**ARM926EJ-S Based**

**32-bit Microcontroller**

# NUC970 Series
# Technical Reference Manual

*The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.*

*Nuvoton is providing this document only for reference purposes of NuMicro microcontroller based system design. Nuvoton assumes no responsibility for errors or omissions.*

*All data and specifications are subject to change without notice.*

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

www.nuvoton.com



Publication Release Date: Dec. 15, 2015



Revision V1.30

**TABLE OF CONTENTS**

TABLE OF CONTENTS ..... 3
LIST OF FIGURES ..... 12
LIST OF TABLES ..... 17
1 GENERAL DESCRIPTION ..... 18
2 FEATURES ..... 19
2.1 NUC970 Series Features ..... 19
3 PARTS INFORMATION LIST AND PIN CONFIGURATION ..... 27
3.1 NUC970 Series Part Number Naming Guide ..... 27
3.2 NUC970 Series Part Selection Guide ..... 28
3.3 Pin Configuration ..... 29
3.3.1 NUC972DFxxY Pin Diagram ..... 29
3.3.2 NUC976DKxxY Pin Diagram ..... 30
3.3.3 NUC977DKxxY Pin Diagram ..... 31
3.4 Pin Description ..... 32
4 BLOCK DIAGRAM ..... 58
4.1 NUC970 Series Block Diagram ..... 58
5 FUNCTIONAL DESCRIPTION ..... 59
5.1 ARM® ARM926EJ-S CPU Core ..... 59
5.1.1 Overview ..... 59
5.1.2 System Control Coprocessor (CP15) ..... 60
5.1.3 Memory Management Unit (MMU) ..... 60
5.1.4 Caches and Write Buffer ..... 61
5.1.5 Bus Interface Unit ..... 62
5.1.6 Power Management ..... 62
5.2 System Manager ..... 63
5.2.1 Overview ..... 63
5.2.2 System Reset ..... 63
5.2.3 System Power Distribution ..... 64
5.2.4 System Memory Map ..... 65
5.2.5 Power-On Setting ..... 69
5.2.6 Register Map ..... 71
5.2.7 Register Description ..... 73
5.3 Clock Controller (CLK_CTL) ..... 139

5.3.1 Overview ..... 139
5.3.2 Features ..... 139
5.3.3 Block diagram ..... 140
5.3.4 Functional description ..... 149
5.3.5 Registers Map ..... 151
5.3.6 Register description ..... 152
5.4 Advanced Interrupt Controller (AIC) ..... 186
5.4.1 Overview ..... 186
5.4.2 Features ..... 186
5.4.3 Block Diagram ..... 187
5.4.4 Functional Description ..... 187
5.4.5 Register Map ..... 192
5.4.6 Register Description ..... 194
5.5 SDRAM Interface Controller (SDIC) ..... 217
5.5.1 Overview ..... 217
5.5.2 Features ..... 217
5.5.3 Block Diagram ..... 218
5.5.4 Basic Configuration ..... 219
5.5.5 Functional Description ..... 219
5.5.6 Register Map ..... 223
5.5.7 Register Description ..... 223
5.6 MTP Controller (MTP) ..... 246
5.6.1 Overview ..... 246
5.6.2 Features ..... 246
5.6.3 Block Diagram ..... 247
5.6.4 Basic Configuration ..... 248
5.6.5 Functional Description ..... 249
5.6.6 Software Programming Flow ..... 250
5.6.7 Register Map ..... 251
5.6.8 Register Description ..... 252
5.7 External Bus Interface (EBI) ..... 261
5.7.1 Overview ..... 261
5.7.2 Features ..... 261
5.7.3 Block Diagram ..... 261
5.7.4 Basic Configuration ..... 262
5.7.5 Functional Description ..... 262

5.7.6 Register Map ..... 263
5.7.7 Register Description ..... 264
5.8 General Purpose I/O (GPIO) ..... 272
5.8.1 Overview ..... 272
5.8.2 Features ..... 272
5.8.3 Block Diagram ..... 273
5.8.4 Basic Configuration ..... 273
5.8.5 Functional Description ..... 273
5.8.6 Register Map ..... 274
5.8.7 Register Description ..... 279
5.9 General DMA Controller (GDMA) ..... 299
5.9.1 Overview ..... 299
5.9.2 Features ..... 299
5.9.3 Block Diagram ..... 299
5.9.4 Basic Configuration ..... 300
5.9.5 Functional Description ..... 300
5.9.6 Register Map ..... 304
5.9.7 Register Description ..... 305
5.10 Timer Controller (TMR) ..... 322
5.10.1 Overview ..... 322
5.10.2 Features ..... 322
5.10.3 Block Diagram ..... 323
5.10.4 Basic Configuration ..... 323
5.10.5 Functional Description ..... 323
5.10.6 Register Map ..... 325
5.10.7 Register Description ..... 326
5.11 Enhance Timer Controller (ETMR) ..... 332
5.11.1 Overview ..... 332
5.11.2 Features ..... 332
5.11.3 Block Diagram ..... 332
5.11.4 Basic Configuration ..... 333
5.11.5 Functional Description ..... 333
5.11.6 Register Map ..... 336
5.11.7 Register Description ..... 336
5.12 Pulse Width Modulation (PWM) ..... 347
5.12.1 Overview ..... 347

5.12.2 Features ..... 347
5.12.3 Block Diagram ..... 348
5.12.4 Basic Configuration ..... 348
5.12.5 Functional Description ..... 348
5.12.6 Register Map ..... 352
5.12.7 Register Description ..... 353
5.13 Watchdog Timer (WDT) ..... 363
5.13.1 Overview ..... 363
5.13.2 Features ..... 363
5.13.3 Block Diagram ..... 363
5.13.4 Basic Configuration ..... 363
5.13.5 Functional Description ..... 364
5.13.6 Register Map ..... 367
5.13.7 Register Description ..... 368
5.14 Windowed Watchdog Timer (WWDT) ..... 372
5.14.1 Overview ..... 372
5.14.2 Features ..... 372
5.14.3 Block Diagram ..... 372
5.14.4 Basic Configuration ..... 372
5.14.5 Function Description ..... 373
5.14.6 Register Map ..... 377
5.14.7 Register Description ..... 378
5.15 Real Time Clock (RTC) ..... 384
5.15.1 Overview ..... 384
5.15.2 Features ..... 384
5.15.3 Block Diagram ..... 385
5.15.4 Basic Configuration ..... 385
5.15.5 Functional Description ..... 385
5.15.6 Register Map ..... 387
5.15.7 Register Description ..... 389
5.16 UART Interface Controller (UART) ..... 411
5.16.1 Overview ..... 411
5.16.2 Features ..... 412
5.16.3 Block Diagram ..... 413
5.16.4 Basic Configuration ..... 414
5.16.5 Functional Description ..... 414

5.16.6 Register Map ..... 443
5.16.7 Register Description ..... 445
5.17 Smart Card Host Interface (SC) ..... 474
5.17.1 Overview ..... 474
5.17.2 Features ..... 474
5.17.3 Block Diagram ..... 475
5.17.4 Basic Configuration ..... 476
5.17.5 Functional description ..... 476
5.17.6 Register Map ..... 486
5.17.7 Register Description ..... 488
5.18 I2C Synchronous Serial Interface Controller (I2C) ..... 516
5.18.1 Overview ..... 516
5.18.2 Features ..... 516
5.18.3 Block Diagram ..... 517
5.18.4 Basic Configuration ..... 517
5.18.5 Functional Description ..... 518
5.18.6 Register Map ..... 526
5.18.7 Register Description ..... 527
5.19 SPI Interface Controller (SPI) ..... 535
5.19.1 Overview ..... 535
5.19.2 Features ..... 535
5.19.3 Block Diagram ..... 536
5.19.4 Basic Configuration ..... 536
5.19.5 Function Description ..... 537
5.19.6 Registers Map ..... 542
5.19.7 Register Description ..... 543
5.20 I2S Controller (I2S) ..... 550
5.20.1 Overview ..... 550
5.20.2 Features ..... 550
5.20.3 Block Diagram ..... 551
5.20.4 Basic Configuration ..... 551
5.20.5 Functional Description ..... 552
5.20.6 Register Map ..... 555
5.20.7 Register Description ..... 556
5.21 Ethernet MAC Controller (EMAC) ..... 581
5.21.1 Overview ..... 581

5.21.2 Features ..... 581
5.21.3 Block Diagram ..... 582
5.21.4 Basic Configuration ..... 582
5.21.5 Functional Description ..... 583
5.21.6 DMA Descriptors Data Structure ..... 586
5.21.7 Register Map ..... 599
5.21.8 Register Description ..... 603
5.22 USB 2.0 Device Controller (USBD) ..... 656
5.22.1 Overview ..... 656
5.22.2 Features ..... 656
5.22.3 Block Diagram ..... 657
5.22.4 Basic Configuration ..... 657
5.22.5 Functional Description ..... 657
5.22.6 Registers Map ..... 660
5.22.7 Register Description ..... 665
5.23 USB Host Controller (USBH) ..... 717
5.23.1 Overview ..... 717
5.23.2 Features ..... 717
5.23.3 Block Diagram ..... 718
5.23.4 Basic Configuration ..... 718
5.23.5 Functional Description ..... 718
5.23.6 Register Map ..... 721
5.23.7 Register Description ..... 723
5.24 Controller Area Network (CAN) ..... 771
5.24.1 Overview ..... 771
5.24.2 Features ..... 771
5.24.3 Block Diagram ..... 772
5.24.4 Basic Configuration ..... 773
5.24.5 Functional Description ..... 773
5.24.6 Test Mode ..... 774
5.24.7 CAN Communications ..... 777
5.24.8 Register Map ..... 795
5.24.9 Register Description ..... 801
5.25 Flash Memory Interface (FMI) ..... 836
5.25.1 Overview ..... 836
5.25.2 Features ..... 836

5.25.3 Block Diagram ..... 837
5.25.4 Basic Configuration ..... 837
5.25.5 Functional Description ..... 838
5.25.6 Register Map ..... 843
5.25.7 Register Description ..... 846
5.26 Secure Digital Host Controller (SDH) ..... 907
5.26.1 Overview ..... 907
5.26.2 Features ..... 907
5.26.3 Block Diagram ..... 907
5.26.4 Basic Configuration ..... 908
5.26.5 Functional Description ..... 908
5.26.6 Registers Map ..... 910
5.26.7 Register Description ..... 910
5.27 Cryptographic Accelerator (CRYPTO) ..... 933
5.27.1 Overview ..... 933
5.27.2 Features ..... 933
5.27.3 Block Diagram ..... 934
5.27.4 Basic Configuration ..... 934
5.27.5 Functional description ..... 935
5.27.6 Register Map ..... 947
5.27.7 Register Description ..... 953
5.28 2D Graphic Engine (GE2D) ..... 1002
5.28.1 Overview ..... 1002
5.28.2 Features ..... 1002
5.28.3 Block Diagram ..... 1003
5.28.4 Basic Configuration ..... 1003
5.28.5 Functional Description ..... 1003
5.28.6 Register Map ..... 1008
5.28.7 Register Description ..... 1010
5.29 JPEG Codec (JPEG) ..... 1039
5.29.1 Overview ..... 1039
5.29.2 Features ..... 1039
5.29.3 Block Diagram ..... 1040
5.29.4 Basic Configuration ..... 1041
5.29.5 Functional Description ..... 1042
5.29.6 Register Map ..... 1047

5.29.7 Register Description ..... 1051
5.30 LCD Display Interface Controller (LCM) ..... 1105
5.30.1 Overview ..... 1105
5.30.2 Features ..... 1105
5.30.3 Block Diagram ..... 1106
5.30.4 Basic Configuration ..... 1106
5.30.5 Functional Description ..... 1107
5.30.6 Register Map ..... 1110
5.30.7 Register Description ..... 1112
5.31 Capture Sensor Interface Controller (CAP) ..... 1161
5.31.1 Overview ..... 1161
5.31.2 Feature ..... 1161
5.31.3 Block Diagram ..... 1162
5.31.4 Basic Configuration ..... 1162
5.31.5 Functional Description ..... 1163
5.31.6 Register Map ..... 1166
5.31.7 Register Description ..... 1167
5.32 Analog to Digital Converter (ADC) ..... 1198
5.32.1 Overview ..... 1198
5.32.2 Features ..... 1198
5.32.3 Block Diagram ..... 1199
5.32.4 Basic Configuration ..... 1200
5.32.5 Functional Description ..... 1200
5.32.6 Register Map ..... 1206
5.32.7 Register Description ..... 1207
5.33 Keypad Interface (KPI) ..... 1231
5.33.1 Overview ..... 1231
5.33.2 Features ..... 1231
5.33.3 Block Diagram ..... 1232
5.33.4 Basic Configuration ..... 1233
5.33.5 Functional Description ..... 1233
5.33.6 Register Map ..... 1235
5.33.7 Register Description ..... 1236
6 PACKAGE DIMENSIONS ..... 1248
6.1 LQFP 216L (24x24x1.4mm footprint 2.0mm) ..... 1248
6.2 LQFP 128L (14x14x1.4mm footprint 2.0mm) ..... 1249

7 REVISION HISTORY.....1250

## LIST OF FIGURES

Figure 3.1-1 NUC970 Series Part Number Naming Guide ..... 27
Figure 3.3-1 NUC972DFxxY LQFP 216-pin Pin Diagram ..... 29
Figure 3.3-2 NUC976DKxxY LQFP 128-pin Pin Diagram ..... 30
Figure 3.3-3 NUC977DKxxY LQFP 128-pin Pin Diagram ..... 31
Figure 4.1-1 NUC970 Series Block Diagram ..... 58
Figure 5.1-1 ARM926EJ-S Block Diagram ..... 60
Figure 5.2-1 NUC970 Series Power Distribution Diagram ..... 64
Figure 5.3-1 Clock Controller Block Diagram ..... 141
Figure 5.3-2 ADC Controller Clock Divider Block Diagram ..... 142
Figure 5.3-3 eMMC Host Controller Clock Divider Block Diagram ..... 142
Figure 5.3-4 Enhanced Timer Clock Divider Clock Diagram ..... 142
Figure 5.3-5 Ethernet MAC Controller Clock Divider Block Diagram ..... 143
Figure 5.3-6 GPIO Clock Divider Block Diagram ..... 143
Figure 5.3-7 $I^2S$ Controller Clock Divider Block Diagram ..... 143
Figure 5.3-8 KPI Controller Clock Divider Block Diagram ..... 144
Figure 5.3-9 LCD Display Controller Clock Divider Block Diagram ..... 144
Figure 5.3-10 Reference Clock Output Divider Block Diagram ..... 144
Figure 5.3-11 SD Card Host Controller Clock Divider Block Diagram ..... 145
Figure 5.3-12 Smart Card Host Controller Clock Divider Block Diagram ..... 145
Figure 5.3-13 CMOS Sensor Controller Divider Block Diagram ..... 145
Figure 5.3-14 UART Clock Divider Block Diagram ..... 146
Figure 5.3-15 USB 1.1 Host Controller 48 MHz Clock Divider Block Diagram ..... 146
Figure 5.3-16 Watchdog Timer Clock Divider Block Diagram ..... 146
Figure 5.3-17 Windowed Watchdog Timer Clock Divider Block Diagram ..... 147
Figure 5.3-18 CPU_HCLK Clock Generator Block Diagram ..... 148
Figure 5.4-1 Advanced Interrupt Controller (AIC) Block Diagram ..... 187
Figure 5.5-1 SDRAM Controller Block Diagram ..... 218
Figure 5.5-2 Clock Delay Circuit ..... 242
Figure 5.6-1 MTP Controller Block Diagram ..... 247
Figure 5.6-2 MTP INTERFACE PIN DESCRIPTION ..... 248
Figure 5.7-1 EBI Block Diagram ..... 261
Figure 5.7-2 External I/O Write operation timing ..... 271
Figure 5.7-3 External I/O Read operation timing ..... 271
Figure 5.8-1 GPIO Block Diagram ..... 273
Figure 5.9-1 GDMA Block Diagram ..... 299

Figure 5.10-1 Timer Controller Block Diagram ..... 323
Figure 5.11-1 Enhance Timer Controller Block Diagram ..... 332
Figure 5.11-2 Enhance Timer Clock Controller Diagram ..... 333
Figure 5.11-3 Timer Clock Controller Diagram ..... 335
Figure 5.12-1 Two channels of PWM in one pair ..... 348
Figure 5.12-2 Legend of Internal Comparator Output of PWM-Timer ..... 349
Figure 5.12-3 PWM Double Buffer Illustration ..... 349
Figure 5.12-4 PWM Controller Output Duty Ratio ..... 350
Figure 5.12-5 Paired PWM Output with Dead Zone Generation Operation ..... 350
Figure 5.13-1 Watchdog Timer Block Diagram ..... 363
Figure 5.13-2 Watchdog Timer Clock Control ..... 364
Figure 5.13-3 Watchdog Timer Time-out Interval and Reset Period Timing ..... 365
Figure 5.14-1 WWDT Block Diagram ..... 372
Figure 5.14-2 WWDT Clock Control ..... 372
Figure 5.14-3 WWDT Reset and Reload Behavior ..... 375
Figure 5.15-1 RTC Functional Block Diagram ..... 385
Figure 5.16-1 UART Block Diagram ..... 413
Figure 5.16-2 Auto Flow Control Block Diagram ..... 414
Figure 5.16-3 UART Line Control of Word and Stop Length Setting ..... 415
Figure 5.16-4 UART Line Control of Parity Bit Setting ..... 415
Figure 5.16-5 UART Auto-Flow Control Block Diagram ..... 417
Figure 5.16-6 UART CTSn Wake-Up Case 1 ..... 417
Figure 5.16-7 UART CTSn Wake-Up Case 2 ..... 418
Figure 5.16-8 IrDA Block Diagram ..... 418
Figure 5.16-9 IrDA TX/RX Timing Diagram ..... 421
Figure 5.16-10 RS-485 Frame Structure ..... 425
Figure 5.16-11 LIN Frame Structure ..... 426
Figure 5.16-12 Break Detection in LIN Mode ..... 429
Figure 5.16-13 Relationship between Break Detection and Frame Error Detection ..... 430
Figure 5.16-14 LIN Sync Field Measurement ..... 438
Figure 5.16-15 UA_BAUD Update Method ..... 439
Figure 5.17-1 SC Clock Control Diagram (4-bit Prescale Counter in Clock Controller) ..... 475
Figure 5.17-2 SC Controller Block Diagram ..... 475
Figure 5.17-3 SC Data Character ..... 476
Figure 5.17-4 SC Activation Sequence ..... 477
Figure 5.17-5 SC Warm Reset Sequence ..... 479

Figure 5.17-6 SC Deactivation Sequence ..... 480
Figure 5.17-7 Basic Operation Flow ..... 481
Figure 5.17-8 Initial Character TS ..... 482
Figure 5.17-9 SC Error Signal ..... 482
Figure 5.18-1 I²C Block Diagram ..... 517
Figure 5.18-2 I²C Bus Timing ..... 518
Figure 5.18-3 Data transfer on the I²C-bus ..... 518
Figure 5.18-4 START and STOP conditions ..... 520
Figure 5.18-5 Bit transfer on the I²C-bus ..... 521
Figure 5.18-6 Acknowledge on the I²C-bus ..... 521
Figure 5.18-7 Master Transmits Data to Slave ..... 521
Figure 5.18-8 Master Reads Data from Slave ..... 522
Figure 5.18-9 Write 1 byte of data to a slave ..... 523
Figure 5.18-10 Read a byte of data ..... 524
Figure 5.18-11 Implementation of Software I²C ..... 524
Figure 5.19-1 SPI Block Diagram ..... 536
Figure 5.19-2 Normal SPI Timing ..... 538
Figure 5.19-3 Alternate Phase SCLK Clock Timing ..... 539
Figure 5.19-4 Dual-IO output Sequence ..... 539
Figure 5.19-5 Dual-IO Input Sequence ..... 540
Figure 5.20-1 I²S Controller Block Diagram ..... 551
Figure 5.20-2 I²S Interface Signal of Master Mode ..... 552
Figure 5.20-3 I²S Interface Signal of Slave Mode ..... 552
Figure 5.20-4 I²S MSB-Justified Format ..... 553
Figure 5.20-5 PCM Mode Signal Interface ..... 553
Figure 5.20-6 PCM Mode Interface Waveform ..... 554
Figure 5.21-1 Ethernet MAC Controller Block Diagram ..... 582
Figure 5.21-2 Ethernet Frame Format ..... 584
Figure 5.21-3 64-bit Reference Timing Counter ..... 585
Figure 5.21-4 RXDMA Descriptor Data Structure ..... 586
Figure 5.21-5 TXDMA Descriptor Data Structure ..... 592
Figure 5.21-6 MII Management Frame Format ..... 621
Figure 5.22-1 USB Device Controller Block Diagram ..... 657
Figure 5.23-1 USB Host Controller Block Diagram ..... 718
Figure 5.24-1 CAN Peripheral Block Diagram ..... 772
Figure 5.24-2 CAN Core in Silent Mode ..... 775

Figure 5.24-3 CAN Core in Loop Back Mode ..... 775
Figure 5.24-4 CAN Core in Loop Back Mode Combined with Silent Mode ..... 776
Figure 5.24-5 Data transfer between IFn Registers and Message ..... 779
Figure 5.24-6 Application Software Handling of a FIFO Buffer ..... 784
Figure 5.24-7 Bit Timing ..... 786
Figure 5.24-8 Propagation Time Segment ..... 787
Figure 5.24-9 Synchronization on “late” and “early” Edges ..... 789
Figure 5.24-10 Filtering of Short Dominant Spikes ..... 790
Figure 5.24-11 Structure of the CAN Core’s CAN Protocol Controller ..... 791
Figure 5.25-1 FMI Block Diagram ..... 837
Figure 5.25-2 Data Arrangement for 2 kB Page Size NAND Flash ..... 841
Figure 5.25-3 Data Arrangement for 4 kB Page Size NAND Flash ..... 841
Figure 5.25-4 Data Arrangement for 8 kB Page Size NAND Flash ..... 842
Figure 5.25-5 PAD (Physical Address Descriptor) Table Format ..... 848
Figure 5.25-6 PAD (Physical Address Descriptor) Table Fetch Modes ..... 849
Figure 5.26-1 PAD (Physical Address Descriptor) Table Format ..... 912
Figure 5.26-2 PAD (Physical Address Descriptor) Table Fetch Modes ..... 913
Figure 5.27-1 Cryptographic Accelerator Block Diagram ..... 934
Figure 5.27-2 PRNG Function Diagram ..... 936
Figure 5.27-3 Electronic Codebook Mode ..... 937
Figure 5.27-4 Cipher Block Chaining Mode ..... 938
Figure 5.27-5 Cipher Feedback Mode ..... 939
Figure 5.27-6 Output Feedback Mode ..... 940
Figure 5.27-7 Counter Mode ..... 941
Figure 5.27-8 CBC-CS1 Encryption ..... 942
Figure 5.27-9 CBC-CS1 Decryption ..... 942
Figure 5.28-1 2-D GE Engine Architecture ..... 1003
Figure 5.28-2 BitBLT Direction ..... 1005
Figure 5.28-3 Bresenham Line Drawing Octant ..... 1006
Figure 5.28-4 Rotation Operation Octant ..... 1007
Figure 5.29-1 JPEG Codec block diagram (Encode) ..... 1040
Figure 5.29-2 JPEG Codec block diagram (Decode) ..... 1041
Figure 5.29-3 Image starting address and stride ..... 1043
Figure 5.29-4 Primary and thumbnail encode ..... 1044
Figure 5.29-5 Single mode and continue mode ..... 1045
Figure 5.29-6 Specified window decode mode ..... 1046

Figure 5.30-1 LCD Display Interface Controller Functional Block Diagram ..... 1106
Figure 5.30-2 Display and Overlay Control Timing ..... 1107
Figure 5.31-1 Image Capture Functional Block Diagram ..... 1162
Figure 5.32-1 ADC Functional Block Diagram ..... 1199
Figure 5.32-2 ADC Transfer Function ..... 1200
Figure 5.32-3 Simplified Sampling Diagram ..... 1201
Figure 5.32-4 ADC Battery Voltage Detection Diagram ..... 1202
Figure 5.32-5 Key Pad Detection Diagram ..... 1203
Figure 5.32-6 4-wire Touch Screen Connection Diagram ..... 1204
Figure 5.32-7 5-wire Touch Screen Connection Diagram ..... 1204
Figure 5.33-1 Keypad Controller Block Diagram ..... 1232
Figure 5.33-2 Keypad Connection ..... 1233
Figure 5.33-3 Keypad Interface ..... 1233
Figure 5.33-4 4x8 Keypad Scan Timing Diagram ..... 1234

## LIST OF TABLES

Table 5.2-1 Address Space Assignments for On-Chip Controllers ..... 68
Table 5.13-1 Watchdog Time-out Interval Period Selection ..... 365
Table 5.13-2 Watchdog Reset Period Selection ..... 365
Table 5.14-1 Window Watchdog Prescaler Value Selection ..... 373
Table 5.14-2 CMPDA Setting Limitation ..... 376
Table 5.16-1 Baud Rate Equation Table ..... 416
Table 5.16-2 Baud Rate Equation Table ..... 416
Table 5.16-3 UART Interrupt Sources and Flag List in Software Mode ..... 442
Table 5.17-1 SC Host Controller Pin Description ..... 476
Table 5.17-2 UART Pin Description ..... 476
Table 5.21-1 Arbiter Arbitration Results ..... 583
Table 5.21-2 Different CAMCMR Setting and Type of Received Packet ..... 606
Table 5.21-3 MII Management Function Configure Sequence ..... 621
Table 5.24-1 Initialization of a Transmit Object ..... 781
Table 5.24-2 Initialization of a Receive Object ..... 782
Table 5.24-3 CAN Bit Time Parameters ..... 786
Table 5.24-4 CAN Register Map for Each Bit Function ..... 800
Table 5.24-5 Error Code ..... 805
Table 5.24-6 Source of Interrupts ..... 808
Table 5.24-7 IF1 and IF2 Message Interface Register ..... 811
Table 5.24-8 Structure of a Message Object in the Message Memory ..... 825
Table 5.25-1 Number of Parity (Byte) for Each BCH Algorithm ..... 840

# 1 GENERAL DESCRIPTION

The NUC970 series targeted for general purpose 32-bit microcontroller embeds an outstanding CPU core ARM926EJ-S, a RISC processor designed by Advanced RISC Machines Ltd., runs up to 300 MHz, with 16 KB I-cache, 16 KB D-cache and MMU, 56KB embedded SRAM and 16 KB IBR (Internal Boot ROM) for booting from USB, NAND and SPI FLASH.

The NUC970 series integrates two 10/100 Mb Ethernet MAC controllers, USB 2.0 HS HOST/Device controller with HS transceiver embedded, TFT type LCD controller, CMOS sensor I/F controller, 2D graphics engine, DES/3DES/AES crypto engine, $I^2S$ I/F controller, SD/MMC/NAND FLASH controller, GDMA and 8 channels 12-bit ADC controller with resistance touch screen functionality. It also integrates UART, SPI/MICROWIRE, $I^2C$, CAN, LIN, PWM, Timer, WDT/Windowed-WDT, GPIO, Keypad, Smart Card I/F, 32.768 KHz XTL and RTC (Real Time Clock).

In addition, the NUC970 series integrates a DRAM I/F, that runs up to 150MHz with supporting DDR2 type SDRAM, and an External Bus Interface (EBI) that supports SRAM and external device with DMA request and ack.

# 2 FEATURES

## 2.1 NUC970 Series Features

- Core
  - ARM® ARM926EJ-S™ processor core runs up to 300 MHz
  - Support 16 KB instruction cache and 16 KB data cache
  - Support MMU
  - Support JTAG Debug interface
- External Bus Interface (EBI)
  - Support SRAM and external I/O devices
  - Support 8/16-bit data bus width
  - Support up to five chip selects for SRAM and external I/O devices
  - Support programmable access cycle
  - Support four 32-bit write buffers
- DDR SDRAM Controller
  - Support DDR, DDR2 and LPDDR SDRAM
  - Clock speed up to 150 MHz
  - Support 16-bit data bus width
  - Support two chip selects
  - Support total memory size up to 256M bytes (each chip select for 128M bytes)
- Embedded SRAM and ROM
  - Support 56K bytes embedded SRAM
  - Support 16K bytes Internal Boot ROM (IBR)
  - Support up to four booting modes
  - Boot from USB
  - Boot from eMMC
  - Boot from NAND Flash
  - Boot from SPI Flash
- Clock Control
  - Support two PLLs, up to 500 MHz, for high performance system operation
  - External 12 MHz high speed crystal input for precise timing operation
  - External 32.768 kHz low speed crystal input for RTC function and low speed clock source
- Ethernet MAC Controller
  - Support up to 2 Ethernet MAC controllers
  - Support IEEE Std. 802.3 CSMA/CD protocol
  - Support packet time stamping for IEEE Std. 1588 protocol
  - Support 10 and 100 Mbps operations
  - Support Half- and Full-duplex operations
  - Support RMII interface to Ethernet physical layer PHY
  - Support Ethernet physical layer PHY management through MDC and MDIO interface
  - Support flow control in Full-duplex mode to receive, recognize and transmit PAUSE frame
  - Support CAM-like function to recognize 48-bit Ethernet MAC address
  - Support Wake-On-LAN by detecting Magic Packet
  - Support 256 bytes transmit FIFO and 256 bytes receive FIFO
  - Support DMA function
  - Support internal loop back mode for diagnostic
- USB 2.0 Controller

- Support USB Revision 2.0 specification
- Support one set of USB 2.0 High-Speed (HS) Device/Host with embedded transceiver
- Support one set of USB 2.0 High-Speed (HS) Host with embedded transceiver
- Support Control, Bulk, Interrupt, Isochronous and Split transfers
- Support USB host function compliant to Enhanced Host Controller Interface (EHCI) 1.0 specification to connect with USB 2.0 High-Speed (HS) device.
- Support USB host function compliant to Open Host Controller Interface (OHCI) 1.0 specification to connect with USB 1.1 Full-Speed (FS) and Low-Speed (LS) devices
- Support USB High-Speed (HS) and Full-Speed (FS) device function
- Support USB device function with 1 endpoint for Control IN/OUT transfers and 12 programmable endpoints for Bulk, Interrupt and Isochronous IN/OUT transfers
- Support suspend, resume and remote wake-up capability
- Support DMA function
- Support 2048 Bytes internal SRAM for USB host function and 4096 Bytes internal SRAM for USB device function

- Flash Memory Interface
  - Support NAND flash interface
  - Support 8-bit data bus width
  - Support SLC and MLC type NAND flash device
  - Support 512 B, 2 KB, 4 KB and 8 KB page size NAND flash device
  - Support ECC4, ECC8, ECC12, ECC15 and ECC24 BCH algorithm for ECC code generation, error detection and error correction.
  - Support eMMC flash interface
  - Support DMA function to accelerate the data transfer between system memory and NAND and eMMC flash.
- $I^2S$ Controller
  - Support $I^2S$ interface
    - Support both mono and stereo
    - Support both record and playback
    - Support 8-bit, 16-bit 20-bit and 24-bit data precision
    - Support master and slave mode
  - Support PCM interface
    - Support 2 slots mode to connect 2 device
    - Support 8-bit, 16-bit 20-bit and 24-bit data precision
    - Support master mode
  - Support four 8x24 (8 24-bit) buffer for left/right channel record and left/right playback
  - Support DMA function to accelerate the data transfer between system memory and internal buffer
  - Support 2 buffer address for left/right channel and 2 slots data transfer
- LCD Display Controller
  - Support 8/9/16/18/24-bit data with to connect with 80/68 series MPU type LCD module
  - Support resolution up to 1024x768
  - Support data format conversion from RGB444, RGB565, RGB666, RGB888, YUV422 and YUV444 to RGB444, RGB565, RGB666, RGB888, YUV422 and YUV444 for display output
  - Support CCIR-656 (with VSYNC, HSYNC and data enable sync signal) 8/16-bit YUV data output to connect with external TV encoder
  - Support 8/16 bpp OSD data with video overlay function to facilitate the diverse graphic UI
  - Support linear 1X to 8X image scaling up function
  - Support Picture-In-Picture display function
  - Support hardware cursor

- Capture (CMOS Sensor Interface)
  - Support CCIR601 & CCIR656 interfaces to connect with CMOS image sensor
  - Support resolution up to 3M pixels
  - Support YUV422 and RGB565 color format for data output by CMOS image sensor
  - Support YUV422, RGB565, RGB555 and Y-only color format for data storing to system memory
  - Support planar and packet data format for data storing to system memory
  - Support image cropping and the cropping window is up to 4096x2048
  - Support image scaling-down:
  - Support vertical and horizontal scaling-down for preview mode
  - Support N/M scaling factor where N is equal to or less than M
  - Support 2 pairs of configurable 16-bit N and 16-bit M
  - Support to combine two interlace-fields to a single frame for data output by TV-decoder.
  - Support 3 color processing effects
  - Negative picture
  - Sepia picture
  - Posterization
- 2D Graphic Engine
  - Support 2D Bit Block Transfer (BitBLT) functions defined in Microsoft GDI
  - Support Host BLT
  - Support Pattern BLT
  - Support Color/Font Expanding BLT
  - Support Transparent BLT
  - Support Tile BLT
  - Support Block Move BLT
  - Support Copy File BLT
  - Support Color/Font Expansion
  - Support Rectangle Fill
  - Support RGB332/RGB565/RGB888 data format.
  - Support fore/background colors and all Microsoft 256 ternary raster-operation codes (ROP)
  - Support both inside and outside clipping function
  - Support alpha-blending for source/destination picture overlaying
  - Support fast Bresenham line drawing algorithm to draw solid/textured line
  - Support rectangular border and frame drawing
  - Support picture re-sizing
  - Support down-scaling from 1/255 to 254/255
  - Support up-scaling from 1 to 1.996 (1+254/255)
  - Support object rotation with different degree
  - Support L45 (45 degree left rotation) and L90 (90 degree left rotation)
  - Support R45 (45 degree right rotation) and R90 (90 degree right rotation)
  - Support M180 (mirror/flop)
  - Support F180 (up-side-down (flip) and X180 (180 degree rotation)
- JPEG Codec
  - Support Baseline Sequential mode JPEG codec function compliant with ISO/IEC 10918-1 international JPEG standard
  - Planar Format
  - Support to encode interleaved YCbCr 4:2:2/4:2:0 and gray-level (Y only) format image
  - Support to decode interleaved YCbCr 4:4:4/4:2:2/4:2:0/4:1:1 and gray-level (Y only) format image
  - Support to decode YCbCr 4:2:2 transpose format

- Support arbitrary width and height image encode and decode
- Support three programmable quantization-tables
- Support standard default Huffman-table and programmable Huffman-table for decode
- Support arbitrarily 1X~8X image up-scaling function for encode mode
- Support down-scaling function for encode and decode modes
- Support specified window decode mode
- Support quantization-table adjustment for bit-rate and quality control in encode mode
- Support rotate function in encode mode
- Packet Format
- Support to encode interleaved YUYV format input image, output bit stream 4:2:2 and 4:2:0 format
- Support to decode interleaved YCbCr 4:4:4/4:2:2/4:2:0 format image
- Support decoded output image RGB555, RGB565 and RGB888 formats.
- The encoded JPEG bit-stream format is fully compatible with JFIF and EXIF standards
- Support arbitrary width and height image encode and decode
- Support three programmable quantization-tables
- Support standard default Huffman-table and programmable Huffman-table for decode
- Support arbitrarily 1X~8X image up-scaling function for encode mode
- Support down-scaling function 1X~ 16X for Y422 and Y420, 1X~ 8X for Y444 for decode mode
- Support specified window decode mode
- Support quantization-table adjustment for bit-rate and quality control in encode mode

- Crypto Engine
  - PRNG
  - Support 64-bit, 128-bit, 192-bit and 256-bit key generation
  - DES
  - Support FIPS 46-3
  - Support both encryption and decryption
  - Support ECB, CBC, CFB, OFB and CTR modes
  - 3DES
  - Support FIPS NIST 800-67
  - Implements according to the X9.52 standard
  - Support 112-bit and 168-bit key
  - Support both encryption and decryption
  - Support ECB, CBC, CFB, OFB and CTR modes
  - AES
  - Support FIPS NIST 197
  - Support SP800-38A & addendum
  - Support 128-bit, 192-bit and 256-bit key
  - Support both encryption and decryption
  - Support ECB, CBC, CFB, OFB , CTR, CBC-CS1, CBC-CS2 and CBC-CS3 modes
  - Support Key Expander
  - SHA/HMAC
  - Support FIPS NIST 180, 180-1, 180-2
  - Support SHA-160, SHA-224, SHA-256, SHA-384, SHA-512 and corresponding HMAC algorithm
  - Support 128-bit MTP key
- GDMA (General DMA)
  - Support 2 channels GDMA for memory-to-memory data transfer without CPU intervention
  - Support increment and decrement for source and destination address calculation
  - Support 8-bit, 16-bit and 32-bit width data transfer
  - Support four 8-bit/16-bit/32-bit burst transfer

- UART
  - Support up to 11 UART controllers
  - Support 1 UART (UART 1) port with full model function (TXD, RXD, CTS, RTS, CDn, RIn, DTR and DSR) and 64-byte FIFO
  - Support 5 UART (UART 2/4/6/8/10) ports with flow control (TXD, RXD, CTS and RTS) and 64-byte FIFO
  - Support 5 TXD/RXD only UART ports (UART 0/3/5/7/9) with 16-byte FIFO for standard device
  - Support IrDA (SIR) and LIN function
  - Support RS-485 9-bit mode and direction control
  - Support programmable baud-rate generator up to 1/16 system clock
- C-CAN
  - Supports CAN protocol version 2.0 part A and B
  - Bit rates up to 1M bit/s
  - 32 Message Objects
  - Each Message Object has its own identifier mask
  - Programmable FIFO mode (concatenation of Message Object)
  - Maskable interrupt
  - Disabled Automatic Re-transmission mode for Time Triggered CAN applications
  - Support power down wake-up function
- Smart Card Host (SC)
  - Compliant to ISO-7816-3 T=0, T=1
  - Supports up to two ISO-7816-3 ports
  - Separate receive / transmit 4 bytes entry FIFO for data payloads
  - Programmable transmission clock frequency
  - Programmable receiver buffer trigger level
  - 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
  - Supports auto inverse convention function
  - Supports transmitter and receiver error retry and error limit function
  - Supports hardware activation sequence process
  - Supports hardware warm reset sequence process
  - Supports hardware deactivation sequence process
  - Supports hardware auto deactivation sequence when detecting the card removal
- Timer
  - Support 5 sets of 32-bit timers with 24-bit up-timer and one 8-bit pre-scale counter
  - Independent clock source for each timer
  - Support one-shot, periodic, toggle and continuous operation modes
- Enhanced Timer
  - Support 4 sets of 32-bit timers with 24-bit up-timer and one 8-bit pre-scale counter
  - Independent clock source for each timer
  - Support one-shot, periodic, toggle and continuous operation modes
  - Supports external pin capture for interval measurement
  - Supports external pin capture for timer counter reset
- Watchdog Timer
  - Multiple clock sources
  - 8 selectable time out period from 1.333us ~ 14.316sec (depends on clock source)
  - WDT can wake-up from power down or idle mode
  - Interrupt or reset selectable on watchdog timer time-out

- Windowed-Watchdog Timer
  - 6-bit down counter with 11-bit pre-scale for wide range window selected
  - Interrupt on windowed-watchdog timer time-out
  - Reset on windowed-watchdog timer time out or reload in an unexpected time window
- Real Time Clock (RTC)
  - Supports software compensation by setting frequency compensate register (FCR)
  - Supports RTC counter (second, minute, hour) and calendar counter (day, month, year)
  - Supports Alarm registers (second, minute, hour, day, month, year)
  - Selectable 12-hour or 24-hour mode
  - Automatic leap year recognition
  - Supports periodic time tick interrupt with 8 period options 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 and 1 second
  - Supports battery power pin (VBAT)
  - Supports wake-up function
- PWM
  - Built-in up to two 16-bit PWM generators provide four PWM outputs
  - Each PWM generator equipped with one clock source selector, one clock divider, one 8-bit pre-scale, two 16-bit counters, and one Dead-Zone generator
- SPI
  - Built-in up to two sets of SPI controller
  - Support SPI master mode
  - Support single/dual/quad bit data bus width
  - Full duplex synchronous serial data transfer
  - Variable length of transfer data from 8 to 32 bits
  - MSB or LSB first data transfer
  - Support burst mode operation that transmission and reception can be executed up to four times in a transfer
  - Support 2 slave/device select lines
- $I^2C$
  - Up to two sets of $I^2C$ device
  - Support master mode
  - Bidirectional data transfer between masters and slaves
  - Multi-master bus (no central master)
  - Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
  - Serial clock synchronization allows devices with different bit rates to communicate via one serial bus
  - Serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer
  - Programmable clocks allow versatile rate control
  - Support software mode to generate $I^2C$ signaling
- Advanced Interrupt Controller
  - Support 58 interrupt sources, including 8 external interrupt sources
  - Support programmable normal or fast interrupt mode (IRQ, FIQ)
  - Support programmable edge-triggered or level-sensitive for 8 external interrupt sources
  - Support programmable low-active or high-active for 8 external interrupt sources
  - Support encoded priority methodology to allow for interrupt daisy-chaining
  - Support lower priority interrupt automatically mask out for nested interrupt
  - Support to clear interrupt flag automatically if interrupt source is programmed as edge-

triggered

- GPIO
  - TTL/Schmitt trigger input selectable
  - I/O pin can be configured as interrupt source with edge/level setting
  - Support pull-up and pull-down control
- ADC
  - 12-bit SAR ADC with 1M/200K SPS
  - Up to 8-ch single-end input
  - Support up to 1M SPS in channel 1 and up to 200K SPS in others channels.
  - Support 4-wire or 5-wire resistance touch screen interface
  - Support touch pressure measurement for 4-wire touch screen application
  - Support pen down detection
  - Support battery measurement
  - Support keypad scan
- KPI
  - Matrix keypad interface supported.
  - Maximum 4X8 and minimum 3X3 keypad matrix supported.
  - Configurable key de-bounce supported.
  - Low power wakeup mode supported.
  - Configurable three-key reset supported.
- MTP
  - Support 256-bit programmable memory for key of Crypto functionality
  - Support up to 15 times of programming and erase.
- Low Voltage Detect (LVD) and Low Voltage Reset (LVR)
  - Support two, 2.6V and 2.8V, voltage detection levels
  - Interrupt when low voltage detected
  - Reset when low voltage detected
  - Low voltage reset threshold voltage levels: 2.4 V
- Power Management
  - Advanced power management including Power Down, Deep Standby, CPU Standby and Normal Operating modes
  - Normal Operating mode
    - CPU run normally and all clocks on, the current consumption of CORE_VDD is around 185 mA (at CPU/DRAM clock is 300/150 MHz CPU).
  - CPU Standby mode
    - CPU clock stop, and all other clocks on.
  - Deep Standby mode
    - All clocks stop, except LXT, with SRAM retention, and the current consumption of CORE_VDD is typicaly 3 mA
  - Power Down mode
    - All powers are off except RTC_VDD (3.3V) and the current consumption of RTC_VDD is typicaly 7uA with RTC functionality on.
- Operating Voltage
  - 1.2V for core logic operating
  - 1.8V for DDR2 SDRAM I/O operating
  - 3.3V for normal I/O operating
- Operating Temperature: -40℃~85℃
- Packages:

