the conversion.

When DAC detects a rising edge on the selected trigger event input, the last data stored in DACDAT is transferred into the DACn_DATOUT[11:0] and DAC starts converting after one PCLK (APB clock) clock cycle. Note that n=0,1.

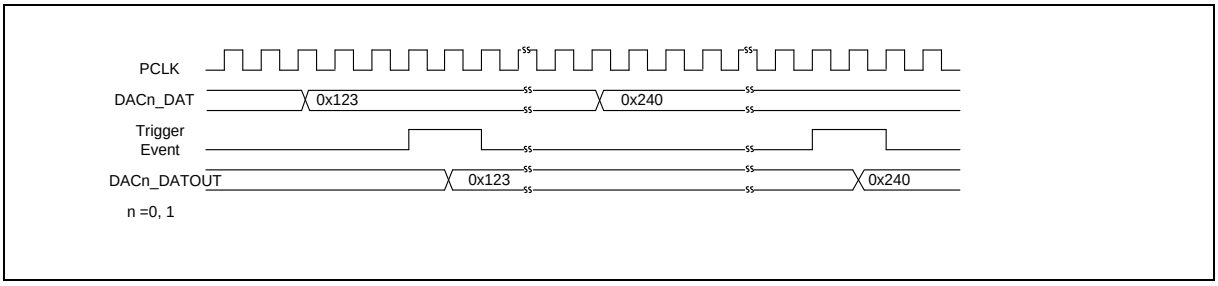


Figure 6.42-4 DAC Conversion Started by Hardware Trigger Event

6.42.5.7 DAC Group Mode

The DAC0 and DAC1 can be grouped together by setting GRPEN (DAC0_CTL[16]) to synchronize the update of each DAC output. Hardware ensures that these two DACs will be updated simultaneously in group mode. In group mode, DAC1_CTL and DAC1_TCTL has no effect. DAC1's behavior is controlled by DAC0_CTL and DAC0_TCTL. Figure 6.42-5 shows an example of group mode and compared with normal mode.

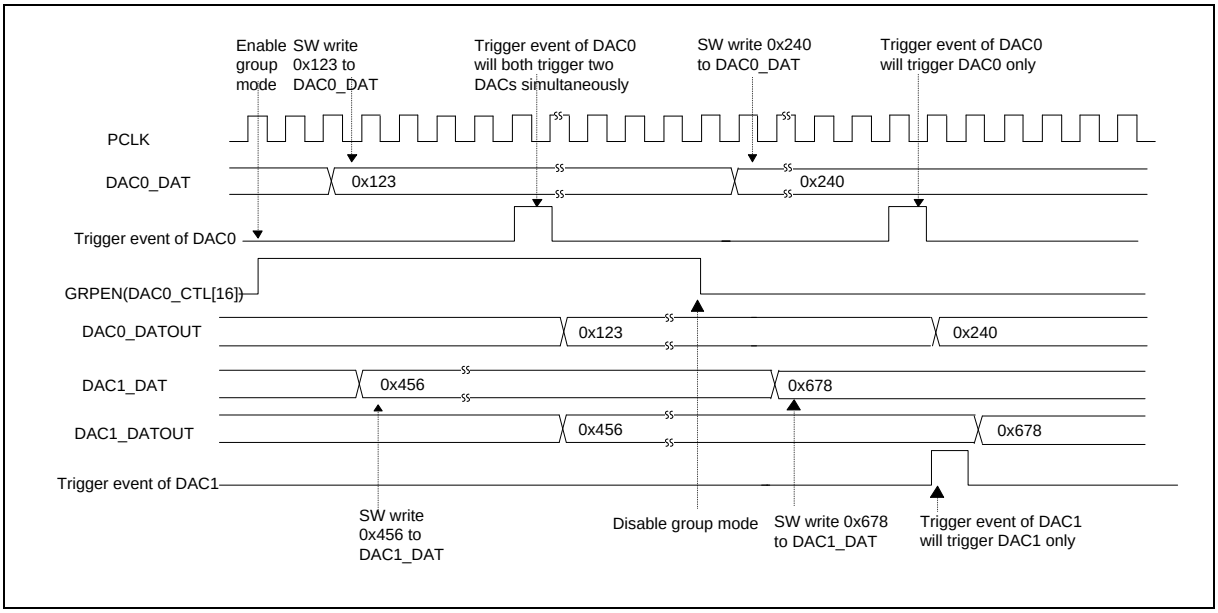


Figure 6.42-5 DAC0 and DAC1 Group and Ungroup Update Example

6.42.5.8 DMA Operation

A DAC DMA request is generated when a hardware trigger event occurs while DMAEN (DACn_CTL[2]) is set. The content of DACn_DAT is transferred to the DACn_DATOUT[11:0] and DAC starts data conversion after one PCLK (APB clock) clock cycle. The new transferred data by PDMA in DACn_DAT will be converted when next trigger event arrives. Figure 6.42-6 shows the DAC PDMA

under-run condition, when the second DMA request trigger event arrives before the first conversion finish, then no new PDMA request is issued and DMA under-run flag DMAUDR (DACn_STATUS[1]) is set 1 to report the error condition. DMA data transfers are then disabled and no further DMA request is treated and DAC continues to convert last data. An interrupt is also generated if the corresponding DMAURIEN (DACn_CTL[3]) is enabled. User has to change the trigger event frequency in timer or EPWM timer and then start DAC conversion again. Note that n=0,1.

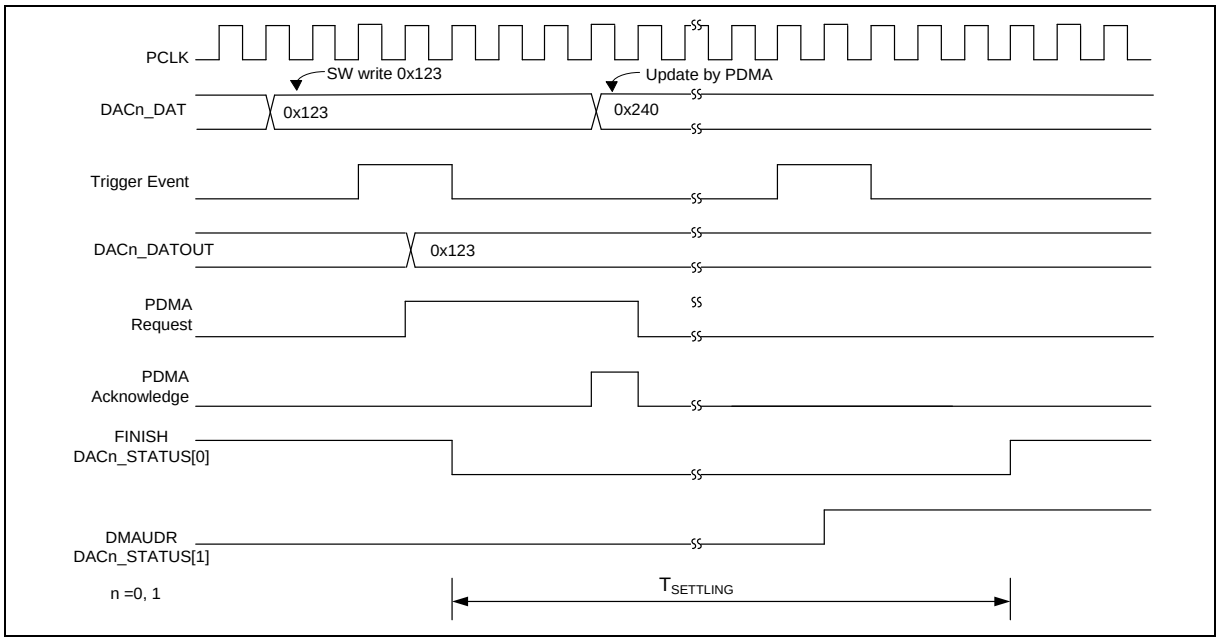


Figure 6.42-6 DAC PDMA Under-Run Condition Example

The DMA request can also be generated by software enable, user sets DMAEN (DACn_CTL[2]) to 1 and TRGEN (DACn_CTL[4]) to 0, DMA request is generated periodically according to the conversion time defined by SETTLET (DACn_TCTL[9:0]) value. DAC output is updated periodically. When user clears DMAEN (DACn_CTL[2]) to 0, DAC controller will stop issuing next new PDMA transfer request. Figure 6.42-7 provides an example of DAC continuous conversion with software PDMA mode. Note that n = 0,1.

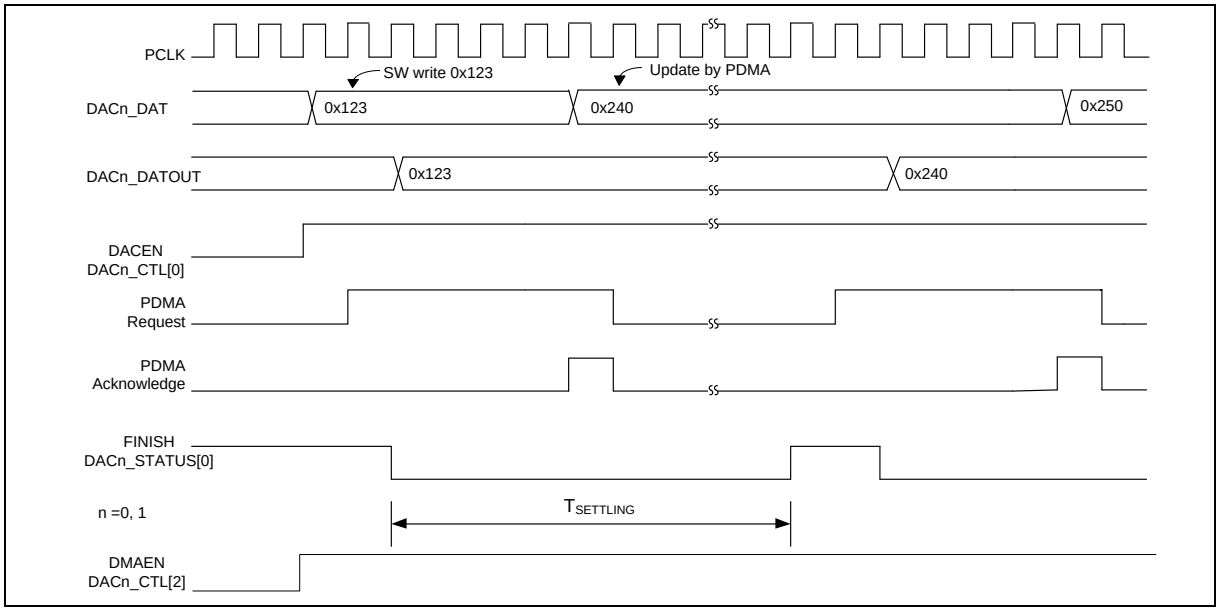


Figure 6.42-7 DAC Continuous Conversion with Software PDMA Mode

6.42.5.9 Interrupt Sources

There are two interrupt sources in the DAC controller, one is DAC data conversion finish interrupt and the other is DMA under-run interrupt as shown in Figure 6.42-8. When DAC conversion is finished, the FINISH (DACn_STATUS[0]) is set to 1 and an interrupt occurs while DACIEN (DACn_CTL[1]) is enabled. If a new DMA trigger event occurs during DAC data conversion period, the DMA under-run flag DMAUDR (DACn_STATUS[1]) is generated and an interrupt occurs if DMAURIEN (DACn_CTL[3]) is enabled. Note that n = 0,1.