- All Green package (RoHS)
- LQFP 216-pin
- LQFP 128-pin

# 3 PARTS INFORMATION LIST AND PIN CONFIGURATION

## 3.1 NUC970 Series Part Number Naming Guide

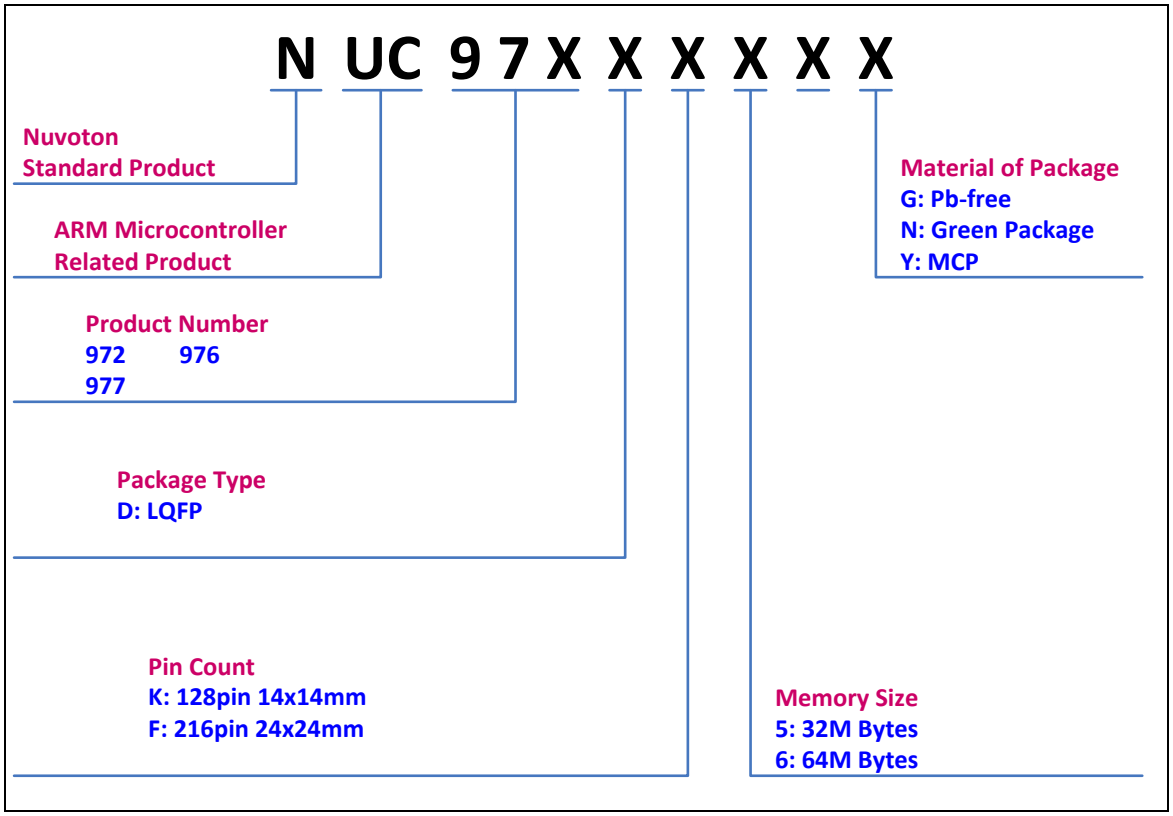



Figure 3.1-1 NUC970 Series Part Number Naming Guide

## 3.2 NUC970 Series Part Selection Guide

| Part No. | Memory I/F | | Storage | | | MAC | USB | | GFX | LCD | Timer | | | | | | ADC (12-bit) | | | | Peripheral | | | | | | | | | | | | Package 2 | Operating Temp. Range1 (°C) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DDR2 RAM(MB) | SPI Flash, No. of I/O Pins | NAND Flash, No. of ECC bits | EMMC | SD / SDIO | Ethernet 10/100 MAC | USB 2.0 Host (480M bps) | USB 2.0 HS Device | 2D Graphics | TFT LCD | Real-Time Clock (RTC) | Timer (32-bit) | Enhanced Timer | Watchdog Timer | Window Watchdog Timer | PWM | Touch Screen Controller | No. of Channels | Speed (Samples per second) | LVD/LVR | JTAG | External Bus Interface | GPIO (Max) | UART | CAN BUS | I2C (Master Only) | SPI | ISO-7816-3 | KPI | PS2 | CMOS Interface | I2S/AC97 | | |
| ARM9 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| NUC972DF62Y | 64 | 1 | 24 | √ | 2 | 2 | 2 | 1 | √ | √ | √ | 5 | 4 | √ | √ | 4 | √ | 8 | 200K* | √ | √ | √ | 146 | 11 | 2 | 2 | 2 | 2 | √ | - | √ | 1 | LQFP216 | -40 to +85 |
| NUC972DF51Y | 32 | 1 | 24 | √ | 2 | 2 | 2 | 1 | √ | √ | √ | 5 | 4 | √ | √ | 4 | √ | 8 | 200K* | √ | √ | √ | 146 | 11 | 2 | 2 | 2 | 2 | √ | - | √ | 1 | LQFP216 | -40 to +85 |
| NUC976DK62Y | 64 | 1 | - | √ | 2 | 1 | 2 | 1 | √ | √ | √ | 5 | 1 | √ | √ | 4 | √ | 4 | 200K* | √ | √ | - | 80 | 6 | 1 | 2 | 2 | 2 | √ | - | √ | 1 | LQFP128 | -40 to +85 |
| NUC976DK51Y | 32 | 1 | - | √ | 2 | 1 | 2 | 1 | √ | √ | √ | 5 | 1 | √ | √ | 4 | √ | 4 | 200K* | √ | √ | - | 80 | 6 | 1 | 2 | 2 | 2 | √ | - | √ | 1 | LQFP128 | -40 to +85 |
| NUC977DK62Y | 64 | 1 | 24 | √ | 2 | 1 | 2 | 1 | √ | √ | √ | 5 | 2 | √ | √ | 4 | - | - | - | √ | √ | - | 87 | 8 | 1 | 2 | 2 | 2 | √ | - | √ | 1 | LQFP128 | -40 to +85 |
| NUC977DK51Y | 32 | 1 | 24 | √ | 2 | 1 | 2 | 1 | √ | √ | √ | 5 | 2 | √ | √ | 4 | - | - | - | √ | √ | - | 87 | 8 | 1 | 2 | 2 | 2 | √ | - | √ | 1 | LQFP128 | -40 to +85 |

## 3.3 Pin Configuration

### 3.3.1 NUC972DFxxY Pin Diagram

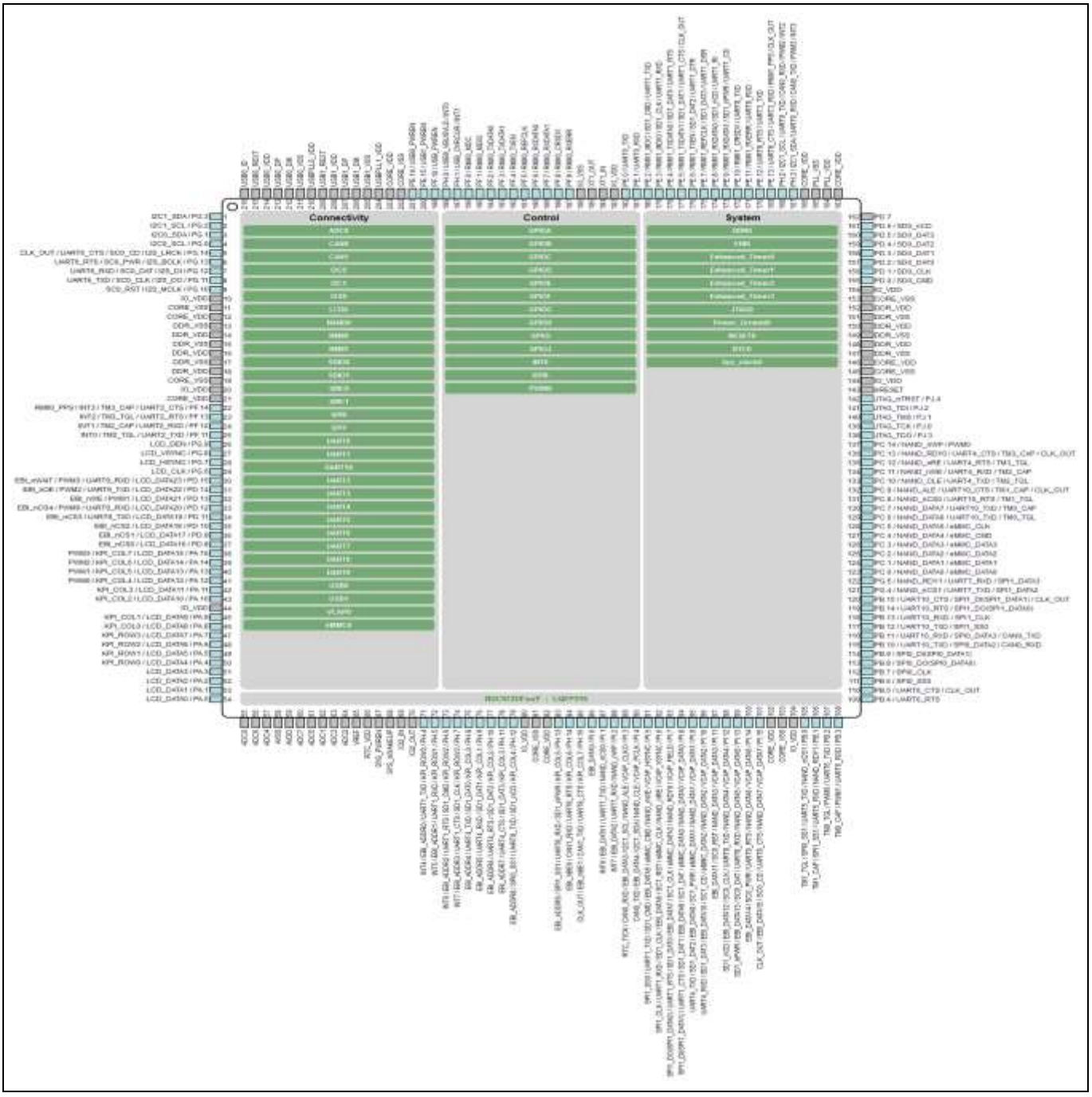



Figure 3.3-1 NUC972DFxxY LQFP 216-pin Pin Diagram

### 3.3.2 NUC976DKxxY Pin Diagram

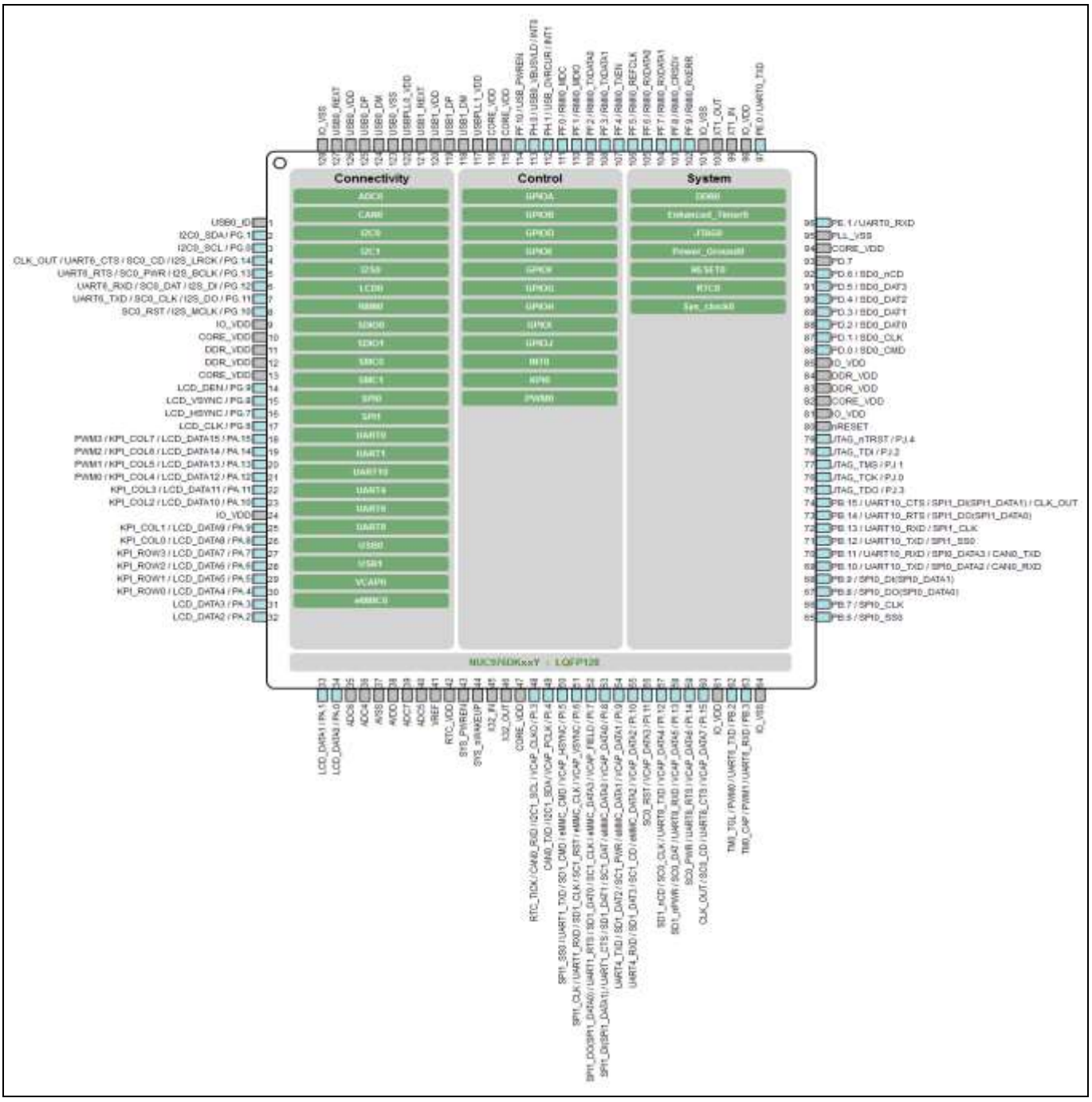



Figure 3.3-2 NUC976DKxxY LQFP 128-pin Pin Diagram

### 3.3.3 NUC977DKxxY Pin Diagram

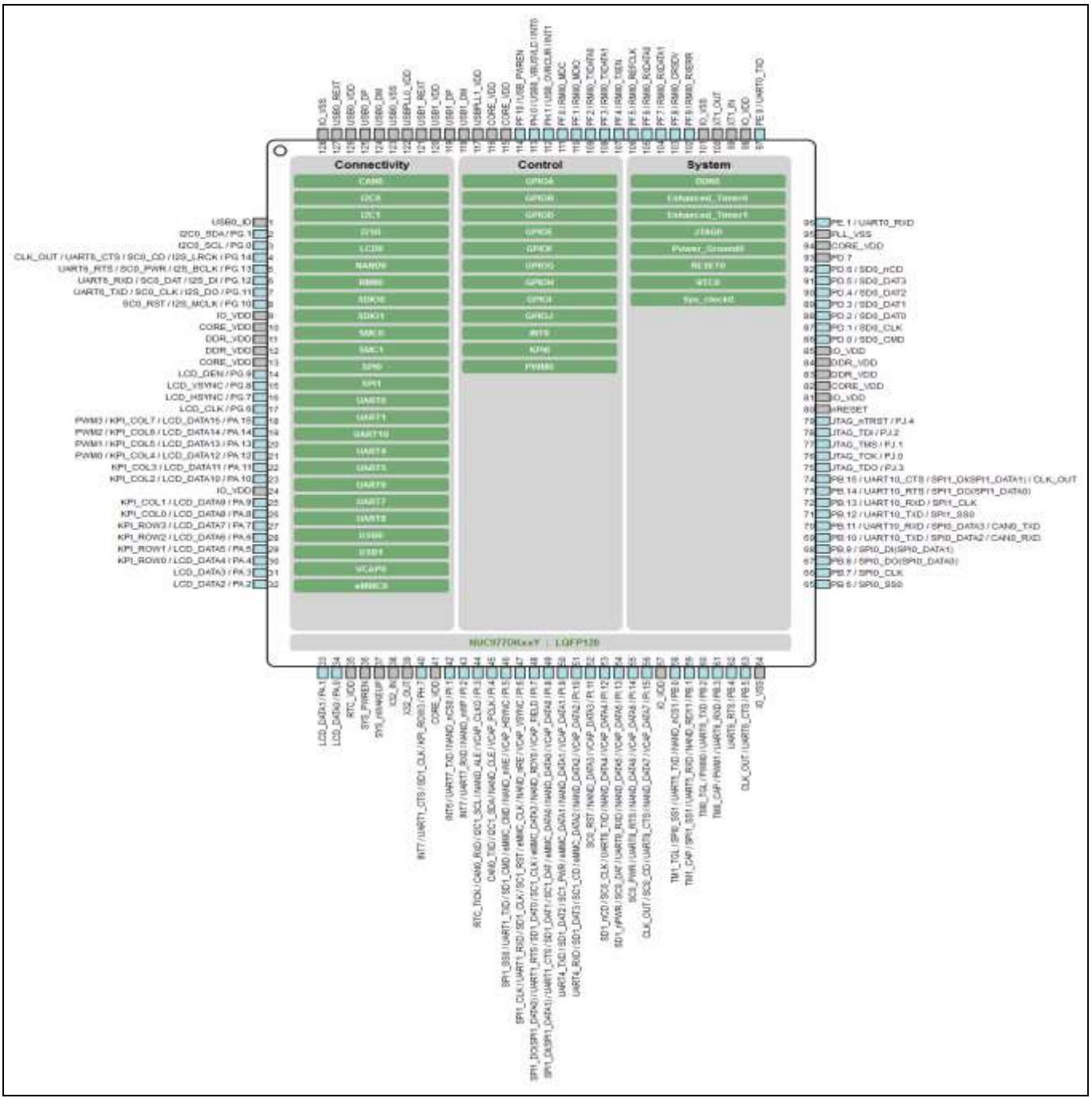



Figure 3.3-3 NUC977DKxxY LQFP 128-pin Pin Diagram

## 3.4 Pin Description

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| 1 | - | - | PG.3 | I/O | General purpose digital I/O pin Port G Pin 3. |
| | | | I2C1_SDA | I/O | $I^2C1$ data input/output pin. |
| 2 | - | - | PG.2 | I/O | General purpose digital I/O pin Port G Pin 2. |
| | | | I2C1_SCL | O | $I^2C1$ clock pin. |
| 3 | 2 | 2 | PG.1 | I/O | General purpose digital I/O pin Port G Pin 1. |
| | | | I2C0_SDA | I/O | $I^2C0$ data input/output pin. |
| 4 | 3 | 3 | PG.0 | I/O | General purpose digital I/O pin Port G Pin 0. |
| | | | I2C0_SCL | O | $I^2C0$ clock pin. |
| 5 | 4 | 4 | PG.14 | I/O | General purpose digital I/O pin Port G Pin 14. |
| | | | I2S_LRCK | O | $I^2S$ left right channel clock. |
| | | | UART6_CTS | I | Clear to send input pin for UART6. |
| | | | SC0_CD | I | SmartCard0 card detect pin. |
| | | | CLK_OUT | O | Reference Clock Output. |
| 6 | 5 | 5 | PG.13 | I/O | General purpose digital I/O pin Port G Pin 13. |
| | | | I2S_BCLK | I | $I^2S$ bit clock pin. |
| | | | UART6_RTS | O | Request to send output pin for UART6. |
| | | | SC0_PWR | O | SmartCard0 power pin. |
| 7 | 6 | 6 | PG.12 | I/O | General purpose digital I/O pin Port G Pin 12. |
| | | | I2S_DI | I | $I^2S$ data input. |
| | | | UART6_RXD | I | Data receiver input pin for UART6. |
| | | | SC0_DAT | I/O | SmartCard0 data pin. |
| 8 | 7 | 7 | PG.11 | I/O | General purpose digital I/O pin Port G Pin 11. |
| | | | I2S_DO | O | $I^2S$ data output. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| | | | UART6_TXD | O | Data transmitter output pin for UART6. |
| | | | SC0_CLK | O | SmartCard0 clock pin. |
| 9 | 8 | 8 | PG.10 | I/O | General purpose digital I/O pin Port G Pin 10. |
| | | | I2S_MCLK | O | $I^2S$ master clock output pin. |
| | | | SC0_RST | O | SmartCard0 reset pin. |
| 10 | 9 | 9 | IO_VDD | P | MCU I/O power pin. |
| 11 | - | - | CORE_VSS | P | MCU internal core ground pin. |
| 12 | 10 | 10 | CORE_VDD | P | MCU internal core power pin. |
| 13 | - | - | DDR_VSS | P | DDR ground pin. |
| 14 | 11 | 11 | DDR_VDD | P | DDR power pin. |
| 15 | - | - | DDR_VSS | P | DDR ground pin. |
| 16 | 11 | 11 | DDR_VDD | P | DDR power pin. |
| 17 | - | - | DDR_VSS | P | DDR ground pin. |
| 18 | 12 | 12 | DDR_VDD | P | DDR power pin. |
| 19 | - | - | CORE_VSS | P | MCU internal core ground pin. |
| 20 | - | - | IO_VDD | P | MCU I/O power pin. |
| 21 | 13 | 13 | CORE_VDD | P | MCU internal core power pin. |
| - | - | - | PF.15 | I/O | General purpose digital I/O pin Port F Pin 15. |
| | | | INT4 | I | External interrupt 4 input pin. |
| 22 | - | - | PF.14 | I/O | General purpose digital I/O pin Port F Pin 14. |
| | | | UART2_CTS | I | Clear to send input pin for UART2. |
| | | | TM3_CAP | I | Enhanced TIMER capture input pin. |
| | | | INT3 | I | External interrupt 3 input pin. |
| 23 | - | - | PF.13 | I/O | General purpose digital I/O pin Port F Pin 13. |
| | | | UART2_RTS | O | Request to send output pin for UART2. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| | | | TM3_TGL | O | Enhanced TIMER toggle output pin. |
| | | | INT2 | I | External interrupt 2 input pin. |
| 24 | - | - | PF.12 | I/O | General purpose digital I/O pin Port F Pin 12. |
| | | | UART2_RXD | I | Data receiver input pin for UART2. |
| | | | TM2_CAP | I | Enhanced TIMER capture input pin. |
| | | | INT1 | I | External interrupt 1 input pin. |
| 25 | - | - | PF.11 | I/O | General purpose digital I/O pin Port F Pin 11. |
| | | | UART2_TXD | O | Data transmitter output pin for UART2. |
| | | | TM2_TGL | O | Enhanced TIMER toggle output pin. |
| | | | INT0 | I | External interrupt 0 input pin. |
| 26 | 14 | 14 | PG.9 | I/O | General purpose digital I/O pin Port G Pin 9. |
| | | | LCD_DEN | O | Data enable or display control signal. |
| 27 | 15 | 15 | PG.8 | I/O | General purpose digital I/O pin Port G Pin 8. |
| | | | LCD_VSYNC | O | Vertical sync or frame sync. |
| 28 | 16 | 16 | PG.7 | I/O | General purpose digital I/O pin Port G Pin 7. |
| | | | LCD_HSYNC | O | Horizontal sync or line sync. |
| 29 | 17 | 17 | PG.6 | I/O | General purpose digital I/O pin Port G Pin 6. |
| | | | LCD_CLK | O | Pixel clock output. |
| 30 | - | - | PD.15 | I/O | General purpose digital I/O pin Port D Pin 15. |
| | | | LCD_DATA23 | O | LCD pixel data output bit 23. |
| | | | UART9_RXD | I | Data receiver input pin for UART9. |
| | | | PWM3 | O | PWM3 output pin. |
| | | | EBI_nWAIT | I | External I/O wait control. |
| 31 | - | - | PD.14 | I/O | General purpose digital I/O pin Port D Pin 14. |
| | | | LCD_DATA22 | O | LCD pixel data output bit 22. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| | | | UART9_TXD | O | Data transmitter output pin for UART9. |
| | | | PWM2 | O | PWM2 output pin. |
| | | | EBI_nOE | O | External I/O output enable. |
| 32 | - | - | PD.13 | I/O | General purpose digital I/O pin Port D Pin 13. |
| | | | LCD_DATA21 | O | LCD pixel data output bit 21. |
| | | | PWM1 | O | PWM1 output pin. |
| | | | EBI_nWE | O | External I/O chip write enable. |
| 33 | - | - | PD.12 | I/O | General purpose digital I/O pin Port D Pin 12. |
| | | | LCD_DATA20 | O | LCD pixel data output bit 20. |
| | | | UART9_RXD | I | Data receiver input pin for UART9. |
| | | | PWM0 | O | PWM0 output pin. |
| | | | EBI_nCS4 | O | External I/O chip select bank 4. |
| 34 | - | - | PD.11 | I/O | General purpose digital I/O pin Port D Pin 11. |
| | | | LCD_DATA19 | O | LCD pixel data output bit 19. |
| | | | UART9_TXD | O | Data transmitter output pin for UART9. |
| | | | EBI_nCS3 | O | External I/O chip select bank 3. |
| 35 | - | - | PD.10 | I/O | General purpose digital I/O pin Port D Pin 10. |
| | | | LCD_DATA18 | O | LCD pixel data output bit 18. |
| | | | EBI_nCS2 | O | External I/O chip select bank 2. |
| 36 | - | - | PD.9 | I/O | General purpose digital I/O pin Port D Pin 9. |
| | | | LCD_DATA17 | O | LCD pixel data output bit 17. |
| | | | EBI_nCS1 | O | External I/O chip select bank 1. |
| 37 | - | - | PD.8 | I/O | General purpose digital I/O pin Port D Pin 8. |
| | | | LCD_DATA16 | O | LCD pixel data output bit 16. |
| | | | EBI_nCS0 | O | External I/O chip select bank 0. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| 38 | 18 | 18 | PA.15 | I/O | General purpose digital I/O pin Port A Pin 15. |
| | | | LCD_DATA15 | O | LCD pixel data output bit 15. |
| | | | KPI_COL7 | I | Keypad Column Scan Input Bus 7. |
| | | | PWM3 | O | PWM3 output pin. |
| 39 | 19 | 19 | PA.14 | I/O | General purpose digital I/O pin Port A Pin 14. |
| | | | LCD_DATA14 | O | LCD pixel data output bit 14. |
| | | | KPI_COL6 | I | Keypad Column Scan Input Bus 6. |
| | | | PWM2 | O | PWM2 output pin. |
| 40 | 20 | 20 | PA.13 | I/O | General purpose digital I/O pin Port A Pin 13. |
| | | | LCD_DATA13 | O | LCD pixel data output bit 13. |
| | | | KPI_COL5 | I | Keypad Column Scan Input Bus 5. |
| | | | PWM1 | O | PWM1 output pin. |
| 41 | 21 | 21 | PA.12 | I/O | General purpose digital I/O pin Port A Pin 12. |
| | | | LCD_DATA12 | O | LCD pixel data output bit 12. |
| | | | KPI_COL4 | I | Keypad Column Scan Input Bus 4. |
| | | | PWM0 | O | PWM0 output pin. |
| 42 | 22 | 22 | PA.11 | I/O | General purpose digital I/O pin Port A Pin 11. |
| | | | LCD_DATA11 | O | LCD pixel data output bit 11. |
| | | | KPI_COL3 | I | Keypad Column Scan Input Bus 3. |
| 43 | 23 | 23 | PA.10 | I/O | General purpose digital I/O pin Port A Pin 10. |
| | | | LCD_DATA10 | O | LCD pixel data output bit 10. |
| | | | KPI_COL2 | I | Keypad Column Scan Input Bus 2. |
| 44 | 24 | 24 | IO_VDD | P | MCU I/O power pin. |
| 45 | 25 | 25 | PA.9 | I/O | General purpose digital I/O pin Port A Pin 9. |
| | | | RMII0_RXERR | I | RMII0 receive data error. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| | | | LCD_DATA9 | O | LCD pixel data output bit 9. |
| | | | KPI_COL1 | I | Keypad Column Scan Input Bus 1. |
| | | | PWRON_SET9 | I | Power On Setting bit 9. |
| 46 | 26 | 26 | PA.8 | I/O | General purpose digital I/O pin Port A Pin 8. |
| | | | RMII0_CRSDV | I | RMII0 carrier sense / receive data valid. |
| | | | LCD_DATA8 | O | LCD pixel data output bit 8. |
| | | | KPI_COL0 | I | Keypad Column Scan Input Bus 0. |
| | | | PWRON_SET8 | I | Power On Setting bit 8. |
| 47 | 27 | 27 | PA.7 | I/O | General purpose digital I/O pin Port A Pin 7. |
| | | | RMII0_RXDATA1 | I | RMII0 receive data bus bit 1. |
| | | | LCD_DATA7 | O | LCD pixel data output bit 7. |
| | | | KPI_ROW3 | O | Keypad Row Scan Output Bus 3. |
| | | | PWRON_SET7 | I | Power On Setting bit 7. |
| 48 | 28 | 28 | PA.6 | I/O | General purpose digital I/O pin Port A Pin 6. |
| | | | RMII0_RXDATA0 | I | RMII0 receive data bus bit 0. |
| | | | LCD_DATA6 | O | LCD pixel data output bit 6. |
| | | | KPI_ROW2 | O | Keypad Row Scan Output Bus 2. |
| | | | PWRON_SET6 | I | Power On Setting bit 6. |
| 49 | 29 | 29 | PA.5 | I/O | General purpose digital I/O pin Port A Pin 5. |
| | | | RMII0_REFCLK | I | RMII0 reference clock. |
| | | | LCD_DATA5 | O | LCD pixel data output bit 5. |
| | | | KPI_ROW1 | O | Keypad Row Scan Output Bus 1. |
| | | | PWRON_SET5 | I | Power On Setting bit 5. |
| 50 | 30 | 30 | PA.4 | I/O | General purpose digital I/O pin Port A Pin 4. |
| | | | RMII0_TXEN | O | RMII0 transmit enable. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| | | | LCD_DATA4 | O | LCD pixel data output bit 4. |
| | | | KPI_ROW0 | O | Keypad Row Scan Output Bus 0. |
| | | | PWRON_SET4 | I | Power On Setting bit 4. |
| 51 | 31 | 31 | PA.3 | I/O | General purpose digital I/O pin Port A Pin 3. |
| | | | RMII0_TXDATA1 | O | RMII0 transmit data bus bit 1. |
| | | | LCD_DATA3 | O | LCD pixel data output bit 3. |
| | | | PWRON_SET3 | I | Power On Setting bit 3. |
| 52 | 32 | 32 | PA.2 | I/O | General purpose digital I/O pin Port A Pin 2. |
| | | | RMII0_TXDATA0 | O | RMII0 Transmit Data bus bit 0. |
| | | | LCD_DATA2 | O | LCD pixel data output bit 2. |
| | | | PWRON_SET2 | I | Power On Setting bit 2. |
| 53 | 33 | 33 | PA.1 | I/O | General purpose digital I/O pin Port A Pin 1. |
| | | | RMII0_MDIO | I/O | RMII0 Management Data I/O |
| | | | LCD_DATA1 | O | LCD pixel data output bit 1. |
| | | | PWRON_SET1 | I | Power On Setting bit 1. |
| 54 | 34 | 34 | PA.0 | I/O | General purpose digital I/O pin Port A Pin 0. |
| | | | RMII0_MDC | O | RMII0 Management Data Clock |
| | | | LCD_DATA0 | O | LCD pixel data output bit 0. |
| | | | PWRON_SET0 | I | Power On Setting bit 0. |
| 55 | - | - | ADC0 | I | ADC input channel 0 or VBAT. |
| 56 | 35 | - | ADC6 | I | ADC input channel 6 or XM. |
| 57 | 36 | - | ADC4 | I | ADC input channel 4 or YM. |
| 58 | 37 | - | AVSS | P | Ground pin for analog circuit. |
| 59 | 38 | - | AVDD | P | Power supply for internal analog circuit. |
| 60 | 39 | - | ADC7 | I | ADC input channel 7 or XP. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| 61 | 40 | - | ADC5 | I | ADC input channel 5 or YP. |
| 62 | - | - | ADC1 | I | ADC input channel 1. |
| 63 | - | - | ADC3 | I | ADC input channel 3 or VSENSE. |
| 64 | - | - | ADC2 | I | ADC input channel 2. |
| 65 | 41 | - | VREF | I | ADC voltage reference pin. |
| 66 | 42 | 35 | RTC_VDD | P | RTC power input. |
| 67 | 43 | 36 | SYS_PWREN | O | RTC wake-up output pin for external DC/DC enable pin control. |
| 68 | 44 | 37 | SYS_nWAKEUP | I | RTC wake-up interrupt input with internal pull-high. |
| 69 | 45 | 38 | X32_IN | I | External 32.768kHz crystal input. |
| 70 | 46 | 39 | X32_OUT | O | External 32.768kHz crystal output. |
| 71 | - | - | PH.4 | I/O | General purpose digital I/O pin Port H Pin 4. |
| | | | KPI_ROW0 | O | Keypad Row Scan Output Bus 0. |
| | | | UART1_TXD | O | Data transmitter output pin for UART1. |
| | | | EBI_ADDR0 | O | External I/O address bus bit 0. |
| | | | INT4 | I | External interrupt 4 input pin. |
| 72 | - | - | PH.5 | I/O | General purpose digital I/O pin Port H Pin 5. |
| | | | KPI_ROW1 | O | Keypad Row Scan Output Bus 1. |
| | | | UART1_RXD | I | Data receiver input pin for UART1. |
| | | | EBI_ADDR1 | O | External I/O address bus bit 1. |
| | | | INT5 | I | External interrupt 5 input pin. |
| 73 | - | - | PH.6 | I/O | General purpose digital I/O pin Port H Pin 6. |
| | | | KPI_ROW2 | O | Keypad Row Scan Output Bus 2. |
| | | | SD1_CMD | O | SD/SDIO Port 1 – command/response. |
| | | | UART1_RTS | O | Request to send output pin for UART1. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| | | | EBI_ADDR2 | O | External I/O address bus bit 2. |
| | | | INT6 | I | External interrupt 6 input pin. |
| 74 | - | 40 | PH.7 | I/O | General purpose digital I/O pin Port H Pin 7. |
| | | | KPI_ROW3 | O | Keypad Row Scan Output Bus 3. |
| | | | SD1_CLK | O | SD/SDIO Port 1 – clock. |
| | | | UART1_CTS | I | Clear to send input pin for UART1. |
| | | | EBI_ADDR3 | O | External I/O address bus bit 3. |
| | | | INT7 | I | External interrupt 7 input pin. |
| 75 | - | - | PH.8 | I/O | General purpose digital I/O pin Port H Pin 8. |
| | | | KPI_COL0 | I | Keypad Column Scan Input Bus 0. |
| | | | SD1_DAT0 | I/O | SD/SDIO mode #1 data line bit 0. |
| | | | UART4_TXD | O | Data transmitter output pin for UART4. |
| | | | EBI_ADDR4 | O | External I/O address bus bit 4. |
| 76 | - | - | PH.9 | I/O | General purpose digital I/O pin Port H Pin 9. |
| | | | KPI_COL1 | I | Keypad Column Scan Input Bus 1. |
| | | | SD1_DAT1 | I/O | SD/SDIO mode #1 data line bit 1. |
| | | | UART4_RXD | I | Data receiver input pin for UART4. |
| | | | EBI_ADDR5 | O | External I/O address bus bit 5. |
| 77 | - | - | PH.10 | I/O | General purpose digital I/O pin Port H Pin 10. |
| | | | KPI_COL2 | I | Keypad Column Scan Input Bus 2. |
| | | | SD1_DAT2 | I/O | SD/SDIO mode #1 data line bit 2. |
| | | | UART4_RTS | O | Request to send output pin for UART4. |
| | | | EBI_ADDR6 | O | External I/O address bus bit 6. |
| 78 | - | - | PH.11 | I/O | General purpose digital I/O pin Port H Pin 11. |
| | | | KPI_COL3 | I | Keypad Column Scan Input Bus 3. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| | | | SD1_DAT3 | I/O | SD/SDIO mode #1 data line bit 3. |
| | | | UART4_CTS | I | Clear to send input pin for UART4. |
| | | | EBI_ADDR7 | O | External I/O address bus bit 7. |
| 79 | - | - | PH.12 | I/O | General purpose digital I/O pin Port H Pin 12. |
| | | | KPI_COL4 | I | Keypad Column Scan Input Bus 4. |
| | | | SD1_nCD | I | SD/SDIO mode #1 – card detect. |
| | | | UART8_TXD | O | Data transmitter output pin for UART8. |
| | | | SPI0_SS1 | O | 2nd SPI0 chip select pin. |
| | | | EBI_ADDR8 | O | External I/O address bus bit 8. |
| 80 | - | - | IO_VDD | P | MCU I/O power pin. |
| 81 | - | - | CORE_VSS | P | MCU internal core ground pin. |
| 82 | 47 | 41 | CORE_VDD | P | MCU internal core power pin. |
| 83 | - | - | PH.13 | I/O | General purpose digital I/O pin Port H Pin 13. |
| | | | KPI_COL5 | I | Keypad Column Scan Input Bus 5. |
| | | | SD1_nPWR | O | SD/SDIO mode #1 – power enable. |
| | | | UART8_RXD | I | Data receiver input pin for UART8. |
| | | | SPI1_SS1 | O | 2nd SPI1 chip select pin. |
| | | | EBI_ADDR9 | O | External I/O address bus bit 9. |
| 84 | - | - | PH.14 | I/O | General purpose digital I/O pin Port H Pin 14. |
| | | | KPI_COL6 | I | Keypad Column Scan Input Bus 6. |
| | | | UART8_RTS | O | Request to send output pin for UART8. |
| | | | CAN1_RXD | I | CAN bus receiver1 input. |
| | | | EBI_nBE0 | O | External I/O low byte enable. |
| 85 | - | - | PH.15 | I/O | General purpose digital I/O pin Port H Pin 15. |
| | | | KPI_COL7 | I | Keypad Column Scan Input Bus 7. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| | | | UART8_CTS | I | Clear to send input pin for UART8. |
| | | | CAN1_TXD | O | CAN bus transmitter1 output. |
| | | | EBI_nBE1 | O | External I/O high byte enable. |
| 86 | - | - | PI.0 | I/O | General purpose digital I/O pin Port I Pin 0. |
| | | | EBI_DATA0 | I/O | External I/O data bus bit 0. |
| 87 | - | 42 | PI.1 | I/O | General purpose digital I/O pin Port I Pin 1. |
| | | | NAND_nCS0 | O | NAND flash chip select 0. |
| | | | UART7_TXD | O | Data transmitter output pin for UART7. |
| | | | EBI_DATA1 | I/O | External I/O data bus bit 1. |
| | | | INT6 | I | External interrupt 6 input pin. |
| 88 | - | 43 | PI.2 | I/O | General purpose digital I/O pin Port I Pin 2. |
| | | | NAND_nWP | O | NAND flash srite protect. |
| | | | UART7_RXD | I | Data receiver input pin for UART7. |
| | | | EBI_DATA2 | I/O | External I/O data bus bit 2. |
| | | | INT7 | I | External interrupt 7 input pin. |
| 89 | 48 | 44 | PI.3 | I/O | General purpose digital I/O pin Port I Pin 3. |
| | | | VCAP_CLKO | O | Sensor interface system clock. |
| | | | NAND_ALE | O | NAND flash address latch enable. |
| | | | I2C1_SCL | O | $I^2C1$ clock pin. |
| | | | EBI_DATA3 | I/O | External I/O data bus bit 3. |
| | | | CAN0_RXD | I | CAN bus receiver0 input. |
| | | | RTC_TICK | O | RTC tick output |
| 90 | 49 | 45 | PI.4 | I/O | General purpose digital I/O pin Port I Pin 4. |
| | | | VCAP_PCLK | I | Sensor interface pixel clock. |
| | | | NAND_CLE | O | NAND flash command latch enable. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| | | | I2C1_SDA | I/O | $I^2C1$ data input/output pin. |
| | | | EBI_DATA4 | I/O | External I/O data bus bit 4. |
| | | | CAN0_TX | O | CAN bus transmitter0 output. |
| 91 | 50 | 46 | PI.5 | I/O | General purpose digital I/O pin Port I Pin 5. |
| | | | VCAP_HSYNC | I | Sensor interface HSYNC. |
| | | | NAND_nWE | O | NAND flash write enable. |
| | | | eMMC_CMD | I/O | eMMC command/Response. |
| | | | EBI_DATA5 | I/O | External I/O data bus bit 5. |
| | | | SD1_CMD | O | SD/SDIO mode #1 – command/response. |
| | | | UART1_TXD | O | Data transmitter output pin for UART1. |
| | | | SPI1_SS0 | O | 1st SPI1 chip select pin. |
| 92 | 51 | 47 | PI.6 | I/O | General purpose digital I/O pin Port I Pin 6. |
| | | | VCAP_VSYNC | I | Sensor interface VSYNC. |
| | | | NAND_nRE | O | NAND flash read enable. |
| | | | eMMC_CLK | O | eMMC clock output. |
| | | | SC1_RST | O | SmartCard1 reset pin. |
| | | | EBI_DATA6 | I/O | External I/O data bus bit 6. |
| | | | SD1_CLK | O | SD/SDIO mode #1– clock. |
| | | | UART1_RXD | I | Data receiver input pin for UART1. |
| | | | SPI1_CLK | O | SPI1 serial clock pin. |
| 93 | 52 | 48 | PI.7 | I/O | General purpose digital I/O pin Port I Pin 7. |
| | | | VCAP_FIELD | I | Sensor interface even/odd field indicator. |
| | | | NAND_RDY0 | I | NAND flash ready/busy channel 0. |
| | | | eMMC_DATA3 | I/O | eMMC data line bit 3. |
| | | | SC1_CLK | O | SmartCard1 clock pin. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| | | | EBI_DATA7 | I/O | External I/O data bus bit 7. |
| | | | SD1_DAT0 | I/O | SD/SDIO mode #1 data line bit 0. |
| | | | UART1_RTS | O | Request to send output pin for UART1. |
| | | | SPI1_DO (SPI1_DATA0) | I (I/O) | SPI1 Data out pin. SPI1 data 0 in dual/quad mode. |
| 94 | 53 | 49 | PI.8 | I/O | General purpose digital I/O pin Port I Pin 8. |
| | | | VCAP_DATA0 | I | Sensor interface data bus bit 0. |
| | | | NAND_DATA0 | I/O | NAND flash data bus bit 0. |
| | | | eMMC_DATA0 | I/O | eMMC data line bit 0. |
| | | | SC1_DAT | I/O | SmartCard1 data pin. |
| | | | EBI_DATA8 | I/O | External I/O data bus bit 8. |
| | | | SD1_DAT1 | I/O | SD/SDIO mode #1 data line bit 1. |
| | | | UART1_CTS | I | Clear to send input pin for UART1. |
| | | | SPI1_DI (SPI1_DATA1) | I (I/O) | SPI1 Data input pin. SPI1 data 1 in dual/quad mode. |
| 95 | 54 | 50 | PI.9 | I/O | General purpose digital I/O pin Port I Pin 9. |
| | | | VCAP_DATA1 | I | Sensor interface data bus bit 1. |
| | | | NAND_DATA1 | I/O | NAND flash data bus bit 1. |
| | | | eMMC_DATA1 | I/O | eMMC data line bit 1. |
| | | | SC1_PWR | O | SmartCard1 power pin. |
| | | | EBI_DATA9 | I/O | External I/O data bus bit 9. |
| | | | SD1_DAT2 | I/O | SD/SDIO mode #1 data line bit 2. |
| | | | UART4_TXD | O | Data transmitter output pin for UART4. |
| 96 | 55 | 51 | PI.10 | I/O | General purpose digital I/O pin Port I Pin 10. |
| | | | VCAP_DATA2 | I | Sensor interface data bus bit 2. |
| | | | NAND_DATA2 | I/O | NAND flash data bus bit 2. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| | | | eMMC_DATA2 | I/O | eMMC data line bit 2. |
| | | | SC1_CD | I | SmartCard1 card detect pin. |
| | | | EBI_DATA10 | I/O | External I/O data bus bit 10. |
| | | | SD1_DAT3 | I/O | SD/SDIO mode #1 data line bit 3. |
| | | | UART4_RXD | I | Data receiver input pin for UART4. |
| 97 | 56 | 52 | PI.11 | I/O | General purpose digital I/O pin Port I Pin 11. |
| | | | VCAP_DATA3 | I | Sensor interface data bus bit 3. |
| | | | NAND_DATA3 | I/O | NAND flash data bus bit 3. |
| | | | SC0_RST | O | SmartCard0 reset pin. |
| | | | EBI_DATA11 | I/O | External I/O data bus bit 11. |
| 98 | 57 | 53 | PI.12 | I/O | General purpose digital I/O pin Port I Pin 12. |
| | | | VCAP_DATA4 | I | Sensor interface data bus bit 4. |
| | | | NAND_DATA4 | I/O | NAND flash data bus bit 4. |
| | | | UART8_TXD | O | Data transmitter output pin for UART8. |
| | | | SC0_CLK | O | SmartCard0 clock pin. |
| | | | EBI_DATA12 | I/O | External I/O data bus bit 12. |
| | | | SD1_nCD | I | SD/SDIO mode #1 – card detect. |
| 99 | 58 | 54 | PI.13 | I/O | General purpose digital I/O pin Port I Pin 13. |
| | | | VCAP_DATA5 | I | Sensor interface data bus bit 5. |
| | | | NAND_DATA5 | I/O | NAND flash data bus bit 5. |
| | | | UART8_RXD | I | Data receiver input pin for UART8. |
| | | | SC0_DAT | I/O | SmartCard0 data pin. |
| | | | EBI_DATA13 | I/O | External I/O data bus bit 13. |
| | | | SD1_nPWR | O | SD/SDIO mode #1 – power enable. |
| 100 | 59 | 55 | PI.14 | I/O | General purpose digital I/O pin Port I Pin 14. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| | | | VCAP_DATA6 | I | Sensor interface data bus bit 6. |
| | | | NAND_DATA6 | I/O | NAND flash data bus bit 6. |
| | | | UART8_RTS | O | Request to send output pin for UART8. |
| | | | SC0_PWR | O | SmartCard0 power pin. |
| | | | EBI_DATA14 | I/O | External I/O data bus bit 14. |
| 101 | 60 | 56 | PI.15 | I/O | General purpose digital I/O pin Port I Pin 15. |
| | | | VCAP_DATA7 | I | Sensor interface data bus bit 7. |
| | | | NAND_DATA7 | I/O | NAND flash data bus bit 7. |
| | | | UART8_CTS | I | Clear to send input pin for UART8. |
| | | | SC0_CD | I | SmartCard0 card detect pin. |
| | | | EBI_DATA15 | I/O | External I/O data bus bit 15. |
| | | | CLK_OUT | O | Clock output pin. |
| 102 | - | - | CORE_VDD | P | MCU internal core power pin. |
| 103 | - | - | CORE_VSS | P | MCU internal core ground pin. |
| 104 | 61 | 57 | IO_VDD | P | MCU I/O power pin. |
| - | - | - | PG.15 | I/O | General purpose digital I/O pin Port G Pin 15. |
| | | | INT5 | I | External interrupt 5 input pin. |
| 105 | - | 58 | PB.0 | I/O | General purpose digital I/O pin Port B Pin 0. |
| | | | NAND_nCS1 | O | NAND flash chip select 1. |
| | | | UART5_TXD | O | Data transmitter output pin for UART5. |
| | | | SPI0_SS1 | O | 2nd SPI0 chip select pin. |
| | | | TM1_TGL | O | Enhanced TIMER toggle output pin. |
| 106 | - | 59 | PB.1 | I/O | General purpose digital I/O pin Port B Pin 1. |
| | | | NAND_RDY1 | I | NAND flash ready/busy channel 1. |
| | | | UART5_RXD | I | Data receiver input pin for UART5. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| | | | SPI1_SS1 | O | 2nd SPI1 chip select pin. |
| | | | TM1_CAP | I | Enhanced TIMER capture input pin. |
| 107 | 62 | 60 | PB.2 | I/O | General purpose digital I/O pin Port B Pin 2. |
| | | | UART6_TXD | O | Data transmitter output pin for UART6. |
| | | | PWM0 | O | PWM0 output pin. |
| | | | TM0_TGL | O | Enhanced TIMER toggle output pin. |
| 108 | 63 | 61 | PB.3 | I/O | General purpose digital I/O pin Port B Pin 3. |
| | | | UART6_RXD | I | Data receiver input pin for UART6. |
| | | | PWM1 | O | PWM1 output pin. |
| | | | TM0_CAP | I | Enhanced TIMER capture input pin. |
| 109 | - | 62 | PB.4 | I/O | General purpose digital I/O pin Port B Pin 4. |
| | | | UART6_RTS | O | Request to send output pin for UART6. |
| 110 | - | 63 | PB.5 | I/O | General purpose digital I/O pin Port B Pin 5. |
| | | | UART6_CTS | I | Clear to send input pin for UART6. |
| - | 64 | 64 | IO_VSS | P | MCU I/O ground pin. |
| 111 | 65 | 65 | PB.6 | I/O | General purpose digital I/O pin Port B Pin 6. |
| | | | SPI0_SS0 | O | 1st SPI0 chip select pin. |
| 112 | 66 | 66 | PB.7 | I/O | General purpose digital I/O pin Port B Pin 7. |
| | | | SPI0_CLK | O | SPI0 serial clock pin. |
| 113 | 67 | 67 | PB.8 | I/O | General purpose digital I/O pin Port B Pin 8. |
| | | | SPI0_DO (SPI0_DATA0) | O (I/O) | SPI0 Data out pin. SPI0 data 0 in dual/quad mode. |
| 114 | 68 | 68 | PB.9 | I/O | General purpose digital I/O pin Port B Pin 9. |
| | | | SPI0_DI (SPI0_DATA1) | I (I/O) | SPI0 Data input pin. SPI0 data 1 in dual/quad mode. |
| 115 | 69 | 69 | PB.10 | I/O | General purpose digital I/O pin Port B Pin 10. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| | | | UART10_TXD | O | Data transmitter output pin for UART10. |
| | | | SPI0_DATA2 | I/O | SPI0 data 2 in dual/quad mode. |
| | | | CAN0_RXD | I | CAN bus receiver0 input. |
| 116 | 70 | 70 | PB.11 | I/O | General purpose digital I/O pin Port B Pin 11. |
| | | | UART10_RXD | I | Data receiver input pin for UART10. |
| | | | SPI0_DATA3 | I/O | SPI0 data 3 in dual/quad mode. |
| | | | CAN0_TXD | O | CAN bus transmitter0 output. |
| 117 | 71 | 71 | PB.12 | I/O | General purpose digital I/O pin Port B Pin 12. |
| | | | UART10_TXD | O | Data transmitter output pin for UART10. |
| | | | SPI1_SS0 | O | 1st SPI1 chip select pin. |
| 118 | 72 | 72 | PB.13 | I/O | General purpose digital I/O pin Port B Pin 13. |
| | | | UART10_RXD | I | Data receiver input pin for UART10. |
| | | | SPI1_CLK | O | SPI1 serial clock pin. |
| 119 | 73 | 73 | PB.14 | I/O | General purpose digital I/O pin Port B Pin 14. |
| | | | UART10_RTS | O | Request to send output pin for UART10. |
| | | | SPI1_DO (SPI1_DATA0) | O (I/O) | SPI1 Data out pin. SPI1 data 0 in dual/quad mode. |
| 120 | 74 | 74 | PB.15 | I/O | General purpose digital I/O pin Port B Pin 15. |
| | | | UART10_CTS | I | Clear to send input pin for UART10. |
| | | | SPI1_DI (SPI1_DATA1) | I (I/O) | SPI1 Data input pin. SPI1 data 1 in dual/quad mode. |
| 121 | - | - | PG.4 | I/O | General purpose digital I/O pin Port G Pin 4. |
| | | | NAND_nCS1 | O | NAND flash chip select 1. |
| | | | UART7_TXD | O | Data transmitter output pin for UART7. |
| | | | SPI1_DATA2 | I/O | SPI1 data 2 in dual/quad mode. |
| 122 | - | - | PG.5 | I/O | General purpose digital I/O pin Port G Pin 5. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| | | | NAND_RDY1 | I | NAND flash ready/busy channel 1. |
| | | | UART7_RXD | I | Data receiver input pin for UART7. |
| | | | SPI1_DATA3 | I/O | SPI1 data 3 in dual/quad mode. |
| 123 | - | - | PC.0 | I/O | General purpose digital I/O pin Port C Pin 0. |
| | | | NAND_DATA0 | I/O | NAND flash data bus bit 0. |
| | | | eMMC_DATA0 | I/O | eMMC data line bit 0. |
| 124 | - | - | PC.1 | I/O | General purpose digital I/O pin Port C Pin 1. |
| | | | NAND_DATA1 | I/O | NAND flash data bus bit 1. |
| | | | eMMC_DATA1 | I/O | eMMC data line bit 1. |
| 125 | - | - | PC.2 | I/O | General purpose digital I/O pin Port C Pin 2. |
| | | | NAND_DATA2 | I/O | NAND flash data bus bit 2. |
| | | | eMMC_DATA2 | I/O | eMMC data line bit 2. |
| 126 | - | - | PC.3 | I/O | General purpose digital I/O pin Port C Pin 3. |
| | | | NAND_DATA3 | I/O | NAND flash data bus bit 3. |
| | | | eMMC_DATA3 | I/O | eMMC data line bit 3. |
| 127 | - | - | PC.4 | I/O | General purpose digital I/O pin Port C Pin 4. |
| | | | NAND_DATA4 | I/O | NAND flash data bus bit 4. |
| | | | eMMC_CMD | I/O | eMMC command/Response. |
| 128 | - | - | PC.5 | I/O | General purpose digital I/O pin Port C Pin 5. |
| | | | NAND_DATA5 | I/O | NAND flash data bus bit 5. |
| | | | eMMC_CLK | O | eMMC clock output. |
| 129 | - | - | PC.6 | I/O | General purpose digital I/O pin Port C Pin 6. |
| | | | NAND_DATA6 | I/O | NAND flash data bus bit 6. |
| | | | UART10_TXD | O | Data transmitter output pin for UART10. |
| | | | TM0_TGL | O | Enhanced TIMER toggle output pin. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| 130 | - | - | PC.7 | I/O | General purpose digital I/O pin Port C Pin 7. |
| | | | NAND_DATA7 | I/O | NAND flash data bus bit 7. |
| | | | UART10_RXD | I | Data receiver input pin for UART10. |
| | | | TM0_CAP | I | Enhanced TIMER capture input pin. |
| 131 | - | - | PC.8 | I/O | General purpose digital I/O pin Port C Pin 8. |
| | | | NAND_nCS0 | O | NAND flash chip select 0. |
| | | | UART10_RTS | O | Request to send output pin for UART10. |
| | | | TM1_TGL | O | Enhanced TIMER toggle output pin. |
| 132 | - | - | PC.9 | I/O | General purpose digital I/O pin Port C Pin 9. |
| | | | NAND_ALE | O | NAND flash address latch enable. |
| | | | UART10_CTS | I | Clear to send input pin for UART10. |
| | | | TM1_CAP | I | Enhanced TIMER capture input pin. |
| 133 | - | - | PC.10 | I/O | General purpose digital I/O pin Port C Pin 10. |
| | | | NAND_CLE | O | NAND flash command latch enable. |
| | | | UART4_TXD | O | Data transmitter output pin for UART4. |
| | | | TM2_TGL | O | Enhanced TIMER toggle output pin. |
| 134 | - | - | PC.11 | I/O | General purpose digital I/O pin Port C Pin 11. |
| | | | NAND_nWE | O | NAND flash write enable. |
| | | | UART4_RXD | I | Data receiver input pin for UART4. |
| | | | TM2_CAP | I | Enhanced TIMER capture input pin. |
| 135 | - | - | PC.12 | I/O | General purpose digital I/O pin Port C Pin 12. |
| | | | NAND_nRE | O | NAND flash read enable. |
| | | | UART4_RTS | O | Request to send output pin for UART4. |
| | | | TM3_TGL | O | Enhanced TIMER toggle output pin. |
| 136 | - | - | PC.13 | I/O | General purpose digital I/O pin Port C Pin 13. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| | | | NAND_RDY0 | I | NAND flash ready/busy channel 0. |
| | | | UART4_CTS | I | Clear to send input pin for UART4. |
| | | | TM3_CAP | I | Enhanced TIMER capture input pin. |
| 137 | - | - | PC.14 | I/O | General purpose digital I/O pin Port C Pin 14. |
| | | | NAND_nWP | O | NAND flash write protect. |
| | | | PWM0 | O | PWM0 output pin. |
| 138 | 75 | 75 | PJ.3 | I/O | General purpose digital I/O pin Port J Pin 3. |
| | | | JTAG_TDO | O | JTAG test data out. |
| 139 | 76 | 76 | PJ.0 | I/O | General purpose digital I/O pin Port J Pin 0. |
| | | | JTAG_TCK | O | JTAG test clock. |
| 140 | 77 | 77 | PJ.1 | I/O | General purpose digital I/O pin Port J Pin 1. |
| | | | JTAG_TMS | O | JTAG test mode select. |
| 141 | 78 | 78 | PJ.2 | I/O | General purpose digital I/O pin Port J Pin 2. |
| | | | JTAG_TDI | I | JTAG test data in. |
| 142 | 79 | 79 | PJ.4 | I/O | General purpose digital I/O pin Port J Pin 4. |
| | | | JTAG_nTRST | O | JTAG Reset. |
| 143 | 80 | 80 | nRESET | I | External reset input: active LOW, with an internal pull-up. Set this pin low reset to initial state. |
| | | | WDT_nRST | O | Watch dog timer external reset output pin. Open-drain. |
| 144 | 81 | 81 | IO_VDD | P | MCU I/O power pin. |
| 145 | - | - | CORE_VSS | P | MCU internal core ground pin. |
| 146 | 82 | 82 | CORE_VDD | P | MCU internal core power pin. |
| 147 | - | - | DDR_VSS | P | DDR ground pin. |
| 148 | 83 | 83 | DDR_VDD | P | DDR power pin. |
| 149 | - | - | DDR_VSS | P | DDR ground pin. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| 150 | 83 | 83 | DDR_VDD | P | DDR power pin. |
| 151 | - | - | DDR_VSS | P | DDR ground pin. |
| 152 | 84 | 84 | DDR_VDD | P | DDR power pin. |
| - | - | - | DDR_VSS | P | DDR ground pin. |
| 153 | - | - | CORE_VSS | P | MCU internal core ground pin. |
| 154 | 85 | 85 | IO_VDD | P | MCU I/O power pin. |
| 155 | 86 | 86 | PD.0 | I/O | General purpose digital I/O pin Port D Pin 0. |
| | | | SD0_CMD | O | SD/SDIO mode #0 command/response. |
| 156 | 87 | 87 | PD.1 | I/O | General purpose digital I/O pin Port D Pin 1. |
| | | | SD0_CLK | O | SD/SDIO mode #0 clock. |
| 157 | 88 | 88 | PD.2 | I/O | General purpose digital I/O pin Port D Pin 2. |
| | | | SD0_DAT0 | I/O | SD/SDIO mode #0 data line bit 0. |
| 158 | 89 | 89 | PD.3 | I/O | General purpose digital I/O pin Port D Pin 3. |
| | | | SD0_DAT1 | I/O | SD/SDIO mode #0 data line bit 1. |
| 159 | 90 | 90 | PD.4 | I/O | General purpose digital I/O pin Port D Pin 4. |
| | | | SD0_DAT2 | I/O | SD/SDIO mode #0 data line bit 2. |
| 160 | 91 | 91 | PD.5 | I/O | General purpose digital I/O pin Port D Pin 5. |
| | | | SD0_DAT3 | I/O | SD/SDIO mode #0 data line bit 3. |
| 161 | 92 | 92 | PD.6 | I/O | General purpose digital I/O pin Port D Pin 6. |
| | | | SD0_nCD | I | SD/SDIO mode #0 card detect. |
| 162 | 93 | 93 | PD.7 | I/O | General purpose digital I/O pin Port D Pin 7. |
| 163 | 94 | 94 | CORE_VDD | P | MCU internal core power pin. |
| 164 | - | - | PLL_VDD | P | PLL power input pin. |
| 165 | 95 | 95 | PLL_VSS | P | PLL ground. |
| 166 | - | - | CORE_VDD | P | MCU internal core power pin. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| 167 | - | - | PH.3 | I/O | General purpose digital I/O pin Port H Pin 3. |
| | | | I2C1_SDA | I/O | $I^2C1$ data input/output pin. |
| | | | UART9_RXD | I | Data receiver input pin for UART9. |
| | | | CAN0_TXD | O | CAN bus transmitter0 output. |
| | | | PWM3 | O | PWM3 output pin. |
| | | | INT3 | I | External interrupt 3 input pin. |
| 168 | - | - | PH.2 | I/O | General purpose digital I/O pin Port H Pin 2. |
| | | | I2C1_SCL | O | $I^2C1$ clock pin. |
| | | | UART9_TXD | O | Data transmitter output pin for UART9. |
| | | | CAN0_RXD | I | CAN bus receiver0 input. |
| | | | PWM2 | O | PWM2 output pin. |
| | | | INT2 | I | External interrupt 2 input pin. |
| 169 | - | - | PE.13 | I/O | General purpose digital I/O pin Port E Pin 13. |
| | | | UART8_CTS | I | Clear to send input pin for UART8. |
| | | | UART3_RXD | I | Data receiver input pin for UART3. |
| | | | CLK_OUT | O | Reference Clock Output. |
| 170 | - | - | PE.12 | I/O | General purpose digital I/O pin Port E Pin 12. |
| | | | UART8_RTS | O | Request to send output pin for UART8. |
| | | | UART3_TXD | O | Data transmitter output pin for UART3. |
| 171 | - | - | PE.11 | I/O | General purpose digital I/O pin Port E Pin 11. |
| | | | RMII1_RXERR | I | RMII1 receive data error. |
| | | | UART8_RXD | I | Data receiver input pin for UART8. |
| 172 | - | - | PE.10 | I/O | General purpose digital I/O pin Port E Pin 10. |
| | | | RMII1_CRSDV | I | RMII1 carrier sense / receive data valid. |
| | | | UART8_TXD | O | Data transmitter output pin for UART8. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| 173 | - | - | PE.9 | I/O | General purpose digital I/O pin Port E Pin 9. |
| | | | RMII1_RXDATA1 | I | RMII1 receive data bus bit 1. |
| | | | SD1_nPWR | O | SD/SDIO mode #1 power enable. |
| | | | UART1_CD | I | Carrier detect input pin for UART1. |
| 174 | - | - | PE.8 | I/O | General purpose digital I/O pin Port E Pin 8. |
| | | | RMII1_RXDATA0 | I | RMII1 receive data bus bit 0. |
| | | | SD1_nCD | I | SD/SDIO mode #1 card detect. |
| | | | UART1_RI | I | Ring indicator input pin for UART1. |
| 175 | - | - | PE.7 | I/O | General purpose digital I/O pin Port E Pin 7. |
| | | | RMII1_REFCLK | I | RMII1 reference clock. |
| | | | SD1_DAT3 | I/O | SD/SDIO mode #1 data line bit 3. |
| | | | UART1_DSR | I | Data set ready input pin for UART1. |
| 176 | - | - | PE.6 | I/O | General purpose digital I/O pin Port E Pin 6. |
| | | | RMII1_TXEN | O | RMII1 transmit enable. |
| | | | SD1_DAT2 | I/O | SD/SDIO mode #1 data line bit 2. |
| | | | UART1_DTR | O | Data terminal ready output pin for UART1. |
| 177 | - | - | PE.5 | I/O | General purpose digital I/O pin Port E Pin 5. |
| | | | RMII1_TXDATA1 | O | RMII1 transmit data bus bit 1. |
| | | | SD1_DAT1 | I/O | SD/SDIO mode #1 data line bit 1. |
| | | | UART1_CTS | I | Clear to send input pin for UART1. |
| | | | CLK_OUT | O | Reference Clock Output. |
| 178 | - | - | PE.4 | I/O | General purpose digital I/O pin Port E Pin 4. |
| | | | RMII1_TXDATA0 | O | RMII1 Transmit Data bus |
| | | | SD1_DAT0 | I/O | SD/SDIO mode #1  ata line bit 0. |
| | | | UART1_RTS | O | Request to send output pin for UART1. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| 179 | - | - | PE.3 | I/O | General purpose digital I/O pin Port E Pin 3. |
| | | | RMII1_MDIO | I/O | RMII1 Management Data I/O |
| | | | SD1_CLK | O | SD/SDIO mode #1 clock. |
| | | | UART1_RXD | I | Data receiver input pin for UART1. |
| 180 | - | - | PE.2 | I/O | General purpose digital I/O pin Port E Pin 2. |
| | | | RMII1_MDC | O | RMII1 Management Data Clock |
| | | | SD1_CMD | O | SD/SDIO mode #1 command/response. |
| | | | UART1_TXD | O | Data transmitter output pin for UART1. |
| 181 | 96 | 96 | PE.1 | I/O | General purpose digital I/O pin Port E Pin 1. |
| | | | UART0_RXD | I | Data receiver input pin for UART0. |
| 182 | 97 | 97 | PE.0 | I/O | General purpose digital I/O pin Port E Pin 0. |
| | | | UART0_TXD | O | Data transmitter output pin for UART0. |
| 183 | 98 | 98 | IO_VDD | P | MCU I/O power pin. |
| 184 | 99 | 99 | XT1_IN | I | External 12MHz crystal input pin. |
| 185 | 100 | 100 | XT1_OUT | O | External 12MHz crystal output pin. |
| 186 | 101 | 101 | IO_VSS | P | MCU I/O ground pin. |
| 187 | 102 | 102 | PF.9 | I/O | General purpose digital I/O pin Port F Pin 9. |
| | | | RMII0_RXERR | I | RMII0 receive data error. |
| 188 | 103 | 103 | PF.8 | I/O | General purpose digital I/O pin Port F Pin 8. |
| | | | RMII0_CRSDV | I | RMII0 carrier sense / receive data valid. |
| 189 | 104 | 104 | PF.7 | I/O | General purpose digital I/O pin Port F Pin 7. |
| | | | RMII0_RXDATA1 | I | RMII0 receive data bus bit 1. |
| 190 | 105 | 105 | PF.6 | I/O | General purpose digital I/O pin Port F Pin 6. |
| | | | RMII0_RXDATA0 | I | RMII0 receive data bus bit 0. |
| 191 | 106 | 106 | PF.5 | I/O | General purpose digital I/O pin Port F Pin 5. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| | | | RMII0_REFCLK | I | RMII0 reference clock. |
| 192 | 107 | 107 | PF.4 | I/O | General purpose digital I/O pin Port F Pin 4. |
| | | | RMII0_TXEN | O | RMII0 transmit enable. |
| 193 | 108 | 108 | PF.3 | I/O | General purpose digital I/O pin Port F Pin 3. |
| | | | RMII0_TXDATA1 | O | RMII0 transmit data bus bit 1. |
| 194 | 109 | 109 | PF.2 | I/O | General purpose digital I/O pin Port F Pin 2. |
| | | | RMII0_TXDATA0 | O | RMII0 Transmit Data bus bit 0. |
| 195 | 110 | 110 | PF.1 | I/O | General purpose digital I/O pin Port F Pin 1. |
| | | | RMII0_MDIO | I/O | RMII0 Management Data I/O |
| 196 | 111 | 111 | PF.0 | I/O | General purpose digital I/O pin Port F Pin 0. |
| | | | RMII0_MDC | O | RMII0 Management Data Clock |
| 197 | 112 | 112 | PH.1 | I/O | General purpose digital I/O pin Port H Pin 1. |
| | | | USB_OVRCUR | I | USB overcurrent |
| | | | INT1 | I | External interrupt 1 input pin. |
| 198 | 113 | 113 | PH.0 | I/O | General purpose digital I/O pin Port H Pin 0. |
| | | | USB0_VBUSVLD | I | USB0 VBUS valid. |
| | | | INT0 | I | External interrupt 0 input pin. |
| 199 | 114 | 114 | PF.10 | I/O | General purpose digital I/O pin Port F Pin 10. |
| | | | USB_PWREN | O | USB host output power control pin for LQFP128 package only. |
| 200 | - | - | PE.15 | I/O | General purpose digital I/O pin Port E Pin 15. |
| | | | USB1_PWREN | O | USB1 host output power control pin. |
| 201 | - | - | PE.14 | I/O | General purpose digital I/O pin Port E Pin 14. |
| | | | USB0_PWREN | O | USB0 host output power control pin. |
| 202 | - | - | CORE_VSS | P | MCU internal core ground pin. |