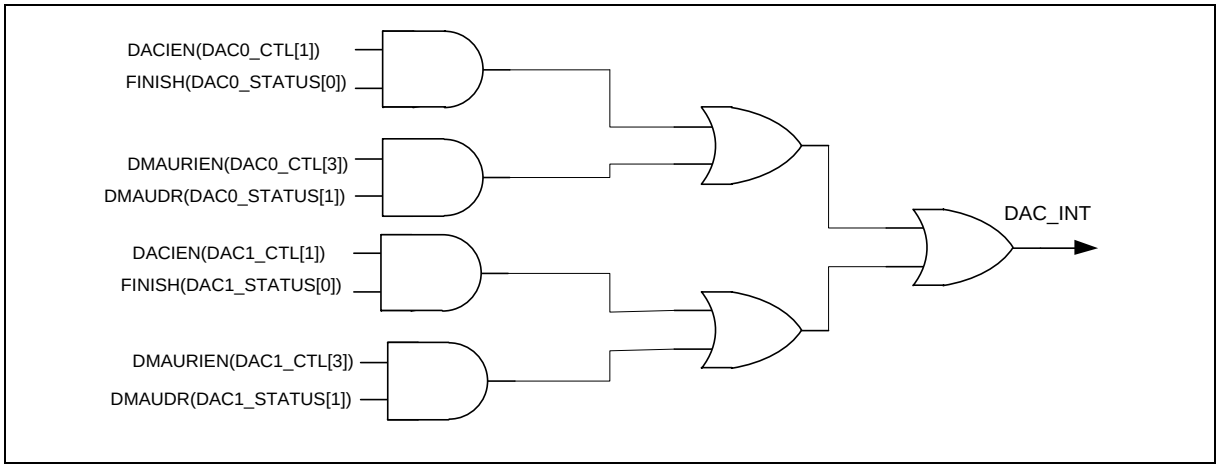


Figure 6.42-8 DAC Interrupt Source

6.42.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
DAC Base Address: **DAC_BA = 0x4004_7000** **DAC non-secure base address is DAC_BA + 0x1000_0000**				
DAC0_CTL	DAC_BA+0x00	R/W	DAC0 Control Register	0x0000_0000
DAC0_SWTRG	DAC_BA+0x04	R/W	DAC0 Software Trigger Control Register	0x0000_0000
DAC0_DAT	DAC_BA+0x08	R/W	DAC0 Data Holding Register	0x0000_0000
DAC0_DATOUT	DAC_BA+0x0C	R	DAC0 Data Output Register	0x0000_0000
DAC0_STATUS	DAC_BA+0x10	R/W	DAC0 Status Register	0x0000_0000
DAC0_TCTL	DAC_BA+0x14	R/W	DAC0 Timing Control Register	0x0000_0000
DAC1_CTL	DAC_BA+0x40	R/W	DAC1 Control Register	0x0000_0000
DAC1_SWTRG	DAC_BA+0x44	R/W	DAC1 Software Trigger Control Register	0x0000_0000
DAC1_DAT	DAC_BA+0x48	R/W	DAC1 Data Holding Register	0x0000_0000
DAC1_DATOUT	DAC_BA+0x4C	R	DAC1 Data Output Register	0x0000_0000
DAC1_STATUS	DAC_BA+0x50	R/W	DAC1 Status Register	0x0000_0000
DAC1_TCTL	DAC_BA+0x54	R/W	DAC1 Timing Control Register	0x0000_0000

6.42.7 Register Description

DAC0 Control Register (DAC0_CTL)

Register	Offset	R/W	Description	Reset Value
DAC0_CTL	DAC_BA+0x00	R/W	DAC0 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							GRPEN
15	14	13	12	11	10	9	8
BWSEL		ETRGSEL		Reserved	LALIGN	Reserved	BYPASS
7	6	5	4	3	2	1	0
TRGSEL			TRGEN	DMAURIEN	DMAEN	DACIEN	DACEN

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**GRPEN**	**DAC Group Mode Enable Bit** 0 = DAC0 and DAC1 are not grouped. 1 = DAC0 and DAC1 are grouped.
[15:14]	**BWSEL**	**DAC Data Bit-width Selection** 00 = Data is 12 bits. 01 = Data is 8 bits. Others = Reserved.
[13:12]	**ETRGSEL**	**External Pin Trigger Selection** 00 = Low level trigger. 01 = High level trigger. 10 = Falling edge trigger. 11 = Rising edge trigger.
[11]	**Reserved**	Reserved.
[10]	**LALIGN**	**DAC Data Left-aligned Enabled Bit** 0 = Right alignment. 1 = Left alignment.
[9]	**Reserved**	Reserved.
[8]	**BYPASS**	**Bypass Buffer Mode** 0 = Output voltage buffer Enabled. 1 = Output voltage buffer Disabled.
[7:5]	**TRGSEL**	**Trigger Source Selection** 000 = Software trigger.

		001 = External pin DAC0_ST trigger. 010 = Timer 0 trigger. 011 = Timer 1 trigger. 100 = Timer 2 trigger. 101 = Timer 3 trigger. 110 = EPWM0 trigger. 111 = EPWM1 trigger.
[4]	**TRGEN**	**Trigger Mode Enable Bit** 0 = DAC event trigger mode Disabled. 1 = DAC event trigger mode Enabled.
[3]	**DMAURIEN**	**DMA Under-run Interrupt Enable Bit** 0 = DMA under-run interrupt Disabled. 1 = DMA under-run interrupt Enabled.
[2]	**DMAEN**	**DMA Mode Enable Bit** 0 = DMA mode Disabled. 1 = DMA mode Enabled.
[1]	**DACIEN**	**DAC Interrupt Enable Bit** 0 = DAC interrupt Disabled. 1 = DAC interrupt Enabled.
[0]	**DACEN**	**DAC Enable Bit** 0 = DAC Disabled. 1 = DAC Enabled.

DAC0 Software Trigger Control Register (DAC0_SWTRG)

Register	Offset	R/W	Description	Reset Value
DAC0_SWTRG	DAC_BA+0x04	R/W	DAC0 Software Trigger Control Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**SWTRG**	**Software Trigger** 0 = Software trigger Disabled. 1 = Software trigger Enabled. **Note:** User writes this bit to generate one shot pulse and it is cleared to 0 by hardware automatically; reading this bit will always get 0.

DAC0 Data Holding Register (DAC0_DAT)

Register	Offset	R/W	Description	Reset Value
DAC0_DAT	DAC_BA+0x08	R/W	DAC0 Data Holding Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**DACDAT**	**DAC 12-bit Holding Data** These bits are written by user software which specifies 12-bit conversion data for DAC output. The unused bits (DAC0_DAT[3:0] in left-alignment mode and DAC0_DAT[15:12] in right alignment mode) are ignored by DAC controller hardware. 12 bit left alignment: user has to load data into DAC0_DAT[15:4] bits. 12 bit right alignment: user has to load data into DAC0_DAT[11:0] bits.

DAC0 Data Output Register (DAC0_DATOUT)

Register	Offset	R/W	Description	Reset Value
DAC0_DATOUT	DAC_BA+0x0C	R	DAC0 Data Output Register	0x0000_0000

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

Bits	Description	
[31:12]	Reserved	Reserved.
[11:0]	DATOUT	DAC 12-bit Output Data These bits are current digital data for DAC output conversion. It is loaded from DAC0_DAT register and user cannot write it directly.

DAC0 Status Register (DAC0_STATUS)

Register	Offset	R/W	Description	Reset Value
DAC0_STATUS	DAC_BA+0x10	R/W	DAC0 Status Register	0x0000_0000

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

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**BUSY**	**DAC Busy Flag (Read Only)** 0 = DAC is ready for next conversion. 1 = DAC is busy in conversion.
[7:2]	**Reserved**	Reserved.
[1]	**DMAUDR**	**DMA Under-run Interrupt Flag** 0 = No DMA under-run error condition occurred. 1 = DMA under-run error condition occurred. **Note:** User writes 1 to clear this bit.
[0]	**FINISH**	**DAC Conversion Complete Finish Flag** 0 = DAC is in conversion state. 1 = DAC conversion finish. **Note:** This bit is set to 1 when conversion time counter counts to SETTLET. It is cleared to 0 when DAC starts a new conversion. User writes 1 to clear this bit to 0.

DAC0 Timing Control Register (DAC0_TCTL)

Register	Offset	R/W	Description	Reset Value
DAC0_TCTL	DAC_BA+0x14	R/W	DAC0 Timing Control Register	0x0000_0000

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

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9:0]	**SETTLET**	**DAC Output Settling Time** User software needs to write appropriate value to these bits to meet DAC conversion settling time base on PCLK (APB clock) speed. For example, DAC controller clock speed is 80 MHz and DAC conversion settling time is 1 us, SETTLET value must be greater than 0x50. SELTTLET = DAC controller clock speed x settling time.

DAC1 Control Register (DAC1_CTL)

Register	Offset	R/W	Description	Reset Value
DAC1_CTL	DAC_BA+0x40	R/W	DAC1 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
BWSEL		ETRGSEL		Reserved	LALIGN	Reserved	BYPASS
7	6	5	4	3	2	1	0
TRGSEL			TRGEN	DMAURIEN	DMAEN	DACIEN	DACEN

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:14]	**BWSEL**	**DAC Data Bit-width Selection** 00 = Data is 12 bits. 01 = Data is 8 bits. Others = reserved.
[13:12]	**ETRGSEL**	**External Pin Trigger Selection** 00 = Low level trigger. 01 = High level trigger. 10 = Falling edge trigger. 11 = Rising edge trigger.
[11]	**Reserved**	Reserved.
[10]	**LALIGN**	**DAC Data Left-aligned Enable Control** 0 = Right alignment. 1 = Left alignment.
[9]	**Reserved**	Reserved.
[8]	**BYPASS**	**Bypass Buffer Mode** 0 = Output voltage buffer Enabled. 1 = Output voltage buffer Disabled.
[7:5]	**TRGSEL**	**Trigger Source Selection** 000 = Software trigger. 001 = External pin DAC1_ST trigger. 010 = Timer 0 trigger. 011 = Timer 1 trigger. 100 = Timer 2 trigger.

		101 = Timer 3 trigger. 110 = EPWM0 trigger. 111 = EPWM1 trigger.
[4]	**TRGEN**	**Trigger Mode Enable Bit** 0 = DAC event trigger mode Disabled. 1 = DAC event trigger mode Enabled.
[3]	**DMAURIEN**	**DMA Under-run Interrupt Enable Bit** 0 = DMA under-run interrupt Disabled. 1 = DMA under-run interrupt Enabled.
[2]	**DMAEN**	**DMA Mode Enable Bit** 0 = DMA mode Disabled. 1 = DMA mode Enabled.
[1]	**DACIEN**	**DAC Interrupt Enable Bit** 0 = DAC interrupt Disabled. 1 = DAC interrupt Enabled.
[0]	**DACEN**	**DAC Enable Bit** 0 = DAC Disabled. 1 = DAC Enabled.

DAC1 Software Trigger Control Register (DAC1_SWTRG)

Register	Offset	R/W	Description	Reset Value
DAC1_SWTRG	DAC_BA+0x44	R/W	DAC1 Software Trigger Control Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**SWTRG**	**Software Trigger** 0 = Software trigger Disabled. 1 = Software trigger Enabled. **Note:** User writes this bit to generate one shot pulse and it is cleared to 0 by hardware automatically; Reading this bit will always get 0.

DAC1 Data Holding Register (DAC1_DAT)

Register	Offset	R/W	Description	Reset Value
DAC1_DAT	DAC_BA+0x48	R/W	DAC1 Data Holding Register	0x0000_0000

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

Bits	Description	
[31:16]	Reserved	Reserved.
[15:0]	DACDAT	**DAC 12-bit Holding Data** These bits are written by user software which specifies 12-bit conversion data for DAC output. The unused bits (DAC1_DAT[3:0] in left-alignment mode and DAC1_DAT[15:12] in right alignment mode) are ignored by DAC controller hardware. 12 bit left alignment: user has to load data into DAC1_DAT[15:4] bits. 12 bit right alignment: user has to load data into DAC1_DAT[11:0] bits.

DAC1 Data Output Register (DAC1_DATOUT)

Register	Offset	R/W	Description	Reset Value
DAC1_DATOUT	DAC_BA+0x4C	R	DAC1 Data Output Register	0x0000_0000

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

Bits	Description	
[31:12]	Reserved	Reserved.
[11:0]	DATOUT	**DAC 12-bit Output Data** These bits are current digital data for DAC output conversion. It is loaded from DAC1_DAT register and user cannot write it directly.

DAC1 Status Register (DAC1_STATUS)

Register	Offset	R/W	Description	Reset Value
DAC1_STATUS	DAC_BA+0x50	R/W	DAC1 Status Register	0x0000_0000

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

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**BUSY**	**DAC Busy Flag (Read Only)** 0 = DAC is ready for the next conversion. 1 = DAC is busy in conversion.
[7:2]	**Reserved**	Reserved.
[1]	**DMAUDR**	**DMA Under-run Interrupt Flag** 0 = No DMA under-run error condition occurred. 1 = DMA under-run error condition occurred. **Note:** User writes 1 to clear this bit.
[0]	**FINISH**	**DAC Conversion Complete Finish Flag** 0 = DAC is in conversion state. 1 = DAC conversion finished. **Note:** This bit set to 1 when conversion time counter counts to SETTLET. It is cleared to 0 when DAC starts a new conversion. User writes 1 to clear this bit to 0.

DAC1 Timing Control Register (DAC1_TCTL)

Register	Offset	R/W	Description	Reset Value
DAC1_TCTL	DAC_BA+0x54	R/W	DAC1 Timing Control Register	0x0000_0000

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

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9:0]	**SETTLET**	**DAC Output Settling Time** User software needs to write appropriate value to these bits to meet DAC conversion settling time base on PCLK (APB clock) speed. For example, DAC controller clock speed is 80 MHz and DAC conversion settling time is 1 us, SETTLET value must be greater than 0x50. SELTTLET = DAC controller clock speed x settling time.

6.43 Analog Comparator Controller (ACMP)

6.43.1 Overview

The chip provides two comparators. The comparator output is logic 1 when positive input is greater than negative input; otherwise, the output is 0. Each comparator can be configured to generate an interrupt when the comparator output value changes.

6.43.2 Features

- Analog input voltage range: 0 ~ AV_{DD} (voltage of AV_{DD} pin)
- Up to two rail-to-rail analog comparators
- Supports hysteresis function
  - Supports programmable hysteresis window: 0mV, 10mV, 20mV and 30mV
- Supports wake-up function
- Supports programmable propagaion speed and low power consumption
- Selectable input sources of positive input and negative input
- ACMP0 supports:
  - 4 multiplexed I/O pins at positive sources:
    - ACMP0_P0, ACMP0_P1, ACMP0_P2, or ACMP0_P3
  - 4 negative sources:
    - ACMP0_N
    - Comparator Reference Voltage (CRV)
    - Internal band-gap voltage (VBG)
    - DAC0 output (DAC0_OUT)
- ACMP1 supports
  - 4 multiplexed I/O pins at positive sources:
    - ACMP1_P0, ACMP1_P1, ACMP1_P2, or ACMP1_P3
  - 4 negative sources:
    - ACMP1_N
    - Comparator Reference Voltage (CRV)
    - Internal band-gap voltage (VBG)
    - DAC0 output (DAC0_OUT)
- Shares one ACMP interrupt vector for all comparators
- Interrupts generated when compare results change (Interrupt event condition is programmable)
- Supports triggers for break events and cycle-by-cycle control for PWM
- Supports window compare mode and window latch mode

6.43.3 Block Diagram

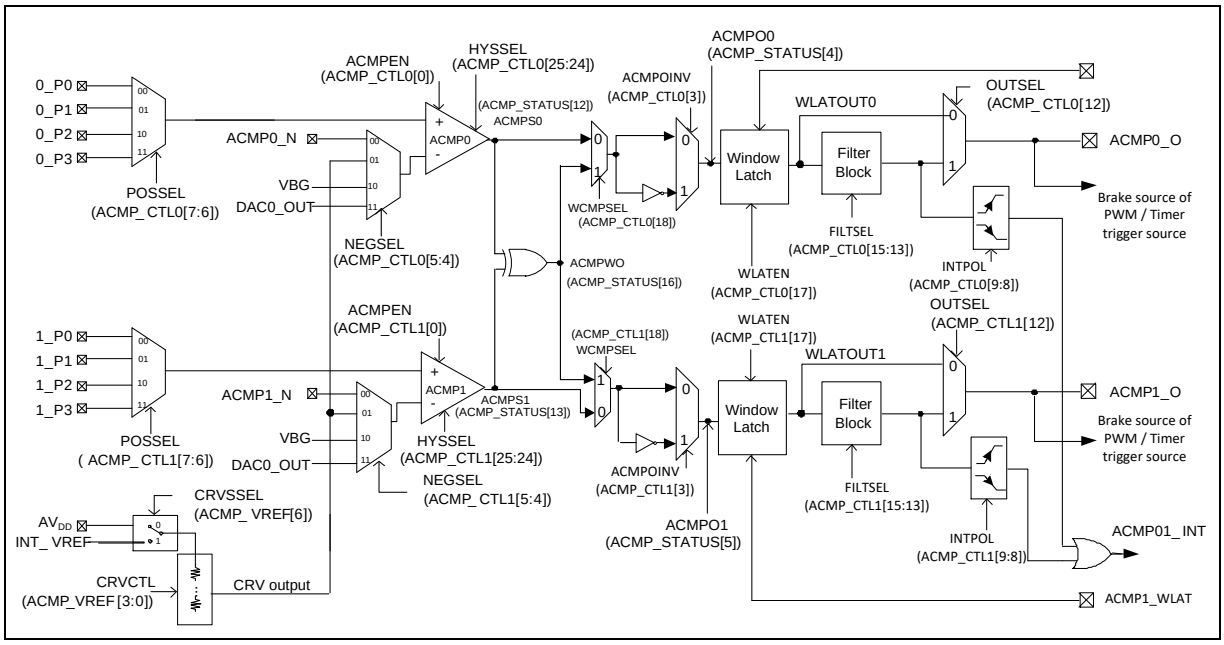


Figure 6.43-1 Analog Comparator Block Diagram

6.43.4 Basic Configuration

6.43.4.1 ACMP0 Basic Configuration

- Clock source Configuration
  - Enable ACMP0 peripheral clock in ACMP01CKEN (CLK_APBCLK0[7]).
- Reset Configuration
  - Reset ACMP0 controller in ACMP01RST (SYS_IPRST1[7]).
- Pin configuration

Group	Pin Name	GPIO	MFP
ACMP0	ACMP0_N	PB.3	MFP1
	ACMP0_O	PC.1, PC.12	MFP14
		PB.7	MFP15
	ACMP0_P0	PA.11	MFP1
	ACMP0_P1	PB.2	MFP1
	ACMP0_P2	PB.12	MFP1
	ACMP0_P3	PB.13	MFP1
	ACMP0_WLAT	PA.7	MFP13

6.43.4.2 ACMP1 Basic Configuration

- Clock source Configuration
  - Enable ACMP1 peripheral clock in ACMP01CKEN (CLK_APBCLK0[7]).

- Reset Configuration
  - Reset ACMP1 controller in ACMP01RST (SYS_IPRST1[7]).
- Pin configuration

Group	Pin Name	GPIO	MFP
ACMP1	ACMP1_N	PB.5	MFP1
	ACMP1_O	PC.0, PC.11	MFP14
		PB.6	MFP15
	ACMP1_P0	PA.10	MFP1
	ACMP1_P1	PB.4	MFP1
	ACMP1_P2	PB.12	MFP1
	ACMP1_P3	PB.13	MFP1
	ACMP1_WLAT	PA.6	MFP13

6.43.5 Functional Description

6.43.5.1 Hysteresis Function

The analog comparator provides the hysteresis function to make the comparator have a stable output transition as shown in Figure 6.43-2. If comparator output is 0, it will not be changed to 1 until the positive input voltage exceeds the negative input voltage by a high threshold voltage. Similarly, if the comparator output is 1, it will not be changed to 0 until the positive input voltage drops below the negative input voltage by a low threshold voltage.

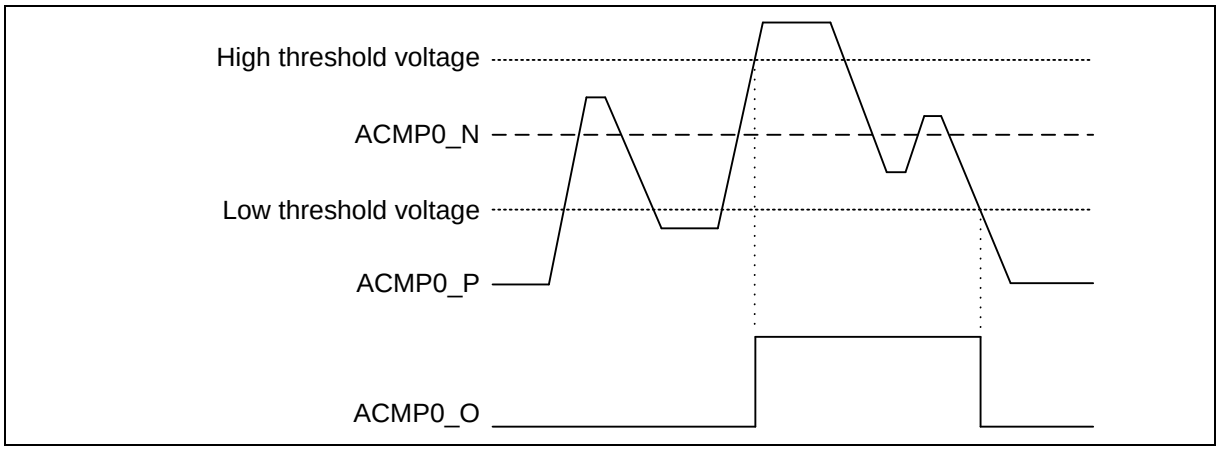


Figure 6.43-2 Comparator Hysteresis Function of ACMP0

6.43.5.2 Window Latch Mode

Figure 6.43-3 shows the comparator operation in window latch mode. Window latch mode can be enabled by setting WLATEN (ACMP_CTL0/1[17]) to 1. When window latch function is enabled, ACMP0/1_WLAT pin is used to control the output WLATOUT0/1 .When ACMP0/1_WLAT pin is high, ACMPO0/1 passes through to WLATOUT0/1. When ACMP0/1_WLAT pin is low, WLATOUT0/1 will keep the last state of WLATOUT0/1.

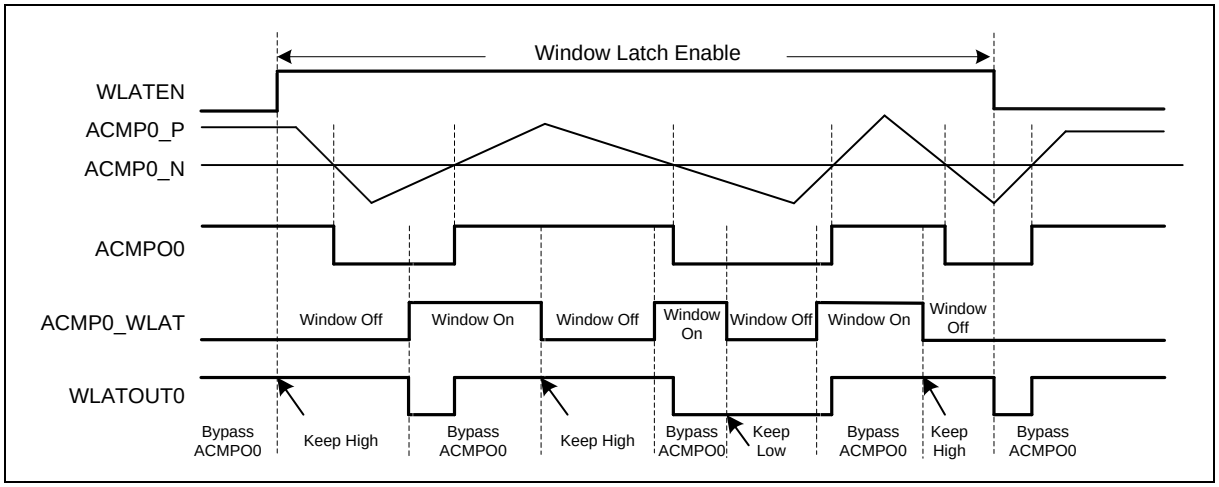


Figure 6.43-3 Window Latch Mode

6.43.5.3 Filter Function

The analog comparator provides filter function to avoid the un-stable state of comparator output.