| NUC972D FxxY | NUC976D KxxY | NUC977D KxxY | Pin Name | Pin Type | Description |
|---|---|---|---|---|---|
| - | 115 | 115 | CORE_VDD | P | MCU internal core power pin. |
| 203 | 116 | 116 | CORE_VDD | P | MCU internal core power pin. |
| 204 | 117 | 117 | USBPLL1_VDD | P | USB1 PLL power pin. |
| 205 | - | - | USB1_VSS | P | USB1 ground pin. |
| 206 | 118 | 118 | USB1_DM | I/O | USB1 differential signal D-. |
| 207 | 119 | 119 | USB1_DP | I/O | USB1 differential signal D+. |
| 208 | 120 | 120 | USB1_VDD | P | USB1 I/O power pin. |
| 209 | 121 | 121 | USB1_REXT | I | USB1 module reference Resister. |
| 210 | 122 | 122 | USBPLL0_VDD | P | USB0 PLL power pin. |
| 211 | 123 | 123 | USB0_VSS | P | USB0 ground pin. |
| 212 | 124 | 124 | USB0_DM | I/O | USB0 differential signal D-. |
| 213 | 125 | 125 | USB0_DP | I/O | USB0 differential signal D+. |
| 214 | 126 | 126 | USB0_VDD | P | USB0 I/O power pin. |
| 215 | 127 | 127 | USB0_REXT | I | USB0 module reference Resister. |
|  | 128 | 128 | IO_VSS | P | MCU I/O ground pin. |
| 216 | 1 | 1 | USB0_ID | I | USB0 Host/Device select. |

**Note:** Pin Type I=Digital Input, O=Digital Output; AI=Analog Input; P=Power Pin; AP=Analog Power

# 4 BLOCK DIAGRAM

## 4.1 NUC970 Series Block Diagram

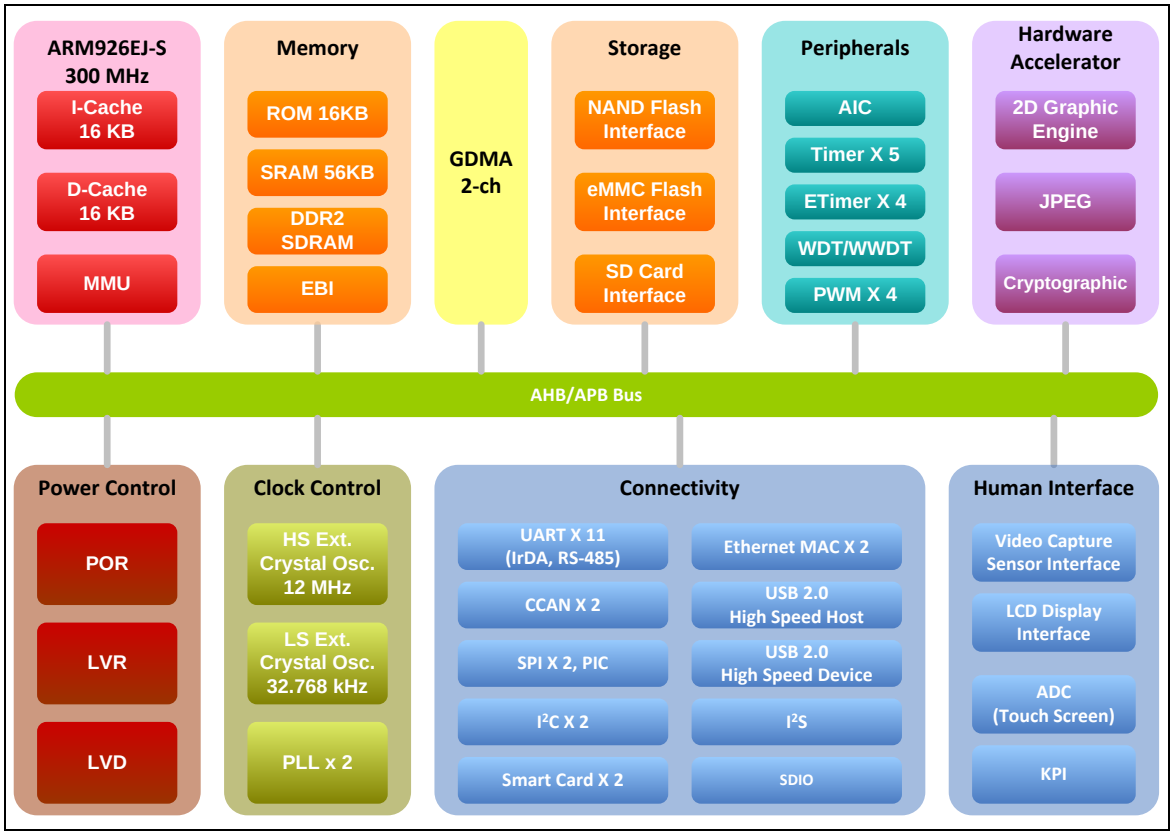



Figure 4.1-1 NUC970 Series Block Diagram

# 5 FUNCTIONAL DESCRIPTION

## 5.1 ARM® ARM926EJ-S CPU Core

### 5.1.1 Overview

The ARM926EJ-S CPU core is a member of the ARM9 family of general-purpose microprocessors. The ARM926EJ-S CPU core is targeted at multi-tasking applications where full memory management, high performance, and low power are all important.

The ARM926EJ-S CPU core supports the 32-bit ARM and 16-bit Thumb instruction sets, enabling the user to choose between high performance and high code density. The ARM926EJ-S CPU core includes features for efficient execution of Java byte codes, providing Java performance similar to JIT, but without the associated code overhead.

The ARM926EJ-S processor provides support for external coprocessor enabling floating-point or other application-specific hardware acceleration to be added. The ARM926EJ-S CPU core implements ARM architecture version 5TEJ.

The ARM926EJ-S processor has a Harvard cached architecture and provides a complete high-performance processor subsystem, including:

- An ARM9EJ-S integer core.
- A Memory Management Unit (MMU).
- Separate instruction and data cache.
- Separate instruction and data AMBA AHB bus interfaces.

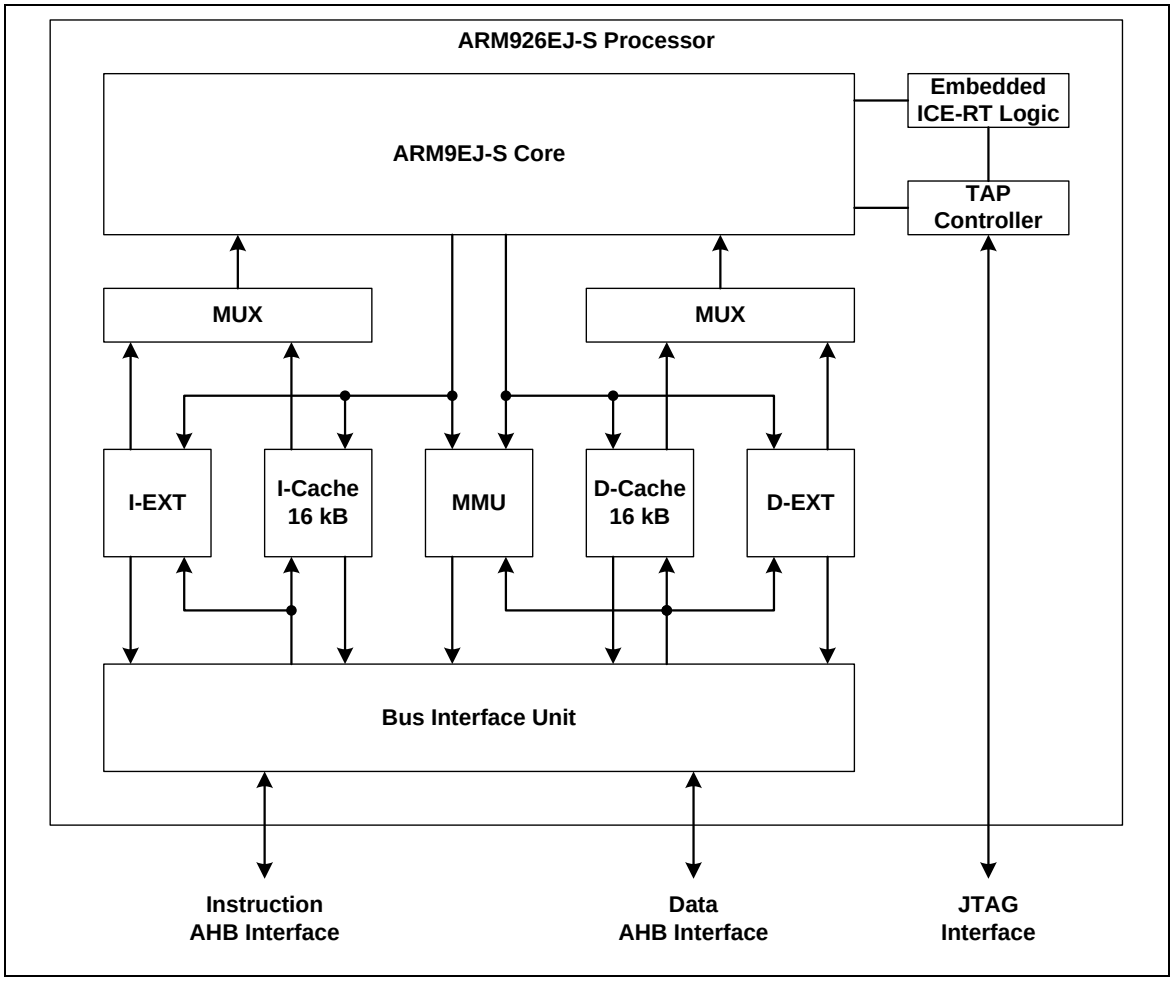



Figure 5.1-1 ARM926EJ-S Block Diagram

### 5.1.2 System Control Coprocessor (CP15)

The system control coprocessor (CP15) is used to configure and control the ARM926EJ-S processor. The caches, Memory Management Unit (MMU), and most other system options are controlled using CP15 registers. User can only access CP15 registers with MRC and MCR instruction in a privileged mode. Access CP15 registers with CDP, LDC, STC, MCRR, and MRRC instructions and unprivileged MRC or MCR instruction causes the undefined instruction exception to be taken.

### 5.1.3 Memory Management Unit (MMU)

The ARM926EJ-S MMU is an ARM architecture v5 MMU. It provides virtual memory features required by systems operating on platforms such as Symbian OS, WindowsCE, and Linux. A single set of two-level page tables stored in main memory is used to control the address translation, permission checks, and memory region attributes for both data and instruction accesses.

The MMU uses a single unified Translation Lookaside Buffer (TLB) to cache the information held in the page tables. To support both sections and pages, there are two levels of address

translation. The MMU puts the translated physical addresses into the MMU Translation Lookaside Buffer TLB.

The MMU TLB has two parts, the main TLB and the lockdown TLB. The main TLB is a two-way, set-associative cache for page table information. It has 32 entries per way for a total of 64 entries. The lockdown TLB is an eight-entry fully-associative cache that contains locked TLB entries. Locking TLB entries can ensure that a memory access to a given region never incurs the penalty of a page table walk.

The MMU features are:

- Standard ARM architecture v4 and v5 MMU mapping sizes, domains, and access protection scheme
- Mapping sizes are 1MB (sections), 64KB (large pages), 4KB (small pages), and 1KB (tiny pages)
- Access permissions for large pages and small pages can be specified separately for each quarter of the page (subpage permissions)
- Hardware pag e table walks
- Invalidate entire TLB using CP15 c8
- Invalidate TLB entry selected by MVA, using CP15 c8
- Lockdown of TLB entries using CP15 c10.

### 5.1.4 Caches and Write Buffer

The ARM926EJ-S processor includes an Instruction Cache (I-Cache), a Data Cache (D-Cache) and a write buffer. The size of I-Cache and D-Cache in this chip is 16 KB, respectively.

The caches features are:

- The caches are virtual index, virtual tag, addressed using the Modified Virtual Address (MVA). This enables the avoidance of cache cleaning and/or invalidating on context switch.
- The caches are four-way set associative, with a cache line length of eight words per line (32 bytes per line), and with two dirty bits in the D-Cache.
- The D-Cache supports write-through and write-back (or copy back) cache operations, selected by memory region using the C and B bits in the MMU translation tables.
- Allocate on read-miss is supported. The caches perform critical-word first cache refilling.
- Pseudo-random or round-robin replacement selectable by the RR bit in CP15 c1.
- Cache lockdown registers enable control over which cache ways are used for allocation on a linefill, providing a mechanism for both lockdown and controlling cache pollution.
- The D-Cache stores the Physical Address (PA) tag corresponding to each D-Cache entry in the tag RAM for use during cache line write-backs, in addition to the Virtual Address tag stored in the tag RAM. This means that the MMU is not involved in D-Cache write-back operations, removing the possibility of TLB misses related to the write-back address.
- The PLD data preload instruction does not cause data cache linefills. It is treated as a NOP instruction.

### 5.1.5 Bus Interface Unit

The ARM926EJ-S Bus Interface Unit (BIU) arbitrates and schedules AHB requests. The BIU contains separate masters for both instruction and data access enabling complete AHB system flexibility. Each master is a fully compliant AHB bus master and implements the master functions as defined in the AMBA Specification (Rev 2.0).

To increase system performance, write buffers are used to prevent AHB writes stalling the ARM926EJ-S system.

### 5.1.6 Power Management

The ARM926EJ-S processor can be put into a low-power state by the wait for interrupt instruction:

MCR p15, 0, <Rd>, c7, c0, 4

This instruction switches the ARM926EJ-S processor into a low-power state until either an interrupt (IRQ or FIQ) or a debug request occurs.

In wait for interrupt mode, all internal ARM926EJ-S clocks are stopped. The switch into the low-power state is delayed until all write buffers have been drained, and the ARM926EJ-S memory system is in a quiescent state.

## 5.2 System Manager

### 5.2.1 Overview

The system management describes following information and functions.

- System Resets
- System Power Architecture
- System Memory Map
- System management registers for Product Identifier (PDID), Power-On Setting, System Wake-Up, Reset Control for on-chip controllers/peripherals, and multi-function pin control.
- System Control registers

### 5.2.2 System Reset

The system reset can be issued by one of the below listed events. For these reset event flags can be read by RSTSTS register.

- Power-On Reset
- Low level on the /RESET pin
- Watchdog Time Out Reset
- Low Voltage Reset
- CPU Reset
- System Reset

### 5.2.3 System Power Distribution

In this chip, the power distribution is divided into six segments.

- Analog power from AVDD provides 3.3V voltage to analog components operation. These analog components including POR33, 12-bit SAR-ADC, LVD and LVR.
- Digital power from CORE_VDD provides 1.2V voltage to POR12, APLL, APLL, SRAM (56 kB) and all digital logic except RTC.
- Digital power from RTC_VDD provides 3.3V voltage to LXT and RTC logic.
- USB PHY power from USB0_VDD, USBPLL0_VDD provides 3.3V and 1.2 respectively to USB 2.0 PHY 0, while USB1_VDD, USBPLL1_VDD provides 3.3V and 1.2 respectively to USB 2.0 PHY 1.
- IO power from DDR_VDD provides 1.8V to I/O pins used to connect DDR2 SDRAM.
- IO power from IO_VDD provides 3.3V to MTP memory, HXT and I/O pins (PA ~ PJ).

The following diagram shows the power distribution of the NUC970 series.

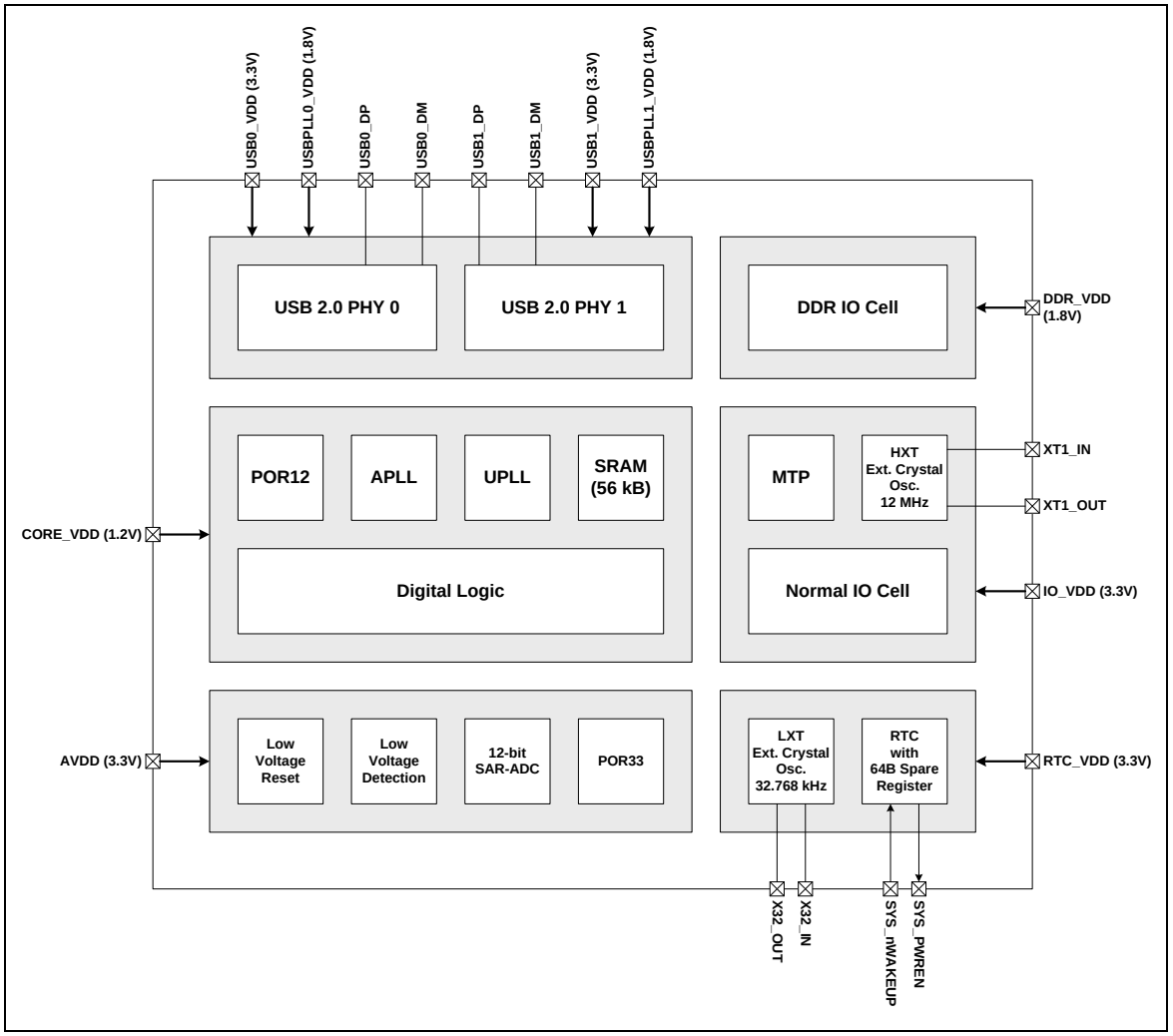



Figure 5.2-1 NUC970 Series Power Distribution Diagram

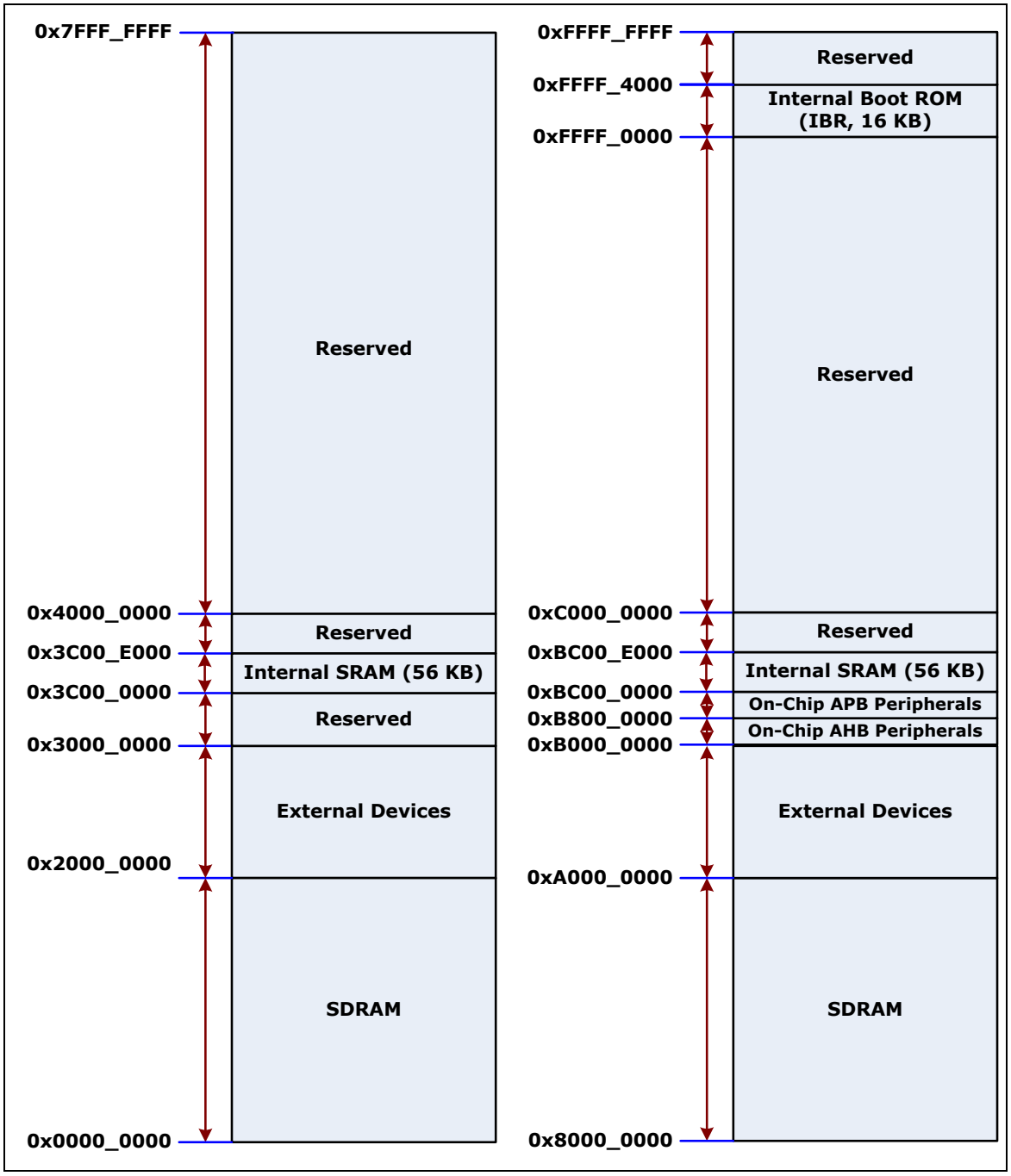
### 5.2.4 System Memory Map

This chip support only little-endian data format and provides 4G-byte addressing space. The diagram shown below describes the memory space definition.