By setting FILTSEL (ACMP_CTL0[15:13], ACMP_CTL1[15:13]), the comparator output would be sampled by consecutive PCLKs. With longer sample clocks, the comparator output would be more stable. But the sensitivity of comparator output would be reduced.

Figure 6.43-4 shows an example of filter function of ACMP0 with FILTSEL = 3 (4 PCLK). In this example, the comparing result is sampled by PCLK. All result must keep for 4 PCLK clocks before it can be output to ACMPO0. If the comparing result is shorter than 4 PCLK, it will be filtered.

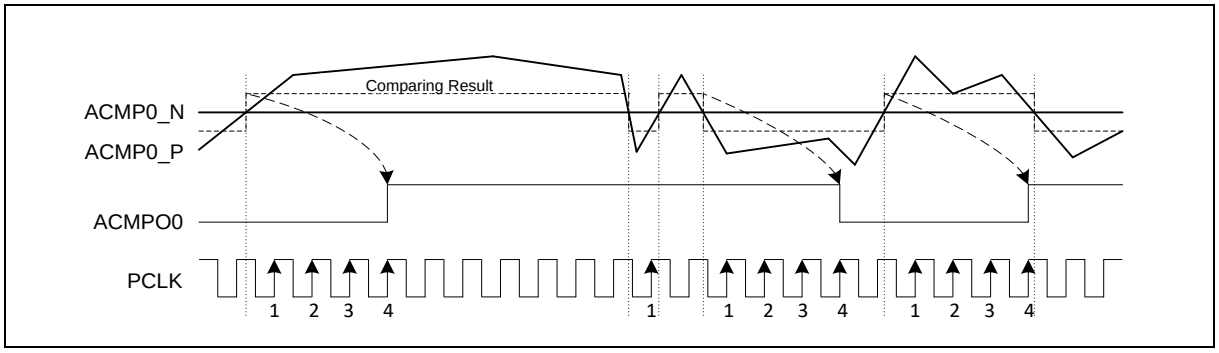


Figure 6.43-4 Example of Filter Function

6.43.5.4 Interrupt Sources

The outputs of ACMP0 and ACMP1 are reflected at ACMPO0 (ACMP_STATUS[4]) and ACMPO1 (ACMP_STATUS[5]) respectively. Then they are processed by window latch and filter functions. Finally, the output signal could be utilized to assert interrupts. Refer to Figure 6.43-5, if ACMPIE of ACMP_CTL0/1 register is set to 1, the interrupt will be enabled. If the output state ACMPO0/1 is changed as the setting of INTPOL (ACMP_CTL0/1[9:8]), the comparator interrupt will be asserted and the corresponding flag, ACMPIF0 (ACMP_STATUS[0]) and ACMPIF1 (ACMP_STATUS[1]), will be set to 1. The interrupt flag can be cleared to 0 by writing 1.

WKIF(ACMP_STATUS[8], ACMP_STATUS[9]) will be set according to the setting of INTPOL (ACMP_CTL0/1[9:8]) if ACMP wakeup function is enabled. These two flags also cause interrupt rising to make system wake up from power down by ACMP. Figure 6.43-5 shows the interrupt sources of ACMP.

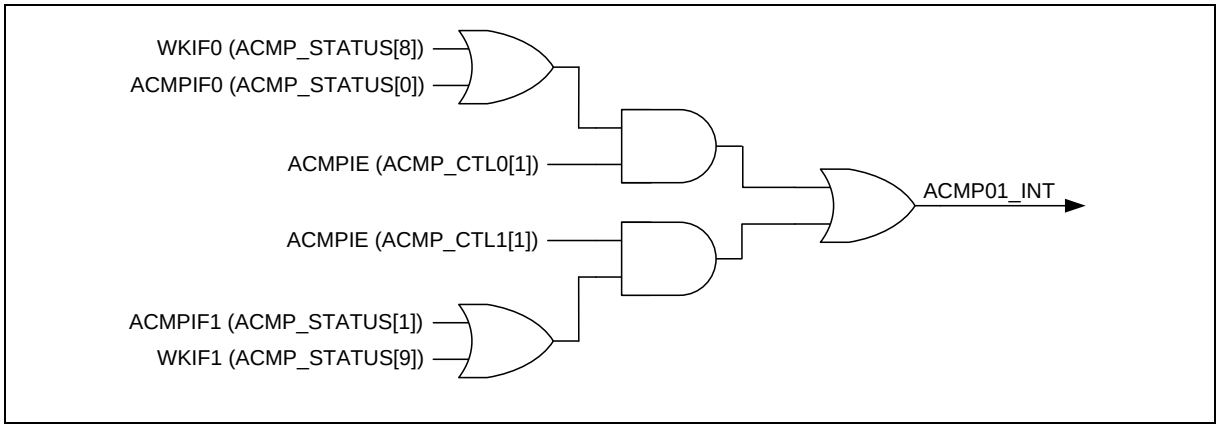


Figure 6.43-5 Comparator Controller Interrupt

6.43.5.5 Comparator Reference Voltage (CRV)

The comparator reference voltage (CRV) module is responsible for generating reference voltage for comparators. The CRV module consists of resistor ladder and analog switch. User can set the CRV output voltage by setting CRVCTL (ACMP_VREF[3:0]). The CRV output voltage can be selected as the negative input of comparator by setting NEGSEL (ACMP_CTL0[5:4], ACMP_CTL1[5:4]). Figure 6.43-6 shows the block diagram of Comparator Reference Voltage.

The resistor ladder will be disabled by hardware to reduce power consumption when NEGSEL (ACMP_CTL0[5:4], ACMP_CTL1[5:4]) is not selected to CRV module. The reference voltage of resistor ladder can be the voltage of AVDD pin or the INT_VREF voltage which is controlled by SYS_VREFCTL register.

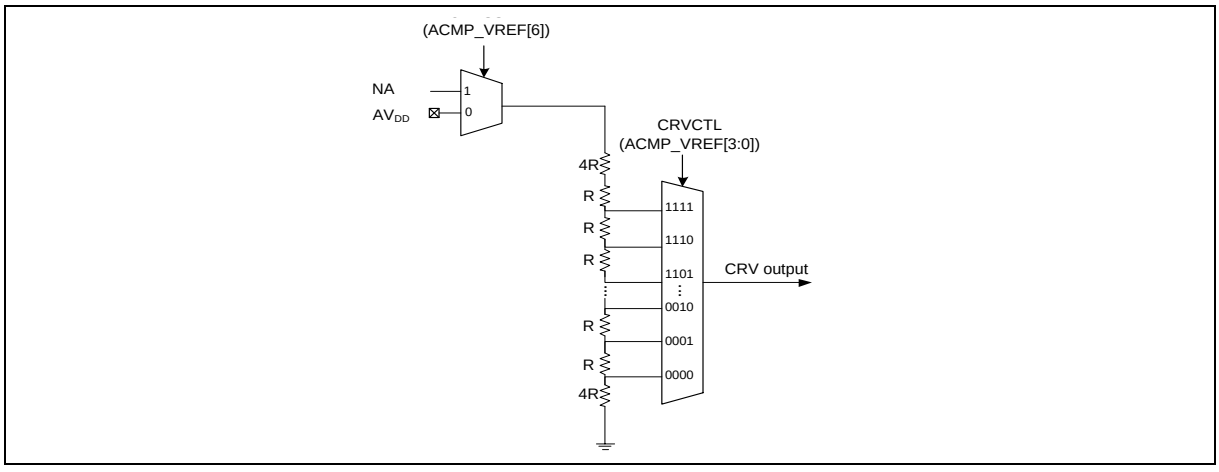


Figure 6.43-6 Comparator Reference Voltage Block Diagram

6.43.5.6 Window Compare Mode

The comparator provides window compare mode. When window compare mode is enabled by setting WCMPSEL (ACMP_CTL0/1[18]) to 1, user can monitor a specific analog voltage source with a designated range. User can connect the specific analog voltage source to either the positive inputs of both comparators or the negative inputs of both comparators. The upper bound and lower bound of the designated range are determined by the voltages applied to the other inputs of both comparators. If the output of a comparator is low and the other comparator outputs high, which means two comparators implies the upper and lower bound. User can directly monitor a specific analog voltage source via ACMPWO (ACMP_STATUS[16]). If ACMPWO is high, it implies a specific analog voltage

source is in the range of upper and lower bound, which is called as the analog voltage in the window.

Figure 6.43-7 illustrates an example of window compare mode. In this example, once window compare mode is selected, user can choose one of four positive input sources of each comparator and connect these two inputs together outside the chip.

If ACMPS0 outputs high and ACMPS1 outputs low, it means the voltage source is in the range of lower bound and upper bound, which is called as the voltage source in the window. Otherwise, the voltage source is outside the window.

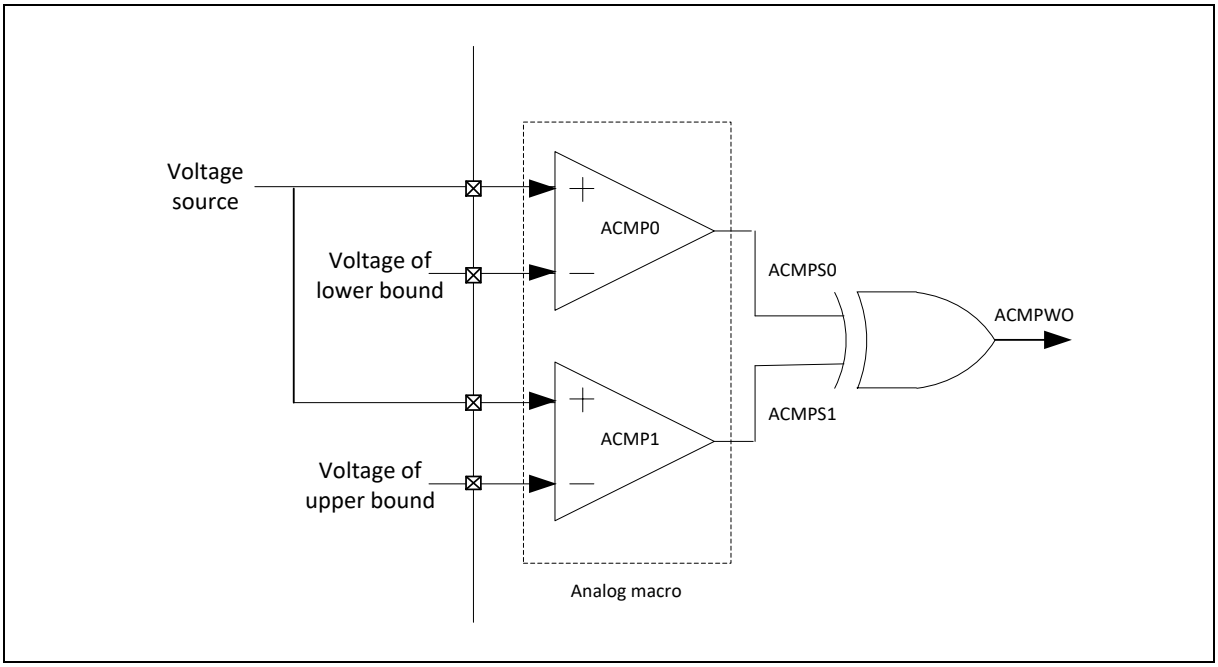


Figure 6.43-7 Example of Window Compare Mode

The comparator window output (ACMPWO) can be shown in ACMP_STATUS[16] and the truth table of window compare logic is shown in Table 6.43-1.

ACMPS0	ACMPS1	ACMPWO
0	0	0
0	1	1
1	1	0
1	0	1

Table 6.43-1 Truth Table of Window Compare Logic

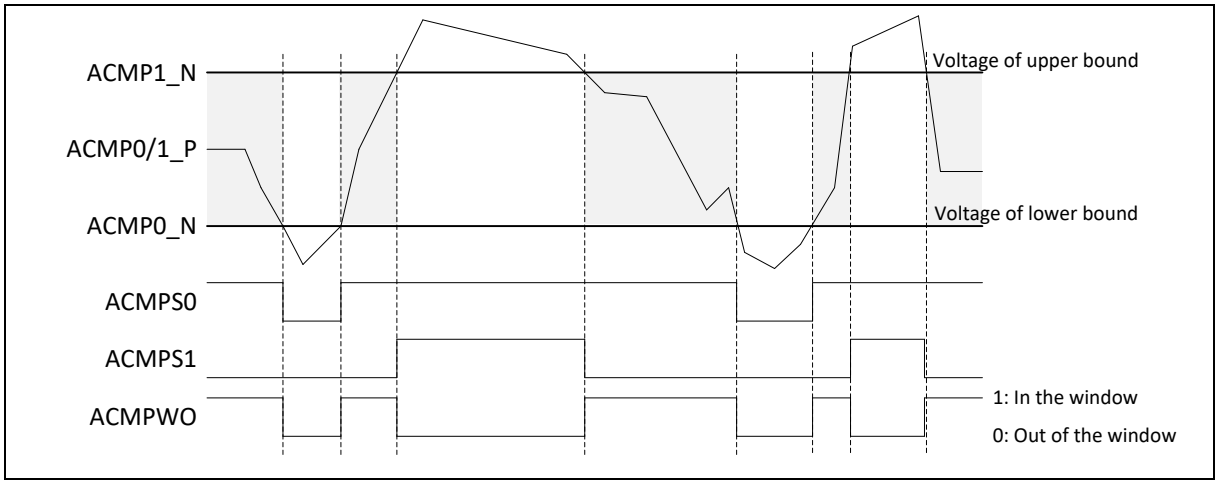


Figure 6.43-8 Example of Window Compare Mode

As shown in Figure 6.43-8, if ACMPWO equals 1, it means positive input voltage is inside the window. Otherwise, the positive input voltage is outside the window. Therefore, ACMPWO can be used to monitor voltage transition of external analog pin. Furthermore, ACMPWO can still be applied to window latch, filter functions and interrupt of ACMP.

Note that negative inputs must choose different source. Otherwise, the function will be meaningless.

6.43.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
ACMP Base Address: **ACMP01_BA = 0x4004_5000** **ACMP01 non-secure base address is ACMP01+ 0x1000_0000**				
ACMP_CTL0	ACMP01_BA+0x00	R/W	Analog Comparator 0 Control Register	0x0000_0000
ACMP_CTL1	ACMP01_BA+0x04	R/W	Analog Comparator 1 Control Register	0x0000_0000
ACMP_STATUS	ACMP01_BA+0x08	R/W	Analog Comparator Status Register	0x0000_0000
ACMP_VREF	ACMP01_BA+0x0C	R/W	Analog Comparator Reference Voltage Control Register	0x0000_0000

6.43.7 Register Description

Analog Comparator 0 Control Register (ACMP_CTL0)

Register	Offset	R/W	Description	Reset Value
ACMP_CTL0	ACMP01_BA+0x00	R/W	Analog Comparator 0 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		MODESEL		Reserved		HYSSEL	
23	22	21	20	19	18	17	16
Reserved					WCMPSEL	WLATEN	WKEN
15	14	13	12	11	10	9	8
FILTSEL			OUTSEL	Reserved		INTPOL	
7	6	5	4	3	2	1	0
POSSEL		NEGSEL		ACMPOINV	Reserved	ACMPIE	ACMPEN

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:28]	**MODESEL**	**Propagation Delay Mode Selection** 00 = Max propagation delay is 4.5uS, operation current is 1.2uA. 01 = Max propagation delay is 2uS, operation current is 3uA. 10 = Max propagation delay is 600nS, operation current is 10uA. 11 = Max propagation delay is 200nS, operation current is 75uA.
[27:26]	**Reserved**	Reserved.
[25:24]	**HYSSEL**	**Hysteresis Mode Selection** 00 = Hysteresis is 0mV. 01 = Hysteresis is 10mV. 10 = Hysteresis is 20mV. 11 = Hysteresis is 30mV.
[23:19]	**Reserved**	Reserved.
[18]	**WCMPSEL**	**Window Compare Mode Selection** 0 = Window Compare Mode Disabled. 1 = Window Compare Mode Selected.
[17]	**WLATEN**	**Window Latch Mode Enable Bit** 0 = Window Latch Mode Disabled. 1 = Window Latch Mode Enabled.
[16]	**WKEN**	**Power-down Wake-up Enable Bit** 0 = Wake-up function Disabled. 1 = Wake-up function Enabled.
[15:13]	**FILTSEL**	**Comparator Output Filter Count Selection**

		000 = Filter function is Disabled. 001 = ACMP0 output is sampled 1 consecutive PCLK. 010 = ACMP0 output is sampled 2 consecutive PCLKs. 011 = ACMP0 output is sampled 4 consecutive PCLKs. 100 = ACMP0 output is sampled 8 consecutive PCLKs. 101 = ACMP0 output is sampled 16 consecutive PCLKs. 110 = ACMP0 output is sampled 32 consecutive PCLKs. 111 = ACMP0 output is sampled 64 consecutive PCLKs.
[12]	**OUTSEL**	**Comparator Output Select** 0 = Comparator 0 output to ACMP0_O pin is unfiltered comparator output. 1 = Comparator 0 output to ACMP0_O pin is from filter output.
[11:10]	**Reserved**	Reserved.
[9:8]	**INTPOL**	**Interrupt Condition Polarity Selection** ACMPIF0 will be set to 1 when comparator output edge condition is detected. 00 = Rising edge or falling edge. 01 = Rising edge. 10 = Falling edge. 11 = Reserved.
[7:6]	**POSSEL**	**Comparator Positive Input Selection** 00 = Input from ACMP0_P0. 01 = Input from ACMP0_P1. 10 = Input from ACMP0_P2. 11 = Input from ACMP0_P3.
[5:4]	**NEGSEL**	**Comparator Negative Input Selection** 00 = ACMP0_N pin. 01 = Internal comparator reference voltage (CRV). 10 = Band-gap voltage. 11 = DAC output.
[3]	**ACMPOINV**	**Comparator Output Inverse** 0 = Comparator 0 output inverse Disabled. 1 = Comparator 0 output inverse Enabled.
[2]	**Reserved**	Reserved.
[1]	**ACMPIE**	**Comparator Interrupt Enable Bit** 0 = Comparator 0 interrupt Disabled. 1 = Comparator 0 interrupt Enabled. If WKEN (ACMP_CTL0[16]) is set to 1, the wake-up interrupt function will be enabled as well.
[0]	**ACMPEN**	**Comparator Enable Bit** 0 = Comparator 0 Disabled. 1 = Comparator 0 Enabled.

Analog Comparator 1 Control Register (ACMP_CTL1)

Register	Offset	R/W	Description	Reset Value
ACMP_CTL1	ACMP01_BA+0x04	R/W	Analog Comparator 1 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		MODESEL		Reserved		HYSSEL	
23	22	21	20	19	18	17	16
Reserved					WCMPSEL	WLATEN	WKEN
15	14	13	12	11	10	9	8
FILTSEL			OUTSEL	Reserved		INTPOL	
7	6	5	4	3	2	1	0
POSSEL		NEGSEL		ACMPOINV	Reserved	ACMPIE	ACMPEN

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:28]	**MODESEL**	**Propagation Delay Mode Selection** 00 = Max propagation delay is 4.5uS, operation current is 1.2uA. 01 = Max propagation delay is 2uS, operation current is 3uA. 10 = Max propagation delay is 600nS, operation current is 10uA. 11 = Max propagation delay is 200nS, operation current is 75uA.
[27:26]	**Reserved**	Reserved.
[25:24]	**HYSSEL**	**Hysteresis Mode Selection** 00 = Hysteresis is 0mV. 01 = Hysteresis is 10mV. 10 = Hysteresis is 20mV. 11 = Hysteresis is 30mV.
[23:19]	**Reserved**	Reserved.
[18]	**WCMPSEL**	**Window Compare Mode Selection** 0 = Window Compare Mode Disabled. 1 = Window Compare Mode Selected.
[17]	**WLATEN**	**Window Latch Mode Enable Bit** 0 = Window Latch Mode Disabled. 1 = Window Latch Mode Enabled.
[16]	**WKEN**	**Power-down Wake-up Enable Bit** 0 = Wake-up function Disabled. 1 = Wake-up function Enabled.

Bits	Description	
[15:13]	**FILTSEL**	**Comparator Output Filter Count Selection** 000 = Filter function is Disabled. 001 = ACMP1 output is sampled 1 consecutive PCLK. 010 = ACMP1 output is sampled 2 consecutive PCLKs. 011 = ACMP1 output is sampled 4 consecutive PCLKs. 100 = ACMP1 output is sampled 8 consecutive PCLKs. 101 = ACMP1 output is sampled 16 consecutive PCLKs. 110 = ACMP1 output is sampled 32 consecutive PCLKs. 111 = ACMP1 output is sampled 64 consecutive PCLKs.
[12]	**OUTSEL**	**Comparator Output Select** 0 = Comparator 1 output to ACMP1_O pin is unfiltered comparator output. 1 = Comparator 1 output to ACMP1_O pin is from filter output.
[11:10]	**Reserved**	Reserved.
[9:8]	**INTPOL**	**Interrupt Condition Polarity Selection** ACMPIF1 will be set to 1 when comparator output edge condition is detected. 00 = Rising edge or falling edge. 01 = Rising edge. 10 = Falling edge. 11 = Reserved.
[7:6]	**POSSEL**	**Comparator Positive Input Selection** 00 = Input from ACMP1_P0. 01 = Input from ACMP1_P1. 10 = Input from ACMP1_P2. 11 = Input from ACMP1_P3.
[5:4]	**NEGSEL**	**Comparator Negative Input Selection** 00 = ACMP1_N pin. 01 = Internal comparator reference voltage (CRV). 10 = Band-gap voltage. 11 = DAC output.
[3]	**ACMPOINV**	**Comparator Output Inverse Control** 0 = Comparator 1 output inverse Disabled. 1 = Comparator 1 output inverse Enabled.
[2]	**Reserved**	Reserved.
[1]	**ACMPIE**	**Comparator Interrupt Enable Bit** 0 = Comparator 1 interrupt Disabled. 1 = Comparator 1 interrupt Enabled. If WKEN (ACMP_CTL1[16]) is set to 1, the wake-up interrupt function will be enabled as well.
[0]	**ACMPEN**	**Comparator Enable Bit** 0 = Comparator 1 Disabled. 1 = Comparator 1 Enabled.