The memory space from 0x0000_0000 to 0x2FFF_FFFF is for SDRAM and external devices. The memory space from 0x3C00_0000 to 0x3C00_DFFF is for embedded 56 k-byte SRAM. The memory space for On-Chip Controllers and Peripherals is from 0xB000_0000 to 0xBBFF_FFFF while the memory space from 0xFFFF_0000 to 0xFFFF_FFFF is for 16 k-byte internal Boot ROM.

This chip provides the shadow memory function. The memory space from 0x8000_0000 to 0xAFFF_FFFF is the shadow memory space for memory space from 0x0000_0000 to 0x2FFF_FFFF. The memory space from 0xBC00_0000 to 0xBC00_DFFF is the shadow memory space for memory space from 0x3C00_0000 to 0x3C00_DFFF. If the DMA of On-Chip Controller wants to access this 56 k-byte embedded SRAM, it's necessary to use memory space from 0xBC00_0000 to 0xBC00_DFFF

The reserved memory space is un-accessible. Chip's behavior is undefined and unpredictable while accessing to reserved memory space.


0x7FFF_FFFF
Reserved
0x4000_0000
Reserved
0x3C00_E000
Internal SRAM (56 KB)
0x3C00_0000
Reserved
0x3000_0000
External Devices
0x2000_0000
SDRAM
0x0000_0000
0xFFFF_FFFF
Reserved
0xFFFF_4000
Internal Boot ROM (IBR, 16 KB)
0xFFFF_0000
Reserved
0xC000_0000
Reserved
0xBC00_E000
Internal SRAM (56 KB)
0xBC00_0000
On-Chip APB Peripherals
0xB800_0000
On-Chip AHB Peripherals
0xB000_0000
External Devices
0xA000_0000
SDRAM
0x8000_0000

The addressing space assigned to each on-chip controller or peripheral described in the table shown below. The detailed register definition, addressing space, and programming details will be described in the following sections.

| Addressing Space | Token | Modules |
|---|---|---|
| **SDRAM, External Devices and SRAM Memory Space** | | |
| 0x0000_0000 – 0x1FFF_FFFF | SDRAM_BA | SDRAM Memory Space |
| 0x2000_0000 – 0x2FFF_FFFF | EXDEV_BA | External Devices Memory Space |
| 0x3C00_0000 – 0x3C00_DFFF | SRAM_BA | SRAM Memory Space (56 KB) |
| **Internal Boot ROM (IBR) Memory Space (0xFFFF_0000 ~ 0xFFFF_3FFF)** | | |
| 0xFFFF_0000 – 0xFFFF_3FFF | IBR_BA | Internal Boot ROM (IBR) Memory Space (16 KB) |
| **AHB Modules Memory Space (0xB000_0000 – 0xB7FF_FFFF)** | | |
| 0xB000_0000 – 0xB000_01FF | SYS_BA | System Global Control Registers |
| 0xB000_0200 – 0xB000_02FF | CLK_BA | Clock Control Registers |
| 0xB000_1000 – 0xB000_17FF | EBI_BA | EBI Control Registers |
| 0xB000_1800 – 0xB000_1FFF | SDIC_BA | SDRAM (SDR/DDR/DDR2) Control Registers |
| 0xB000_2000 – 0xB000_2FFF | EMAC0_BA | Ethernet MAC 0 Control Registers |
| 0xB000_3000 – 0xB000_3FFF | EMAC1_BA | Ethernet MAC 1 Control Registers |
| 0xB000_4000 – 0xB000_4FFF | GDMA_BA | GDMA Control Registers |
| 0xB000_5000 – 0xB000_5FFF | EHCI_BA | USB EHCI Host Control Registers |
| 0xB000_6000 – 0xB000_6FFF | USBD_BA | USB Device Control Registers |
| 0xB000_7000 – 0xB000_7FFF | OHCI_BA | USB OHCI Host Control Registers |
| 0xB000_8000 – 0xB000_8FFF | LCD_BA | LCD Display Control Registers |
| 0xB000_9000 – 0xB000_9FFF | I2S_BA | $I^2S$ Interface Control Registers |
| 0xB000_A000 – 0xB000_AFFF | JPEG_BA | JPEG Codec Control Registers |
| 0xB000_B000 – 0xB000_BFFF | GE2D_BA | 2D Graphic Engine Control Registers |
| 0xB000_C000 – 0xB000_CFFF | SDH_BA | SD/SDIO Host Control Registers |
| 0xB000_D000 – 0xB000_DFFF | FMI_BA | Flash Memory Interface (FMI) Control Registers |
| 0xB000_E000 – 0xB000_EFFF | CAP_BA | Capture Sensor Interface Control Registers |
| 0xB000_F000 – 0xB000_FFFF | CRYPTO_BA | Cryptographic Accelerator Control Registers |
| **APB Modules Memory Space (0xB800_0000 ~ 0xBBFF_FFFF)** | | |
| 0xB800_0000 – 0xB800_00FF | UART0_BA | UART 0 Control Registers |
| 0xB800_0100 – 0xB800_01FF | UART1_BA | UART 1 Control Registers (High-Speed UART) |
| 0xB800_0200 – 0xB800_02FF | UART2_BA | UART 2 Control Registers (High-Speed UART) |
| 0xB800_0300 – 0xB800_03FF | UART3_BA | UART 3 Control Registers |
| 0xB800_0400 – 0xB800_04FF | UART4_BA | UART 4 Control Registers (High-Speed UART) |

| | | |
|---|---|---|
| 0xB800_0500 – 0xB800_05FF | UART5_BA | UART 5 Control Registers |
| 0xB800_0600 – 0xB800_06FF | UART6_BA | UART 6 Control Registers (High-Speed UART) |
| 0xB800_0700 – 0xB800_07FF | UART7_BA | UART 7 Control Registers |
| 0xB800_0800 – 0xB800_08FF | UART8_BA | UART 8 Control Registers (High-Speed UART) |
| 0xB800_0900 – 0xB800_09FF | UART9_BA | UART 9 Control Registers |
| 0xB800_0A00 – 0xB800_0AFF | UART10_BA | UART 10 Control Registers (High-Speed UART) |
| 0xB800_1000 – 0xB800_10FF | TIMER_BA | Timer Control Registers |
| 0xB800_1400 – 0xB800_14FF | ETIMER0_BA | Enhance Timer 0 Control Registers |
| 0xB800_1500 – 0xB800_15FF | ETIMER1_BA | Enhance Timer 1 Control Registers |
| 0xB800_1600 – 0xB800_16FF | ETIMER2_BA | Enhance Timer 2 Control Registers |
| 0xB800_1700 – 0xB800_17FF | ETIMER3_BA | Enhance Timer 3 Control Registers |
| 0xB800_1800 – 0xB800_18FF | WDT_BA | Watch-Dog Timer Control Registers |
| 0xB800_1900 – 0xB800_19FF | WWDT_BA | Windowed Watch-Dog Timer Control Registers |
| 0xB800_2000 – 0xB800_2FFF | AIC_BA | Advance Interrupt Control Registers |
| 0xB800_3000 – 0xB800_3FFF | GPIO_BA | GPIO Control Registers |
| 0xB800_4000 – 0xB800_4FFF | RTC_BA | Real Time Clock (RTC) Control Registers |
| 0xB800_5000 – 0xB800_53FF | SC0_BA | Smart Card 0 Control Registers |
| 0xB800_5400 – 0xB800_57FF | SC1_BA | Smart Card 1 Control Registers |
| 0xB800_6000 – 0xB800_60FF | I2C0_BA | $I^2C$ 0 Control Registers |
| 0xB800_6100 – 0xB800_61FF | I2C1_BA | $I^2C$ 1 Control Registers |
| 0xB800_6200 – 0xB800_62FF | SPI0_BA | SPI 0 Control Registers |
| 0xB800_6300 – 0xB800_63FF | SPI1_BA | SPI 1 Control Registers |
| 0xB800_7000 – 0xB800_7FFF | PWM_BA | PWM Control Registers |
| 0xB800_8000 – 0xB800_8FFF | KPI_BA | KPI Control Registers |
| 0xB800_A000 – 0xB800_AFFF | ADC_BA | ADC Control Registers |
| 0xB800_B000 – 0xB800_B3FF | CAN0_BA | CAN 0 Control Registers |
| 0xB800_B400 – 0xB800_B7FF | CAN1_BA | CAN 1 Control Registers |
| 0xB800_C000 – 0xB800_CFFF | MTP_BA | MTP Control Registers |

Table 5.2-1 Address Space Assignments for On-Chip Controllers

### 5.2.5 Power-On Setting

After power on reset, Power-On setting registers are latched to configure this chip. The table shown below describes the definition of each power-on setting bit.

| Power-On Setting Pin | Description | Power-On Setting Register Bit |
|---|---|---|
| USB0_ID | **USB Port 0 Role Selection**<br>0 = USB Port 0 act as a USB host.<br>1 = USB Port 0 act as a USB device. | PWRON[16] |
| PA[1:0] | **Boot Source Selection**<br>00 = Boot from USB and pin PH.0 used as the USB0_VBUSVLD.<br>01 = Boor from eMMC and pin PC[5:0] or PI[10:5] used as the eMMC functionality[1].<br>10 = Boot from NAND Flash and pin PC[14:0] or PI[15:1] used as the NAND functionality[2,3].<br>11 = Boot from SPI Flash and pin PB[9:6] used as the SPI0 functionality.<br><br>Note 1: PC[5:0] used as the eMMC functionality in NUC972 while PI[10:5] used as the eMMC functionality in NUC976 and NUC977.<br>Note 2: PC[14:0] used as the NAND functionality in NUC972 while PI[15:1] used as the NAND functionality in NUC977.<br>Note 3: NUC976 doesn't support NAND interface. | PWRON[1:0] |
| PA.2 | **System Clock Source Selection**<br>0 = System clock is from 12 MHz crystal.<br>1 = System clock is from UPLL output. | PWRON[2] |
| PA.3 | **Watchdog Timer (WDT) Enabled/Disabled Selection**<br>0 = WDT Disabled after power-on.<br>1 = WDT Enabled after power-on. | PWRON[3] |
| PA.4 | **JTAG Interface ON/OFF Selection**<br>0 = Pin PJ[4:0] used as GPIO pin.<br>1 = Pin PJ[4:0] used as JTAG interface. | PWRON[4] |
| PA.5 | **UART 0 Debug Message Output ON/OFF Selection**<br>0 = UART 0 debug message output ON and pin PE[1:0] used as the UART0 functionality.<br>1 = UART 0 debug message output OFF and pin PE[1:0] used as the GPIO functionality. | PWRON[5] |
| PA[7:6] | **NAND Flash Page Size selection**<br>00 = NAND Flash page size is 2KB.<br>01 = NAND Flash page size is 4KB.<br>10 = NAND Flash page size is 8KB.<br>11 = Ignore Power-On Setting. | PWRON[7:6] |

| | | |
|---|---|---|
| PA[9:8] | **NAND Flash ECC Type Selection**<br>00 = NAND Flash ECC type is BCH T12.<br>01 = NAND Flash ECC type is BCH T15.<br>10 = NAND Flash ECC type is BCH T24.<br>11 = Ignore Power-On Setting. | PWRON[9:8] |

### 5.2.6 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS Base Address:**<br>**SYS_BA = 0xB000_0000** | | | | |
| **SYS_PDID** | SYS_BA+0x000 | R | Product Identifier Register | 0x0X30_D008[1] |
| **SYS_PWRON** | SYS_BA+0x004 | R/W | Power-On Setting Register | 0xXXXX_XXXX[2] |
| **SYS_LVRDCR** | SYS_BA+0x020 | R/W | Low Voltage Reset & Detect Control Register | 0x0000_0001 |
| **SYS_MISCFCR** | SYS_BA+0x030 | R/W | Miscellaneous Function Control Register | 0x0000_0200 |
| **SYS_MISCIER** | SYS_BA+0x040 | R/W | Miscellaneous Interrupt Enable Register | 0x0000_0000 |
| **SYS_MISCISR** | SYS_BA+0x044 | R/W | Miscellaneous Interrupt Status Register | 0x0001_0000 |
| **SYS_WKUPSER** | SYS_BA+0x058 | R/W | System Wakeup Source Enable Register | 0x0000_0000 |
| **SYS_WKUPSSR** | SYS_BA+0x05C | R/W | System Wakeup Source Status Register | 0x0000_0000 |
| **SYS_AHBIPRST** | SYS_BA+0x060 | R/W | AHB IP Reset Control Register | 0x0000_0000 |
| **SYS_APBIPRST0** | SYS_BA+0x064 | R/W | APB IP Reset Control Register 0 | 0x0000_0000 |
| **SYS_APBIPRST1** | SYS_BA+0x068 | R/W | APB IP Reset Control Register 1 | 0x0000_0000 |
| **SYS_RSTSTS** | SYS_BA+0x06C | R/W | Reset Source Active Status Register | 0x0000_00XX |
| **SYS_GPA_MFPL** | SYS_BA+0x070 | R/W | GPIOA Low Byte Multiple Function Control Register | 0x0000_0000 |
| **SYS_GPA_MFPH** | SYS_BA+0x074 | R/W | GPIOA High Byte Multiple Function Control Register | 0x0000_0000 |
| **SYS_GPB_MFPL** | SYS_BA+0x078 | R/W | GPIOB Low Byte Multiple Function Control Register | 0x0000_0000 |
| **SYS_GPB_MFPH** | SYS_BA+0x07C | R/W | GPIOB High Byte Multiple Function Control Register | 0x0000_0000 |
| **SYS_GPC_MFPL** | SYS_BA+0x080 | R/W | GPIOC Low Byte Multiple Function Control Register | 0x0000_0000 |
| **SYS_GPC_MFPH** | SYS_BA+0x084 | R/W | GPIOC High Byte Multiple Function Control Register | 0x0000_0000 |
| **SYS_GPD_MFPL** | SYS_BA+0x088 | R/W | GPIOD Low Byte Multiple Function Control Register | 0x0000_0000 |
| **SYS_GPD_MFPH** | SYS_BA+0x08C | R/W | GPIOD High Byte Multiple Function Control Register | 0x0000_0000 |
| **SYS_GPE_MFPL** | SYS_BA+0x090 | R/W | GPIOE Low Byte Multiple Function Control Register | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **SYS_GPE_MFPH** | SYS_BA+0x094 | R/W | GPIOE High Byte Multiple Function Control Register | 0x0000_0000 |
| **SYS_GPF_MFPL** | SYS_BA+0x098 | R/W | GPIOF Low Byte Multiple Function Control Register | 0x0000_0000 |
| **SYS_GPF_MFPH** | SYS_BA+0x09C | R/W | GPIOF High Byte Multiple Function Control Register | 0x0000_0000 |
| **SYS_GPG_MFPL** | SYS_BA+0x0A0 | R/W | GPIOG Low Byte Multiple Function Control Register | 0x0000_0000 |
| **SYS_GPG_MFPH** | SYS_BA+0x0A4 | R/W | GPIOG High Byte Multiple Function Control Register | 0x0000_0000 |
| **SYS_GPH_MFPL** | SYS_BA+0x0A8 | R/W | GPIOH Low Byte Multiple Function Control Register | 0x0000_0000 |
| **SYS_GPH_MFPH** | SYS_BA+0x0AC | R/W | GPIOH High Byte Multiple Function Control Register | 0x0000_0000 |
| **SYS_GPI_MFPL** | SYS_BA+0x0B0 | R/W | GPIOI Low Byte Multiple Function Control Register | 0x0000_0000 |
| **SYS_GPI_MFPH** | SYS_BA+0x0B4 | R/W | GPIOI High Byte Multiple Function Control Register | 0x0000_0000 |
| **SYS_GPJ_MFPL** | SYS_BA+0x0B8 | R/W | GPIOJ Low Byte Multiple Function Control Register | 0x000X_XXXX[2] |
| **SYS_DDR_DSCTL** | SYS_BA+0x0F0 | R/W | DDR I/O Driving Strength Control Register | 0x0000_0000 |
| **SYS_PORDISCR** | SYS_BA+0x100 | R/W | Power-On-Reset Disable Control Register | 0x0000_00XX |
| **SYS_REGWPCTL** | SYS_BA+0x1FC | R/W | Register Write-Protection Control Register | 0x0000_0000 |

**Note:** [1] Dependents on part number.

**Note:** [2] Dependents on power-on setting.

### 5.2.7 Register Description

### Product Identifier Register (SYS_PDID)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_PDID** | SYS_BA+0x000 | R | Product Identifier Register | 0xXX30_D008 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **PRDNUML6** | | | | **PRDNUML5** | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **PRDNUML4** | | | | **PRDNUML3** | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **PRDNUML2** | | | | **PRDNUML1** | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:20] | **PRDNUML6** | **Product Number Letter 6**<br>0 = D.<br>1 = F.<br>2 = G.<br>3 = H. |
| [19:16] | **PRDNUML5** | **Product Number Letter 5**<br>0 = A.<br>1 = B. |
| [15:12] | **PRDNUML4** | **Product Number Letter 4**<br>0xD |
| [11:8] | **PRDNUML3** | **Product Number Letter 3**<br>0x0 |
| [7:4] | **PRDNUML2** | **Product Number Letter 2**<br>0x0 |
| [3:0] | **PRDNUML1** | **Product Number Letter 1**<br>0x8 |

## Power-On Setting Register (SYS_PWRON)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_PWRON | SYS_BA+0x004 | R/W | Power-On Setting Register | 0xXXXX_XXXX |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | DID | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | TICMOD | USBID |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | NECCSEL | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| NPAGESEL | | URDBGON | JTAGON | WDTON | SYSCKSEL | BTSSEL | |

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27:24] | **DID** | **Device ID (Read Only)**<br>0000 = NUC970.<br>0010 = NUC972.<br>1110 = NUC976.<br>1111 = NUC977.<br><br>DID [3] is also used to indicate NAND interface pin out location.<br>1'b0 = Pin PC[14:0] is used as NAND interface for boot from NAND.<br>1'b1 = Pin PI[15:1] is used as NAND interface for boot from NAND. |
| [17] | **TICMOD** | **TIC Mode Enable**<br>0= TIC disabled.<br>1= TIC enabled. |
| [16] | **USBID** | **USB ID Pin Status**<br>0= USB port 0 used as a USB device.<br>1= USB port 0 used as a USB host. |
| [9:8] | **NECCSEL** | **NAND Flash ECC Type Selection**<br>When pin nRESET transited from low to high, the value of pin PA[9:8] latched to NECCSEL.<br>00= NAND Flash ECC type is BCH T12.<br>01= NAND Flash ECC type is BCH T15.<br>10= NAND Flash ECC type is BCH T24.<br>11= Ignore power-on setting. |

| | | |
|---|---|---|
| [7:6] | **NPAGESEL** | **NAND Flash Page Size Selection**<br>When pin nRESET transited from low to high, the value of pin PA[7:6] latched to NPAGESEL.<br>00= NAND Flash page size is 2KB.<br>01= NAND Flash page size is 4KB.<br>10= NAND Flash page size is 8KB.<br>11= Ignore power-on setting. |
| [5] | **URDBGON** | **UART 0 Debug Message Output ON/OFF Selection**<br>When pin nRESET transited from low to high, the value of pin PA[5] latched to URDBGON.<br>0= UART 0 debug message output ON.<br>1= UART 0 debug message output OFF. |
| [4] | **JTAGON** | **JTAG Interface ON/OFF Selection**<br>When pin nRESET transited from low to high, the value of pin PA[4] latched to JTAGON.<br>0= Pin PJ[4:0] used as GPIO pin.<br>1= Pin PJ[4:0] used as JTAG interface. |
| [3] | **WDTON** | **Watchdog Timer (WDT) ON/OFF Selection**<br>When pin nRESET transited from low to high, the value of pin PA[3] latched to WDTON.<br>0= WDT is OFF after power-on.<br>1= WDT is ON after power-on. |
| [2] | **SYSCKSEL** | **System Clock Source Selection**<br>When pin nRESET transited from low to high, the value of pin PA[2] latched to BTCKSEL.<br>0 = System clock is from 12 MHz crystal.<br>1 = System clock is from UPLL output. |
| [1:0] | **BTSSEL** | **Boot Source Selection**<br>When pin nRESET transited from low to high, the value of pin PA[1:0] latched to BTSSEL.<br>00= Boot from USB.<br>01= Boot from eMMC.<br>10= Boot from NAND Flash.<br>11= Boot from SPI Flash. |

## Low Voltage Reset & Detect Control Register (SYS_LVRDCR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_LVRDCR** | SYS_BA+0x020 | R/W | Low Voltage Reset & Detect 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** | | | | | | **LVD_SEL** | **LVD_EN** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | | | | | **LVR_EN** |

| Bits | Description | |
|---|---|---|
| [31:10] | **Reserved** | Reserved. |
| [9] | **LVD_SEL** | **Low Voltage Detect Threshold Selection**<br>0: Low voltage detection level is 2.6V<br>1: Low voltage detection level is 2.8V |
| [8] | **LVD_EN** | **Low Voltage Detect Enable**<br>0: Disable low voltage detect function<br>1: Enable low voltage detect function |
| [7:1] | **Reserved** | Reserved. |
| [0] | **LVR_EN** | **Low Voltage Reset Enable**<br>0: Disable low voltage reset function<br>1: Enable low voltage reset function |

## Miscellaneous Function Control Register (SYS_MISCFCR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_MISCFCR | SYS_BA+0x030 | R/W | Miscellaneous Function 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 |
| UPHY1MEN | UPHY0MEN | Reserved | GPIOLBEN | USRHDSEN | CAPEMAC1SWAP | HDSPUEN | WDTRSTEN |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | TDESKYS | AESKYS |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15] | **UPHY1MEN** | **USB PHY 1 Monitor Enable**<br>0 = USB 2.0 port 1 PHY monitor mode Disabled.<br>1 = USB 2.0 port 1 PHY monitor mode Enabled. |
| [14] | **UPHY0MEN** | **USB PHY 0 Monitor Enable**<br>0 = USB 2.0 port 0 PHY monitor mode Disabled.<br>1 = USB 2.0 port 0 PHY monitor mode Enabled. |
| [13] | **Reserved** | Reserved. |
| [12] | **GPIOLBEN** | **GPIO Pin Loop-back Enable**<br>0 = GPIO input status didn't reflect pin status if the GPIO configured as functional pin.<br>1 = GPIO input status did reflect pin status even if the GPIO configured as functional pin. |
| [11] | **USRHDSEN** | **User Configurable USB Host Device Role Selection Enable**<br>0 = USB host/device role selection decided by HDS pin.<br>1 = USB host/device role selection decided by register PWRON[16]. |
| [10] | **CAPEMAC1 SWAP** | **CAP and EMAC1 Bus Location Swap**<br>0 = CAP is in AHB4 and EMAC1 is in AHB3.<br>1 = CAP is in AHB3 and EMAC1 is in AHB4. |
| [9] | **HDSPUEN** | **HDS Pin Internal Pull-up Enable**<br>0 = Disable HDS pin internal pull-up resister.<br>1 = Enable HDS pin internal pull-up resister. |

| | | |
|---|---|---|
| [8] | **WDTRSTEN** | **WatchDog Timer Reset Connection Enable**<br>This bit is use to enable the function that connect watch-dog timer reset to nRESET pin. If this bit is enabled, the watch-dog timer reset is connected to nRESET pin internally<br>0 = Watch-dog timer reset not connected to nRESET pin internally.<br>1 = Watch-dog timer reset connected to nRESET pin internally. |
| [7:2] | **Reserved** | Reserved. |
| [1] | **TDESKYS** | **DES/3DES Engine Key Selection**<br>This bit is only valid when key is 256-bit.<br>0 = The key is from crypto engine control register.<br>1 = The key is from OTP. |
| [0] | **AESKYS** | **AES Engine Key Selection**<br>This bit is only valid when key is 256-bit.<br>0 = The key is from crypto engine control register.<br>1 = The key is from OTP. |

### Miscellaneous Interrupt Enable Register (SYS_MISCIER)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_MISCIER** | SYS_BA+0x040 | R/W | Miscellaneous 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** | | | | | | **USBIDC_IEN** | **LVD_IEN** |

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [1] | **USBIDC_IEN** | **USB0_ID Pin Status Change Interrupt Enable**<br>0 = Disable HDS status change interrupt.<br>1 = Enable HDS status change interrupt. |
| [0] | **LVD_IEN** | **Low Voltage Detect Interrupt Enable**<br>0 = Disable low voltage detect interrupt.<br>1 = Enable low voltage detect interrupt. |

## Miscellaneous Interrupt Status Register (SYS_MISCISR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_MISCISR | SYS_BA+0x044 | R/W | Miscellaneous Interrupt Status Register | 0x0001_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | USB0_IDS | IBR_RUN_F |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | USBIDC_IS | LVD_IS |

| Bits | Description | |
|---|---|---|
| [31:18] | **Reserved** | Reserved. |
| [17] | **USB0_IDS** | **USB0_ID Status**<br>0 = USB port 0 used as a USB device port.<br>1 = USB port 0 used as a USB host port. |
| [16] | **IBR_RUN_F** | **IBR Run Flag**<br>0 = CPU didn't execute instruction in 0xFFFF_0000 yet.<br>1 = CPU executed instruction in 0xFFFF_0000. |
| [15:2] | **Reserved** | Reserved. |
| [1] | **USBIDC_IS** | **USB0_ID Pin State Change Interrupt Status**<br>0 = USB0_ID state didn't change.<br>1 = USB0_ID state changed from low to high or from high to low. |
| [0] | **LVD_IS** | **Low Voltage Detect Interrupt Status**<br>0 = No low voltage event.<br>1 = Low voltage event detected. |

### System Wakeup Source Enable Register (SYS_WKUPSER)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_WKUPSER | SYS_BA+0x058 | R/W | System Wakeup Source Enable Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| USBH | USBD | Reserved | WDT | KPI | ADC | GPIO | RTC |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ETIMER3 | ETIMER2 | ETIMER1 | ETIMER0 | CAN1 | CAN0 | EMAC1 | EMAC0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LVD | Reserved | UART10 | UART8 | UART6 | UART4 | UART2 | UART1 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EINT7 | EINT6 | EINT5 | EINT4 | EITN3 | EINT2 | EINT1 | EINT0 |

| Bits | Description | |
|---|---|---|
| [31] | **USBH** | **USB Host Wake System Up Enable**<br>0 = USB host wake system up function disabled.<br>1 = USB host wake system up function enabled. |
| [30] | **USBD** | **USB Device Wake System Up Enable**<br>0 = USB device wake system up function disabled.<br>1 = USB device wake system up function enabled. |
| [29] | **Reserved** | Reserved. |
| [28] | **WDT** | **Watch-dog Timer Wake System Up Enable**<br>0 = Watch-dog Timer wake system up function disabled.<br>1 = Watch-dog Timer wake system up function enabled. |
| [27] | **KPI** | **KPI Wake System Up Enable**<br>0 = KPI wake system up function disabled.<br>1 = KPI wake system up function enabled. |
| [26] | **ADC** | **ADC Wake System Up Enable**<br>0 = ADC wake system up function disabled.<br>1 = ADC wake system up function enabled. |
| [25] | **GPIO** | **GPIO Wake System Up Enable**<br>0 = GPIO wake system up function disabled.<br>1 = GPIO wake system up function enabled. |
| [24] | **RTC** | **RTC Wake System Up Enable**<br>0 = RTC wake system up function disabled.<br>1 = RTC wake system up function enabled. |

| | | |
|---|---|---|
| [23] | **ETIMER3** | **Enhanced Timer 3 Wake System Up Enable**<br>0 = Enhanced Timer 3 wake system up function disabled.<br>1 = Enhanced Timer 3 wake system up function enabled. |
| [22] | **ETIMER2** | **Enhanced Timer 2 Wake System Up Enable**<br>0 = Enhanced Timer 2 wake system up function disabled.<br>1 = Enhanced Timer 2 wake system up function enabled. |
| [21] | **ETIMER1** | **Enhanced Timer 1 Wake System Up Enable**<br>0 = Enhanced Timer 1 wake system up function disabled.<br>1 = Enhanced Timer 1 wake system up function enabled. |
| [20] | **ETIMER0** | **Enhanced Timer 0 System Up Enable**<br>0 = Enhanced Timer 0 wake system up function disabled.<br>1 = Enhanced Timer 0 wake system up function enabled. |
| [19] | **CAN1** | **CAN 1 Wake System Up Enable**<br>0 = CAN 1 wake system up function disabled.<br>1 = CAN 1 wake system up function enabled. |
| [18] | **CAN0** | **CAN 0 Wake System Up Enable**<br>0 = CAN 0 wake system up function disabled.<br>1 = CAN 0 wake system up function enabled. |
| [17] | **EMAC1** | **Ethernet MAC 1 Wake System Up Enable**<br>0 = Ethernet MAC 1 wake system up function disabled.<br>1 = Ethernet MAC 1 wake system up function enabled. |
| [16] | **EMAC0** | **Ethernet MAC 0 Wake System Up Enable**<br>0 = Ethernet MAC 0 wake system up function disabled.<br>1 = Ethernet MAC 0 wake system up function enabled. |
| [15] | **LVD** | **Low Voltage Detect Wake System Up Enable**<br>0 = Low Voltage Detect wake system up function disabled.<br>1 = Low Voltage Detect wake system up function enabled. |
| [14] | **Reserved** | Reserved. |
| [13] | **UART10** | **UART 10 Wake System Up Enable**<br>0 = UART 10 wake system up function disabled.<br>1 = UART 10 wake system up function enabled. |
| [12] | **UART8** | **UART 8 Wake System Up Enable**<br>0 = UART 8 wake system up function disabled.<br>1 = UART 8 wake system up function enabled. |
| [11] | **UART6** | **UART 6 Wake System Up Enable**<br>0 = UART 6 wake system up function disabled.<br>1 = UART 6 wake system up function enabled. |
| [10] | **UART4** | **UART 4 Wake System Up Enable**<br>0 = UART 4 wake system up function disabled.<br>1 = UART 4 wake system up function enabled. |

| | | |
|---|---|---|
| [9] | **UART2** | **UART 2 Wake System Up Enable**<br>0 = UART 2 wake system up function disabled.<br>1 = UART 2 wake system up function enabled. |
| [8] | **UART1** | **UART 1 Wake System Up Enable**<br>0 = UART 1 wake system up function disabled.<br>1 = UART 1 wake system up function enabled. |
| [7] | **EINT7** | **External Interrupt 7 Wake System Up Enable**<br>0 = External Interrupt 7 wake system up function disabled.<br>1 = External Interrupt 7 wake system up function enabled. |
| [6] | **EINT6** | **External Interrupt 6 Wake System Up Enable**<br>0 = External Interrupt 6 wake system up function disabled.<br>1 = External Interrupt 6 wake system up function enabled. |
| [5] | **EINT5** | **External Interrupt 5 Wake System Up Enable**<br>0 = External Interrupt 5 wake system up function disabled.<br>1 = External Interrupt 5 wake system up function enabled. |
| [4] | **EINT4** | **External Interrupt 4 Wake System Up Enable**<br>0 = External Interrupt 4 wake system up function disabled.<br>1 = External Interrupt 4 wake system up function enabled. |
| [3] | **EINT3** | **External Interrupt 3 Wake System Up Enable**<br>0 = External Interrupt 3 wake system up function disabled.<br>1 = External Interrupt 3 wake system up function enabled. |
| [2] | **EITN2** | **External Interrupt 2 Wake System Up Enable**<br>0 = External Interrupt 2 wake system up function disabled.<br>1 = External Interrupt 2 wake system up function enabled. |
| [1] | **EINT1** | **External Interrupt 1 Wake System Up Enable**<br>0 = External Interrupt 1 wake system up function disabled.<br>1 = External Interrupt 1 wake system up function enabled. |
| [0] | **EINT0** | **External Interrupt 0 Wake System Up Enable**<br>0 = External Interrupt 0 wake system up function disabled.<br>1 = External Interrupt 0 wake system up function enabled. |

### System Wakeup Source Status Register (SYS_WKUPSSR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_WKUPSSR | SYS_BA+0x05C | R/W | System Wakeup Source Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| USBH | USBD | Reserved | WDT | KPI | ADC | GPIO | RTC |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ETIMER3 | ETIMER2 | ETIMER1 | ETIMER0 | CAN1 | CAN0 | EMAC1 | EMAC0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LVD | Reserved | UART10 | UART8 | UART6 | UART4 | UART2 | UART1 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EINT7 | EINT6 | EINT5 | EINT4 | EITN3 | EINT2 | EINT1 | EINT0 |

| Bits | Description | |
|---|---|---|
| [31] | **USBH** | **USB Host Wake System Up Status**<br>0 = USB host didn't wake system up.<br>1 = USB host waked system up. |
| [30] | **USBD** | **USB Device Wake System Up Status**<br>0 = USB device didn't wake system up.<br>1 = USB device waked system up. |
| [29] | **Reserved** | Reserved. |
| [28] | **WDT** | **Watch-dog Timer Wake System Up Status**<br>0 = Watch-dog Timer didn't wake system up.<br>1 = Watch-dog Timer waked system up. |
| [27] | **KPI** | **KPI Wake System Up Status**<br>0 = KPI didn't wake system up.<br>1 = KPI waked system up. |
| [26] | **ADC** | **ADC Wake System Up Status**<br>0 = ADC didn't wake system up.<br>1 = ADC waked system up. |
| [25] | **GPIO** | **GPIO Wake System Up Status**<br>0 = GPIO didn't wake system up.<br>1 = GPIO waked system up. |
| [24] | **RTC** | **RTC Wake System Up Status**<br>0 = RTC didn't wake system up.<br>1 = RTC waked system up. |

| | | |
|---|---|---|
| [23] | **ETIMER3** | **Enhanced Timer 3 Wake System Up Status**<br>0 = Enhanced Timer 3 didn't wake system up.<br>1 = Enhanced Timer 3 waked system up. |
| [22] | **ETIMER2** | **Enhanced Timer 2 Wake System Up Status**<br>0 = Enhanced Timer 2 didn't wake system up.<br>1 = Enhanced Timer 2 waked system up. |
| [21] | **ETIMER1** | **Enhanced Timer 1 Wake System Up Status**<br>0 = Enhanced Timer 1 didn't wake system up.<br>1 = Enhanced Timer 1 waked system up. |
| [20] | **ETIMER0** | **Enhanced Timer 0 System Up Status**<br>0 = Enhanced Timer 0 didn't wake system up.<br>1 = Enhanced Timer 0 waked system up. |
| [19] | **CAN1** | **CAN 1 Wake System Up Status**<br>0 = CAN 1 didn't wake system up.<br>1 = CAN 1 waked system up. |
| [18] | **CAN0** | **CAN 0 Wake System Up Status**<br>0 = CAN 0 didn't wake system up.<br>1 = CAN 0 waked system up. |
| [17] | **EMAC1** | **Ethernet MAC 1 Wake System Up Status**<br>0 = Ethernet MAC 1 didn't wake system up.<br>1 = Ethernet MAC 1 waked system up. |
| [16] | **EMAC0** | **Ethernet MAC 0 Wake System Up Status**<br>0 = Ethernet MAC 0 didn't wake system up.<br>1 = Ethernet MAC 0 waked system up. |
| [15] | **LVD** | **Low Voltage Detect Wake System Up Status**<br>0 = Low Voltage Detect didn't wake system up.<br>1 = Low Voltage Detect waked system up. |
| [14] | **Reserved** | Reserved. |
| [13] | **UART10** | **UART 10 Wake System Up Status**<br>0 = UART 10 didn't wake system up.<br>1 = UART 10 waked system up. |
| [12] | **UART8** | **UART 8 Wake System Up Status**<br>0 = UART 8 didn't wake system up.<br>1 = UART 8 waked system up. |
| [11] | **UART6** | **UART 6 Wake System Up Status**<br>0 = UART 6 didn't wake system up.<br>1 = UART 6 waked system up. |
| [10] | **UART4** | **UART 4 Wake System Up Status**<br>0 = UART 4 didn't wake system up.<br>1 = UART 4 waked system up. |

| | | |
|---|---|---|
| [9] | **UART2** | **UART 2 Wake System Up Status**<br>0 = UART 2 didn't wake system up.<br>1 = UART 2 waked system up. |
| [8] | **UART1** | **UART 1 Wake System Up Status**<br>0 = UART 1 didn't wake system up.<br>1 = UART 1 waked system up. |
| [7] | **EINT7** | **External Interrupt 7 Wake System Up Status**<br>0 = External Interrupt 7 didn't wake system up.<br>1 = External Interrupt 7 waked system up. |
| [6] | **EINT6** | **External Interrupt 6 Wake System Up Status**<br>0 = External Interrupt 6 didn't wake system up.<br>1 = External Interrupt 6 waked system up. |
| [5] | **EINT5** | **External Interrupt 5 Wake System Up Status**<br>0 = External Interrupt 5 didn't wake system up.<br>1 = External Interrupt 5 waked system up. |
| [4] | **EINT4** | **External Interrupt 4 Wake System Up Status**<br>0 = External Interrupt 4 didn't wake system up.<br>1 = External Interrupt 4 waked system up. |
| [3] | **EINT3** | **External Interrupt 3 Wake System Up Status**<br>0 = External Interrupt 3 didn't wake system up.<br>1 = External Interrupt 3 waked system up. |
| [2] | **EITN2** | **External Interrupt 2 Wake System Up Status**<br>0 = External Interrupt 2 didn't wake system up.<br>1 = External Interrupt 2 waked system up. |
| [1] | **EINT1** | **External Interrupt 1 Wake System Up Status**<br>0 = External Interrupt 1 didn't wake system up.<br>1 = External Interrupt 1 waked system up. |
| [0] | **EINT0** | **External Interrupt 0 Wake System Up Status**<br>0 = External Interrupt 0 didn't wake system up.<br>1 = External Interrupt 0 waked system up. |

## AHB IP Reset Control Register (SYS_AHBIPRST)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_AHBIPRST | SYS_BA+0x060 | R/W | AHB IP Reset Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | SDIO |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CRYPTO | JPEG | GE2D | FMI | USBD | USBH | EMAC1 | EMAC0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | CAP | LCD | I2S |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | GDMA | CPU_PLS | Reserved | CHIP |

| Bits | Description | |
|---|---|---|
| [31:25] | **Reserved** | Reserved. |
| [24] | **SDIO** | **SDIO Controller Reset Enable**<br>0 = SDIO controller reset disabled.<br>1 = SDIO controller reset enabled. |
| [23] | **CRYPTO** | **Cryptographic Accelerator Reset Enable**<br>0 = Cryptographic Accelerator reset disabled.<br>1 = Cryptographic Accelerator reset enabled. |
| [22] | **JPEG** | **JPEG Codec Reset Enable**<br>0 = JPEG codec reset disabled.<br>1 = JPEG codec reset enabled. |
| [21] | **GE2D** | **2D Graphic Engine Reset Enable**<br>0 = 2D graphic engine reset disabled.<br>1 = 2D graphic engine reset enabled. |
| [20] | **FMI** | **FMI Controller Reset Enable**<br>0 = FMI controller reset disabled.<br>1 = FMI controller reset enabled. |
| [19] | **USBD** | **USB Device Controller Reset Enable**<br>0 = USB device controller reset disabled.<br>1 = USB device controller reset enabled. |
| [18] | **USBH** | **USB Host Controller (EHCI/OHCI) Reset Enable**<br>0 = USB host controller (EHCI/OHCI) reset disabled.<br>1 = USB host controller (EHCI/OHCI) reset enabled. |

| | | |
|---|---|---|
| [17] | **EMAC1** | **Ethernet MAC 1 Reset Enable**<br>0 = Ethernet MAC 1 reset disabled.<br>1 = Ethernet MAC 1 reset enabled. |
| [16] | **EMAC0** | **Ethernet MAC 0 Reset Enable**<br>0 = Ethernet MAC 0 reset disabled.<br>1 = Ethernet MAC 0 reset enabled. |
| [15:11] | **Reserved** | Reserved. |
| [10] | **CAP** | **Capture Sensor Interface Reset Enable**<br>0 = Capture sensor interface reset disabled.<br>1 = Capture sensor interface reset enabled. |
| [9] | **LCD** | **LCD Controller Reset Enable**<br>0 = LCD controller reset disabled.<br>1 = LCD controller reset enabled. |
| [8] | **I2S** | **$I^2S$ Controller Reset Enable**<br>0 = $I^2S$ controller reset disabled.<br>1 = $I^2S$ controller reset enabled. |
| [7:4] | **Reserved** | Reserved. |
| [3] | **GDMA** | **GDMA Reset Enable**<br>0 = GDMA reset disabled.<br>1 = GDMA reset enabled. |
| [2] | **CPU_PLS** | **CPU Pulse Reset Enable**<br>This bit is used to generate a reset pulse to ARM926EJ-S CPU.<br>When set this bit high, reset controller generates a 6 system clock long reset pulse to ARM926EJ-S CPU. After the reset completed, this bit will be clear to low automatically.<br>0 = CPU pulse reset disabled.<br>1 = CPU pulse reset enabled. |
| [1] | **Reserved** | Reserved. |
| [0] | **CHIP** | **Chip Reset Enable**<br>0 = Chip reset disabled.<br>1 = Chip reset enabled. |

### APB IP Reset Control Register 0 (SYS_APBIPRST0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_APBIPRST0 | SYS_BA+0x064 | R/W | APB IP Reset Control Register 0 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | UART10 | UART9 | UART8 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| UART7 | UART6 | UART5 | UART4 | UART3 | UART2 | UART1 | UART0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | TIMER4 | TIMER3 | TIMER2 | TIMER1 | TIMER0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ETIMER3 | ETIMER2 | ETIMER1 | ETIMER0 | GPIO | Reserved | | |