Analog Comparator Status Register (ACMP_STATUS)

Register	Offset	R/W	Description	Reset Value
ACMP_STATUS	ACMP01_BA+0x08	R/W	Analog Comparator Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							ACMPWO
15	14	13	12	11	10	9	8
Reserved		ACMPS1	ACMPS0	Reserved		WKIF1	WKIF0
7	6	5	4	3	2	1	0
Reserved		ACMPO1	ACMPO0	Reserved		ACMPIF1	ACMPIF0

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**ACMPWO**	**Comparator Window Output** This bit shows the output status of window compare mode 0 = The positive input voltage is outside the window. 1 = The positive input voltage is in the window.
[15:14]	**Reserved**	Reserved.
[13]	**ACMPS1**	**Comparator 1 Status** Synchronized to the PCLK to allow reading by software. Cleared when the comparator 1 is disabled, i.e. ACMPEN (ACMP_CTL1[0]) is cleared to 0.
[12]	**ACMPS0**	**Comparator 0 Status** Synchronized to the PCLK to allow reading by software. Cleared when the comparator 0 is disabled, i.e. ACMPEN (ACMP_CTL0[0]) is cleared to 0.
[11:10]	**Reserved**	Reserved.
[9]	**WKIF1**	**Comparator 1 Power-down Wake-up Interrupt Flag** This bit will be set to 1 when ACMP1 wake-up interrupt event occurs. 0 = No power-down wake-up occurred. 1 = Power-down wake-up occurred. **Note:** Write 1 to clear this bit to 0.
[8]	**WKIF0**	**Comparator 0 Power-down Wake-up Interrupt Flag** This bit will be set to 1 when ACMP0 wake-up interrupt event occurs. 0 = No power-down wake-up occurred. 1 = Power-down wake-up occurred. **Note:** Write 1 to clear this bit to 0.
[7:6]	**Reserved**	Reserved.

Bits	Description	
[5]	**ACMPO1**	**Comparator 1 Output** Synchronized to the PCLK to allow reading by software. Cleared when the comparator 1 is disabled, i.e. ACMPEN (ACMP_CTL1[0]) is cleared to 0.
[4]	**ACMPO0**	**Comparator 0 Output** Synchronized to the PCLK to allow reading by software. Cleared when the comparator 0 is disabled, i.e. ACMPEN (ACMP_CTL0[0]) is cleared to 0.
[3:2]	**Reserved**	Reserved.
[1]	**ACMPIF1**	**Comparator 1 Interrupt Flag** This bit is set by hardware when the edge condition defined by INTPOL (ACMP_CTL1[9:8]) is detected on comparator 1 output. This will cause an interrupt if ACMPIE (ACMP_CTL1[1]) is set to 1. **Note:** Write 1 to clear this bit to 0.
[0]	**ACMPIF0**	**Comparator 0 Interrupt Flag** This bit is set by hardware when the edge condition defined by INTPOL (ACMP_CTL0[9:8]) is detected on comparator 0 output. This will generate an interrupt if ACMPIE (ACMP_CTL0[1]) is set to 1. **Note:** Write 1 to clear this bit to 0.

ACMP Reference Voltage Control Register (ACMP_VREF)

Register	Offset	R/W	Description	Reset Value
ACMP_VREF	ACMP01_BA+0x0C	R/W	Analog Comparator Reference Voltage Control Register	0x0000_0000

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

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**CRVSSEL**	**CRV Source Voltage Selection** 0 = AV_{DD} is selected as CRV source voltage. 1 = The reference voltage defined by SYS_VREFCTL register is selected as CRV source voltage.
[5:4]	**Reserved**	Reserved.
[3:0]	**CRVCTL**	**Comparator Reference Voltage Setting** CRV = CRV source voltage * (1/6+CRVCTL/24).

6.44 Peripherals Interconnection

6.44.1 Overview

Some peripherals have interconnections which allow autonomous communication or synchronous action between peripherals without needing to involve the CPU. Peripherals interact without CPU saves CPU resources, reduces power consumption, operates with no software latency and fast response.

6.44.2 Peripherals Interconnect Matrix Table

Note: These figures in this table represent the following responding sections to describe the detail. For instance, section 6.44.3.1 describes the detail of figure 1 and section 6.44.3.2 describes the detail of figure 2.

Source	Destination									
	EADC	ACMP	DAC	Timer0~3	Timer4~5	BPWM	EPWM	IRCTRIM	QEI	ECAP
EADC			-							
ACMP			-	5, 6	5, 6		13			
DAC		3								
ACMP_CRV		3								
VBG	2	3								
Temp sensor	2									
V_{BAT}	2									
Internal V_{REF}										
LXT				5	5			14		
HXT				5	5					
HIRC				5	5					
LIRC32K				5	5					
MIRC				5	5					
Timer0~3	1		4	7		10	10		16	
Timer4~5	1		4		7	10	10			
BPWM	1					11,	11,			
EPWM	1		4			11,	11,12			
USB Device				8	8			14		
BOD				6	6		9			
CPU Lockup				6	6		9			
Clock fail				6	6		9			
SRAM parity error				6	6		9			

QEI										15

Table 6.44-1 Peripherals Interconnect Matrix Table

6.44.3 Functional Description

6.44.3.1 From BPWM/EPWM and Timer to EADC

BPWM/EPWM can be one of the EADC conversion trigger source.

Setting the EADC external hardware trigger input source from BPWM/EPWM trigger is described in section 6.37.5.8.

The detailed BPWM trigger conditions are described in section 6.16.5.16 .

The detailed EPWM trigger conditions are described in section 6.15.5.27.

Timer Trigger EADC Conversion

Timer0 ~ Timer5 can be one of the EADC conversion trigger source. When timer counter value matches the timer compared value or when the TMx_EXT pin edge transition meets setting, timer will trigger the EADC to start the conversion.

Setting the EADC external hardware trigger input source from timer trigger is described in section 6.37.5.7.

The detailed Timer trigger conditions are described in section 6.9.5.10 .

6.44.3.2 VBG, Temperature sensor and V_{BAT} as EADC input

EADC supports 3 internal channels, band-gap voltage (V_{BG}), temperature sensor (V_{TEMP}), and Battery power (V_{BAT}).

6.44.3.3 VBG, DAC and ACMP_CRV as ACMP input

ACMP supports 3 internal inputs, Internal band-gap voltage (VBG), Comparator Reference Voltage (CRV) and DAC0 output (DAC0_OUT).

6.44.3.4 From TIMER and EPWM to DAC

Timer Trigger DAC Conversion

Timer0 ~ Timer3 can be one of the DAC conversion trigger source. When timer counter value matches the timer compared value or when the TMx_EXT pin edge transition meets setting, timer will trigger the DAC to start the conversion.

Setting the DAC external hardware trigger input source from timer trigger is described in section 6.42.5.6 .

The detailed Timer trigger conditions are described in section 6.9.5.10.

From EPWM to DAC

EPWM can also be used to trigger DAC conversion.

Setting the DAC hardware trigger input source from EPWM trigger is described in section .

The detailed EPWM trigger conditions are described in section 6.15.5.27.

6.44.3.5 From ACMP and clocks to Timer Capture Function

Sets the timer capture source from ACMP0/1 output signal or clocks(HXT, LXT, HIRC, LIRC, MIRC) and measures the time interval of the signal by using timer capture function..

The detail of time capture function setting is described in section 6.9.5.8 and 6.9.5.9 .

6.44.3.6 From ACMP, BOD, Clock Fail,SRAM Parity Error and CPU Lockup to Timer

Timer Brake Source

Timer0 ~ Timer5 brake sources can be ACMP0/1_O output signal or several different system fail conditions include clock fail, Brown-out detect, and Core lockup and SRAM Parity Error. When system fault, Timer brake signal generated, timer output will be set to protect the power switch controlled by Timer.

The detailed setting of Timer brake function is described in section 6.9.6.17 .

6.44.3.7 From Timer0/2/4 to Timer1/3/5

Inter-Timer Trigger Capture Mode

Timer0/2/4 will be forced in event counting mode, counting with external event, and will generate an internal signal (INTR_TMR_TRG) to trigger Timer1/3/5 start or stop counting. The Timer1/3/5 will be forced in capture mode and start/stop trigger-counting by Timer0/2/4 counter status.

The detail of inter-timer trigger capture mode is described in section 6.9.5.11 .

6.44.3.8 From USBD SOF to Timer0~Timer5

Timer0 ~ Timer5 supports event counting source from internal USB SOF signal.

6.44.3.9 From ACMP, BOD, Clock Fail,SRAM Parity Error and CPU Lockup to EPWM

Timer Brake Source

EPWM brake sources can be ACMP0/1_O output signal or several different system fail conditions include clock fail, Brown-out detect, and Core lockup and SRAM Parity Error. When system fault, Timer brake signal generated, timer output will be set to protect the power switch controlled by Timer.

The detailed setting of Timer brake function is described in section 6.15.5.23 .

6.44.3.10 From TIMER to BPWM/EPWM

Timer Generates Trigger Pulses as BPWM/EPWM External Clock Source

Timer0 ~ Timer5 can generate trigger pules as BPWM/EPWM external clock source.

When the timer counter value matches the timer compared value or when the TMx_EXT pin edge transition meets setting, timer can generate a trigger pulse by setting described in section 6.9.5.10.

The setting of BPWM clock source is described in section 6.15.3 .

The setting of EPWM clock source is described in section 6.16.3 .

6.44.3.11 From BPWM/EPWM to BPWM/EPWM

EPWM Synchronous Start Function

Select synchronous source from EPWM0 or EPWM1 or BPWM0 or BPWM1, and select EPWM channels. The chosen EPWM channels will start counting at the same time once the synchronous start function is enabled and CNTSEN(EPWM_SSTRG[0]) is set.

The detailed setting of EPWM synchronous start function is described in section 6.15.5.19.

BPWM Synchronous Start Function

Select synchronous source from EPWM0 or EPWM1 or BPWM0 or BPWM1, and select BPWM channels. The chosen BPWM channels will start counting at the same time once the synchronous start function is enabled and CNTSEN(BPWM_SSTRG[0]) is set.

The detailed setting of BPWM synchronous start function is described in section 6.16.5.11.

6.44.3.12 From SYNC_IN to EPWM

The SYNC_IN signal for the first EPWM0 pair counter comes from EPWM0_SYNC_IN pin, and the others come from the SYNC_OUT signal of the previous EPWM pair counter.

The detailed setting of time capture function is described in section 6.15.5.19 .

6.44.3.13 From ACMP to EPWM LEB

Support Leading Edge Blanking (LEB) function for brake source from analog comparaton.

The detailed setting of EPWM Leading Edge Blanking (LEB) function is described in section 6.15.5.24 .

6.44.3.14 From Use LXT or USB Synchronous Mode to System Auto-trim HIRC Circuit

This chip supports auto-trim function: the HIRC trim (12 MHz RC oscillator) and RC 48 MHz oscillator, according to the accurate external 32.768 kHz crystal oscillator or internal USB synchronous mode, automatically gets accurate output frequency, 0.25 % deviation within all temperature ranges.

The detail of HIRC trim setting is described in section 6.2.9 .

6.44.3.15 From QEI to ECAP

ECAP Input Noise Filter

The architecture of ECAP input noise filter is similar to that one used for QEI. With 6 sampling-rate options, it supports a wide range of filtering noise, the duration of filtered noise and the duration of the signal that is guaranteed to be sampled.

The detailed setting of modulation is described in section 6.18.5.1 .

6.44.3.16 From Timer to QEI

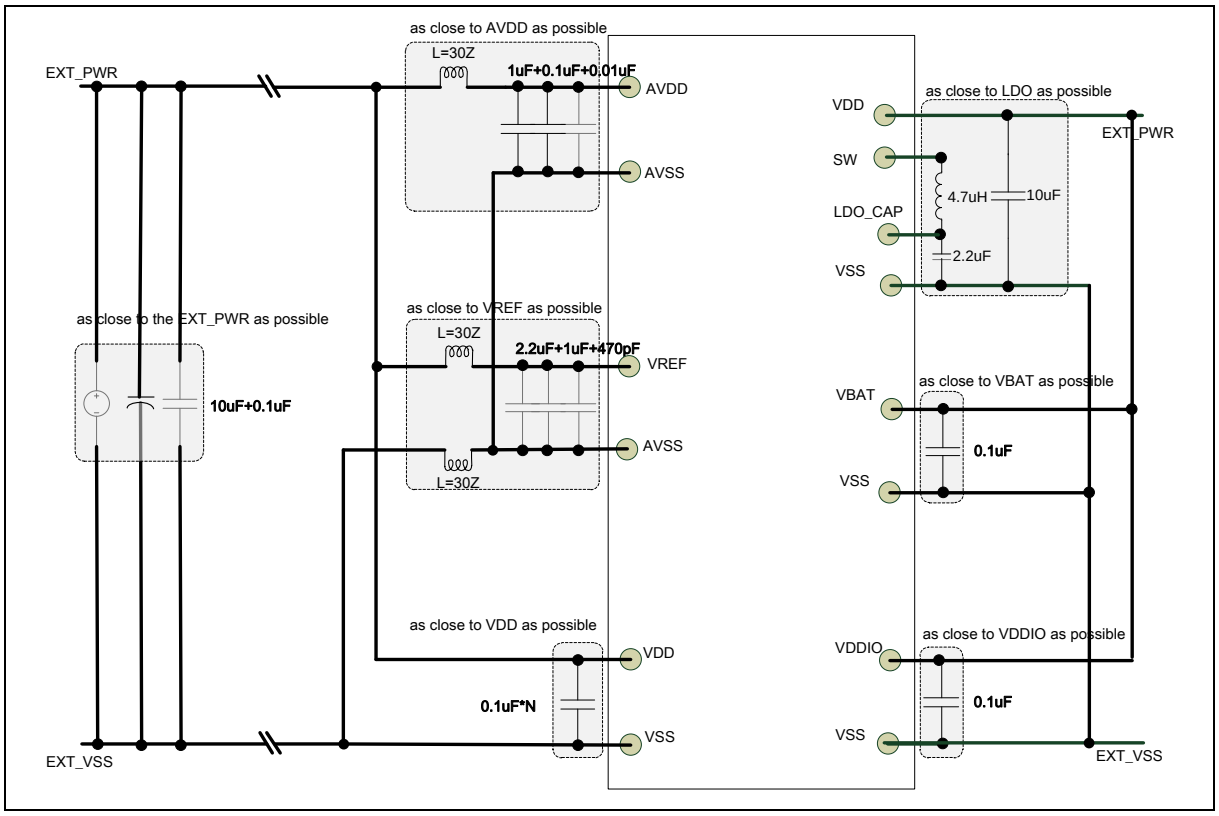
TIMER TIF Event to QEI

When QEI bit HOLDCNT(QEI_CTL[24]) set, the CNT(QEI_CNT[31:0]) content will be captured into QEI Counter Hold Register CNTHOLD(QEI_CNTHOLD[31:0]), the data will be held until the next HOLDCNT (QEI_CTL[24]) trigger comes. The bit HOLDCNT can be set by writing 1 to it or the rising edge of timers interrupt flags TIF (TIMERx_INTSTS[0])The detailed setting of modulation is described in section 6.17.5.11 .

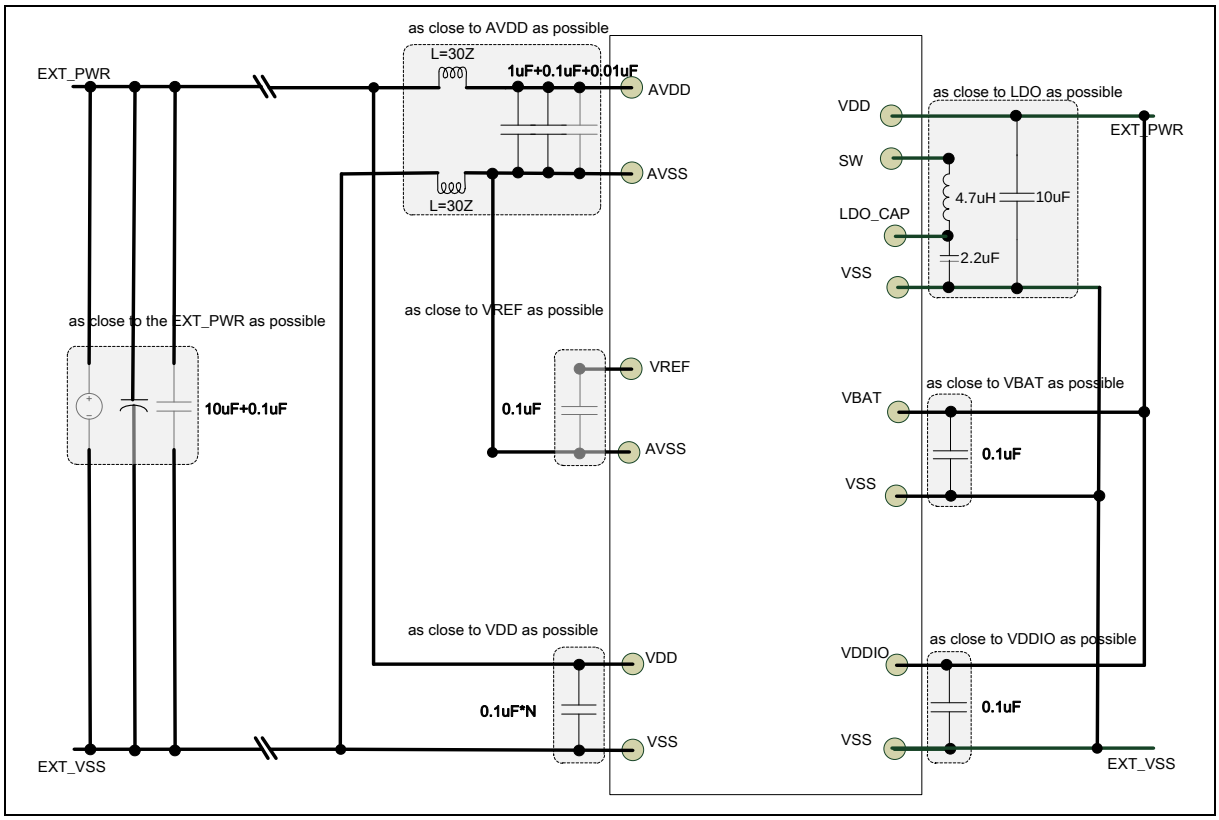
The detailed setting of modulation is described in section 6.9.5.1.

7 APPLICATION CIRCUIT

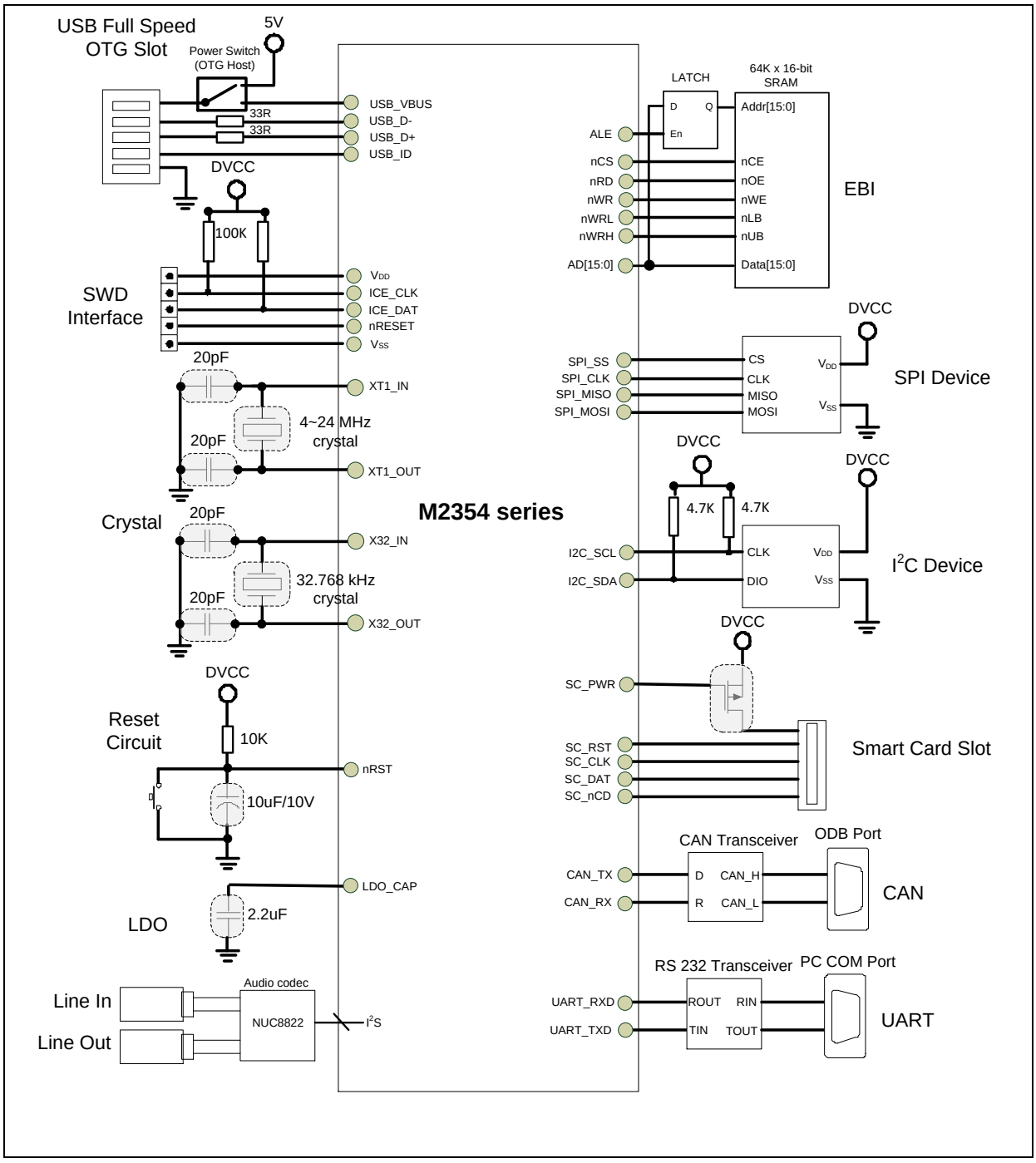
7.1 Power supply scheme with External V_{REF}