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26] | **UART10** | **UART 10 Reset Enable**<br>0 = UART 10 reset disabled.<br>1 = UART 10 reset enabled. |
| [25] | **UART9** | **UART 9 Reset Enable**<br>0 = UART 9 reset disabled.<br>1 = UART 9 reset enabled. |
| [24] | **UART8** | **UART 8 Reset Enable**<br>0 = UART 8 reset disabled.<br>1 = UART 8 reset enabled. |
| [23] | **UART7** | **UART 7 Reset Enable**<br>0 = UART 7 reset disabled.<br>1 = UART 7 reset enabled. |
| [22] | **UART6** | **UART 6 Reset Enable**<br>0 = UART 6 reset disabled.<br>1 = UART 6 reset enabled. |
| [21] | **UART5** | **UART 5 Reset Enable**<br>0 = UART 5 reset disabled.<br>1 = UART 5 reset enabled. |
| [20] | **UART4** | **UART 4 Reset Enable**<br>0 = UART 4 reset disabled.<br>1 = UART 4 reset enabled. |

| | | |
|---|---|---|
| [19] | **UART3** | **UART 3 Reset Enable**<br>0 = UART 3 reset disabled.<br>1 = UART 3 reset enabled. |
| [18] | **UART2** | **UART 2 Reset Enable**<br>0 = UART 2 reset disabled.<br>1 = UART 2 reset enabled. |
| [17] | **UART1** | **UART 1 Reset Enable**<br>0 = UART 1 reset disabled.<br>1 = UART 1 reset enabled. |
| [16] | **UART0** | **UART 0 Reset Enable**<br>0 = UART 0 reset disabled.<br>1 = UART 0 reset enabled. |
| [15:13] | **Reserved** | Reserved. |
| [12] | **TIMER4** | **TIMER 4 Reset Enable**<br>0 = TIMER 4 reset disabled.<br>1 = TIMER 4 reset enabled. |
| [11] | **TIMER3** | **TIMER 3 Reset Enable**<br>0 = TIMER 3 reset disabled.<br>1 = TIMER 3 reset enabled. |
| [10] | **TIMER2** | **TIMER 2 Reset Enable**<br>0 = TIMER 2 reset disabled.<br>1 = TIMER 2 reset enabled. |
| [9] | **TIMER1** | **TIMER 1 Reset Enable**<br>0 = TIMER 1 reset disabled.<br>1 = TIMER 1 reset enabled. |
| [8] | **TIMER0** | **TIMER 0 Reset Enable**<br>0 = TIMER 0 reset disabled.<br>1 = TIMER 0 reset enabled. |
| [7] | **ETIMER3** | **ETIMER 3 Reset Enable**<br>0 = ETIMER 3 reset disabled.<br>1 = ETIMER 3 reset enabled. |
| [6] | **ETIMER2** | **ETIMER 2 Reset Enable**<br>0 = ETIMER 2 reset disabled.<br>1 = ETIMER 2 reset enabled. |
| [5] | **ETIMER1** | **ETIMER 1 Reset Enable**<br>0 = ETIMER 1 reset disabled.<br>1 = ETIMER 1 reset enabled. |
| [4] | **ETIMER0** | **ETIMER 0 Reset Enable**<br>0 = ETIMER 0 reset disabled.<br>1 = ETIMER 0 reset enabled. |

| | | |
|---|---|---|
| [3] | **GPIO** | **GPIO Reset Enable**<br>0 = GPIO reset disabled.<br>1 = GPIO reset enabled. |
| [2:0] | **Reserved** | Reserved. |

## APB IP Reset Control Register 1 (SYS_APBIPRST1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_APBIPRST1 | SYS_BA+0x068 | R/W | APB IP Reset Control Register 1 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | PWM | MTPC | KPI | ADC |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | SMC1 | SMC0 | Reserved | | CAN1 | CAN0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | SPI1 | SPI0 | Reserved | | I2C1 | I2C0 |

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27] | **PWM** | **PWM Reset Enable**<br>0 = PWM reset disabled.<br>1 = PWM reset enabled. |
| [26] | **MTPC** | **MTP Controller Reset Enable**<br>0 = MTP Controller reset disabled.<br>1 = MTP Controller reset enabled. |
| [25] | **KPI** | **KPI Reset Enable**<br>0 = KPI reset disabled.<br>1 = KPI reset enabled. |
| [24] | **ADC** | **ADC Reset Enable**<br>0 = ADC reset disabled.<br>1 = ADC reset enabled. |
| [23:14] | **Reserved** | Reserved. |
| [13] | **SMC1** | **SMC 1 Reset Enable**<br>0 = SMC 1 reset disabled.<br>1 = SMC 1 reset enabled. |
| [12] | **SMC0** | **SMC 0 Reset Enable**<br>0 = SMC 0 reset disabled.<br>1 = SMC 0 reset enabled. |
| [11:10] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [9] | **CAN1** | **CAN 1 Reset Enable**<br>0 = CAN 1 reset disabled.<br>1 = CAN 1 reset enabled. |
| [8] | **CNA0** | **CAN 0 Reset Enable**<br>0 = CAN 0 reset disabled.<br>1 = CAN 0 reset enabled. |
| [7:6] | **Reserved** | Reserved. |
| [5] | **SPI1** | **SPI 1 Reset Enable**<br>0 = SPI 1 reset disabled.<br>1 = SPI 1 reset enabled. |
| [4] | **SPI0** | **SPI 0 Reset Enable**<br>0 = SPI 0 reset disabled.<br>1 = SPI 0 reset enabled. |
| [3:2] | **Reserved** | Reserved. |
| [1] | **I2C1** | **$I^2C$ 1 Reset Enable**<br>0 = $I^2C$ 1 reset disabled.<br>1 = $I^2C$ 1 reset enabled. |
| [0] | **I2C0** | **$I^2C$ 0 Reset Enable**<br>0 = $I^2C$ 0 reset disabled.<br>1 = $I^2C$ 1 reset enabled. |

## Reset Source Active Status Register (SYS_RSTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_RSTSTS | SYS_BA+0x06C | R/W | Reset Source Active Status 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 | KPIRSTS | WDTRSTS | CPURSTS | CHIPRSTS | LVRRSTS | PINRSTS | PORRSTS |

| Bits | Description | |
|---|---|---|
| [31:7] | **Reserved** | Reserved. |
| [6] | **KPIRSTS** | **Chip Reset by KPI Status**<br>0 = No reset from KPI.<br>1 = The KPI had issued reset signal to reset the chip. |
| [5] | **WDTRSTS** | **Chip Reset by Watchdog Timer Status**<br>0 = No reset from watchdog timer.<br>1 = Watchdog timer had issued reset signal to reset the chip. |
| [4] | **CPURSTS** | **CPU Reset by CPU_LVL (AHBIPRST[1]) or CPU_PLS (AHBIPRST[2]) Status**<br>0 = No CPU reset from CPU_LVL (AHBIPRST[1]) or CPU_PLS (AHBIPRST[2]).<br>1 = CPU_LVL (AHBIPRST[1]) or CPU_PLS (AHBIPRST[2]) has been high to reset the CPU. |
| [3] | **CHIPRSTS** | **Chip Reset by CHIP (AHBIPRST[0]) Status**<br>0 = No reset from CHIP (AHBIPRST[0]).<br>1 = CHIP (AHBIPRST[0]) has been high to reset CPU. |
| [2] | **LVRRSTS** | **Chip Reset by LVRD Status**<br>0 = No reset from LVRD.<br>1 = LVRD had issued reset signal to reset the chip. |
| [1] | **PINRSTS** | **Chip Reset by NRESET Pin Status**<br>0 = No reset from nRESET pin.<br>1 = nRESET pin had issued reset signal to reset the chip. |
| [0] | **PORRSTS** | **Chip Reset by POR Status**<br>0 = No reset from POR.<br>1 = POR had issued reset signal to reset the chip. |

## GPIOA Low Byte Multiple Function Control Register (SYS_GPA_MFPL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_GPA_MFPL | SYS_BA+0x070 | R/W | GPIOA Low Byte Multiple Function Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MFP_GPA7 | | | | MFP_GPA6 | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MFP_GPA5 | | | | MFP_GPA4 | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MFP_GPA3 | | | | MFP_GPA2 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MFP_GPA1 | | | | MFP_GPA0 | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **MFP_GPA7** | **Pin PA.7 Multi Function Selection**<br>0000 = PA.7.<br>0010 = LCD_DATA7.<br>0100 = KPI_ROW3.<br>Others = PA.7. |
| [27:24] | **MFP_GPA6** | **Pin PA.6 Multi Function Selection**<br>0000 = PA.6.<br>0010 = LCD_DATA6.<br>0100 = KPI_ROW2.<br>Others = PA.6. |
| [23:20] | **MFP_GPA5** | **Pin PA.5 Multi Function Selection**<br>0000 = PA.5.<br>0010 = LCD_DATA5.<br>0100 = KPI_ROW1.<br>Others = PA.5. |
| [19:16] | **MFP_GPA4** | **Pin PA.4 Multi Function Selection**<br>0000 = PA.4.<br>0010 = LCD_DATA4.<br>0100 = KPI_ROW0.<br>Others = PA.4. |
| [15:12] | **MFP_GPA3** | **Pin PA.3 Multi Function Selection**<br>0000 = PA.3.<br>0010 = LCD_DATA3.<br>Others = PA.3. |

| | | |
|---|---|---|
| [11:8] | **MFP_GPA2** | **Pin PA.2 Multi Function Selection**<br>0000 = PA.2.<br>0010 = LCD_DATA2.<br>Others = PA.2. |
| [7:4] | **MFP_GPA1** | **Pin PA.1 Multi Function Selection**<br>0000 = PA.1.<br>0010 = LCD_DATA1.<br>Others = PA.1. |
| [3:0] | **MFP_GPA0** | **Pin PA.0 Multi Function Selection**<br>0000 = PA.0.<br>0010 = LCD_DATA0.<br>Others = PA.0. |

## GPIOA High Byte Multiple Function Control Register (SYS_GPA_MFPH)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_GPA_MFPH | SYS_BA+0x074 | R/W | GPIOA High Byte Multiple Function Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MFP_GPA15 | | | | MFP_GPA14 | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MFP_GPA13 | | | | MFP_GPA12 | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MFP_GPA11 | | | | MFP_GPA10 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MFP_GPA9 | | | | MFP_GPA8 | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **MFP_GPA15** | **Pin PA.15 Multi Function Selection**<br>0000 = PA.15.<br>0010 = LCD_DATA15.<br>0100 = KPI_COL7.<br>1101 = PWM3.<br>Others = PA.15. |
| [27:24] | **MFP_GPA14** | **Pin PA.14 Multi Function Selection**<br>0000 = PA.14.<br>0010 = LCD_DATA14.<br>0100 = KPI_COL6.<br>1101 = PWM2.<br>Others = PA.14. |
| [23:20] | **MFP_GPA13** | **Pin PA.13 Multi Function Selection**<br>0000 = PA.13.<br>0010 = LCD_DATA13.<br>0100 = KPI_COL5.<br>1101 = PWM1.<br>Others = PA.13. |
| [19:16] | **MFP_GPA12** | **Pin PA.12 Multi Function Selection**<br>0000 = PA.12.<br>0010 = LCD_DATA12.<br>0100 = KPI_COL4.<br>1101 = PWM0.<br>Others = PA.12. |

| | | |
|---|---|---|
| [15:12] | **MFP_GPA11** | **Pin PA.11 Multi Function Selection**<br>0000 = PA.11.<br>0010 = LCD_DATA11.<br>0100 = KPI_COL3.<br>Others = PA.11. |
| [11:8] | **MFP_GPA10** | **Pin PA.10 Multi Function Selection**<br>0000 = PA.10.<br>0010 = LCD_DATA10.<br>0100 = KPI_COL2.<br>Others = PA.10. |
| [7:4] | **MFP_GPA9** | **Pin PA.9 Multi Function Selection**<br>0000 = PA.9.<br>0010 = LCD_DATA9.<br>0100 = KPI_COL1.<br>Others = PA.9. |
| [3:0] | **MFP_GPA8** | **Pin PA.8 Multi Function Selection**<br>0000 = PA.8.<br>0010 = LCD_DATA8.<br>0100 = KPI_COL0.<br>Others = PA.8. |

### GPIOB Low Byte Multiple Function Control Register (SYS_GPB_MFPL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_GPB_MFPL | SYS_BA+0x078 | R/W | GPIOB Low Byte Multiple Function Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MFP_GPB7 | | | | MFP_GPB6 | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MFP_GPB5 | | | | MFP_GPB4 | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MFP_GPB3 | | | | MFP_GPB2 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MFP_GPB1 | | | | MFP_GPB0 | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **MFP_GPB7** | **Pin PB.7 Multi Function Selection**<br>0000 = PB.7.<br>1011 = SPI0_CLK.<br>Others = PB.7. |
| [27:24] | **MFP_GPB6** | **Pin PB.6 Multi Function Selection**<br>0000 = PB.6.<br>1011 = SPI0_SS0.<br>Others = PB.6. |
| [23:20] | **MFP_GPB5** | **Pin PB.5 Multi Function Selection**<br>0000 = PB.5.<br>1001 = UR6_CTS.<br>1111 = CLK_OUT.<br>Others = PB.5. |
| [19:16] | **MFP_GPB4** | **Pin PB.4 Multi Function Selection**<br>0000 = PB.4.<br>1001 = UR6_RTS.<br>Others = PB.4. |
| [15:12] | **MFP_GPB3** | **Pin PB.3 Multi Function Selection**<br>0000 = PB.3.<br>1001 = UR6_RXD.<br>1101 = PWM1.<br>1111 = ETMR0_CAP.<br>Others = PB.3. |

| | | |
|---|---|---|
| [11:8] | **MFP_GPB2** | **Pin PB.2 Multi Function Selection**<br>0000 = PB.2.<br>1001 = UR6_TXD.<br>1101 = PWM0.<br>1111 = ETMR0_TGL.<br>Others = PB.2. |
| [7:4] | **MFP_GPB1** | **Pin PB.1 Multi Function Selection**<br>0000 = PB.1.<br>0101 = NAND_RDY1.<br>1001 = UR5_RXD.<br>1011 = SPI1_SS1.<br>1101 = ETMR1_CAP.<br>Others = PB.1. |
| [3:0] | **MFP_GPB0** | **Pin PB.0 Multi Function Selection**<br>0000 = PB.0.<br>0101 = NAND_nCS1.<br>1001 = UR5_TXD.<br>1011 = SPI0_SS1.<br>1101 = ETMR1_TGL.<br>Others = PB.0. |

## GPIOB High Byte Multiple Function Control Register (SYS_GPB_MFPH)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_GPB_MFPH | SYS_BA+0x07C | R/W | GPIOB High Byte Multiple Function Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MFP_GPB15 | | | | MFP_GPB14 | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MFP_GPB13 | | | | MFP_GPB12 | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MFP_GPB11 | | | | MFP_GPB10 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MFP_GPB9 | | | | MFP_GPB8 | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **MFP_GPB15** | **Pin PB.15 Multi Function Selection**<br>0000 = PB.15.<br>1001 = UR10_CTS.<br>1011 = SPI1_DATAI (SPI1_DATA1).<br>1111 = CLK_OUT.<br>Others = PB.15. |
| [27:24] | **MFP_GPB14** | **Pin PB.14 Multi Function Selection**<br>0000 = PB.14.<br>1001 = UR10_RTS.<br>1011 = SPI1_DATAO (SPI1_DATA0).<br>Others = PB.14. |
| [23:20] | **MFP_GPB13** | **Pin PB.13 Multi Function Selection**<br>0000 = PB.13.<br>1001 = UR10_RXD.<br>1011 = SPI1_CLK.<br>Others = PB.13. |
| [19:16] | **MFP_GPB12** | **Pin PB.12 Multi Function Selection**<br>0000 = PB.12.<br>1001 = UR10_TXD.<br>1011 = SPI1_SS0.<br>Others = PB.12. |

| | | |
|---|---|---|
| [15:12] | **MFP_GPB11** | **Pin PB.11 Multi Function Selection**<br>0000 = PB.11.<br>1001 = UR10_RXD.<br>1011 = SPI0_DATA3.<br>1100 = CAN0_TX.<br>Others = PB.11. |
| [11:8] | **MFP_GPB10** | **Pin PB.10 Multi Function Selection**<br>0000 = PB.10.<br>1001 = UR10_TXD.<br>1011 = SPI0_DATA2.<br>1100 = CAN0_RX.<br>Others = PB.10. |
| [7:4] | **MFP_GPB9** | **Pin PB.9 Multi Function Selection**<br>0000 = PB.9.<br>1011 = SPI0_DATAI (SPI0_DATA1).<br>Others = PB.9. |
| [3:0] | **MFP_GPB8** | **Pin PB.8 Multi Function Selection**<br>0000 = PB.8.<br>1011 = SPI0_DATAO (SPI0_DATA0).<br>Others = PB.8. |

## GPIOC Low Byte Multiple Function Control Register (SYS_GPC_MFPL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_GPC_MFPL | SYS_BA+0x080 | R/W | GPIOC Low Byte Multiple Function Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MFP_GPC7 | | | | MFP_GPC6 | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MFP_GPC5 | | | | MFP_GPC4 | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MFP_GPC3 | | | | MFP_GPC2 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MFP_GPC1 | | | | MFP_GPC0 | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **MFP_GPC7** | **Pin PC.7 Multi Function Selection**<br>0000 = PC.7.<br>0101 = NAND_DATA7.<br>1001 = UR10_RXD.<br>1101 = ETMR0_CAP.<br>Others = PC.7. |
| [27:24] | **MFP_GPC6** | **Pin PC.6 Multi Function Selection**<br>0000 = PC.6.<br>0101 = NAND_DATA6.<br>1001 = UR10_TXD.<br>1101 = ETMR0_TGL.<br>Others = PC.6. |
| [23:20] | **MFP_GPC5** | **Pin PC.5 Multi Function Selection**<br>0000 = PC.5.<br>0101 = NAND_DATA5.<br>0110 = eMMC_CLK.<br>Others = PC.5. |
| [19:16] | **MFP_GPC4** | **Pin PC.4 Multi Function Selection**<br>0000 = PC.4.<br>0101 = NAND_DATA4.<br>0110 = eMMC_CMD.<br>Others = PC.4. |

| | | |
|---|---|---|
| [15:12] | **MFP_GPC3** | **Pin PC.3 Multi Function Selection**<br>0000 = PC.3.<br>0101 = NAND_DATA3.<br>0110 = eMMC_DATA3.<br>Others = PC.3. |
| [11:8] | **MFP_GPC2** | **Pin PC.2 Multi Function Selection**<br>0000 = PC.2.<br>0101 = NAND_DATA2.<br>0110 = eMMC_DATA2.<br>Others = PC.2. |
| [7:4] | **MFP_GPC1** | **Pin PC.1 Multi Function Selection**<br>0000 = PC.1.<br>0101 = NAND_DATA1.<br>0110 = eMMC_DATA1.<br>Others = PC.1. |
| [3:0] | **MFP_GPC0** | **Pin PC.0 Multi Function Selection**<br>0000 = PC.0.<br>0101 = NAND_DATA0.<br>0110 = eMMC_DATA0.<br>Others = PC.0. |

## GPIOC High Byte Multiple Function Control Register (SYS_GPC_MFPH)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_GPC_MFPH | SYS_BA+0x084 | R/W | GPIOC High Byte Multiple Function Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | MFP_GPC14 | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MFP_GPC13 | | | | MFP_GPC12 | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MFP_GPC11 | | | | MFP_GPC10 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MFP_GPC9 | | | | MFP_GPC8 | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27:24] | **MFP_GPC14** | **Pin PC.14 Multi Function Selection**<br>0000 = PC.14.<br>0101 = NAND_nWP.<br>1101 = PWM0.<br>Others = PC.14. |
| [23:20] | **MFP_GPC13** | **Pin PC.13 Multi Function Selection**<br>0000 = PC.13.<br>0101 = NAND_RDY0.<br>1001 = UR4_CTS.<br>1101 = ETMR3_CAP.<br>1111 = CLK_OUT.<br>Others = PC.13. |
| [19:16] | **MFP_GPC12** | **Pin PC.12 Multi Function Selection**<br>0000 = PC.12.<br>0101 = NAND_nRE.<br>1001 = UR4_RTS.<br>1101 = ETMR3_TGL.<br>Others = PC.12. |
| [15:12] | **MFP_GPC11** | **Pin PC.11 Multi Function Selection**<br>0000 = PC.11.<br>0101 = NAND_nWE.<br>1001 = UR4_RXD.<br>1101 = ETMR2_CAP.<br>Others = PC.11. |

| | | |
|---|---|---|
| [11:8] | **MFP_GPC10** | **Pin PC.10 Multi Function Selection**<br>0000 = PC.10.<br>0101 = NAND_CLE.<br>1001 = UR4_TXD.<br>1101 = ETMR2_TGL.<br>Others = PC.10. |
| [7:4] | **MFP_GPC9** | **Pin PC.9 Multi Function Selection**<br>0000 = PC.9.<br>0101 = NAND_ALE.<br>1001 = UR10_CTS.<br>1101 = ETMR1_CAP.<br>1111 = CLK_OUT.<br>Others = PC.9. |
| [3:0] | **MFP_GPC8** | **Pin PC.8 Multi Function Selection**<br>0000 = PC.8.<br>0101 = NAND_nCS0.<br>1001 = UR10_RTS.<br>1101 = ETMR1_TGL.<br>Others = PC.8. |

## GPIOD Low Byte Multiple Function Control Register (SYS_GPD_MFPL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_GPD_MFPL | SYS_BA+0x088 | R/W | GPIOD Low Byte Multiple Function Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MFP_GPD7 | | | | MFP_GPD6 | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MFP_GPD5 | | | | MFP_GPD4 | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MFP_GPD3 | | | | MFP_GPD2 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MFP_GPD1 | | | | MFP_GPD0 | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **MFP_GPD7** | **Pin PD.7 Multi Function Selection**<br>0000 = PD.7.<br>Others = PD.7. |
| [27:24] | **MFP_GPD6** | **Pin PD.6 Multi Function Selection**<br>0000 = PD.6.<br>0110 = SD0_nCD.<br>Others = PD.6. |
| [23:20] | **MFP_GPD5** | **Pin PD.5 Multi Function Selection**<br>0000 = PD.5.<br>0110 = SD0_DATA3.<br>Others = PD.5. |
| [19:16] | **MFP_GPD4** | **Pin PD.4 Multi Function Selection**<br>0000 = PD.4.<br>0110 = SD0_DATA2.<br>Others = PD.4. |
| [15:12] | **MFP_GPD3** | **Pin PD.3 Multi Function Selection**<br>0000 = PD.3.<br>0110 = SD0_DATA1.<br>Others = PD.3. |
| [11:8] | **MFP_GPD2** | **Pin PD.2 Multi Function Selection**<br>0000 = PD.2.<br>0110 = SD0_DATA0.<br>Others = PD.2. |

| | | |
|---|---|---|
| [7:4] | **MFP_GPD1** | **Pin PD.1 Multi Function Selection**<br>0000 = PD.1.<br>0110 = SD0_CLK.<br>Others = PD.1. |
| [3:0] | **MFP_GPD0** | **Pin PD.0 Multi Function Selection**<br>0000 = PD.0.<br>0110 = SD0_CMD.<br>Others = PD.0. |

## GPIOD High Byte Multiple Function Control Register (SYS_GPD_MFPH)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_GPD_MFPH | SYS_BA+0x08C | R/W | GPIOD High Byte Multiple Function Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MFP_GPD15 | | | | MFP_GPD14 | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MFP_GPD13 | | | | MFP_GPD12 | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MFP_GPD11 | | | | MFP_GPD10 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MFP_GPD9 | | | | MFP_GPD8 | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **MFP_GPD15** | **Pin PD.15 Multi Function Selection**<br>0000 = PD.15.<br>0010 = LCD_DATA23.<br>1001 = UR9_RXD.<br>1101 = PWM3.<br>1110 = EBI_nWAIT.<br>Others = PD.15. |
| [27:24] | **MFP_GPD14** | **Pin PD.14 Multi Function Selection**<br>0000 = PD.14.<br>0010 = LCD_DATA22.<br>1001 = UR9_TXD.<br>1101 = PWM2.<br>1110 = EBI_nOE.<br>Others = PD.14. |
| [23:20] | **MFP_GPD13** | **Pin PD.13 Multi Function Selection**<br>0000 = PD.13.<br>0010 = LCD_DATA21.<br>1101 = PWM1.<br>1110 = EBI_nWE.<br>Others = PD.13. |

| | | |
|---|---|---|
| [19:16] | **MFP_GPD12** | **Pin PD.12 Multi Function Selection**<br>0000 = PD.12.<br>0010 = LCD_DATA20.<br>1001 = UR9_RXD.<br>1101 = PWM0.<br>1110 = EBI_nCS4.<br>Others = PD.12. |
| [15:12] | **MFP_GPD11** | **Pin PD.11 Multi Function Selection**<br>0000 = PD.11.<br>0010 = LCD_DATA19.<br>1001 = UR9_TXD.<br>1110 = EBI_nCS3.<br>Others = PD.11. |
| [11:8] | **MFP_GPD10** | **Pin PD.10 Multi Function Selection**<br>0000 = PD.10.<br>0010 = LCD_DATA18.<br>1110 = EBI_nCS2.<br>Others = PD.10. |
| [7:4] | **MFP_GPD9** | **Pin PD.9 Multi Function Selection**<br>0000 = PD.9.<br>0010 = LCD_DATA17.<br>1110 = EBI_nCS1.<br>Others = PD.9. |
| [3:0] | **MFP_GPD8** | **Pin PD.8 Multi Function Selection**<br>0000 = PD.8.<br>0010 = LCD_DATA16.<br>1110 = EBI_nCS0.<br>Others = PD.8. |

## GPIOE Low Byte Multiple Function Control Register (SYS_GPE_MFPL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_GPE_MFPL | SYS_BA+0x090 | R/W | GPIOE Low Byte Multiple Function Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MFP_GPE7 | | | | MFP_GPE6 | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MFP_GPE5 | | | | MFP_GPE4 | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MFP_GPE3 | | | | MFP_GPE2 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MFP_GPE1 | | | | MFP_GPE0 | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **MFP_GPE7** | **Pin PE.7 Multi Function Selection**<br>0000 = PE.7.<br>0001 = RMII1_REFCLK.<br>0110 = SD1_DATA3.<br>1001 = UR1_DSR.<br>Others = PE.7. |
| [27:24] | **MFP_GPE6** | **Pin PE.6 Multi Function Selection**<br>0000 = PE.6.<br>0001 = RMII1_TXEN.<br>0110 = SD1_DATA2.<br>1001 = UR1_DTR.<br>Others = PE.6. |
| [23:20] | **MFP_GPE5** | **Pin PE.5 Multi Function Selection**<br>0000 = PE.5.<br>0001 = RMII1_TXDATA1.<br>0110 = SD1_DATA1.<br>1001 = UR1_CTS.<br>1111 = CLK_OUT.<br>Others = PE.5. |
| [19:16] | **MFP_GPE4** | **Pin PE.4 Multi Function Selection**<br>0000 = PE.4.<br>0001 = RMII1_TXDATA0.<br>0110 = SD1_DATA0.<br>1001 = UR1_RTS.<br>Others = PE.4. |

| | | |
|---|---|---|
| [15:12] | **MFP_GPE3** | **Pin PE.3 Multi Function Selection**<br>0000 = PE.3.<br>0001 = RMII1_MDIO.<br>0110 = SD1_CLK.<br>1001 = UR1_RXD.<br>Others = PE.3. |
| [11:8] | **MFP_GPE2** | **Pin PE.2 Multi Function Selection**<br>0000 = PE.2.<br>0001 = RMII1_MDC.<br>0110 = SD1_CMD.<br>1001 = UR1_TXD.<br>Others = PE.2. |
| [7:4] | **MFP_GPE1** | **Pin PE.1 Multi Function Selection**<br>0000 = PE.1.<br>1001 = UR0_RXD.<br>Others = PE.1. |
| [3:0] | **MFP_GPE0** | **Pin PE.0 Multi Function Selection**<br>0000 = PE.0.<br>1001 = UR0_TXD.<br>Others = PE.0. |

## GPIOE High Byte Multiple Function Control Register (SYS_GPE_MFPH)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_GPE_MFPH | SYS_BA+0x094 | R/W | GPIOE High Byte Multiple Function Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MFP_GPE15 | | | | MFP_GPE14 | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MFP_GPE13 | | | | MFP_GPE12 | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MFP_GPE11 | | | | MFP_GPE10 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MFP_GPE9 | | | | MFP_GPE8 | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **MFP_GPE15** | **Pin PE.15 Multi Function Selection**<br>0000 = PE.15.<br>0111 = USBH_PPWR1.<br>Others = PE.15. |
| [27:24] | **MFP_GPE14** | **Pin PE.14 Multi Function Selection**<br>0000 = PE.14.<br>0111 = USBH_PPWR0.<br>Others = PE.14. |
| [23:20] | **MFP_GPE13** | **Pin PE.13 Multi Function Selection**<br>0000 = PE.13.<br>1001 = UR8_CTS.<br>1010 = UR3_RXD.<br>1111 = CLK_OUT.<br>Others = PE.13. |
| [19:16] | **MFP_GPE12** | **Pin PE.12 Multi Function Selection**<br>0000 = PE.12.<br>1001 = UR8_RTS.<br>1010 = UR3_TXD.<br>Others = PE.12. |
| [15:12] | **MFP_GPE11** | **Pin PE.11 Multi Function Selection**<br>0000 = PE.11.<br>0001 = RMII1_RXERR.<br>1001 = UR8_RXD.<br>Others = PE.11. |

| | | |
|---|---|---|
| [11:8] | **MFP_GPE10** | **Pin PE.10 Multi Function Selection**<br>0000 = PE.10.<br>0001 = RMII1_CRSDV.<br>1001 = UR8_TXD.<br>Others = PE.10. |
| [7:4] | **MFP_GPE9** | **Pin PE.9 Multi Function Selection**<br>0000 = PE.9.<br>0001 = RMII1_RXDATA1.<br>0110 = SD1_nPWR.<br>1001 = UR1_nCD.<br>Others = PE.9. |
| [3:0] | **MFP_GPE8** | **Pin PE.8 Multi Function Selection**<br>0000 = PE.8.<br>0001 = RMII1_RXDATA0.<br>0110 = SD1_nCD.<br>1001 = UR1_nRI.<br>Others = PE.8. |

### GPIOF Low Byte Multiple Function Control Register (SYS_GPF_MFPL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_GPF_MFPL | SYS_BA+0x098 | R/W | GPIOF Low Byte Multiple Function Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MFP_GPF7 | | | | MFP_GPF6 | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MFP_GPF5 | | | | MFP_GPF4 | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MFP_GPF3 | | | | MFP_GPF2 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MFP_GPF1 | | | | MFP_GPF0 | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **MFP_GPF7** | **Pin PF.7 Multi Function Selection**<br>0000 = PF.7.<br>0001 = RMII0_RXDATA1.<br>Others = PF.7. |
| [27:24] | **MFP_GPF6** | **Pin PF.6 Multi Function Selection**<br>0000 = PF.6.<br>0001 = RMII0_RXDATA0.<br>Others = PF.6. |
| [23:20] | **MFP_GPF5** | **Pin PF.5 Multi Function Selection**<br>0000 = PF.5.<br>0001 = RMII0_REFCLK.<br>Others = PF.5. |
| [19:16] | **MFP_GPF4** | **Pin PF.4 Multi Function Selection**<br>0000 = PF.4.<br>0001 = RMII0_TXEN.<br>Others = PF.4. |
| [15:12] | **MFP_GPF3** | **Pin PF.3 Multi Function Selection**<br>0000 = PF.3.<br>0001 = RMII0_TXDATA1.<br>Others = PF.3. |
| [11:8] | **MFP_GPF2** | **Pin PF.2 Multi Function Selection**<br>0000 = PF.2.<br>0001 = RMII0_TXDATA0.<br>Others = PF.2. |

| | | |
|---|---|---|
| [7:4] | **MFP_GPF1** | **Pin PF.1 Multi Function Selection**<br>0000 = PF.1.<br>0001 = RMII0_MDIO.<br>Others = PF.1. |
| [3:0] | **MFP_GPF0** | **Pin PF.0 Multi Function Selection**<br>0000 = PF.0.<br>0001 = RMII0_MDC.<br>Others = PF.0. |

### GPIOF High Byte Multiple Function Control Register (SYS_GPF_MFPH)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_GPF_MFPH | SYS_BA+0x09C | R/W | GPIOF High Byte Multiple Function Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MFP_GPF15 | | | | MFP_GPF14 | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MFP_GPF13 | | | | MFP_GPF12 | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MFP_GPF11 | | | | MFP_GPF10 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MFP_GPF9 | | | | MFP_GPF8 | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **MFP_GPF15** | **Pin PF.15 Multi Function Selection**<br>0000 = PF.15.<br>1111 = EINT4.<br>Others = PF.15. |
| [27:24] | **MFP_GPF14** | **Pin PF.14 Multi Function Selection**<br>0000 = PF.14.<br>1001 = UR2_CTS.<br>1101 = ETMR3_CAP.<br>1111 = EINT3.<br>Others = PF.14. |
| [23:20] | **MFP_GPF13** | **Pin PF.13 Multi Function Selection**<br>0000 = PF.13.<br>1001 = UR2_RTS.<br>1101 = ETMR3_TGL.<br>1111 = EINT2.<br>Others = PF.13. |
| [19:16] | **MFP_GPF12** | **Pin PF.12 Multi Function Selection**<br>0000 = PF.12.<br>1001 = UR2_RXD.<br>1101 = ETMR2_CAP.<br>1111 = EINT1.<br>Others = PF.12. |

| | | |
|---|---|---|
| [15:12] | **MFP_GPF11** | **Pin PF.11 Multi Function Selection**<br>0000 = PF.11.<br>1001 = UR2_TXD.<br>1101 = ETMR2_TGL.<br>1111 = EINT0.<br>Others = PF.11. |
| [11:8] | **MFP_GPF10** | **Pin PF.10 Multi Function Selection**<br>0000 = PF.10.<br>0111 = USBH_PPWR.<br>Others = PF.10. |
| [7:4] | **MFP_GPF9** | **Pin PF.9 Multi Function Selection**<br>0000 = PF.9.<br>0001 = RMII0_RXERR.<br>Others = PF.9. |
| [3:0] | **MFP_GPF8** | **Pin PF.8 Multi Function Selection**<br>0000 = PF.8.<br>0001 = RMII0_CRSDV.<br>Others = PF.8. |

## GPIOG Low Byte Multiple Function Control Register (SYS_GPG_MFPL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_GPG_MFPL | SYS_BA+0x0A0 | R/W | GPIOG Low Byte Multiple Function Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MFP_GPG7 | | | | MFP_GPG6 | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MFP_GPG5 | | | | MFP_GPG4 | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MFP_GPG3 | | | | MFP_GPG2 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MFP_GPG1 | | | | MFP_GPG0 | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **MFP_GPG7** | **Pin PG.7 Multi Function Selection**<br>0000 = PG.7.<br>0010 = LCD_HSYNC.<br>Others = PG.7. |
| [27:24] | **MFP_GPG6** | **Pin PG.6 Multi Function Selection**<br>0000 = PG.6.<br>0010 = LCD_CLK.<br>Others = PG.6. |
| [23:20] | **MFP_GPG5** | **Pin PG.5 Multi Function Selection**<br>0000 = PG.5.<br>0101 = NAND_RDY1.<br>1001 = UR7_RXD.<br>1011 = SPI1_DATA3.<br>Others = PG.5. |
| [19:16] | **MFP_GPG4** | **Pin PG.4 Multi Function Selection**<br>0000 = PG.4.<br>0101 = NAND_nCS1.<br>1001 = UR7_TXD.<br>1011 = SPI1_DATA2.<br>Others = PG.4. |
| [15:12] | **MFP_GPG3** | **Pin PG.3 Multi Function Selection**<br>0000 = PG.3.<br>1000 = I2C1_SDA.<br>Others = PG.3. |

| | | |
|---|---|---|
| [11:8] | **MFP_GPG2** | **Pin PG.2 Multi Function Selection**<br>0000 = PG.2.<br>1000 = I2C1_SCL.<br>Others = PG.2. |
| [7:4] | **MFP_GPG1** | **Pin PG.1 Multi Function Selection**<br>0000 = PG.1.<br>1000 = I2C0_SDA.<br>Others = PG.1. |
| [3:0] | **MFP_GPG0** | **Pin PG.0 Multi Function Selection**<br>0000 = PG.0.<br>1000 = I2C0_SCL.<br>Others = PG.0. |

## GPIOG High Byte Multiple Function Control Register (SYS_GPG_MFPH)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_GPG_MFPH | SYS_BA+0x0A4 | R/W | GPIOG High Byte Multiple Function Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MFP_GPG15 | | | | MFP_GPG14 | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MFP_GPG13 | | | | MFP_GPG12 | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MFP_GPG11 | | | | MFP_GPG10 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MFP_GPG9 | | | | MFP_GPG8 | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **MFP_GPG15** | **Pin PG.15 Multi Function Selection**<br>0000 = PG.15.<br>1111 = EINT5.<br>Others = PG.15. |
| [27:24] | **MFP_GPG14** | **Pin PG.14 Multi Function Selection**<br>0000 = PG.14.<br>1000 = I2S_WS.<br>1001 = UR6_CTS.<br>1010 = SMC0_CD.<br>1111 = CLK_OUT.<br>Others = PG.14. |
| [23:20] | **MFP_GPG13** | **Pin PG.13 Multi Function Selection**<br>0000 = PG.13.<br>1000 = I2S_BITCLK.<br>1001 = UR6_RTS.<br>1010 = SMC0_PWR.<br>Others = PG.13. |
| [19:16] | **MFP_GPG12** | **Pin PG.12 Multi Function Selection**<br>0000 = PG.12.<br>1000 = I2S_DATAI.<br>1001 = UR6_RXD.<br>1010 = SMC0_DATA.<br>Others = PG.12. |

| | | |
|---|---|---|
| [15:12] | **MFP_GPG11** | **Pin PG.11 Multi Function Selection**<br>0000 = PG.11.<br>1000 = I2S_DATAO.<br>1001 = UR6_TXD.<br>1010 = SMC0_CLK.<br>Others = PG.11. |
| [11:8] | **MFP_GPG10** | **Pin PG.10 Multi Function Selection**<br>0000 = PG.10.<br>1000 = I2S_SYSCLK.<br>1010 = SMC0_RST.<br>Others = PG.10. |
| [7:4] | **MFP_GPG9** | **Pin PG.9 Multi Function Selection**<br>0000 = PG.9.<br>0010 = LCD_DEN.<br>Others = PG.9. |
| [3:0] | **MFP_GPG8** | **Pin PG.8 Multi Function Selection**<br>0000 = PG.8.<br>0010 = LCD_VSYNC.<br>Others = PG.8. |

## GPIOH Low Byte Multiple Function Control Register (SYS_GPH_MFPL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_GPH_MFPL | SYS_BA+0x0A8 | R/W | GPIOH Low Byte Multiple Function Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MFP_GPH7 | | | | MFP_GPH6 | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MFP_GPH5 | | | | MFP_GPH4 | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MFP_GPH3 | | | | MFP_GPH2 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MFP_GPH1 | | | | MFP_GPH0 | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **MFP_GPH7** | **Pin PH.7 Multi Function Selection**<br>0000 = PH.7.<br>0100 = KPI_ROW3.<br>0110 = SD1_CLK.<br>1001 = UR1_CTS.<br>1110 = EBI_ADDR3.<br>1111 = EINT7.<br>Others = PH.7. |
| [27:24] | **MFP_GPH6** | **Pin PH.6 Multi Function Selection**<br>0000 = PH.6.<br>0100 = KPI_ROW2.<br>0110 = SD1_CMD.<br>1001 = UR1_RTS.<br>1110 = EBI_ADDR2.<br>1111 = EINT6.<br>Others = PH.6. |
| [23:20] | **MFP_GPH5** | **Pin PH.5 Multi Function Selection**<br>0000 = PH.5.<br>0100 = KPI_ROW1.<br>1001 = UR1_RXD.<br>1110 = EBI_ADDR1.<br>1111 = EINT5.<br>Others = PH.5. |

| | | |
|---|---|---|
| [19:16] | **MFP_GPH4** | **Pin PH.4 Multi Function Selection**<br>0000 = PH.4.<br>0100 = KPI_ROW0.<br>1001 = UR1_TXD.<br>1101 = RTC_TICK.<br>1110 = EBI_ADDR0.<br>1111 = EINT4.<br>Others = PH.4. |
| [15:12] | **MFP_GPH3** | **Pin PH.3 Multi Function Selection**<br>0000 = PH.3.<br>1000 = I2C1_SDA.<br>1001 = UR9_RXD.<br>1100 = CAN0_TX.<br>1101 = PWM3.<br>1111 = EINT3.<br>Others = PH.3. |
| [11:8] | **MFP_GPH2** | **Pin PH.2 Multi Function Selection**<br>0000 = PH.2.<br>1000 = I2C1_SCL.<br>1001 = UR9_TXD.<br>1100 = CAN0_RX.<br>1101 = PWM2.<br>1111 = EINT2.<br>Others = PH.2. |
| [7:4] | **MFP_GPH1** | **Pin PH.1 Multi Function Selection**<br>0000 = PH.1.<br>0111 = USBH_OVRCUR.<br>1111 = EINT1.<br>Others = PH.1. |
| [3:0] | **MFP_GPH0** | **Pin PH.0 Multi Function Selection**<br>0000 = PH.0.<br>0111 = USBD_VBUSVLD.<br>1111 = EINT0.<br>Others = PH.0. |

## GPIOH High Byte Multiple Function Control Register (SYS_GPH_MFPH)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_GPH_MFPH | SYS_BA+0x0AC | R/W | GPIOH High Byte Multiple Function Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MFP_GPH15 | | | | MFP_GPH14 | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MFP_GPH13 | | | | MFP_GPH12 | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MFP_GPH11 | | | | MFP_GPH10 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MFP_GPH9 | | | | MFP_GPH8 | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **MFP_GPH15** | **Pin PH.15 Multi Function Selection**<br>0000 = PH.15.<br>0100 = KPI_COL7.<br>1001 = UR8_CTS.<br>1100 = CAN1_TX.<br>1110 = EBI_nBE1.<br>1111 = CLK_OUT.<br>Others = PH.15. |
| [27:24] | **MFP_GPH14** | **Pin PH.14 Multi Function Selection**<br>0000 = PH.14.<br>0100 = KPI_COL6.<br>1001 = UR8_RTS.<br>1100 = CAN1_RX.<br>1110 = EBI_nBE0.<br>Others = PH.14. |
| [23:20] | **MFP_GPH13** | **Pin PH.13 Multi Function Selection**<br>0000 = PH.13.<br>0100 = KPI_COL5.<br>0110 = SD1_nPWR.<br>1001 = UR8_RXD.<br>1011 = SPI1_SS1.<br>1110 = EBI_ADDR9.<br>Others = PH.13. |

| | | |
|---|---|---|
| [19:16] | **MFP_GPH12** | **Pin PH.12 Multi Function Selection**<br>0000 = PH.12.<br>0100 = KPI_COL4.<br>0110 = SD1_nCD.<br>1001 = UR8_TXD.<br>1011 = SPI0_SS1.<br>1110 = EBI_ADDR8.<br>Others = PH.12. |
| [15:12] | **MFP_GPH11** | **Pin PH.11 Multi Function Selection**<br>0000 = PH.11.<br>0100 = KPI_COL3.<br>0110 = SD1_DATA3.<br>1001 = UR4_CTS.<br>1110 = EBI_ADDR7.<br>Others = PH.11. |
| [11:8] | **MFP_GPH10** | **Pin PH.10 Multi Function Selection**<br>0000 = PH.10.<br>0100 = KPI_COL2.<br>0110 = SD1_DATA2.<br>1001 = UR4_RTS.<br>1110 = EBI_ADDR6.<br>Others = PH.10. |
| [7:4] | **MFP_GPH9** | **Pin PH.9 Multi Function Selection**<br>0000 = PH.9.<br>0100 = KPI_COL1.<br>0110 = SD1_DATA1.<br>1001 = UR4_RXD.<br>1110 = EBI_ADDR5.<br>Others = PH.9. |
| [3:0] | **MFP_GPH8** | **Pin PH.8 Multi Function Selection**<br>0000 = PH.8.<br>0100 = KPI_COL0.<br>0110 = SD1_DATA0.<br>1001 = UR4_TXD.<br>1110 = EBI_ADDR4.<br>Others = PH.8. |

## GPIOI Low Byte Multiple Function Control Register (SYS_GPI_MFPL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_GPI_MFPL | SYS_BA+0x0B0 | R/W | GPIOI Low Byte Multiple Function Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MFP_GPI7 | | | | MFP_GPI6 | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MFP_GPI5 | | | | MFP_GPI4 | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MFP_GPI3 | | | | MFP_GPI2 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MFP_GPI1 | | | | MFP_GPI0 | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **MFP_GPI7** | **Pin PI.7 Multi Function Selection**<br>0000 = PI.7.<br>0011 = VCAP_FIELD.<br>0100 = SD1_DATA0.<br>0101 = NAND_RDY0.<br>0110 = eMMC_DATA3.<br>1001 = UR1_RTS.<br>1010 = SMC1_CLK.<br>1011 = SPI1_DATAO (SPI1_DATA0).<br>1110 = EBI_DATA7.<br>Others = PI.7. |
| [27:24] | **MFP_GPI6** | **Pin PI.6 Multi Function Selection**<br>0000 = PI.6.<br>0011 = VCAP_VSYNC.<br>0100 = SD1_CLK.<br>0101 = NAND_nRE.<br>0110 = eMMC_CLK.<br>1001 = UR1_RXD.<br>1010 = SMC1_RST.<br>1011 = SPI1_CLK.<br>1110 = EBI_DATA6.<br>Others = PI.6. |

| | | |
|---|---|---|
| [23:20] | **MFP_GPI5** | **Pin PI.5 Multi Function Selection**<br>0000 = PI.5.<br>0011 = VCAP_HSYNC.<br>0100 = SD1_CMD.<br>0101 = NAND_nWE.<br>0110 = eMMC_CMD.<br>1001 = UR1_TXD.<br>1011 = SPI1_SS0.<br>1110 = EBI_DATA5.<br>Others = PI.5. |
| [19:16] | **MFP_GPI4** | **Pin PI.4 Multi Function Selection**<br>0000 = PI.4.<br>0011 = VCAP_PCLK.<br>0101 = NAND_CLE.<br>1000 = I2C1_SDA.<br>1100 = CAN0_TX.<br>1110 = EBI_DATA4.<br>Others = PI.4. |
| [15:12] | **MFP_GPI3** | **Pin PI.3 Multi Function Selection**<br>0000 = PI.3.<br>0011 = VCAP_CLKO.<br>0101 = NAND_ALE.<br>1000 = I2C1_SCL.<br>1100 = CAN0_RX.<br>1101 = RTC_TICK.<br>1110 = EBI_DATA3.<br>Others = PI.3. |
| [11:8] | **MFP_GPI2** | **Pin PI.2 Multi Function Selection**<br>0000 = PI.2.<br>0101 = NAND_nWP.<br>1001 = UR7_RXD.<br>1110 = EBI_DATA2.<br>1111 = EINT7.<br>Others = PI.2. |
| [7:4] | **MFP_GPI1** | **Pin PI.1 Multi Function Selection**<br>0000 = PI.1.<br>0101 = NAND_nCS0.<br>1001 = UR7_TXD.<br>1110 = EBI_DATA1.<br>1111 = EINT6.<br>Others = PI.1. |
| [3:0] | **MFP_GPI0** | **Pin PI.0 Multi Function Selection**<br>0000 = PI.0.<br>1110 = EBI_DATA0.<br>Others = PI.0. |

### GPIOI High Byte Multiple Function Control Register (SYS_GPI_MFPH)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_GPI_MFPH | SYS_BA+0x0B4 | R/W | GPIOI High Byte Multiple Function Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MFP_GPI15 | | | | MFP_GPI14 | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MFP_GPI13 | | | | MFP_GPI12 | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MFP_GPI11 | | | | MFP_GPI10 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MFP_GPI9 | | | | MFP_GPI8 | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **MFP_GPI15** | **Pin PI.15 Multi Function Selection**<br>0000 = PI.15.<br>0011 = VCAP_DATA7.<br>0101 = NAND_DATA7.<br>1001 = UR8_CTS.<br>1010 = SMC0_CD.<br>1110 = EBI_DATA15.<br>1111 = CLK_OUT.<br>Others = PI.15. |
| [27:24] | **MFP_GPI14** | **Pin PI.14 Multi Function Selection**<br>0000 = PI.14.<br>0011 = VCAP_DATA6.<br>0101 = NAND_DATA6.<br>1001 = UR8_RTS.<br>1010 = SMC0_PWR.<br>1110 = EBI_DATA14.<br>Others = PI.14. |

| | | |
|---|---|---|
| [23:20] | **MFP_GPI13** | **Pin PI.13 Multi Function Selection**<br>0000 = PI.13.<br>0011 = VCAP_DATA5.<br>0100 = SD1_nPWR.<br>0101 = NAND_DATA5.<br>1001 = UR8_RXD.<br>1010 = SMC0_DATA.<br>1110 = EBI_DATA13.<br>Others = PI.13. |
| [19:16] | **MFP_GPI12** | **Pin PI.12 Multi Function Selection**<br>0000 = PI.12.<br>0011 = VCAP_DATA4.<br>0100 = SD1_nCD.<br>0101 = NAND_DATA4.<br>1001 = UR8_TXD.<br>1010 = SMC0_CLK.<br>1110 = EBI_DATA12.<br>Others = PI.12. |
| [15:12] | **MFP_GPI11** | **Pin PI.11 Multi Function Selection**<br>0000 = PI.11.<br>0011 = VCAP_DATA3.<br>0101 = NAND_DATA3.<br>1010 = SMC0_RST.<br>1110 = EBI_DATA11.<br>Others = PI.11. |
| [11:8] | **MFP_GPI10** | **Pin PI.10 Multi Function Selection**<br>0000 = PI.10.<br>0011 = VCAP_DATA2.<br>0100 = SD1_DATA3.<br>0101 = NAND_DATA2.<br>0110 = eMMC_DATA2.<br>1001 = UR4_RXD.<br>1010 = SMC1_CD.<br>1110 = EBI_DATA10.<br>Others = PI.10. |
| [7:4] | **MFP_GPI9** | **Pin PI.9 Multi Function Selection**<br>0000 = PI.9.<br>0011 = VCAP_DATA1.<br>0100 = SD1_DATA2.<br>0101 = NAND_DATA1.<br>0110 = eMMC_DATA1.<br>1001 = UR4_TXD.<br>1010 = SMC1_PWR.<br>1110 = EBI_DATA9.<br>Others = PI.9. |

| | | |
|---|---|---|
| [3:0] | **MFP_GPI8** | **Pin PI.8 Multi Function Selection**<br>0000 = PI.8.<br>0011 = VCAP_DATA0.<br>0100 = SD1_DATA1.<br>0101 = NAND_DATA0.<br>0110 = eMMC_DATA0.<br>1001 = UR1_CTS.<br>1010 = SMC1_DATA.<br>1011 = SPI1_DATAI (SPI1_DATA1).<br>1110 = EBI_DATA8.<br>Others = PI.8. |

## GPIOJ Low Byte Multiple Function Control Register (SYS_GPJ_MFPL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_GPJ_MFPL | SYS_BA+0x0B8 | R/W | GPIOJ Low Byte Multiple Function Control Register | 0x000X_XXXX[1] |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | MFP_GPJ4 | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MFP_GPJ3 | | | | MFP_GPJ2 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MFP_GPJ1 | | | | MFP_GPJ0 | | | |

| Bits | Description | |
|---|---|---|
| [31:20] | **Reserved** | Reserved. |
| [19:16] | **MFP_GPJ4** | **Pin PJ.4 Multi Function Selection**<br>0000 = PJ.4.<br>1111 = JTAG_nTRST.<br>Others = PJ.4. |
| [15:12] | **MFP_GPJ3** | **Pin PJ.3 Multi Function Selection**<br>0000 = PJ.3.<br>1111 = JTAG_TDO.<br>Others = PJ.3. |
| [11:8] | **MFP_GPJ2** | **Pin PJ.2 Multi Function Selection**<br>0000 = PJ.2.<br>1111 = JTAG_TDI.<br>Others = PJ.2. |
| [7:4] | **MFP_GPJ1** | **Pin PJ.1 Multi Function Selection**<br>0000 = PJ.1.<br>1111 = JTAG_TMS.<br>Others = PJ.1. |
| [3:0] | **MFP_GPJ0** | **Pin PJ.0 Multi Function Selection**<br>0000 = PJ.0.<br>1111 = JTAG_TCK.<br>Others = PJ.0. |

**Note: [1] Dependents on power-on setting.**

## DDR I/O Driving Strength Control Register (SYS_DDR_DSCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_DDR_DSCTL | SYS_BA+0x0F0 | R/W | DDR I/O Driving Strength 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 |
| DATA_DS | | ADDR_DS | | CTRL_DS | | CLK_DS | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:6] | **DATA_DS** | **DDR Data I/O Driving Strength Selection**<br>This bit controls the driving strength for DDR I/O used as data.<br><br>00 = Reserved.<br>01 = Reduced Strength.<br>10 = Reserved.<br>11 = Full Strength. |
| [5:4] | **ADDR_DS** | **DDR Address I/O Driving Strength Selection**<br>This bit controls the driving strength for DDR I/O used as address.<br><br>00 = Reserved.<br>01 = Reduced Strength.<br>10 = Reserved.<br>11 = Full Strength. |
| [3:2] | **CTRL_DS** | **DDR Control I/O Driving Strength Selection**<br>This bit controls the driving strength for DDR I/O used as control signals.<br><br>00 = Reserved.<br>01 = Reduced Strength.<br>10 = Reserved.<br>11 = Full Strength. |

| | | |
|---|---|---|
| [1:0] | **CLK_DS** | **DDR Clock I/O Driving Strength Selection**<br>This bit controls the driving strength for DDR I/O used as clock.<br><br>00 = Reserved.<br>01 = Reduced Strength.<br>10 = Reserved.<br>11 = Full Strength. |

### Power-On-Reset Disable Control Register (SYS_PORDISCR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_PORDISCR | SYS_BA+0x100 | R/W | Power-On-Reset Disable Control 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 |
| POR_DIS_CODE | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| POR_DIS_CODE | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **POR_DIS_CODE** | **Power-on-reset Disable Code (Write-protection Bits)**<br>When powered on, the Power-On-Reset (POR) circuit generates a reset signal to reset whole chip function. However, after power is ready, the POR circuit would consume a few power. To minimize the POR circuit power consumption, user to disable POR circuit by writing 0x5AA5 to this field.<br>The POR circuit will become active again when this field is set to other value or chip is reset by other reset source, including /RESET pin, Watchdog, LVR reset and the software chip reset function.<br>This field is protected. It means that before programming it, user has to write “59h”, “16h” and “88h” to address 0xB000_01FC continuously to disable the register protection. Refer to the register REGWRPROT at address SYS_BA+0x1FC for detail. |

## Register Write-Protection Control Register (SYS_REGWPCTL)

Some of the system control registers need to be protected to avoid inadvertent write and disturb the chip operation. These system control registers are protected after the power-on reset till user to disable register protection. For user to program these protected registers, a register protection disable sequence needs to be followed by a special programming. The register protection disable sequence is writing the data “59h”, “16h” “88h” to the register REGWRPROT address at 0xB000_01FC continuously. Any different data value, different sequence or any other write to other address during these three data writing will abort the whole sequence.

After the protection is disabled, user can check the protection disable bit at address 0xB000_01FC bit0, 1 is protection disable, and 0 is protection enable. Then user can update the target protected register value and then write any data to the address “0xB000_01FC” to enable register protection.

This register is write for disable/enable register protection and read for the REGPROTDIS status

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_REGWPCTL** | SYS_BA+0x1FC | R/W | Register Write-Protection 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** |
| **REGWRPRT[7:1]** | | | | | | | **REGWRPRT[0]**<br>**REGPRTDIS** |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [7:0] | **REGWRPRT** | **Register Write-protection Code (Write Only)**<br>Some registers have write-protection function. Writing these registers has to disable the protected function by writing the sequence value “59h”, “16h”, “88h” to this field. After this sequence is completed, the REGPROTDIS bit will be set to 1 and write-protection registers can be normal write. |

| | | |
|---|---|---|
| [0] | **REGPRTDIS** | **Register Write-protection Disable Indicator (Read Only)**<br><br>0 = Write-protection Enabled for writing protected registers. Any write to the protected register is ignored.<br>1 = Write-protection Disabled for writing protected registers.<br><br>The protected registers are:<br>SYS_PDID: Product Identifier Register, address 0xB000_0000<br>SYS_MISCFCR: Miscellaneous Function Control Register, address 0xB000_0030<br>SYS_AHBIPRST: AHB IP Reset Control Register, address 0xB000_0060<br>SYS_APBIPRST0: APB IP Reset Control Register 0, address 0xB000_0064<br>SYS_APBIPRST1: APB IP Reset Control Register 1, address 0xB000_0068<br>SYS_PORDISCR: Power-On-Reset Disable Control Register, address 0xB000_0100. |

## 5.3 Clock Controller (CLK_CTL)

### 5.3.1 Overview

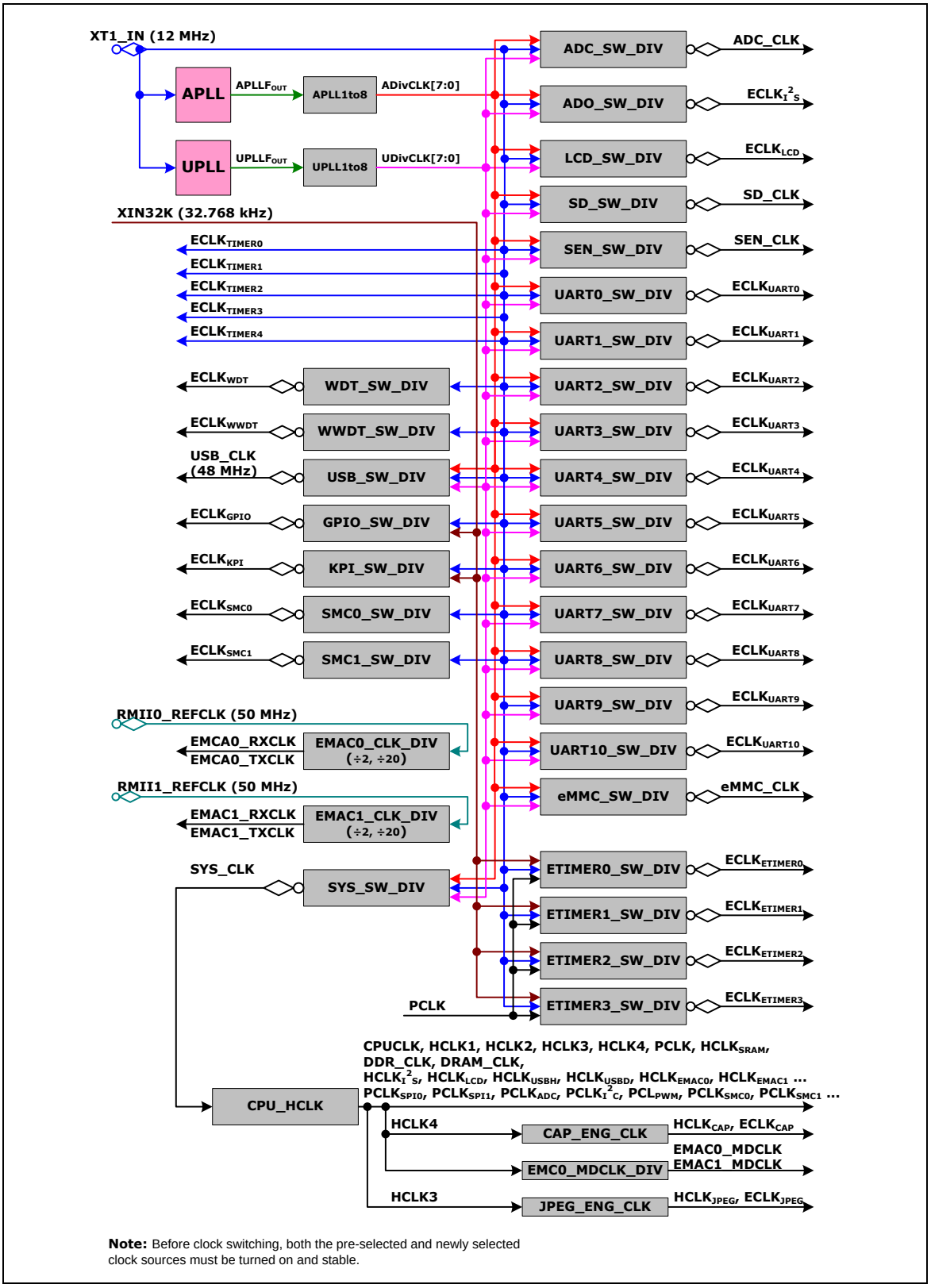
The clock controller generates all clocks for Video, Audio, CPU, system bus and all functionalities. This chip includes two PLL modules. The clock source for each functionality comes from the PLL, or from the external crystal input directly. For each clock there is a bit on the CLKEN register to control the clock ON or OFF individually, and the divider setting is in the CLK_DIVCTL register. The register can also be used to control the clock enable or disable for power control.

### 5.3.2 Features

- Supports two PLLs, up to 500 MHz, for high performance system operation
- External 12 MHz high speed crystal input for precise timing operation
- External 32.768 kHz low speed crystal input for RTC function and low speed clock source

### 5.3.3 Block diagram

5.3.3.1 *Clock Controller Top View*




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

Figure 5.3-1 Clock Controller Block Diagram

### 5.3.3.2 *ADC Controller Clock Divider*

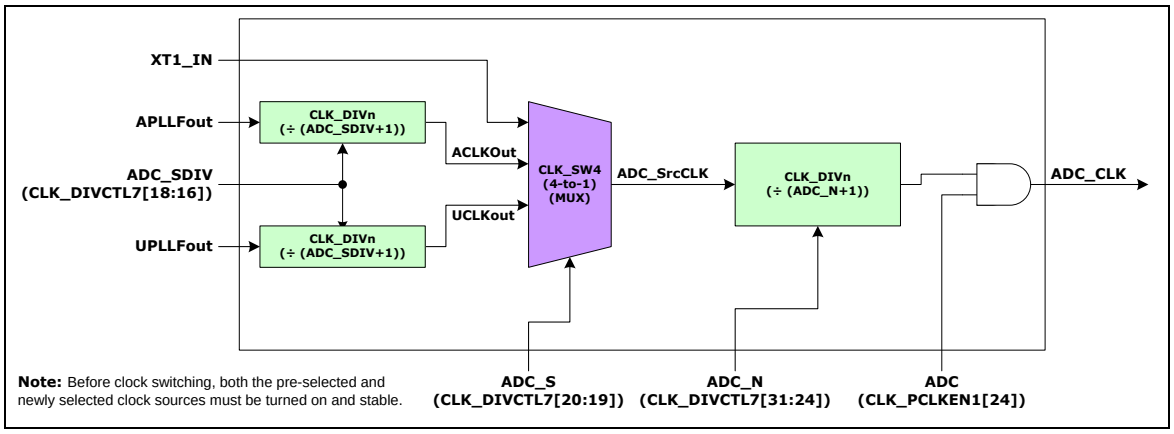



Figure 5.3-2 ADC Controller Clock Divider Block Diagram

### 5.3.3.3 *eMMC Host Controller Clock Divider*

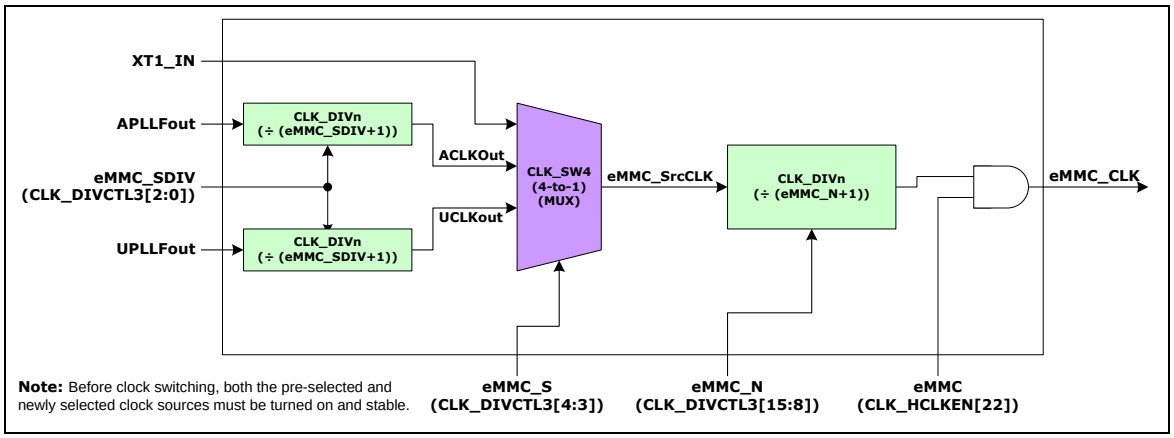



Figure 5.3-3 eMMC Host Controller Clock Divider Block Diagram

### 5.3.3.4 *Enhanced Timer Clock Divider*

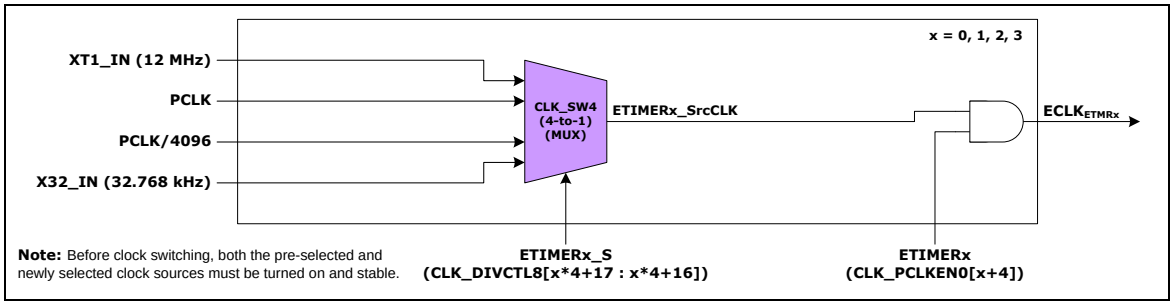



Figure 5.3-4 Enhanced Timer Clock Divider Clock Diagram

### 5.3.3.5 *Ethernet MAC Controller Clock Divider*

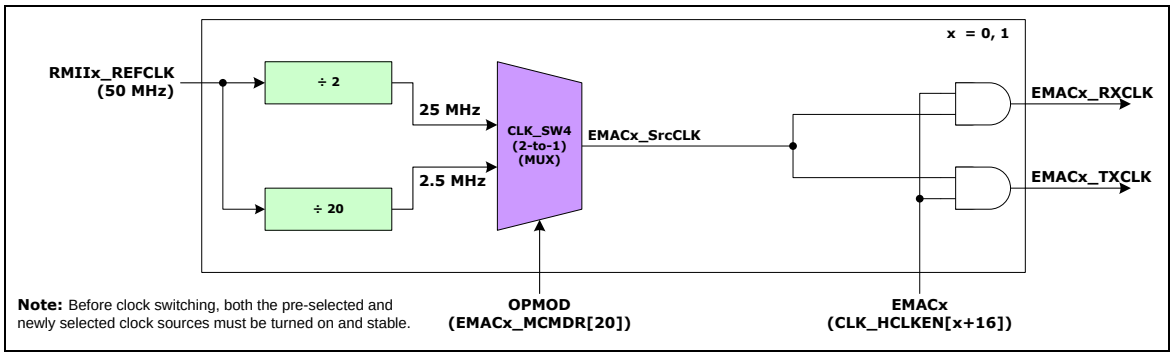



Figure 5.3-5 Ethernet MAC Controller Clock Divider Block Diagram

### 5.3.3.6 *GPIO Clock Divider*

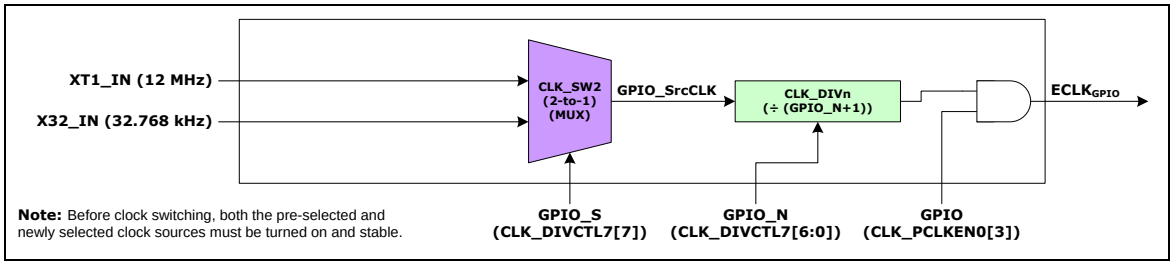



Figure 5.3-6 GPIO Clock Divider Block Diagram

### 5.3.3.7 *I²S Controller Clock Divider*

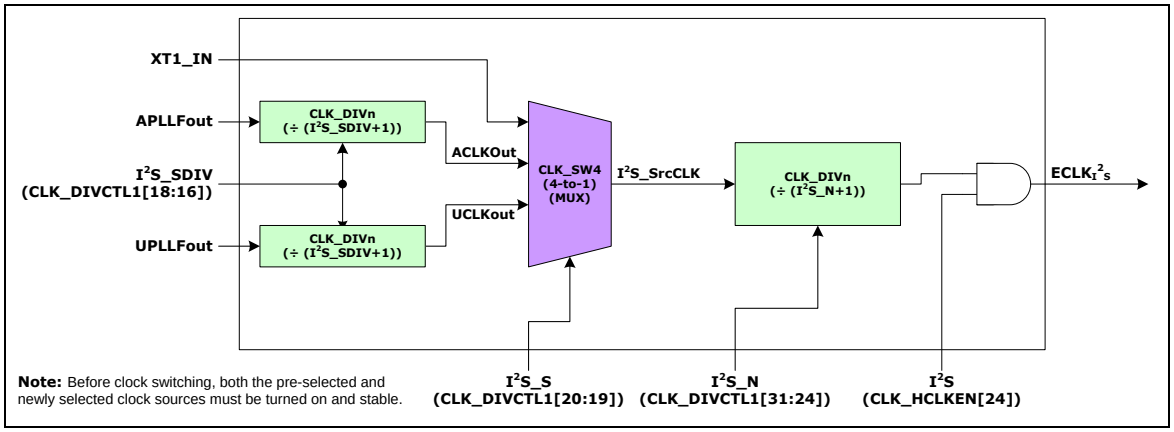



Figure 5.3-7 I²S Controller Clock Divider Block Diagram

### 5.3.3.8 *KPI Controller Clock Divider*

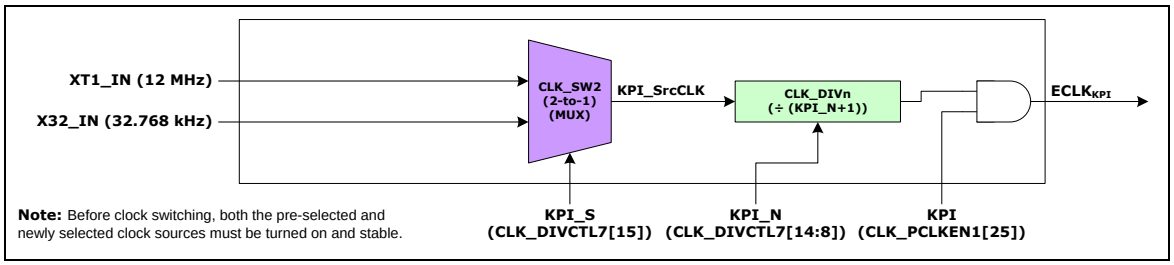



Figure 5.3-8 KPI Controller Clock Divider Block Diagram

### 5.3.3.9 *LCD Display Controller Clock Divider*

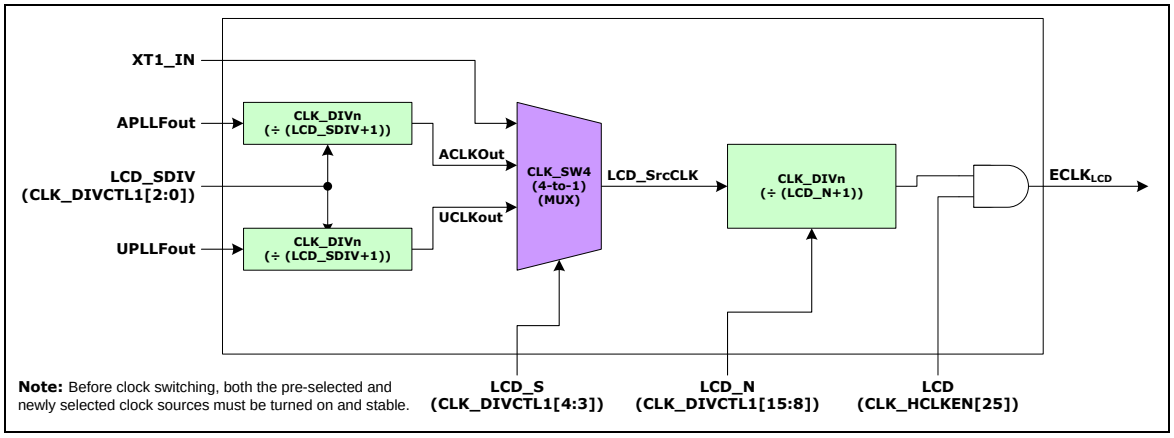



Figure 5.3-9 LCD Display Controller Clock Divider Block Diagram

### 5.3.3.10 *Reference Clock Output Divider*

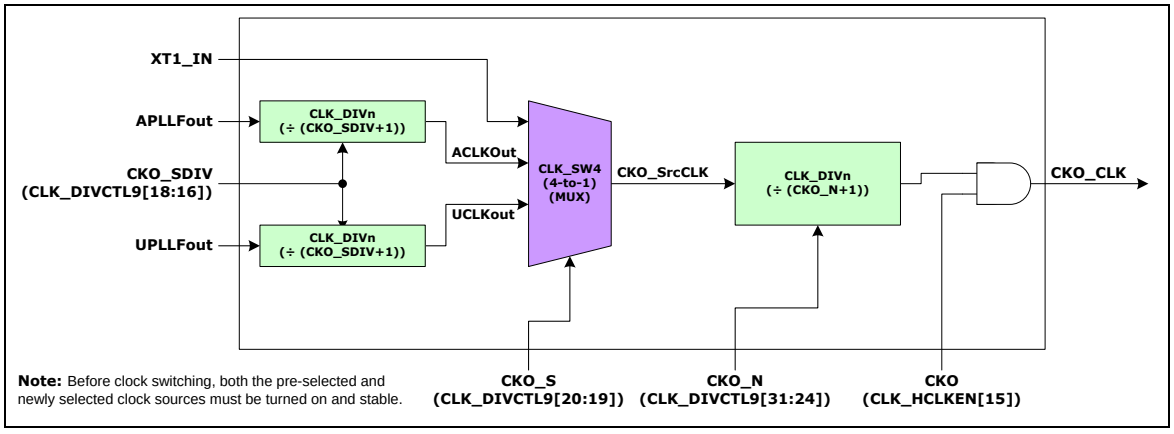



Figure 5.3-10 Reference Clock Output Divider Block Diagram

### 5.3.3.11 *SD Card Host Controller Clock Divider*

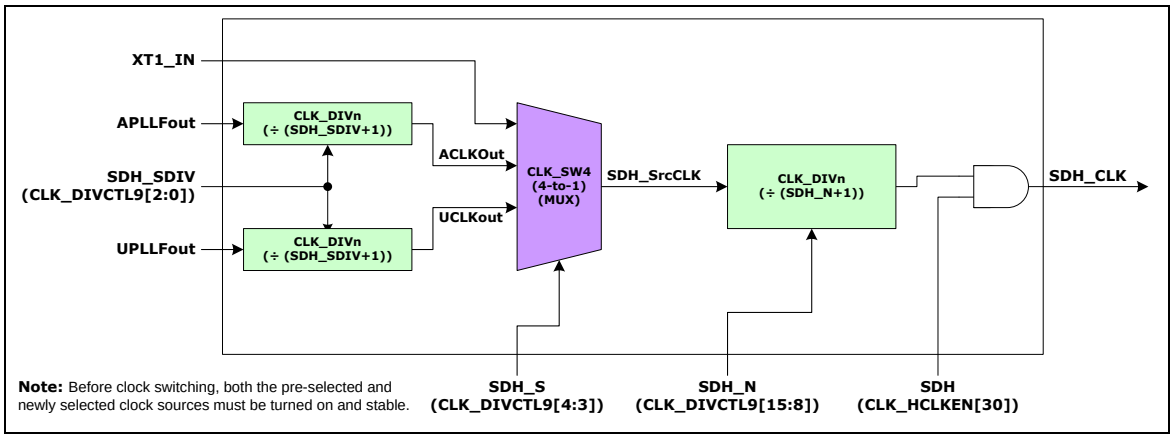



Figure 5.3-11 SD Card Host Controller Clock Divider Block Diagram

### 5.3.3.12 *Smart Card Host Controller Clock Divider*

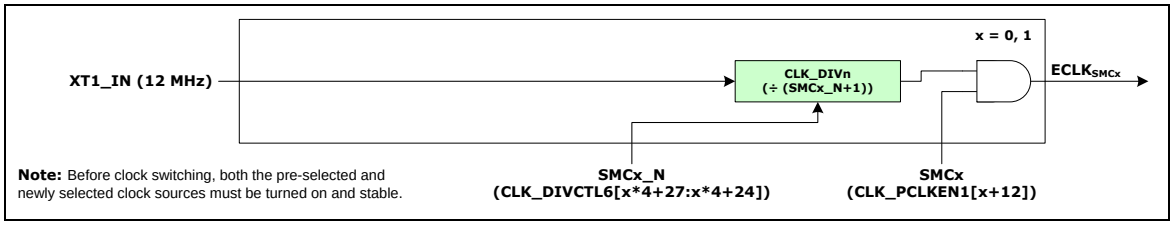



Figure 5.3-12 Smart Card Host Controller Clock Divider Block Diagram

### 5.3.3.13 *CMOS Sensor Clock Divider*

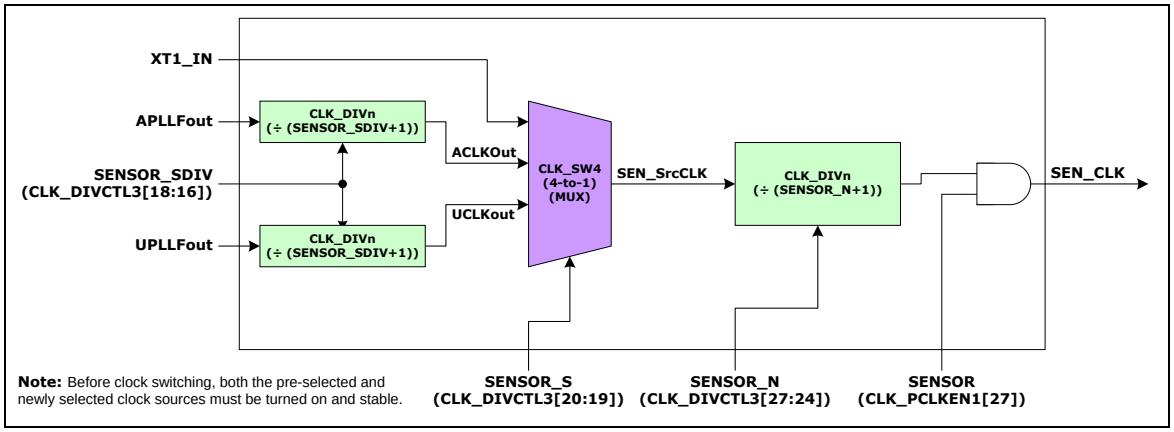



Figure 5.3-13 CMOS Sensor Controller Divider Block Diagram

### 5.3.3.14 *UART Clock Divider*

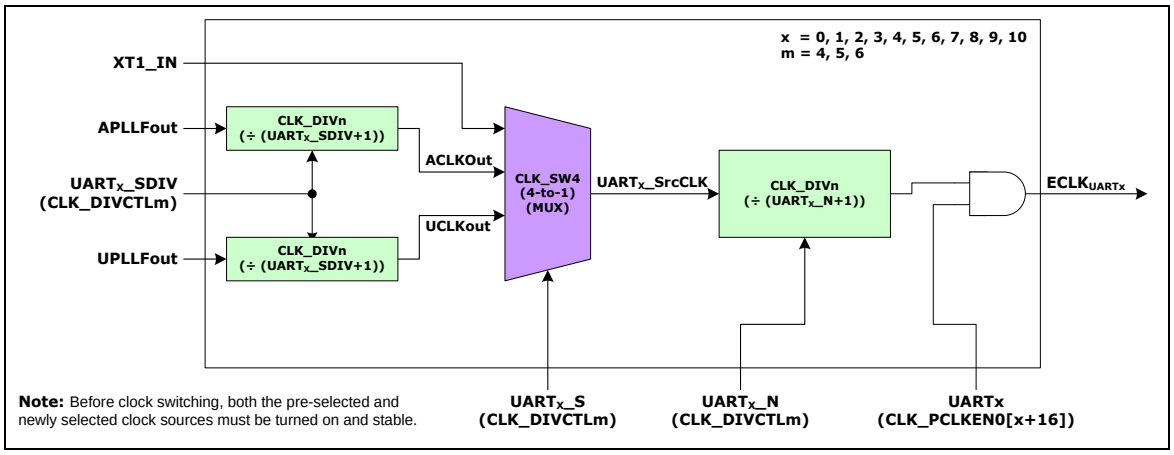



Figure 5.3-14 UART Clock Divider Block Diagram

### 5.3.3.15 *USB 1.1 Host 48 MHz Clock Divider*

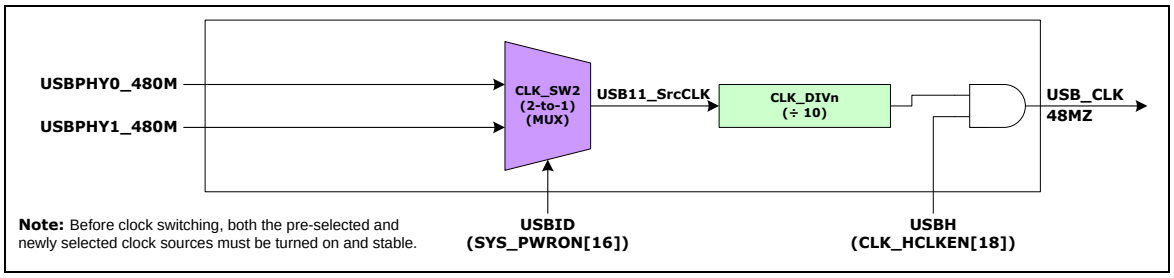



Figure 5.3-15 USB 1.1 Host Controller 48 MHz Clock Divider Block Diagram

### 5.3.3.16 *Watchdog Timer Clock Divider*

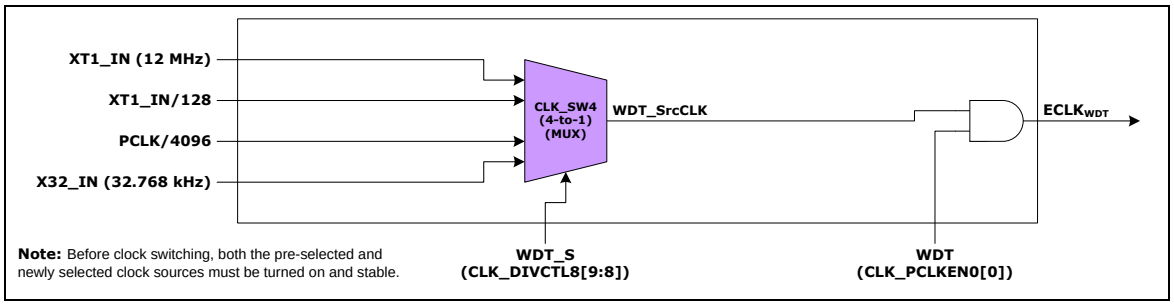



Figure 5.3-16 Watchdog Timer Clock Divider Block Diagram

### 5.3.3.17 *Windowed Watchdog Timer Clock Divider*

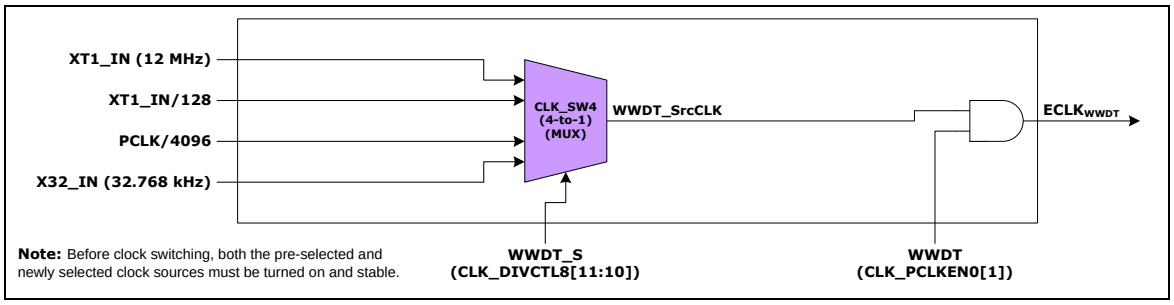
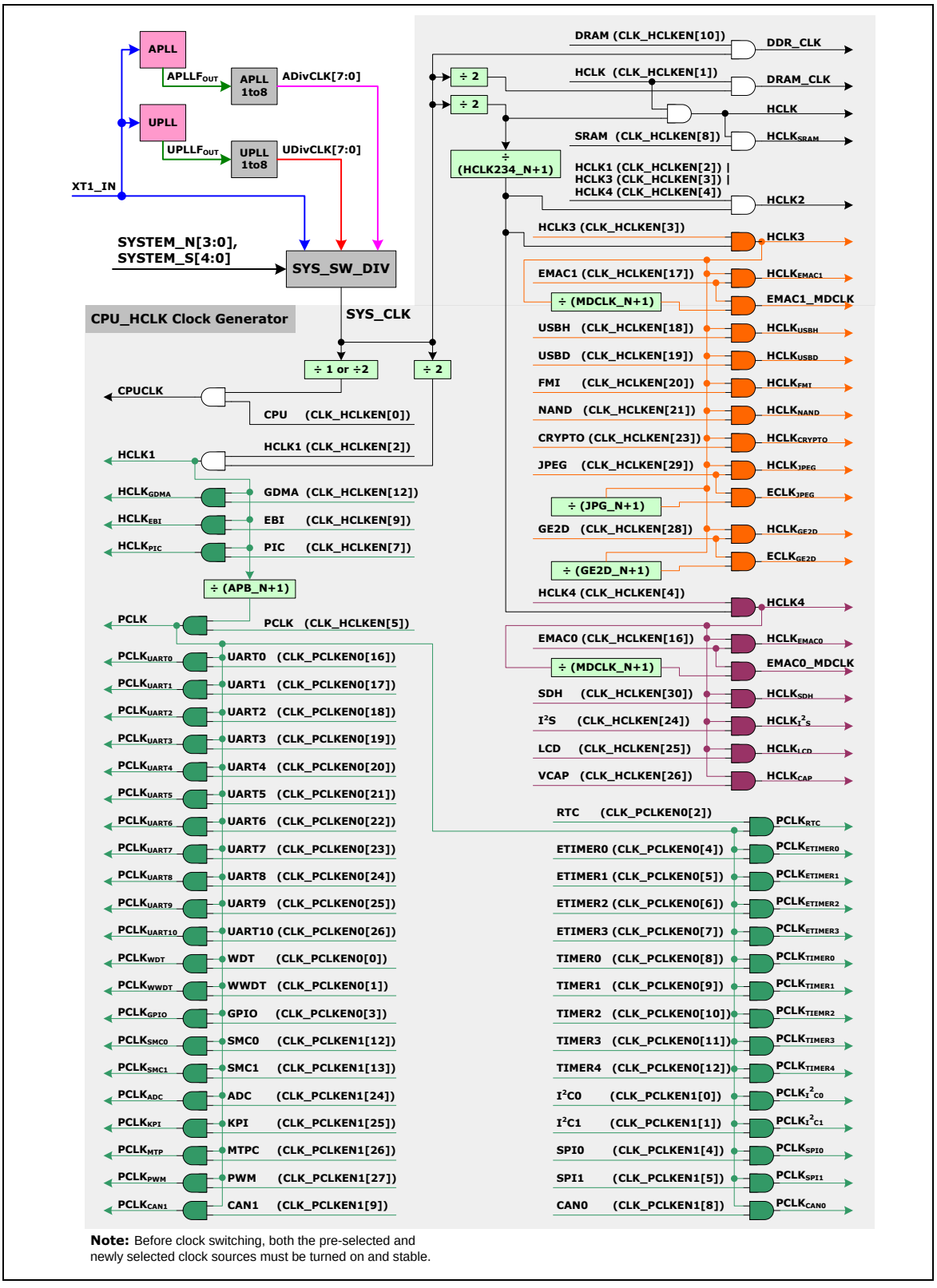



Figure 5.3-17 Windowed Watchdog Timer Clock Divider Block Diagram

### 5.3.3.18 *CPU_HCLK Clock Generator*




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

Figure 5.3-18 CPU_HCLK Clock Generator Block Diagram

### 5.3.4 Functional description

#### 5.3.4.1 *Power management*

This chip provides four power management scenarios, including Deep Power-down, Power-down, Idle and Normal Operating modes, to manage the power consumption. The peripheral clocks can be enabled / disabled individually by controlling the corresponding bit in CLKSEL control register. User can turn-off the unused modules' clock for power saving.