7.2 Power supply scheme with internal V_{REF}



7.3 Peripheral Application scheme



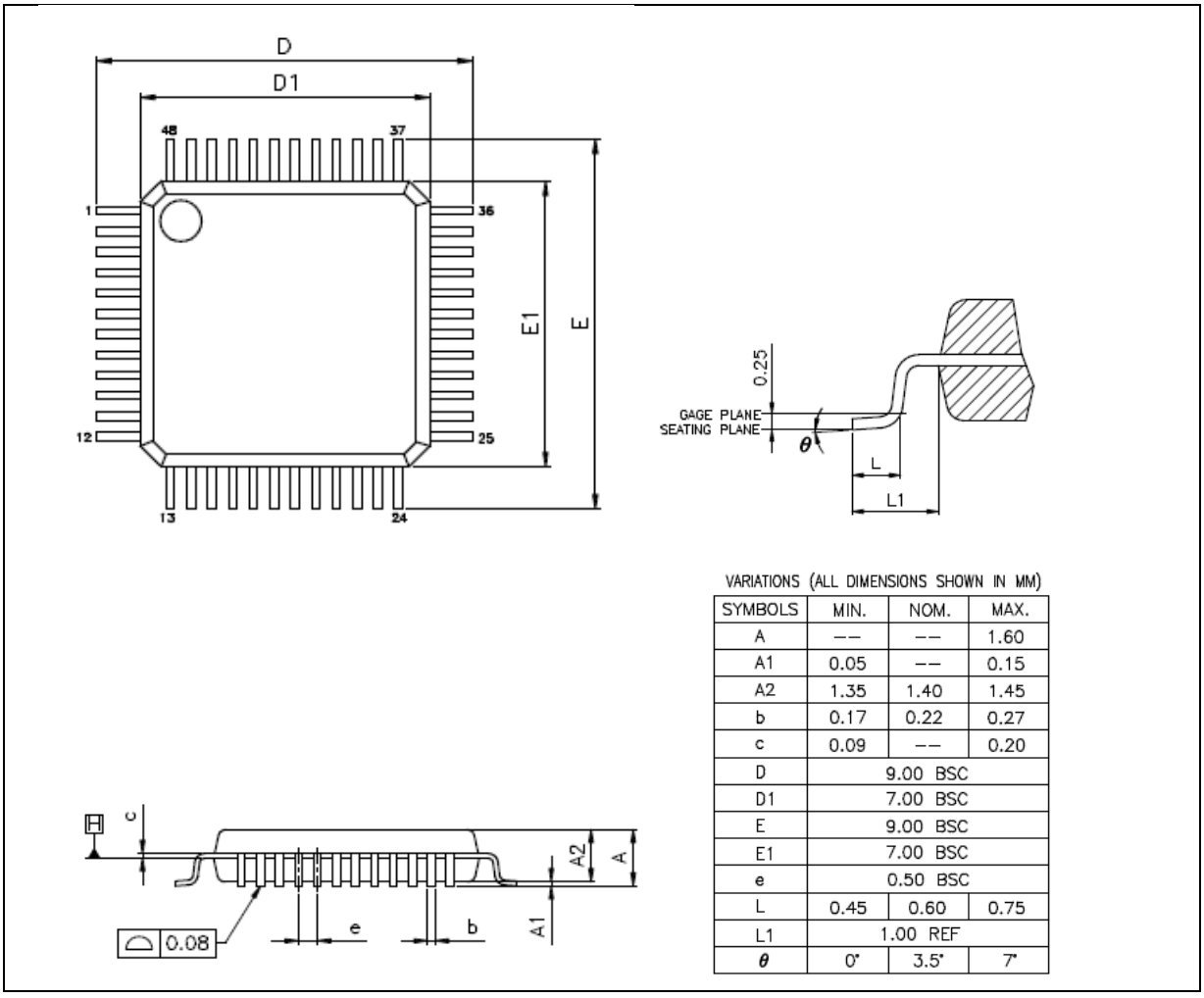
***Note:** USB_ID, HSUSB _ID could be floating using USB or USB HS without OTG.

8 ELECTRICAL CHARACTERISTIC

Please refer to the relative Datasheet for detailed information about the M2354 electrical characteristics.

9 PACKAGE DIMENSIONS

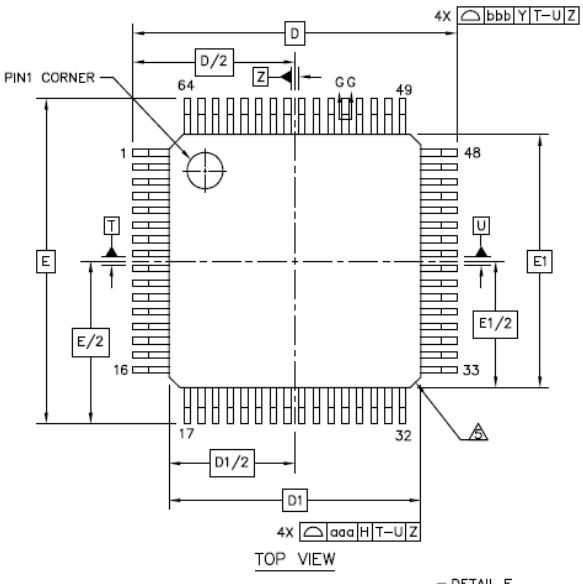
9.1 LQFP 48 (7x7x1.4 mm^3 Footprint 2.0 mm)



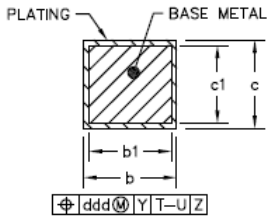
VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

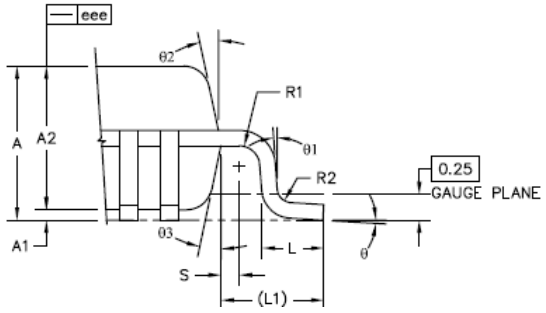
SYMBOLS	MIN.	NOM.	MAX.
A	--	--	1.60
A1	0.05	--	0.15
A2	1.35	1.40	1.45
b	0.17	0.22	0.27
c	0.09	--	0.20
D	9.00 BSC		
D1	7.00 BSC		
E	9.00 BSC		
E1	7.00 BSC		
e	0.50 BSC		
L	0.45	0.60	0.75
L1	1.00 REF		
θ	0°	3.5°	7°

9.2 LQFP 64 (7x7x1.4 mm^3 Footprint 2.0 mm)

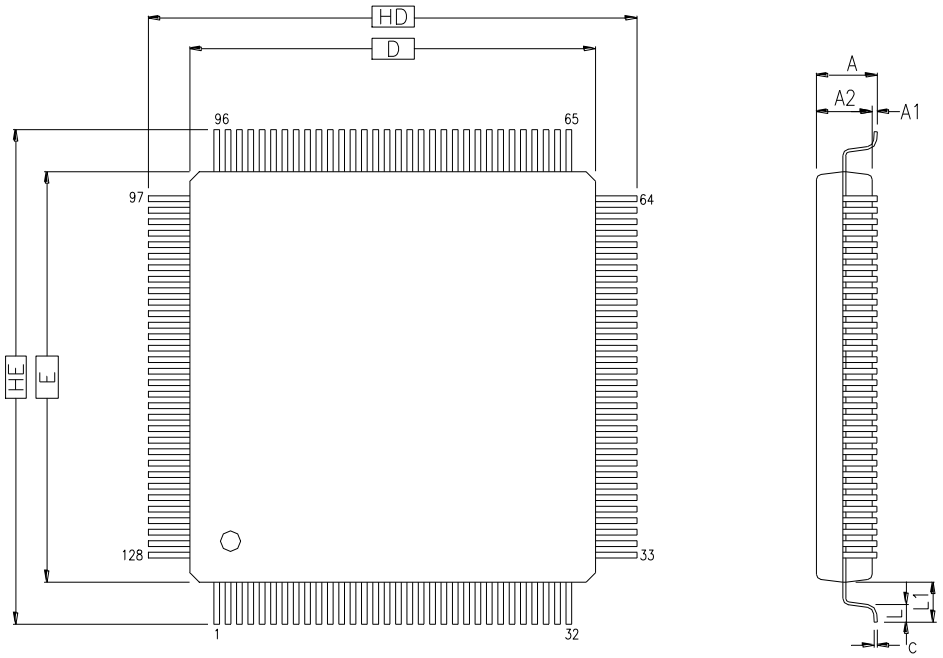


		SYMBOL	MIN	NOM	MAX
TOTAL THICKNESS		A	---	---	1.6
STAND OFF		A1	0.05	---	0.15
MOLD THICKNESS		A2	1.35	1.4	1.45
LEAD WIDTH(PLATING)		b	0.13	0.18	0.23
LEAD WIDTH		b1	0.13	0.16	0.19
L/F THICKNESS(PLATING)		c	0.09	---	0.2
L/F THICKNESS		c1	0.09	---	0.16
BODY SIZE	X	D	9 BSC		
	Y	E	9 BSC		
	X	D1	7 BSC		
	Y	E1	7 BSC		
LEAD PITCH		e	0.4 BSC		
		L	0.45	0.6	0.75
FOOTPRINT		L1	1 REF		
		θ	0°	3.5°	7°
		θ1	0°	---	---
		θ2	11°	12°	13°
		θ3	11°	12°	13°
		R1	0.08	---	---
		R2	0.08	---	0.2
		S	0.2	---	---
PACKAGE EDGE TOLERANCE		aaa	0.2		
LEAD EDGE TOLERANCE		bbb	0.2		
COPLANARITY		ccc	0.08		
LEAD OFFSET		ddd	0.07		
MOLD FLATNESS		eee	0.05		





9.3 LQFP 128 (14x14x1.4 mm^3 Footprint 2.0 mm)



COTROL DIMENSIONS ARE IN MILLIMETERS.

SYMBOL	MILLIMETER			INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	—	—	1.60	—	—	0.063
A_1	0.05	—	0.15	0.002	—	0.006
A_2	1.35	1.40	1.45	0.053	0.055	0.057
HD	16.00 BSC.			0.630 BSC.		
D	14.00 BSC.			0.551 BSC.		
HE	16.00 BSC.			0.630 BSC.		
E	14.00 BSC.			0.551 BSC.		
b	0.13	0.16	0.23	0.005	0.006	0.009
e	0.40 BSC.			0.016 BSC.		
θ	0°	3.5°	7°	0°	3.5°	7°
c	0.09	—	0.20	0.004	—	0.008
L	0.45	0.60	0.75	0.018	0.024	0.030
L_1	1.00 REF			0.039 REF		
y	–	–	0.1	–	–	0.004

10 ABBREVIATIONS

10.1 Abbreviations

Acronym	Description
ACMP	Analog Comparator Controller
ADC	Analog-to-Digital Converter
AES	Advanced Encryption Standard
APB	Advanced Peripheral Bus
AHB	Advanced High-Performance Bus
BOD	Brown-out Detection
CAN	Controller Area Network
DAP	Debug Access Port
DES	Data Encryption Standard
EADC	Enhanced Analog-to-Digital Converter
EBI	External Bus Interface
EMAC	Ethernet MAC Controller
EPWM	Enhanced Pulse Width Modulation
FIFO	First In, First Out
FMC	Flash Memory Controller
FPU	Floating-point Unit
GPIO	General-Purpose Input/Output
HCLK	The Clock of Advanced High-Performance Bus
HIRC	12 MHz Internal High Speed RC Oscillator
HXT	4~24 MHz External High Speed Crystal Oscillator
IAP	In Application Programming
ICP	In Circuit Programming
ISP	In System Programming
LDO	Low Dropout Regulator
LIN	Local Interconnect Network
LIRC	10 kHz internal low speed RC oscillator (LIRC)
MPU	Memory Protection Unit
NVIC	Nested Vectored Interrupt Controller
PCLK	The Clock of Advanced Peripheral Bus
PDMA	Peripheral Direct Memory Access
PLL	Phase-Locked Loop

PWM	Pulse Width Modulation
QEI	Quadrature Encoder Interface
SD	Secure Digital
SPI	Serial Peripheral Interface
SPS	Samples per Second
TDES	Triple Data Encryption Standard
TK	Touch Key
TMR	Timer Controller
UART	Universal Asynchronous Receiver/Transmitter
UCID	Unique Customer ID
USB	Universal Serial Bus
WDT	Watchdog Timer
WWDT	Window Watchdog Timer

Table 10.1-1 List of Abbreviations

11 REVISION HISTORY

Date	Revision	Description
2020.12.25	1.00	Initial version.
2021.09.13	1.01	1. Added internal reference voltage function in chapter 2 and section 6.37.2, 6.42.2 and6.42.5.2. 2. Added power loss detector to indicate power status of VBAT in section 6.41 and 6.41.5.5 3. Added 6.41.5.2 Watchdog Condition. 4. Revised XOM function features in section 6.6.4.7. 5. Added RTC_TEST register in section 6.14.6 and 6.14.7.