#### 5.3.4.2 *Normal Operating Mode*

In this mode, CPU run normally and clocks of all functionalities are on. The clock frequency of CPU, DRAM, AHB peripherals and APB peripherals are 300 MHz, 150 MHz, 150 MHz and 75 MHz, respectively.

#### 5.3.4.3 *Idle Mode*

When CPU is not busy, user can put ARM926EJ-S processor into a low-power state by the wait for interrupt instruction:

MCR p15, 0, <Rd>, c7, c0, 4

This instruction switches the ARM926EJ-S processor into a low-power state until either an interrupt (IRQ or FIQ) or a debug request occurs.

In this mode, the clocks of all functionalities are on. The clock frequency of DRAM, AHB peripherals and APB peripherals are 150 MHz, 150 MHz and 75 MHz.

#### 5.3.4.4 *Power-down Mode*

To reduce power consumption further, user could put the chip into Power-down mode by clearing XTAL_EN (CLK_PMCON[0]) to 0 before wait for interrupt instruction:

MCR p15, 0, <Rd>, c7, c0, 4

In this mode, all clocks (clocks for all functionalities, CPU and the HXT (Ext. Crystall Osc. 12 MHz)) stop, except LXT (Ext. Crystal Osc. 32.768 kHz), with SRAM retention.

The mechanisms shown below could wake chip up from Power-down mode:

- EINT, External Interrupt, pin toggled.
- UART 1/2/4/6/8/10 CTS toggled.
- EMAC 0 or EMAC 1 received a Magic Packet.
- USB device controller detected a VBUS valid event.
- USB host controller detected connect/dis-connect/remote-wakeup event.
- Touch screen touching or Keypad pressing detected by ADC controller.
- Key pressing or releasing detected by KPI, Keypad Interface.
- RTC alarm or relative alarm interrupt is generated.
- Enhanced Timer interrupt is active.
- WDT time-out wake-up.

#### 5.3.4.5 *Deep Power-down Mode*

To extremely reduce the power consumption, user could put the chip inot Deep Power-down mode without SRAM retention by turning off power supply to all power pin except RTC_VDD. In this mode, only RTC circuit and LXT (Ext. Crystal Osc. 32.768 kHz) keep active.

To keep DDR2 SDRAM content in this mode, it's necessary to keep supplying 1.8V to DDR_VDD.

### 5.3.5 Registers Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CLK Base Address:**<br>**CLK_BA = 0xB000_0200** | | | | |
| **CLK_PMCON** | CLK_BA+0x000 | R/W | Power Management Control Register | 0xFFFF_FF03 |
| **CLK_HCLKEN** | CLK_BA+0x010 | R/W | AHB Devices Clock Enable Control Register | 0x0000_0527 |
| **CLK_PCLKEN0** | CLK_BA+0x018 | R/W | APB Devices Clock Enable Control Register 0 | 0x0000_0000 |
| **CLK_PCLKEN1** | CLK_BA+0x01C | R/W | APB Devices Clock Enable Control Register 1 | 0x0000_0000 |
| **CLK_DIVCTL0** | CLK_BA+0x020 | R/W | Clock Divider Control Register 0 | 0x0100_00XX |
| **CLK_DIVCTL1** | CLK_BA+0x024 | R/W | Clock Divider Control Register 1 | 0x0000_0000 |
| **CLK_DIVCTL2** | CLK_BA+0x028 | R/W | Clock Divider Control Register 2 | 0x0000_0000 |
| **CLK_DIVCTL3** | CLK_BA+0x02C | R/W | Clock Divider Control Register 3 | 0x0000_0000 |
| **CLK_DIVCTL4** | CLK_BA+0x030 | R/W | Clock Divider Control Register 4 | 0x0000_0000 |
| **CLK_DIVCTL5** | CLK_BA+0x034 | R/W | Clock Divider Control Register 5 | 0x0000_0000 |
| **CLK_DIVCTL6** | CLK_BA+0x038 | R/W | Clock Divider Control Register 6 | 0x0000_0000 |
| **CLK_DIVCTL7** | CLK_BA+0x03C | R/W | Clock Divider Control Register 7 | 0x0000_0000 |
| **CLK_DIVCTL8** | CLK_BA+0x040 | R/W | Clock Divider Control Register 8 | 0x0000_0500 |
| **CLK_DIVCTL9** | CLK_BA+0x044 | R/W | Clock Divider Control Register 9 | 0x0000_0000 |
| **CLK_APLLCON** | CLK_BA+0x060 | R/W | APLL Control Register | 0x1000_0015 |
| **CLK_UPLLCON** | CLK_BA+0x064 | R/W | UPLL Control Register | 0xX000_0015 |
| **CLK_PLLSTBCNTR** | CLK_BA+0x080 | R/W | PLL Stable Counter and Test Clock Control Register | 0x0000_1800 |

### 5.3.6 Register description

### Power Management Control Register (CLK_PMCON)

The chip clock source is from an external crystal. The crystal oscillator can be control on/off by the register XTAL_EN. When turn off the crystal, the chip into power down state. To avoid outputting an unstable clock to system, clock controller implements a pre-scalar counter. After the clock counter count pre-scalar x 256 crystal cycle, the clock controller starts to output the clock to system.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CLK_PMCON** | CLK_BA+0x000 | R/W | Power Management Control Register | 0xFFFF_FF03 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:8] | **PRESCALE** | **Pre-scalar Counter**<br>Assume the crystal is stable after the Pre-Scalar x 256 crystal cycles. Clock controller wouldn't output clock to system before the counter reaching (pre-scalar x 256). |
| [7:5] | **Reserved** | Reserved. |
| [4] | **SEN_OFF_ST** | **Sensor Clock Level on Clock Off State**<br>0 = sensor clock keep on low level.<br>1 = sensor clock keep on high level. |
| [3:2] | **Reserved** | Reserved. |
| [1] | **XIN_CTL** | **Pre-scalar Counter Enable**<br>Crystal pre-divide control for Wake-up from power down mode The chip will delay 256 x pre-scalar cycles after the reset signal to wait the Crystal to stable<br>0 = Disable the pre-scalar, assume the crystal is stable.<br>1 = Enable the pre-scalar counter. |
| [0] | **XTAL_EN** | **Crystal (Power-down) Control**<br>0 = Crystal off (Power-down mode).<br>1 = Crystal on (Normal operating mode). |

## AHB Devices Clock Enable Control Register (CLK_HCLKEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CLK_HCLKEN | CLK_BA+0x010 | R/W | AHB Devices Clock Enable Control Register | 0x0000_0527 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | SDH | JPEG | GE2D | SENSOR | CAP | LCD | I2S |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CRYPTO | eMMC | NAND | FMI | USBD | USBH | EMAC1 | EMAC0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CKO | Reserved | | GDMA | Reserved | DDR | EBI | SRAM |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TIC | Reserved | PCLK | HCLK4 | HCLK3 | HCLK1 | HCLK | CPU |

| Bits | Description | |
|---|---|---|
| [31] | **Reserved** | Reserved. |
| [30] | **SDH** | **SD Card Controller Clock Enable**<br>0 = SD card controller clock disabled.<br>1 = SD card controller clock enabled. |
| [29] | **JPEG** | **JPEG Codec Clock Enable**<br>0 = JPEG codec clock disabled.<br>1 = JPEG codec clock enabled. |
| [28] | **GE2D** | **2D Graphic Engine Clock Enable**<br>0 = 2D graphic engine clock disabled.<br>1 = 2D graphic engine clock enabled. |
| [27] | **SNESOR** | **CMOS Sensor Reference Clock Output Enable**<br>0 = CMOS sensor reference clock output disabled.<br>1 = CMOS sensor reference clock output enabled. |
| [26] | **CAP** | **CMOS Sensor Interface Controller Clock Enable**<br>0 = CMOS sensor interface controller clock disabled.<br>1 = CMOS sensor interface controller clock enabled. |
| [25] | **LCD** | **LCD Display Controller Clock Enable**<br>0 = LCD display controller clock disabled.<br>1 = LCD display controller clock enabled. |
| [24] | **I2S** | **Audio Controller Clock Enable**<br>0 = Audio controller clock disabled.<br>1 = Audio controller clock enabled. |

| | | |
|---|---|---|
| [23] | **CRYPTO** | **Crypto Engine Clock Enable**<br>0 = Crypto engine clock disabled.<br>1 = Crypto engine clock enabled. |
| [22] | **eMMC** | **eMMC Engine Clock Enable**<br>0 = eMMC controller clock disabled.<br>1 = eMMC controller clock enabled. |
| [21] | **NAND** | **NAND Engine Clock Enable**<br>0 = NAND controller clock disabled.<br>1 = NAND controller clock enabled. |
| [20] | **FMI** | **FMI Controller Clock Enable**<br>0 = FMI controller clock disabled.<br>1 = FMI controller clock enabled. |
| [19] | **USBD** | **USB Device Controller Clock Enable**<br>0 = USB device controller clock disabled.<br>1 = USB device controller clock enabled. |
| [18] | **USBH** | **USB Host Controller Clock Enable**<br>0 = USB host controller clock disabled.<br>1 = USB host controller clock enabled. |
| [17] | **EMAC1** | **Ethernet MAC Controller 1 Clock Enable**<br>0 = Ethernet MAC controller 1 clock disabled.<br>1 = Ethernet MAC controller 1 clock enabled. |
| [16] | **EMAC0** | **Ethernet MAC Controller 0 Clock Enable**<br>0 = Ethernet MAC controller 0 clock disabled.<br>1 = Ethernet MAC controller 0 clock enabled. |
| [15] | **CKO** | **Reference Clock Output Enable**<br>0 = Reference clock output disabled.<br>1 = Reference clock output enabled. |
| [14:13] | **Reserved** | Reserved. |
| [12] | **GDMA** | **GDMA Clock Enable**<br>0 = GDMA clock disabled.<br>1 = GDMA clock enabled. |
| [11] | **Reserved** | Reserved. |
| [10] | **DDR** | **DDR Clock Enable**<br>0 = DDR clock disabled.<br>1 = DDR clock enabled. |
| [9] | **EBI** | **EBI Controller Clock Enable**<br>0 = EBI controller clock disabled.<br>1 = EBI controller clock enabled. |
| [8] | **SRAM** | **SRAM Controller Clock Enable**<br>0 = SRAM controller clock disabled.<br>1 = SRAM controller clock enabled. |

| | | |
|---|---|---|
| [7] | **TIC** | **TIC Clock Enable**<br>0 = TIC clock disabled.<br>1 = TIC clock enabled. |
| [6] | **Reserved** | Reserved. |
| [5] | **PCLK** | **Internal APB Bus Clock Enable**<br>0 = Internal APB bus clock disabled.<br>1 = Internal APB bus clock enabled. |
| [4] | **HCLK4** | **Internal AHB-4 Bus Clock Enable**<br>0 = Internal AHB-4 bus clock disabled.<br>1 = Internal AHB-4 bus clock enabled. |
| [3] | **HCLK3** | **Internal AHB-3 Bus Clock Enable**<br>0 = Internal AHB-3 bus clock disabled.<br>1 = Internal AHB-3 bus clock enabled. |
| [2] | **HCLK1** | **Internal AHB-1 Bus Clock Enable**<br>0 = Internal AHB-1 bus clock disabled.<br>1 = Internal AHB-1 bus clock enabled. |
| [1] | **HCLK** | **Internal AHB Bus Clock Enable**<br>0 = Internal AHB bus clock disabled.<br>1 = Internal AHB bus clock enabled. |
| [0] | **CPU** | **ARM926EJ-s CPU Clock Enable**<br>0 = ARM926EJ-S CPU clock disabled.<br>1 = ARM926EJ-S CPU clock enabled. |

## APB Devices Clock Enable Control Register 0 (CLK_PCLKEN0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CLK_PCLKEN0 | CLK_BA+0x018 | R/W | APB Devices Clock Enable Control Register 0 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | UART10 | UART9 | UART8 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| UART7 | UART6 | UART5 | UART4 | UART3 | UART2 | UART1 | UART0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | TIMER4 | TIMER3 | TIMER2 | TIMER1 | TIMER0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ETIMER3 | ETIMER2 | ETIMER1 | ETIMER0 | GPIO | RTC | WWDT | WDT |

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26] | **UART10** | **UART 10 Clock Enable**<br>0 = UART 10 clock disabled.<br>1 = UART 10 clock enabled. |
| [25] | **UART9** | **UART 9 Clock Enable**<br>0 = UART 9 clock disabled.<br>1 = UART 9 clock enabled. |
| [24] | **UART8** | **UART 8 Clock Enable**<br>0 = UART 8 clock disabled.<br>1 = UART 8 clock enabled. |
| [23] | **UART7** | **UART 7 Clock Enable**<br>0 = UART 7 clock disabled.<br>1 = UART 7 clock enabled. |
| [22] | **UART6** | **UART 6 Clock Enable**<br>0 = UART 6 clock disabled.<br>1 = UART 6 clock enabled. |
| [21] | **UART5** | **UART 5 Clock Enable**<br>0 = UART 5 clock disabled.<br>1 = UART 5 clock enabled. |
| [20] | **UART4** | **UART 4 Clock Enable**<br>0 = UART 4 clock disabled.<br>1 = UART 4 clock enabled. |

| | | |
|---|---|---|
| [19] | **UART3** | **UART 3 Clock Enable**<br>0 = UART 3 clock disabled.<br>1 = UART 3 clock enabled. |
| [18] | **UART2** | **UART 2 Clock Enable**<br>0 = UART 2 clock disabled.<br>1 = UART 2 clock enabled. |
| [17] | **UART1** | **UART 1 Clock Enable**<br>0 = UART 1 clock disabled.<br>1 = UART 1 clock enabled. |
| [16] | **UART0** | **UART 0 Clock Enable**<br>0 = UART 0 clock disabled.<br>1 = UART 0 clock enabled. |
| [15:13] | **Reserved** | Reserved. |
| [12] | **TIMER4** | **Timer 4 Clock Enable**<br>0 = Timer 4 clock disabled.<br>1 = Timer 4 clock enabled. |
| [11] | **TIMER3** | **Timer 3 Clock Enable**<br>0 = Timer 3 clock disabled.<br>1 = Timer 3 clock enabled. |
| [10] | **TIMER2** | **Timer 2 Clock Enable**<br>0 = Timer 2 clock disabled.<br>1 = Timer 2 clock enabled. |
| [9] | **TIMER1** | **Timer 1 Clock Enable**<br>0 = Timer 1 clock disabled.<br>1 = Timer 1 clock enabled. |
| [8] | **TIMER0** | **Timer 0 Clock Enable**<br>0 = Timer 0 clock disabled.<br>1 = Timer 0 clock enabled. |
| [7] | **ETIMER3** | **Enhanced Timer 3 Clock Enable**<br>0 = Enhanced Timer 3 clock disabled.<br>1 = Enhanced Timer 3 clock enabled. |
| [6] | **ETIMER2** | **Enhanced Timer 2 Clock Enable**<br>0 = Enhanced Timer 2 clock disabled.<br>1 = Enhanced Timer 2 clock enabled. |
| [5] | **ETIMER1** | **Enhanced Timer 1 Clock Enable**<br>0 = Enhanced Timer 1 clock disabled.<br>1 = Enhanced Timer 1 clock enabled. |
| [4] | **ETIMER0** | **Enhanced Timer 0 Clock Enable**<br>0 = Enhanced Timer 0 clock disabled.<br>1 = Enhanced Timer 0 clock enabled. |

| | | |
|---|---|---|
| [3] | **GPIO** | **GPIO Controller Clock Enable**<br>0 = GPIO controller clock disabled.<br>1 = GPIO controller clock enabled. |
| [2] | **RTC** | **RTC Clock Enable**<br>0 = RTC clock disabled.<br>1 =RTC clock enabled. |
| [1] | **WWDT** | **Windowed Watch-dog Clock Enable**<br>0 = Windowed Watch-dog clock disabled.<br>1 = Windowed Watch-dog clock enabled. |
| [0] | **WDT** | **Watch-dog Clock Enable**<br>0 = Watch-dog clock disabled.<br>1 = Watch-dog clock enabled. |

## APB Devices Clock Enable Control Register 1 (CLK_PCLKEN1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CLK_PCLKEN1 | CLK_BA+0x01C | R/W | APB Devices Clock Enable Control Register 1 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | PWM | MTPC | KPI | ADC |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | SMC1 | SMC0 | Reserved | | CAN1 | CAN0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | SPI1 | SPI0 | Reserved | | I2C1 | I2C0 |

| Bits | Description | |
|---|---|---|
| [31:26] | **Reserved** | Reserved. |
| [27] | **PWM** | **PWM Clock Enable**<br>0 = PWM clock disabled.<br>1 = PWM clock enabled. |
| [26] | **MTPC** | **MTP Controller Clock Enable**<br>0 = MTP controller clock disabled.<br>1 = MTP controller clock enabled. |
| [25] | **KPI** | **Keypad Controller Clock Enable**<br>0 = Keypad controller clock disabled.<br>1 = Keypad controller clock enabled. |
| [24] | **ADC** | **ADC Controller Clock Enable**<br>0 = ADC controller clock disabled.<br>1 = ADC controller clock enabled. |
| [23:14] | **Reserved** | Reserved. |
| [13] | **SMC1** | **Smart Card Interface 1 Clock Enable**<br>0 = Smart Card interface 1 clock disabled.<br>1 = Smart Card interface 1 clock enabled. |
| [12] | **SMC0** | **Smart Card Interface 0 Clock Enable**<br>0 = Smart Card interface 0 clock disabled.<br>1 = Smart Card interface 0 clock enabled. |
| [11:10] | **Reserved** | Reserved. |
| [9] | **CAN1** | **CAN 1 Clock Enable**<br>0 = CAN 1 clock disabled.<br>1 = CAN 1 clock enabled. |

| | | |
|---|---|---|
| [8] | **CAN0** | **CAN 0 Clock Enable**<br>0 = CAN 0 clock disabled.<br>1 = CAN 0 clock enabled. |
| [7:6] | **Reserved** | Reserved. |
| [5] | **SPI1** | **SPI Interface 1 Clock Enable**<br>0 = SPI Interface 1 clock disabled.<br>1 = SPI Interface 1 clock enabled. |
| [4] | **SPI0** | **SPI Interface 0 Clock Enable**<br>0 = SPI Interface 0 clock disabled.<br>1 = SPI Interface 0 clock enabled. |
| [3:2] | **Reserved** | Reserved. |
| [1] | **I2C1** | **$I^2C$ Interface 1 Clock Enable**<br>0 = $I^2C$ Interface 1 clock disabled.<br>1 = $I^2C$ Interface 1 clock enabled. |
| [0] | **I2C0** | **$I^2C$ Interface 0 Clock Enable**<br>0 = $I^2C$ Interface 0 clock disabled.<br>1 = $I^2C$ Interface 0 clock enabled. |

## Clock Divider Control Register 0 (CLK_DIVCTL0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CLK_DIVCTL0 | CLK_BA+0x020 | R/W | Clock Divider Control Register 0 | 0x0100_00XX |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | PCLK_N | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HCLK234_N | | | | CPU_N | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | SYSTEM_N | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | SYSTEM_S | SYSTEM_SDIV | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27:24] | **PCLK_N** | **APB Clock Divider**<br>This field defines the clock divide number for clock divider to generate the PCLK for APB bus and controllers in APB bus.<br>The actual clock divide number is (PCLK_N + 1). So,<br>PCLK = HCLK1 / (PCLK_N + 1). |
| [23:20] | **HCLK234_N** | **AHB234 Clock Divider**<br>This field defines the clock divide number for clock divider to generate the HCLK for AHB2, AHB3, AHB4 bus and controllers in AHB2, AHB3 and AHB4 bus.<br>The actual clock divide number is (HCLK234_N + 1). So,<br>HCLK2 = HCLK / (HCLK234_N + 1).<br>HCLK3 = HCLK / (HCLK234_N + 1).<br>HCLK4 = HCLK / (HCLK234_N + 1). |
| [19:16] | **CPU_N** | **CPU Clock Divider**<br>This field defines the clock divide number for clock divider to generate the CPUCLK for ARM926EJ-S CPU.<br>The actual clock divide number is (CPU_N + 1). So,<br>CPUCLK = SYS_CLK / (CPU_N + 1).<br>**Note:** The CPU_N only could be set as 0x0, 0x1. Other values are prohibited. |
| [15:12] | **Reserved** | Reserved. |
| [11:8] | **SYSTEM_N** | **System Clock Divider**<br>This field defines the clock divide number for clock divider to generate the system clock SYS_CLK.<br>The actual clock divide number is (SYSTEM_N + 1). So,<br>SYS_CLK = SYSTEM_SrcCLK / (SYSTEM_N + 1). |
| [7:5] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [4:3] | **SYSTEM_S** | **System Clock Source Selection**<br>This field selects which clock is used to be the source of system clock SYS_CLK.<br>00 = SYSTEM_SrcCLK is from XIN.<br>01 = Reserved.<br>10 = SYSTEM_SrcCLK is from ACLKOut.<br>11 = SYSTEM_SrcCLK is from UCLKOut. |
| [2:0] | **SYSTEM_SDIV** | **System Source Clock Divider**<br>This field defines the source clock divide number for clock divider of APLL and UPLL output. This field only takes effect while the SYSTEM_S (CLK_DIVCTL0[4:3]) is 2'b10 (APLL) or 2'b11 (UPLL).<br>If SYSTEM_S (CLK_DIVCTL0[4:3]) is 2'b10,<br>ACLKOut = APLLFout ÷ (SYSTEM_SDIV + 1).<br>If SYSTEM_S (CLK_DIVCTL0[4:3]) is 2'b11,<br>UCLKOut = UPLLFout ÷ (SYSTEM_SDIV + 1). |

## Clock Divider Control Register 1 (CLK_DIVCTL1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CLK_DIVCTL1 | CLK_BA+0x024 | R/W | Clock Divider Control Register 1 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| I2S_N | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | I2S_S | | I2S_SDIV | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LCD_N | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | LCD_S | | LCD_SDIV | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **I2S_N** | **$I^2S$ Controller Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for $I^2S$ controller.<br>The actual clock divide number is (I2S_N + 1). So,<br>ECLKi2s = I2S_SrcCLK / (I2S_N + 1). |
| [23:21] | **Reserved** | Reserved. |
| [20:19] | **I2S_S** | **$I^2S$ Controller Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for $I^2S$ controller.<br>00 = I2S_SrcCLK is from XIN.<br>01 = Reserved.<br>10 = I2S_SrcCLK is from ACLKOut.<br>11 = I2S_SrcCLK is from UCLKOut. |
| [18:16] | **I2S_SDIV** | **$I^2S$ Controller Source Clock Divider**<br>This field defines the source clock divide number for clock divider of APLL and UPLL output. This field only takes effect while the I2S_S (CLK_DIVCTL1[20:19]) is 2'b10 (APLL) or 2'b11 (UPLL).<br>If I2S_S (CLK_DIVCTL1[20:19]) is 2'b10,<br>ACLKOut = APLLFout ÷ (I2S_SDIV + 1).<br>If I2S_S (CLK_DIVCTL1[20:19]) is 2'b11,<br>UCLKOut = UPLLFout ÷ (I2S_SDIV + 1). |
| [15:8] | **LCD_N** | **LCD Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for LCD display controller.<br>The actual clock divide number is (LCD_N + 1). So,<br>ECLKlcd = LCD_SrcCLK / (LCD_N + 1). |
| [7:5] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [4:3] | **LCD_S** | **LCD Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for LCD display controller.<br>00 = LCD_SrcCLK is from XIN.<br>01 = Reserved.<br>10 = LCD_SrcCLK is from ACLKOut.<br>11 = LCD_SrcCLK is from UCLKOut. |
| [2:0] | **LCD_SDIV** | **LCD Engine Source Clock Divider**<br>This field defines the source clock divide number for clock divider of APLL and UPLL output. This field only takes effect while the LCD_S (CLK_DIVCTL1[4:3]) is 2'b10 (APLL) or 2'b11 (UPLL).<br>If LCD_S (CLK_DIVCTL1[4:3]) is 2'b10,<br>ACLKOut = APLLFout ÷ (LCD_SDIV + 1).<br>If LCD_S (CLK_DIVCTL1[4:3]) is 2'b11,<br>UCLKOut = UPLLFout ÷ (LCD_SDIV + 1). |

## Clock Divider Control Register 2 (CLK_DIVCTL2)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CLK_DIVCTL2 | CLK_BA+0x028 | R/W | Clock Divider Control Register 2 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | GE2D_N | | Reserved | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | USB_N | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | USB_S | | Reserved | | |

| Bits | Description | |
|---|---|---|
| [31:30] | **Reserved** | Reserved. |
| [29:28] | **GE2D_N** | **GE2D Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for GE2D codec.<br>The actual clock divide number is (GE2D_N + 1). So,<br>ECLKge2d = HCLK3 / (GE2D_N + 1). |
| [27:12] | **Reserved** | Reserved. |
| [11:8] | **USB_N** | **USB 1.1 Host Controller Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the 48MHz clock for USB 1.1 host controller.<br>The actual clock divide number is (USB_N + 1). So,<br>USB_CLK = USB11_SrcCLK / (USB_N + 1).<br>**Note:** The USB_CLK must be 48MHz. |
| [7:5] | **Reserved** | Reserved. |
| [4:3] | **USB_S** | **USB 1.1 Engine Clock Source Selection**<br>This field selects which clock is used to be the source of 48MHz clock for USB 1.1 host controller.<br>00 = Reserved.<br>01 = Reserved.<br>10 = USB11_SrcCLK is from 480 MHz outputted by USB PHY 0.<br>11 = USB11_SrcCLK is from 480 MHz outputted by USB PHY 1. |
| [2:0] | **Reserved** | Reserved. |

## Clock Divider Control Register 3 (CLK_DIVCTL3)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CLK_DIVCTL3 | CLK_BA+0x02C | R/W | Clock Divider Control Register 3 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | JPG_N | | | SENSOR_N | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | SENSOR_S | | SENSOR_SDIV | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| eMMC_N | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | eMMC_S | | eMMC_SDIV | | |

| Bits | Description | |
|---|---|---|
| [30:28] | **JPG_N** | **JPEG Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for JPEG codec.<br>The actual clock divide number is (JPG_N + 1). So,<br>ECLKjpg = HCLK3 / (JPG_N + 1). |
| [27:24] | **SENSOR_N** | **Sensor Clock Divider**<br>This field defines the clock divide number for clock divider to generate the sensor clock.<br>The actual clock divide number is (SENSOR_N + 1). So,<br>SEN_CLK = SEN_SrcCLK / (SENSOR_N + 1). |
| [20:19] | **SENSOR_S** | **Sensor Clock Source Selection**<br>This field selects which clock is used to be the source of sensor clock.<br>00 = SEN_SrcCLK is from XIN.<br>01 = Reserved.<br>10 = SEN_SrcCLK is from ACLKOut.<br>11 = SEN_SrcCLK is from UCLKOut. |
| [18:16] | **SENSOR_SDIV** | **Sensor Source Clock Divider**<br>This field defines the source clock divide number for clock divider of APLL and UPLL output. This field only takes effect while the SENSOR_S (CLK_DIVCTL3[20:19]) is 2'b10 (APLL) or 2'b11 (UPLL).<br>If SENSOR_S (CLK_DIVCTL3[20:19]) is 2'b10,<br>ACLKOut = APLLFout ÷ (SENSOR_SDIV + 1).<br>If SENSOR_S (CLK_DIVCTL3[20:19]) is 2'b11,<br>UCLKOut = UPLLFout ÷ (SENSOR_SDIV + 1). |

| | | |
|---|---|---|
| [15:8] | **eMMC_N** | **eMMC Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for eMMC controller.<br>The actual clock divide number is (eMMC_N + 1). So,<br>eMMC_CLK = eMMC_SrcCLK / (eMMC_N + 1). |
| [7:5] | **Reserved** | Reserved. |
| [4:3] | **eMMC_S** | **eMMC Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for eMMC controller.<br>00: eMMC_SrcCLK = XIN.<br>01: eMMC_SrcCLK = Reserved.<br>10: eMMC_SrcCLK = ACLKOut.<br>11: eMMC_SrcCLK = UCLKOut. |
| [2:0] | **eMMC_SDIV** | **eMMC Engine Source Clock Divider**<br>This field defines the source clock divide number for clock divider of APLL and UPLL output. This field only takes effect while the eMMC_S (CLK_DIVCTL3[4:3]) is 2'b10 (APLL) or 2'b11 (UPLL).<br>If eMMC_S (CLK_DIVCTL3[4:3]) is 2'b10,<br>ACLKOut = APLLFout ÷ (eMMC_SDIV + 1).<br>If eMMC_S (CLK_DIVCTL3[4:3]) is 2'b11,<br>UCLKOut = UPLLFout ÷ (eMMC_SDIV + 1). |

## Clock Divider Control Register 4 (CLK_DIVCTL4)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CLK_DIVCTL4 | CLK_BA+0x030 | R/W | Clock Divider Control Register 4 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| UART3_N | | | UART3_S | | UART3_SDIV | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| UART2_N | | | UART2_S | | UART2_SDIV | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| UART1_N | | | UART1_S | | UART1_SDIV | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| UART0_N | | | UART0_S | | UART0_SDIV | | |

| Bits | Description | |
|---|---|---|
| [31:29] | UART3_N | **UART3 Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for UART3.<br>The actual clock divide number is (UART3_N + 1). So,<br>ECLKuart3 = UART3_SrcCLK / (UART3_N + 1) |
| [28:27] | UART3_S | **UART3 Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for UART3 controller.<br>00 = UART3_SrcCLK is from XIN<br>01 = Reserved<br>10 = UART3_SrcCLK is from ACLKOut<br>11 = UART3_SrcCLK is from UCLKOut |
| [26:24] | UART3_SDIV | **UART3 Engine Source Clock Divider**<br>This field defines the source clock divide number for clock divider of APLL and UPLL output. This field only takes effect while the UART3_S (CLK_DIVCTL4[28:27]) is 2'b10 (APLL) or 2'b11 (UPLL).<br>If UART3_S (CLK_DIVCTL4[28:27]) is 2'b10,<br>ACLKOut = APLLFout ÷ (UART3_SDIV + 1)<br>If UART3_S (CLK_DIVCTL4[28:27]) is 2'b11,<br>UCLKOut = UPLLFout ÷ (UART3_SDIV + 1) |
| [23:21] | UART2_N | **UART2 Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for UART2.<br>The actual clock divide number is (UART2_N + 1). So,<br>ECLKuart2 = UART2_SrcCLK / (UART2_N + 1) |

| | | |
|---|---|---|
| [20:19] | UART2_S | **UART2 Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for UART2 controller.<br>00 = UART2_SrcCLK is from XIN<br>01 = Reserved<br>10 = UART2_SrcCLK is from ACLKOut<br>11 = UART2_SrcCLK is from UCLKOut |
| [18:16] | UART2_SDIV | **UART2 Engine Source Clock Divider**<br>This field defines the source clock divide number for clock divider of APLL and UPLL output. This field only takes effect while the UART2_S (CLK_DIVCTL4[20:19]) is 2'b10 (APLL) or 2'b11 (UPLL).<br>If UART2_S (CLK_DIVCTL4[20:19]) is 2'b10,<br>ACLKOut = APLLFout ÷ (UART2_SDIV + 1)<br>If UART2_S (CLK_DIVCTL4[20:19]) is 2'b11,<br>UCLKOut = UPLLFout ÷ (UART2_SDIV + 1) |
| [15:13] | UART1_N | **UART1 Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for UART1.<br>The actual clock divide number is (UART1_N + 1). So,<br>ECLKuart1 = UART1_SrcCLK / (UART1_N + 1) |
| [12:11] | UART1_S | **UART1 Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for UART1 controller.<br>00 = UART1_SrcCLK is from XIN<br>01 = Reserved<br>10 = UART1_SrcCLK is from ACLKOut<br>11 = UART1_SrcCLK is from UCLKOut |
| [10:8] | UART1_SDIV | **UART1 Engine Source Clock Divider**<br>This field defines the source clock divide number for clock divider of APLL and UPLL output. This field only takes effect while the UART1_S (CLK_DIVCTL4[12:11]) is 2'b10 (APLL) or 2'b11 (UPLL).<br>If UART1_S (CLK_DIVCTL4[12:11]) is 2'b10,<br>ACLKOut = APLLFout ÷ (UART1_SDIV + 1)<br>If UART1_S (CLK_DIVCTL4[12:11]) is 2'b11,<br>UCLKOut = UPLLFout ÷ (UART1_SDIV + 1) |
| [7:5] | UART0_N | **UART0 Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for UART0.<br>The actual clock divide number is (UART0_N + 1). So,<br>ECLKuart0 = UART0_SrcCLK / (UART0_N + 1) |
| [4:3] | UART0_S | **UART0 Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for UART0 controller.<br>00 = UART0_SrcCLK is from XIN<br>01 = Reserved<br>10 = UART0_SrcCLK is from ACLKOut<br>11 = UART0_SrcCLK is from UCLKOut |

| | | |
|---|---|---|
| [2:0] | | **UART0 Engine Source Clock Divider**<br>This field defines the source clock divide number for clock divider of APLL and UPLL output. This field only takes effect while the UART0_S (CLK_DIVCTL4[4:3]) is 2'b10 (APLL) or 2'b11 (UPLL).<br>If UART0_S (CLK_DIVCTL4[5:4]) is 2'b10,<br>ACLKOut = APLLFout ÷ (UART0_SDIV + 1)<br>If UART0_S (CLK_DIVCTL4[5:4]) is 2'b11,<br>UCLKOut = UPLLFout ÷ (UART0_SDIV + 1) |

## Clock Divider Control Register 5 (CLK_DIVCTL5)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CLK_DIVCTL5 | CLK_BA+0x034 | R/W | Clock Divider Control Register 5 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| UART7_N | | | UART7_S | | UART7_SDIV | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| UART6_N | | | UART6_S | | UART6_SDIV | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| UART5_N | | | UART5_S | | UART5_SDIV | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| UART4_N | | | UART4_S | | UART4_SDIV | | |

| Bits | Description | |
|---|---|---|
| [31:29] | **UART7_N** | **UART7 Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for UART7.<br>The actual clock divide number is (UART7_N + 1). So,<br>ECLKuart7 = UART7_SrcCLK / (UART7_N + 1). |
| [28:27] | **UART7_S** | **UART7 Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for UART7 controller.<br>00 = UART7_SrcCLK is from XIN.<br>01 = Reserved.<br>10 = UART7_SrcCLK is from ACLKOut.<br>11 = UART7_SrcCLK is from UCLKOut. |
| [26:24] | **UART7_SDIV** | **UART7 Engine Source Clock Divider**<br>This field defines the source clock divide number for clock divider of APLL and UPLL output. This field only takes effect while the UART7_S (CLK_DIVCTL5[28:27]) is 2'b10 (APLL) or 2'b11 (UPLL).<br>If UART7_S (CLK_DIVCTL5[28:27]) is 2'b10,<br>ACLKOut = APLLFout ÷ (UART7_SDIV + 1).<br>If UART7_S (CLK_DIVCTL5[28:27]) is 2'b11,<br>UCLKOut = UPLLFout ÷ (UART7_SDIV + 1). |
| [23:21] | **UART6_N** | **UART6 Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for UART6.<br>The actual clock divide number is (UART6_N + 1). So,<br>ECLKuart6 = UART6_SrcCLK / (UART6_N + 1). |

| | | |
|---|---|---|
| [20:19] | **UART6_S** | **UART6 Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for UART6 controller.<br>00 = UART6_SrcCLK is from XIN.<br>01 = Reserved.<br>10 = UART6_SrcCLK is from ACLKOut.<br>11 = UART6_SrcCLK is from UCLKOut. |
| [18:16] | **UART6_SDIV** | **UART6 Engine Source Clock Divider**<br>This field defines the source clock divide number for clock divider of APLL and UPLL output. This field only takes effect while the UART6_S (CLK_DIVCTL5[20:19]) is 2'b10 (APLL) or 2'b11 (UPLL).<br>If UART6_S (CLK_DIVCTL5[20:19]) is 2'b10,<br>ACLKOut = APLLFout ÷ (UART6_SDIV + 1).<br>If UART6_S (CLK_DIVCTL5[20:19]) is 2'b11,<br>UCLKOut = UPLLFout ÷ (UART6_SDIV + 1). |
| [15:13] | **UART5_N** | **UART5 Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for UART5.<br>The actual clock divide number is (UART5_N + 1). So,<br>ECLKuart5 = UART5_SrcCLK / (UART5_N + 1). |
| [12:11] | **UART5_S** | **UART5 Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for UART5 controller.<br>00 = UART5_SrcCLK is from XIN.<br>01 = Reserved.<br>10 = UART5_SrcCLK is from ACLKOut.<br>11 = UART5_SrcCLK is from UCLKOut. |
| [10:8] | **UART5_SDIV** | **UART5 Engine Source Clock Divider**<br>This field defines the source clock divide number for clock divider of APLL and UPLL output. This field only takes effect while the UART5_S (CLK_DIVCTL5[12:11]) is 2'b10 (APLL) or 2'b11 (UPLL).<br>If UART5_S (CLK_DIVCTL5[12:11]) is 2'b10,<br>ACLKOut = APLLFout ÷ (UART5_SDIV + 1).<br>If UART5_S (CLK_DIVCTL5[12:11]) is 2'b11,<br>UCLKOut = UPLLFout ÷ (UART5_SDIV + 1). |
| [7:5] | **UART4_N** | **UART4 Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for UART4.<br>The actual clock divide number is (UART4_N + 1). So,<br>ECLKuart4 = UART4_SrcCLK / (UART4_N + 1). |
| [4:3] | **UART4_S** | **UART4 Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for UART4 controller.<br>00 = UART4_SrcCLK is from XIN.<br>01 = Reserved.<br>10 = UART4_SrcCLK is from ACLKOut.<br>11 = UART4_SrcCLK is from UCLKOut. |

| | | |
|---|---|---|
| [2:0] | **UART4_SDIV** | **UART4 Engine Source Clock Divider**<br>This field defines the source clock divide number for clock divider of APLL and UPLL output. This field only takes effect while the UART4_S (CLK_DIVCTL5[4:3]) is 2'b10 (APLL) or 2'b11 (UPLL).<br>If UART4_S (CLK_DIVCTL5[4:3]) is 2'b10,<br>ACLKOut = APLLFout ÷ (UART4_SDIV + 1).<br>If UART4_S (CLK_DIVCTL5[4:3]) is 2'b11,<br>UCLKOut = UPLLFout ÷ (UART4_SDIV + 1). |

## Clock Divider Control Register 6 (CLK_DIVCTL6)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CLK_DIVCTL6 | CLK_BA+0x038 | R/W | Clock Divider Control Register 6 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| SMC1_N | | | | SMC0_N | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| UART10_N | | | UART10_S | | UART10_SDIV | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| UART9_N | | | UART9_S | | UART9_SDIV | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| UART8_N | | | UART8_S | | UART8_SDIV | | |

| Bits | Description | |
|---|---|---|
| [30:28] | SMC1_N | **Smart Card 1 Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for Smart card controller.<br>The actual clock divide number is (SMC1_N + 1). So,<br>ECLKsmc1 = XIN12M / (SMC1_N + 1). |
| [27:24] | SMC0_N | **Smart Card 0 Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for Smart card controller.<br>The actual clock divide number is (SMC0_N + 1). So,<br>ECLKsmc0 = XIN12M / (SMC0_N + 1). |
| [23:21] | UART10_N | **UART10 Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for UART10.<br>The actual clock divide number is (UART10_N + 1). So,<br>ECLKuart10 = UART10_SrcCLK / (UART10_N + 1). |
| [20:19] | UART10_S | **UART10 Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for UART10 controller.<br>00 = UART10_SrcCLK is from XIN.<br>01 = Reserved.<br>10 = UART10_SrcCLK is from ACLKOut.<br>11 = UART10_SrcCLK is from UCLKOut. |

| | | |
|---|---|---|
| [18:16] | **UART10_SDIV** | **UART10 Engine Source Clock Divider**<br>This field defines the source clock divide number for clock divider of APLL and UPLL output. This field only takes effect while the UART10_S (CLK_DIVCTL6[20:19]) is 2'b10 (APLL) or 2'b11 (UPLL).<br>If UART10_S (CLK_DIVCTL6[20:19]) is 2'b10,<br>ACLKOut = APLLFout ÷ (UART10_SDIV + 1).<br>If UART10_S (CLK_DIVCTL6[20:19]) is 2'b11,<br>UCLKOut = UPLLFout ÷ (UART10_SDIV + 1). |
| [15:13] | **UART9_N** | **UART9 Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for UART9.<br>The actual clock divide number is (UART9_N + 1). So,<br>ECLKuart9 = UART9_SrcCLK / (UART9_N + 1). |
| [12:11] | **UART9_S** | **UART9 Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for UART9 controller.<br>00 = UART9_SrcCLK is from XIN.<br>01 = Reserved.<br>10 = UART9_SrcCLK is from ACLKOut.<br>11 = UART9_SrcCLK is from UCLKOut. |
| [10:8] | **UART9_SDIV** | **UART9 Engine Source Clock Divider**<br>This field defines the source clock divide number for clock divider of APLL and UPLL output. This field only takes effect while the UART9_S (CLK_DIVCTL6[12:11]) is 2'b10 (APLL) or 2'b11 (UPLL).<br>If UART9_S (CLK_DIVCTL6[12:11]) is 2'b10,<br>ACLKOut = APLLFout ÷ (UART9_SDIV + 1).<br>If UART9_S (CLK_DIVCTL6[12:11]) is 2'b11,<br>UCLKOut = UPLLFout ÷ (UART9_SDIV + 1). |
| [7:5] | **UART8_N** | **UART8 Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for UART8.<br>The actual clock divide number is (UART8_N + 1). So,<br>ECLKuart8 = UART8_SrcCLK / (UART8_N + 1). |
| [4:3] | **UART8_S** | **UART8 Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for UART8 controller.<br>00 = UART8_SrcCLK is from XIN.<br>01 = Reserved.<br>10 = UART8_SrcCLK is from ACLKOut.<br>11 = UART8_SrcCLK is from UCLKOut. |
| [2:0] | **UART8_SDIV** | **UART8 Engine Source Clock Divider**<br>This field defines the source clock divide number for clock divider of APLL and UPLL output. This field only takes effect while the UART8_S (CLK_DIVCTL6[5:4]) is 2'b10 (APLL) or 2'b11 (UPLL).<br>If UART8_S (CLK_DIVCTL6[5:4]) is 2'b10,<br>ACLKOut = APLLFout ÷ (UART8_SDIV + 1).<br>If UART8_S (CLK_DIVCTL6[5:4]) is 2'b11,<br>UCLKOut = UPLLFout ÷ (UART8_SDIV + 1). |

## Clock Divider Control Register 7 (CLK_DIVCTL7)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CLK_DIVCTL7 | CLK_BA+0x03C | R/W | Clock Divider Control Register 7 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| ADC_N | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | ADC_S | | ADC_SDIV | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| KPI_S | KPI_N | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO_S | GPIO_N | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | ADC_N | **ADC Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for ADC.<br>The actual clock divide number is (ADC_N + 1). So,<br>ADC_CLK = ADC_SrcCLK / (ADC_N + 1). |
| [23:21] | Reserved | Reserved. |
| [20:19] | ADC_S | **ADC Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for ADC controller.<br>00 = ADC_SrcCLK is from XIN.<br>01 = Reserved.<br>10 = ADC_SrcCLK is from ACLKOut.<br>11 = ADC_SrcCLK is from UCLKOut. |
| [18:16] | ADC_SDIV | **ADC Engine Source Clock Divider**<br>This field defines the source clock divide number for clock divider of APLL and UPLL output. This field only takes effect while the ADC_S (CLK_DIVCTL7[20:19]) is 2'b10 (APLL) or 2'b11 (UPLL).<br>If ADC_S (CLK_DIVCTL7[20:19]) is 2'b10,<br>ACLKOut = APLLFout ÷ (ADC_SDIV + 1).<br>If ADC_S (CLK_DIVCTL7[20:19]) is 2'b11,<br>UCLKOut = UPLLFout ÷ (ADC_SDIV + 1). |
| [15] | KPI_S | **KPI Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for KPI controller.<br>0 = XIN.<br>1 = X32K. |

| | | |
|---|---|---|
| [14:8] | **KPI_N** | **KPI Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for KPI controller.<br>The actual clock divide number is (KPI_N + 1). So,<br>ECLKkpi = KPI_SrcCLK / (KPI_N + 1). |
| [7] | **GPIO_S** | **GPIO Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for GPIO controller.<br>0 = XIN.<br>1 = X32K. |
| [6:0] | **GPIO_N** | **GPIO Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for GPIO controller.<br>The actual clock divide number is (GPIO_N + 1). So,<br>ECLKgpio = GPIO_SrcCLK / (GPIO_N + 1). |

## Clock Divider Control Register 8 (CLK_DIVCTL8)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CLK_DIVCTL8 | CLK_BA+0x040 | R/W | Clock Divider Control Register 8 | 0x0000_0500 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | ETIMER3_S | | Reserved | | ETIMER2_S | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | ETIMER1_S | | Reserved | | ETIMER0_S | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | WWDT_S | | WDT_S | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MDCLK_N | | | | | | | |

| Bits | Description | |
|---|---|---|
| [29:28] | **ETIMER3_S** | **Enhanced Timer 3 Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for Enhanced Timer 3 controller.<br>00: ETIMER3_SrcCLK = XIN.<br>01: ETIMER3_SrcCLK = PCLK.<br>10: ETIMER3_SrcCLK = PCLK/4096.<br>11: ETIMER3_SrcCLK = 32.768 kHz. |
| [25:24] | **ETIMER2_S** | **Enhanced Timer 2 Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for Enhanced Timer 2 controller.<br>00: ETIMER2_SrcCLK = XIN.<br>01: ETIMER2_SrcCLK = PCLK.<br>10: ETIMER2_SrcCLK = PCLK/4096.<br>11: ETIMER2_SrcCLK = 32.768 kHz. |
| [21:20] | **ETIMER1_S** | **Enhanced Timer 1 Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for Enhanced Timer 1 controller.<br>00: ETIMER1_SrcCLK = XIN.<br>01: ETIMER1_SrcCLK = PCLK.<br>10: ETIMER1_SrcCLK = PCLK/4096.<br>11: ETIMER1_SrcCLK = 32.768 kHz. |

| | | |
|---|---|---|
| [17:16] | **ETIMER0_S** | **Enhanced Timer 0 Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for Enhanced Timer 0 controller.<br>00: ETIMER0_SrcCLK = XIN.<br>01: ETIMER0_SrcCLK = PCLK.<br>10: ETIMER0_SrcCLK = PCLK/4096.<br>11: ETIMER0_SrcCLK = 32.768 kHz. |
| [11:10] | **WWDT_S** | **WWDT Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for WWDT controller.<br>00: WWDT_SrcCLK = XIN.<br>01: WWDT_SrcCLK = XIN/128.<br>10: WWDT_SrcCLK = PCLK/4096.<br>11: WWDT_SrcCLK = 32.768 kHz. |
| [9:8] | **WDT_S** | **WDT Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for WDT controller.<br>00: WDT_SrcCLK = XIN.<br>01: WDT_SrcCLK = XIN/128.<br>10: WDT_SrcCLK = PCLK/4096.<br>11: WDT_SrcCLK = 32.768 kHz. |
| [7:0] | **MDCLK_N** | **MII Management Interface Clock**<br>This field defines the clock divide number for clock divider to generate the clock for MII management interface.<br>The actual clock divide number is (MDCLK_N + 1). So,<br>MDCLK = HCLK / (MDCLK_N + 1). |

## Clock Divider Control Register 9 (CLK_DIVCTL9)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CLK_DIVCTL9 | CLK_BA+0x044 | R/W | Clock Divider Control Register 9 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CKO_N | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | CKO_S | | CKO_SDIV | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SDH_N | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | SDH_S | | SDH_SDIV | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **CKO_N** | **Reference Clock Out Divide**<br>This field defines the clock divide number for clock divider to generate the reference clock output<br>The actual clock divide number is (CKO_N + 1). So,<br>CKO_CLK = CKO_SrcCLK / (CKO_N + 1). |
| [23:21] | **Reserved** | Reserved. |
| [20:19] | **CKO_S** | **Reference Clock Out Source Selection**<br>This field selects which clock is used to be the source of reference clock output.<br>00 = CKO_SrcCLK is from XIN.<br>01 = Reserved.<br>10 = CKO_SrcCLK is from ACLKOut.<br>11 = CKO_SrcCLK is from UCLKOut. |
| [18:16] | **CKO_SDIV** | **Reference Clock Out Source Clock Divider**<br>This field defines the source clock divide number for clock divider of APLL and UPLL output. This field only takes effect while the CKO_S (CLK_DIVCTL9[20:19]) is 2'b10 (APLL) or 2'b11 (UPLL).<br>If CKO_S (CLK_DIVCTL9[20:19]) is 2'b10,<br>ACLKOut = APLLFout ÷ (CKO_SDIV + 1).<br>If CKO_S (CLK_DIVCTL9[20:19]) is 2'b11,<br>UCLKOut = UPLLFout ÷ (CKO_SDIV + 1). |
| [15:8] | **SDH_N** | **SD Host Engine Clock Divider**<br>This field defines the clock divide number for clock divider to generate the engine clock for SD host controller.<br>The actual clock divide number is (SDH_N + 1). So,<br>SDH_CLK = SDH_SrcCLK / (SDH_N + 1). |
| [7:5] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [4:3] | **SDH_S** | **SD Host Engine Clock Source Selection**<br>This field selects which clock is used to be the source of engine clock for SD host controller.<br>00 = SDH_SrcCLK is from XIN.<br>01 = Reserved.<br>10 = SDH_SrcCLK is from ACLKOut.<br>11 = SDH_SrcCLK is from UCLKOut. |
| [2:0] | **SDH_SDIV** | **SD Host Engine Source Clock Divider**<br>This field defines the source clock divide number for clock divider of APLL and UPLL output. This field only takes effect while the SDH_S (CLK_DIVCTL9[4:3]) is 2'b10 (APLL) or 2'b11 (UPLL).<br>If SDH_S (CLK_DIVCTL9[4:3]) is 2'b10,<br>ACLKOut = APLLFout ÷ (SDH_SDIV + 1).<br>If SDH_S (CLK_DIVCTL9[4:3]) is 2'b11,<br>UCLKOut = UPLLFout ÷ (SDH_SDIV + 1). |

## APLL Control Register (CLK_APLLCON), UPLL Control Register (CLK_UPLLCON)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CLK_APLLCON | CLK_BA+0x060 | R/W | APLL Control Register | 0x1000_0015 |
| CLK_UPLLCON | CLK_BA+0x064 | R/W | UPLL Control Register | 0xX000_0015 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PLL_STB | RESETN | BYPASS | PD | FRAC | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FRAC | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| OUT_DV | | | IN_DV | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IN_DV | FB_DV | | | | | | |

| Bits | Description | |
|---|---|---|
| [31] | PLL_STB | **PLL Stable Flag**<br>0 = PLL is not stable.<br>1 = PLL is stable (500us after PLL setting changed). |
| [30] | RESETN | **Reset Mode Enable**<br>0 = PLL is in reset mode.<br>1 = PLL is in normal operation mode (Default). |
| [29] | BYPASS | **Bypass Mode Enable**<br>0 = PLL is in normal operation mode (Default).<br>1 = PLL is in bypass mode. |
| [28] | PD | **Power Down Mode Enable**<br>0 = PLL is in normal operation mode.<br>1 = PLL is in power down mode (Default). |
| [27:16] | FRAC | **PLL VCO Output Clock Feedback Divider Fraction Part**<br>Set the fraction part (X) of feedback divider factor.<br>Write a non-zero value to this field enables the fraction mode automatically. Please keep this field in 0x0 if don't want to use the PLL fraction mode.<br>The X = FRAC[11:0] / $2^{12}$. |
| [15:13] | OUT_DV | **PLL Output Divider**<br>Set the output divider factor (P) from 1 to 8.<br>The P = OUT_DV[2:0] + 1. |

| | | |
|---|---|---|
| [12:7] | **IN_DV** | **Reference Input Divider**<br>Set the reference divider factor (M) from 1 to 64.<br><br>The M = IN_DV[5:0] + 1. |
| [6:0] | **FB_DV** | **PLL VCO Output Clock Feedback Divider Integer Part**<br>Set the feedback divider factor (N) from 1 to 128.<br><br>The N = FB_DV[6:0] + 1. |

The formula to calculate the PLL output frequency shown below:

$$F_{pllout} = 12\ \text{MHz} \times \frac{N}{M \times P}$$

$$F_{vco} = 12\ \text{MHz} \times \frac{N}{M}$$

$$200\ \text{MHz} < F_{vco} < 500\ MHz$$

$$F_{pfd} = \frac{12\ MHz}{M} = \frac{F_{vco}}{N}$$

| N | $F_{pfd}$ Range |
|---|---|
| 1 | $11.0 \le F_{pfd} \le 80$ |
| 2 | $7.0 \le F_{pfd} \le 80$ |
| 3 | $5.0 \le F_{pfd} \le 80$ |
| 4 | $4.0 \le F_{pfd} \le 80$ |
| 5 | $3.5 \le F_{pfd} \le 80$ |
| 6 | $3.0 \le F_{pfd} \le 80$ |
| 7 ~ 8 | $2.5 \le F_{pfd} \le 80$ |
| 9 ~ 10 | $3.5 \le F_{pfd} \le 80$ |
| 11 ~ 40 | $3.0 \le F_{pfd} \le 80$ |
| 41 ~ 128 | $2.5 \le F_{pfd} \le 80$ |

## PLL Stable Counter and Test Clock Control Register (CLK_PLLSTBCNTR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CLK_PLLSTBCNTR** | CLK_BA+0x080 | R/W | PLL Stable Counter and Test Clock Control Register | 0x0000_1800 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [15:0] | **PLLSTBCNT** | PLL Stable Counter |

## 5.4 Advanced Interrupt Controller (AIC)

### 5.4.1 Overview

An interrupt temporarily changes the sequence of program execution to react to a particular event such as power failure, watchdog timer timeout, transmit/receive request from Ethernet MAC Controller, and so on. The CPU processor provides two modes of interrupt, the Fast Interrupt (FIQ) mode for critical session and the Interrupt (IRQ) mode for general purpose. The IRQ request is occurred when the nIRQ input is asserted. Similarly, the FIQ request is occurred when the nFIQ input is asserted. The FIQ has privilege over the IRQ and can preempt an ongoing IRQ. It is possible to ignore the FIQ and the IRQ by setting the F and I bits in the current program status register (CPSR).

The Advanced Interrupt Controller (AIC) is capable of processing the interrupt requests up to 64 different sources. Currently, 61 interrupt sources are defined. Each interrupt source is uniquely assigned to an interrupt channel. For example, the watchdog timer interrupt is assigned to channel 1. The AIC implements a proprietary eight-level priority scheme that categories the available 61 interrupt sources into eight priority levels. Interrupt sources within the priority level 0 is the highest priority and the priority level 7 is the lowest. In order to make this scheme work properly, a certain priority level must be specified to each interrupt source during power-on initialization; otherwise, the system shall behave unexpectedly. Within each priority level, interrupt source that is positioned in a lower channel has a higher priority. Interrupt source that is active, enabled, and positioned in the lowest channel with priority level 0 is promoted to the FIQ. Interrupt sources within the priority levels other than 0 are routed to the IRQ. The IRQ can be preempted by the occurrence of the FIQ. Interrupt nesting is performed automatically by the AIC.

Though interrupt sources originated from the chip itself are intrinsically high-level sensitive, the AIC can be configured as either low-level sensitive, high-level sensitive, negative-edge triggered, or positive-edge triggered to each interrupt source.

### 5.4.2 Features

- AMBA APB bus interface
- External interrupts can be programmed as either edge-triggered or level-sensitive
- External interrupts can be programmed as either low-active or high-active
- Flags to reflect the status of each interrupt source
- Individual mask for each interrupt source
- Support proprietary 8-level interrupt scheme to employ the priority scheme.
- Priority methodology is adopted to allow for interrupt daisy-chaining
- Automatically masking out the lower priority interrupt during interrupt nesting
- Automatically clearing the interrupt flag when the external interrupt source is programmed to be edge-triggered

### 5.4.3 Block Diagram

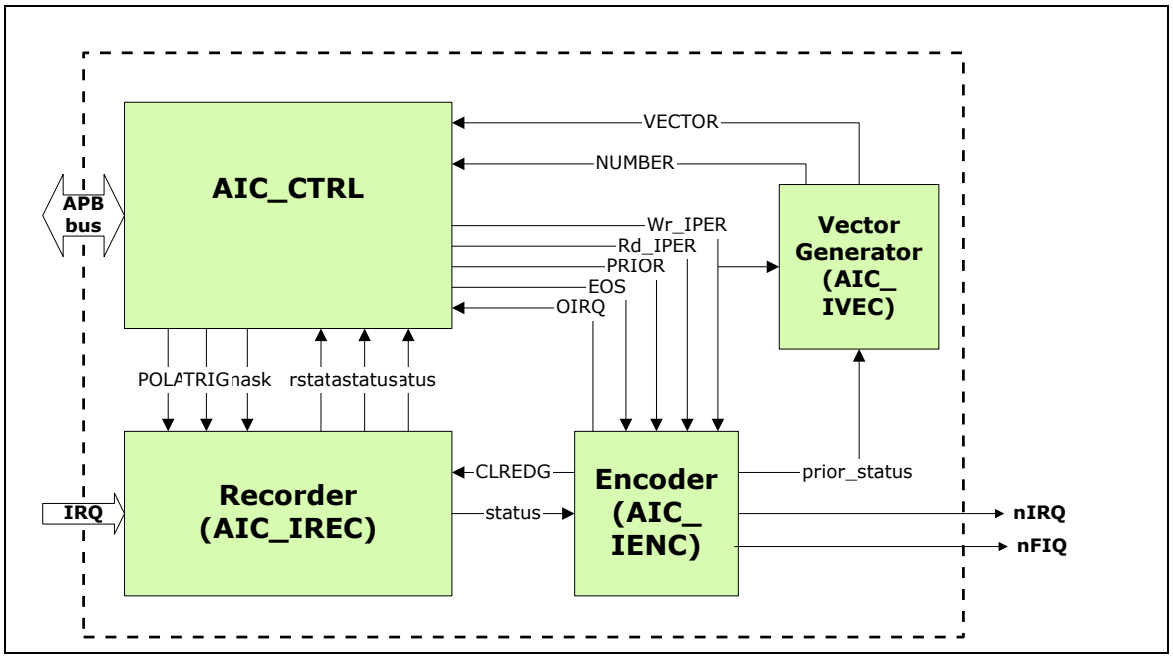



Figure 5.4-1 Advanced Interrupt Controller (AIC) Block Diagram

### 5.4.4 Functional Description

#### 5.4.4.1 *Hardware Interrupt Vectoring*

The hardware interrupt vectoring can be used to shorten the interrupt latency. If it is not be used, priority determination must be carried out by software. When the Interrupt Priority Encoding Register (AIC_IPER) is read, it will return an integer representing the channel that is active and having the highest priority. This integer is equivalent to interrupt number multiplied by 4 (shifted left two bits to word-align it), such that it can be used directly to index into a interrupt vector table to select the appropriate interrupt service routine vector.

#### 5.4.4.2 *Priority Controller*

An 8-level priority encoder controls the NIRQ line. Each interrupt source belongs to priority group between of 0 to 7. Group 0 has the highest priority and group 7 the lowest. When more than one unmasked interrupt channels are active at the same time, the interrupt with the highest priority is serviced first. If all active interrupts have equal priority, the interrupt with the lowest interrupt source number is serviced firstly.

The current priority level is defined as the priority level of the interrupt with the highest priority at the time the register AIC_IPER is read. In the case when a higher priority unmasked interrupt occurs while an interrupt already exits, there are two possible outcomes depending on whether the AIC_IPER has been read.

If the processor has already read the AIC_IPER and caused the NIRQ line to be de-asserted, then the NIRQ line is reasserted. When the processor has enabled nested interrupts and reads the AIC_IPER again, it reads the new, higher priority interrupt vector. At the same time, the current priority level is

updated to the higher priority.

If the AIC_IPER has not been read after the NIRQ line has been asserted, then the processor will read the new higher priority interrupt vector in the AIC_IPER register and the current priority level is updated.

When the End of Service Command Register (AIC_EOSCR) is written, the current interrupt level is updated with the last stored interrupt level from the stack (if any). Therefore, at the end of a higher priority interrupt, the AIC returns to the previous state corresponding to the preceding lower priority interrupt which had been interrupted.

#### 5.4.4.3 *Interrupt Handling*

When the IRQ line is asserted, the interrupt handler must read the AIC_IPER as soon as possible (If the H/W interrupt priority wants to be used). This can de-assert the NIRQ request to the processor and clears the interrupt if it is programmed to be edge triggered. This allows the AIC to assert the NIRQ line again when a higher priority unmasked interrupt occurs.

The AIC_EOSCR (End of Service Command Register) must be written at the end of the interrupt service routine. This permits pending interrupts to be serviced.

#### 5.4.4.4 *Interrupt Masking*

Each interrupt source, including FIQ, can be enabled or disabled individually by using the command registers AIC_MECR and AIC_MDCR. The status of interrupt mask can be read from the read only register AIC_IMR. A disabled interrupt doesn't affect the service of other interrupts.

#### 5.4.4.5 *Interrupt Clearing and Setting*

All interrupt sources (including FIQ) can be individually set or clear by respectively writing to the registers AIC_SSCR and AIC_SCCR when they are programmed to be edge triggered. This feature of the AIC is useful in auto-testing or software debugging.

#### 5.4.4.6 *Fake Interrupt*

When the AIC asserts the NIRQ line, the processor enters interrupt mode and the interrupt handler reads the AIC_IPER, it may happen that AIC de-asserts the NIRQ line after the processor has taken into account the NIRQ assertion and before the read of the AIC_IPER.

This behavior is called a fake interrupt.

The AIC is able to detect these fake interrupts and returns all zero when AIC_IPER is read. The same mechanism of fake interrupt occurs if the processor reads the AIC_IPER (application software or ICE) when there is no interrupt pending. The current priority level is not updated in this situation. Hence, the AIC_EOSCR shouldn't be written.

#### 5.4.4.7 *ICE/Debug Mode*

This mode allows reading of the AIC_IPER without performing the associated automatic operations. This is necessary when working with a debug system. When an ICE or debug monitor reads the AIC user interface, the AIC_IPER can be read. This has the following consequences in normal mode:

If there is no enabled pending interrupt, the fake vector will be returned.

If an enabled interrupt with a higher priority than the current one is pending, it will be stacked.

In the second case, an End-of-Service command would be necessary to restore the state of the AIC. This operation is generally not performed by the debug system. Therefore, the debug system would become strongly intrusive, and could cause the application to enter an undesired state.

This can be avoided by using ICE/Debug Mode. When this mode enable, the AIC performs interrupt stacking only when a write access is performed on the AIC_IPER. Hence, the interrupt service routine must write to the AIC_IPER (any value) just after reading it. When AIC_IPER is written, the new status of AIC, including the value of interrupt source number register (AIC_ISNR), is updated with the value that is kept at previous reading of AIC_IPER. The debug system must not write to the AIC_IPER as this would cause undesirable effects.

The following table shows the main steps of an interrupt and the order in which they are performed according to the mode:

| Action | Normal Mode | ICE/Debug Mode |
|---|---|---|
| Calculate active interrupt | Read AIC_IPER | Read AIC_IPER |
| Determine and return the vector of the active interrupt | Read AIC_IPER | Read AIC_IPER |
| Push on internal stack the current priority level | Read AIC_IPER | Write AIC_IPER |
| Acknowledge the interrupt (Note 1) | Read AIC_IPER | Write AIC_IPER |
| No effect (Note 2) | Read AIC_IPER | |

Notes:

NIRQ de-assertion and automatic interrupt clearing if the source is programmed as level sensitive.

Note that software which has been written and debugged using this mode will run correctly in normal mode without modification. However, in normal mode writing to AIC_IPER has no effect and can be removed to optimize the code

#### 5.4.4.8 *Interrupt Sources*

| Priority | Name | Mode | Source |
|---|---|---|---|
| 1 (Highest) | WDT_INT, | Positive Level | Watch Dog Timer Interrupt |
| 2 | WWDT_INT | Positive Level | Windowed-WDT Interrupt |
| 3 | LVD_INT | Positive Level | Low Voltage Detect Interrupt |
| 4 | External Interrupt 0 | Positive Level | External Interrupt 0 |
| 5 | External Interrupt 1 | Positive Level | External Interrupt 1 |

| | | | |
|---|---|---|---|
| 6 | External Interrupt 2 | Positive Level | External Interrupt 2 |
| 7 | External Interrupt 3 | Positive Level | External Interrupt 3 |
| 8 | External Interrupt 4 | Positive Level | External Interrupt 4 |
| 9 | External Interrupt 5 | Positive Level | External Interrupt 5 |
| 10 | External Interrupt 6 | Positive Level | External Interrupt 6 |
| 11 | External Interrupt 7 | Positive Level | External Interrupt 7 |
| 12 | I2S_INT | Positive Level | $I^2S$ Controller Interrupt |
| 13 | LCD_INT | Positive Level | LCD Controller Interrupt |
| 14 | CAP_INT | Positive Level | Sensor Interface Controller Interrupt |
| 15 | RTC_INT | Positive Level | RTC Interrupt |
| 16 | TMR0_INT | Positive Level | Timer 0 Interrupt |
| 17 | TMR1_INT | Positive Level | Timer 1 Interrupt |
| 18 | ADC_INT | Positive Level | ADC Interrupt |
| 19 | EMC0_RX_INT | Positive Level | EMC 0 RX Interrupt |
| 20 | EMC1_RX_INT | Positive Level | EMC 1 RX Interrupt |
| 21 | EMC0_TX_INT | Positive Level | EMC 0 TX Interrupt |
| 22 | EMC1_TX_INT | Positive Level | EMC 1 TX Interrupt |
| 23 | EHCI_INT | Positive Level | USB 2.0 Host Controller Interrupt |
| 24 | OHCI_INT | Positive Level | USB 1.1 Host Controller Interrupt |
| 25 | GDMA0_INT | Positive Level | GDMA Channel 0 Interrupt |
| 26 | GDMA1_INT | Positive Level | GDMA Channel 1 Interrupt |
| 27 | SDH_INT | Positive Level | SD/SDIO Host Interrupt |
| 28 | SIC_INT | Positive Level | SIC Interrupt |
| 29 | UDC_INT | Positive Level | USB Device Controller Interrupt |
| 30 | TMR2_INT | Positive Level | Timer 2 Interrupt |
| 31 | TMR3_INT | Positive Level | Timer 3 Interrupt |
| 32 | TMR4_INT | Positive Level | Timer 4 Interrupt |
| 33 | JPEG_INT | Positive Level | JPEG Engine Interrupt |
| 34 | GE2D_INT | Positive Level | 2D Graphic Engine Interrupt |
| 35 | CRYPTO_INT | Positive Level | CRYPTO Engine Interrupt |
| 36 | UART0_INT | Positive Level | UART 0 Interrupt |
| 37 | UART1_INT | Positive Level | UART 1 Interrupt |
| 38 | UART2_INT | Positive Level | UART 2 Interrupt |
| 39 | UART4_INT | Positive Level | UART 4 Interrupt |

| | | | |
|---|---|---|---|
| 40 | UART6_INT | Positive Level | UART 6 Interrupt |
| 41 | UART8_INT | Positive Level | UART 8 Interrupt |
| 42 | UART10_INT | Positive Level | UART 10 Interrupt |
| 43 | UART3_INT | Positive Level | UART 3 Interrupt |
| 44 | UART5_INT | Positive Level | UART 5 Interrupt |
| 45 | UART7_INT | Positive Level | UART 7 Interrupt |
| 46 | UART9_INT | Positive Level | UART 9 Interrupt |
| 47 | ETMR0_INT | Positive Level | Enhanced Timer 0 Interrupt |
| 48 | ETMR1_INT | Positive Level | Enhanced Timer 1 Interrupt |
| 49 | ETMR2_INT | Positive Level | Enhanced Timer 2 Interrupt |
| 50 | ETMR3_INT | Positive Level | Enhanced Timer 3 Interrupt |
| 51 | SPI0_INT | Positive Level | SPI 0 Interrupt |
| 52 | SPI1_INT | Positive Level | SPI 1 Interrupt |
| 53 | I2C0_INT | Positive Level | $I^2C$ 0 Interrupt |
| 54 | I2C1_INT | Positive Level | $I^2C$ 1 Interrupt |
| 55 | SMC0_INT | Positive Level | SmartCard 0 Interrupt |
| 56 | SMC1_INT | Positive Level | SmartCard 1 Interrupt |
| 57 | GPIO_INT | Positive Level | GPIO Interrupt |
| 58 | CAN0_INT | Positive Level | CAN 0 Interrupt |
| 59 | CAN1_INT | Positive Level | CAN 1 Interrupt |
| 60 | PWM_INT | Positive Level | PWM Interrupt |
| 61 | KPI_INT | Positive Level | KPI Interrupt |
| 62 ~ 63 | Reserved | | Reserved |

### 5.4.5 Register Map

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **AIC Base Address:**<br>**AIC_BA = 0xB800_2000** | | | | |
| **AIC_SCR1** | AIC_BA+0x000 | R/W | AIC Source Control Register 1 | 0x4747_4747 |
| **AIC_SCR2** | AIC_BA+0x004 | R/W | AIC Source Control Register 2 | 0x4747_4747 |
| **AIC_SCR3** | AIC_BA+0x008 | R/W | AIC Source Control Register 3 | 0x4747_4747 |
| **AIC_SCR4** | AIC_BA+0x00C | R/W | AIC Source Control Register 4 | 0x4747_4747 |
| **AIC_SCR5** | AIC_BA+0x010 | R/W | AIC Source Control Register 5 | 0x4747_4747 |
| **AIC_SCR6** | AIC_BA+0x014 | R/W | AIC Source Control Register 6 | 0x4747_4747 |
| **AIC_SCR7** | AIC_BA+0x018 | R/W | AIC Source Control Register 7 | 0x4747_4747 |
| **AIC_SCR8** | AIC_BA+0x01C | R/W | AIC Source Control Register 8 | 0x4747_4747 |
| **AIC_SCR9** | AIC_BA+0x020 | R/W | AIC Source Control Register 9 | 0x4747_4747 |
| **AIC_SCR10** | AIC_BA+0x024 | R/W | AIC Source Control Register 10 | 0x4747_4747 |
| **AIC_SCR11** | AIC_BA+0x028 | R/W | AIC Source Control Register 11 | 0x4747_4747 |
| **AIC_SCR12** | AIC_BA+0x02C | R/W | AIC Source Control Register 12 | 0x4747_4747 |
| **AIC_SCR13** | AIC_BA+0x030 | R/W | AIC Source Control Register 13 | 0x4747_4747 |
| **AIC_SCR14** | AIC_BA+0x034 | R/W | AIC Source Control Register 14 | 0x4747_4747 |
| **AIC_SCR15** | AIC_BA+0x038 | R/W | AIC Source Control Register 15 | 0x4747_4747 |
| **AIC_SCR16** | AIC_BA+0x03C | R/W | AIC Source Control Register 16 | 0x0000_4747 |
| **AIC_IRSR** | AIC_BA+0x100 | R | AIC Interrupt Raw Status Register | 0x0000_0000 |
| **AIC_IRSRH** | AIC_BA+0x104 | R | AIC Interrupt Raw Status Register (High) | 0x0000_0000 |
| **AIC_IASR** | AIC_BA+0x108 | R | AIC Interrupt Active Status Register | 0x0000_0000 |
| **AIC_IASRH** | AIC_BA+0x10C | R | AIC Interrupt Active Status Register (High) | 0x0000_0000 |
| **AIC_ISR** | AIC_BA+0x110 | R | AIC Interrupt Status Register | 0x0000_0000 |
| **AIC_ISRH** | AIC_BA+0x114 | R | AIC Interrupt Status Register (High) | 0x0000_0000 |
| **AIC_IPER** | AIC_BA+0x118 | R | AIC Interrupt Priority Encoding Register | 0x0000_0000 |
| **AIC_ISNR** | AIC_BA+0x120 | R | AIC Interrupt Source Number Register | 0x0000_0000 |
| **AIC_OISR** | AIC_BA+0x124 | R | AIC Output Interrupt Status Register | 0x0000_0000 |
| **AIC_IMR** | AIC_BA+0x128 | R | AIC Interrupt Mask Register | 0x0000_0000 |
| **AIC_IMRH** | AIC_BA+0x12C | R | AIC Interrupt Mask Register (High) | 0x0000_0000 |
| **AIC_MECR** | AIC_BA+0x130 | W | AIC Mask Enable Command Register | Undefined |
| **AIC_MECRH** | AIC_BA+0x134 | W | AIC Mask Enable Command Register (High) | Undefined |
| **AIC_MDCR** | AIC_BA+0x138 | W | AIC Mask Disable Command Register | Undefined |

| | | | | |
|---|---|---|---|---|
| **AIC_MDCRH** | AIC_BA+0x13C | W | AIC Mask Disable Command Register (High) | Undefined |
| **AIC_SSCR** | AIC_BA+0x140 | W | AIC Source Set Command Register | Undefined |
| **AIC_SSCRH** | AIC_BA+0x144 | W | AIC Source Set Command Register (High) | Undefined |
| **AIC_SCCR** | AIC_BA+0x148 | W | AIC Source Clear Command Register | Undefined |
| **AIC_SCCRH** | AIC_BA+0x14C | W | AIC Source Clear Command Register (High) | Undefined |
| **AIC_EOSCR** | AIC_BA+0x150 | W | AIC End of Service Command Register | Undefined |

### 5.4.6 Register Description

### AIC Source Control Register (AIC_SCR1 ~ AIC_SCR16)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **AIC_SCR1** | AIC_BA+0x000 | R/W | AIC Source Control Register 1 | 0x4747_4747 |
| **AIC_SCR2** | AIC_BA+0x004 | R/W | AIC Source Control Register 2 | 0x4747_4747 |
| **AIC_SCR3** | AIC_BA+0x008 | R/W | AIC Source Control Register 3 | 0x4747_4747 |
| **AIC_SCR4** | AIC_BA+0x00C | R/W | AIC Source Control Register 4 | 0x4747_4747 |
| **AIC_SCR5** | AIC_BA+0x010 | R/W | AIC Source Control Register 5 | 0x4747_4747 |
| **AIC_SCR6** | AIC_BA+0x014 | R/W | AIC Source Control Register 6 | 0x4747_4747 |
| **AIC_SCR7** | AIC_BA+0x018 | R/W | AIC Source Control Register 7 | 0x4747_4747 |
| **AIC_SCR8** | AIC_BA+0x01C | R/W | AIC Source Control Register 8 | 0x4747_4747 |
| **AIC_SCR9** | AIC_BA+0x020 | R/W | AIC Source Control Register 9 | 0x4747_4747 |
| **AIC_SCR10** | AIC_BA+0x024 | R/W | AIC Source Control Register 10 | 0x4747_4747 |
| **AIC_SCR11** | AIC_BA+0x028 | R/W | AIC Source Control Register 11 | 0x4747_4747 |
| **AIC_SCR12** | AIC_BA+0x02C | R/W | AIC Source Control Register 12 | 0x4747_4747 |
| **AIC_SCR13** | AIC_BA+0x030 | R/W | AIC Source Control Register 13 | 0x4747_4747 |
| **AIC_SCR14** | AIC_BA+0x034 | R/W | AIC Source Control Register 14 | 0x4747_4747 |
| **AIC_SCR15** | AIC_BA+0x038 | R/W | AIC Source Control Register 15 | 0x4747_4747 |
| **AIC_SCR16** | AIC_BA+0x03C | R/W | AIC Source Control Register 16 | 0x0000_4747 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **TYPE (Channel 3)** | | **Reserved** | | | **PRIORITY (Channel 3)** | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **TYPE (Channel 2)** | | **Reserved** | | | **PRIORITY (Channel 2)** | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **TYPE (Channel 1)** | | **Reserved** | | | **PRIORITY (Channel 1)** | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **TYPE (channel 0)** | | **Reserved** | | | **PRIORITY (Channel 0)** | | |

| Bits | Description | |
|---|---|---|
| [7:6] | **TYPE** | **Interrupt Type**<br>00: low-active level triggered<br>01: high-active level triggered<br>10: low-active edge triggered<br>11: high-active edge triggered<br>Interrupts other than INT_EXT can be configured as level triggered during normal operation unless in the test mode. |

| | | |
|---|---|---|
| [5:3] | **Reserved** | Reserved. |
| [2:0] | **PRIORITY** | **Priority Level (0 – 7)**<br>The level 0 indicates the highest priority and the level 7 indicates the lowest priority.<br>An interrupt is treated as a FIQ for the priority level 0, and is treated as an IRQ for other levels.<br>If two or more interrupts have the identical priority level, the interrupts located in the upper rows of the interrupt source table, have higher priorities. |

### AIC Interrupt Raw Status Register (AIC_IRSR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **AIC_IRSR** | AIC_BA+0x100 | R | AIC Interrupt Raw Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **IRS31** | **IRS30** | **IRS29** | **IRS28** | **IRS27** | **IRS26** | **IRS25** | **IRS24** |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **IRS23** | **IRS22** | **IRS21** | **IRS20** | **IRS19** | **IRS18** | **IRS17** | **IRS16** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **IRS15** | **IRS14** | **IRS13** | **IRS12** | **IRS11** | **IRS10** | **IRS9** | **IRS8** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **IRS7** | **IRS6** | **IRS5** | **IRS4** | **IRS3** | **IRS2** | **IRS1** | **Reserved** |

| Bits | Description | |
|---|---|---|
| [31:1] | **IRS x** | **Interrupt Status**<br>Indicate the intrinsic status of the corresponding interrupt source<br>0 = Interrupt channel is in the voltage level 0.<br>1 = Interrupt channel is in the voltage level 1. |
| [0] | **Reserved** | Reserved. |

This register records the intrinsic state within each interrupt channel.

### AIC Interrupt Raw Status Register (High) (AIC_IRSRH)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| AIC_IRSRH | AIC_BA+0x104 | R | AIC Interrupt Raw Status Register (High) | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | ISR61 | ISR60 | ISR59 | ISR58 | ISR57 | IRS56 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IRS55 | IRS54 | IRS53 | IRS52 | IRS51 | IRS50 | IRS49 | IRS48 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| IRS47 | IRS46 | IRS45 | IRS44 | IRS43 | IRS42 | IRS41 | IRS40 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IRS39 | IRS38 | IRS37 | IRS36 | IRS35 | IRS34 | IRS33 | IRS32 |

| Bits | Description | |
|---|---|---|
| [31:30] | **Reserved** | Reserved. |
| [29:0] | **IRS x** | **Interrupt Status**<br>Indicate the intrinsic status of the corresponding interrupt source<br>0 = Interrupt channel is in the voltage level 0.<br>1 = Interrupt channel is in the voltage level 1. |

This register records the intrinsic state within each interrupt channel.

### AIC Interrupt Active Status Register (AIC_IASR)

This register indicates the status of each interrupt channel in consideration of the interrupt source type as defined in the corresponding Source Control Register, but regardless of its mask setting.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **AIC_IASR** | AIC_BA+0x108 | R | AIC Interrupt Active Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **IAS31** | **IAS30** | **IAS29** | **IAS28** | **IAS27** | **IAS26** | **IAS25** | **IAS24** |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **IAS23** | **IAS22** | **IAS21** | **IAS20** | **IAS19** | **IAS18** | **IAS17** | **IAS16** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **IAS15** | **IAS14** | **IAS13** | **IAS12** | **IAS11** | **IAS10** | **IAS9** | **IAS8** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **IAS7** | **IAS6** | **IAS5** | **IAS4** | **IAS3** | **IAS2** | **IAS1** | **Reserved** |

| Bits | Description | |
|---|---|---|
| [31:1] | **IAS x** | **Interrupt Active Status**<br>Indicate the status of the corresponding interrupt source<br>0 = Corresponding interrupt channel is inactive.<br>1 = Corresponding interrupt channel is active. |

### AIC Interrupt Active Status Register (High) (AIC_IASRH)

This register indicates the status of each interrupt channel in consideration of the interrupt source type as defined in the corresponding Source Control Register, but regardless of its mask setting.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **AIC_IASRH** | AIC_BA+0x10C | R | AIC Interrupt Active Status Register (High) | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | **IAS61** | **IAS60** | **IAS59** | **IAS58** | **IAS57** | **IAS56** |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **IAS55** | **IAS54** | **IAS53** | **IAS52** | **IAS51** | **IAS50** | **IAS49** | **IAS48** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **IAS47** | **IAS46** | **IAS45** | **IAS44** | **IAS43** | **IAS42** | **IAS41** | **IAS40** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **IAS39** | **IAS38** | **IAS37** | **IAS36** | **IAS35** | **IAS34** | **IAS33** | **IAS32** |

| Bits | Description | |
|---|---|---|
| [31:30] | **Reserved** | Reserved. |
| [29:0] | **IAS x** | **Interrupt Active Status**<br>Indicate the status of the corresponding interrupt source<br>0 = Corresponding interrupt channel is inactive.<br>1 = Corresponding interrupt channel is active. |

### AIC Interrupt Status Register (AIC_ISR)

This register identifies those interrupt channels whose are both active and enabled.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **AIC_ISR** | AIC_BA+0x110 | R | AIC Interrupt Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **IS31** | **IS30** | **IS29** | **IS28** | **IS27** | **IS26** | **IS25** | **IS24** |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **IS23** | **IS22** | **IS21** | **IS20** | **IS19** | **IS18** | **IS17** | **IS16** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **IS15** | **IS14** | **IS13** | **IS12** | **IS11** | **IS10** | **IS9** | **IS8** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **IS7** | **IS6** | **IS5** | **IS4** | **IS3** | **IS2** | **IS1** | **Reserved** |

| Bits | Description | |
|---|---|---|
| [31:1] | **IS x** | **Interrupt Status**<br>Indicates the status of corresponding interrupt channel<br>0 = Two possibilities:.<br>The corresponding interrupt channel is inactive no matter whether it is enabled or disabled;<br>It is active but not enabled<br>1 = Corresponding interrupt channel is both active and enabled (can assert an interrupt). |

### AIC Interrupt Status Register (High) (AIC_ISRH)

This register identifies those interrupt channels whose are both active and enabled.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **AIC_ISRH** | AIC_BA+0x114 | R | AIC Interrupt Status Register (High) | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | **IS61** | **IS60** | **IS59** | **IS58** | **IS57** | **IS56** |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **IS55** | **IS54** | **IS53** | **IS52** | **IS51** | **IS50** | **IS49** | **IS48** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **IS47** | **IS46** | **IS45** | **IS44** | **IS43** | **IS42** | **IS41** | **IS40** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **IS39** | **IS38** | **IS37** | **IS36** | **IS35** | **IS34** | **IS33** | **IS32** |

| Bits | Description | |
|---|---|---|
| [31:30] | **Reserved** | Reserved. |
| [29:0] | **IS x** | **Interrupt Status**<br>Indicates the status of corresponding interrupt channel<br>0 = Two possibilities:.<br>The corresponding interrupt channel is inactive no matter whether it is enabled or disabled;<br>It is active but not enabled<br>1 = Corresponding interrupt channel is both active and enabled (can assert an interrupt). |

## AIC Interrupt Priority Encoding Register (AIC_IPER)

When the AIC generates the interrupt, VECTOR represents the interrupt channel number that is active, enabled, and has the highest priority. If the representing interrupt channel possesses a priority level 0, then the interrupt asserted is FIQ; otherwise, it is IRQ. The value of VECTOR is copied to the register AIC_ISNR thereafter by the AIC. This register was restored a value 0 after it was read by the interrupt handler. This register can help indexing into a branch table to quickly jump to the corresponding interrupt service routine.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **AIC_IPER** | AIC_BA+0x118 | R | AIC Interrupt Priority Encoding 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** |
| **VECTOR** | | | | | | **Reserved** | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:2] | **VECTOR** | **Interrupt Vector**<br>0 = no interrupt occurs.<br>1 ~ 56 = representing the interrupt channel that is active, enabled, and having the highest priority. |
| [1:0] | **Reserved** | Reserved. |

### AIC Interrupt Source Number Register (AIC_ISNR)

The purpose of this register is to record the interrupt channel number that is active, enabled, and has the highest priority.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **AIC_ISNR** | AIC_BA+0x120 | R | AIC Interrupt Source 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** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | **IRQID** | | | | | |

| Bits | Description | |
|---|---|---|
| [31:0] | **Reserved** | Reserved. |
| [5:0] | **IRQID** | **IRQ Identification**<br>Stands for the interrupt channel number |

### AIC Output Interrupt Status Register (AIC_OISR)

The AIC classifies the interrupt into FIQ and IRQ. This register indicates whether the asserted interrupt is FIQ or IRQ. If both IRQ and FIQ are equal to 0, it means there is no interrupt occurred.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **AIC_OISR** | AIC_BA+0x124 | R | AIC Output 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** | | | | | | **IRQ** | **FIQ** |

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [1] | **IRQ** | **Interrupt Request**<br>0 = nIRQ line is inactive.<br>1 = nIRQ line is active. |
| [0] | **FIQ** | **Fast Interrupt Request**<br>0 = nFIQ line is inactive.<br>1 = nFIQ line is active. |

### AIC Interrupt Mask Register (AIC_IMR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| AIC_IMR | AIC_BA+0x128 | R | AIC Interrupt Mask Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| IM31 | IM30 | IM29 | IM28 | IM27 | IM26 | IM25 | IM24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IM23 | IM22 | IM21 | IM20 | IM19 | IM18 | IM17 | IM16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| IM15 | IM14 | IM13 | IM12 | IM11 | IM10 | IM9 | IM8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IM7 | IM6 | IM5 | IM4 | IM3 | IM2 | IM1 | Reserved |

| Bits | Description | |
|---|---|---|
| [31:1] | IM x | **Interrupt Mask**<br>This bit determines whether the corresponding interrupt channel is enabled or disabled. Every interrupt channel can be active no matter whether it is enabled or disabled. If an interrupt channel is enabled, it does not definitely mean it is active. Every interrupt channel can be authorized by the AIC only when it is both active and enabled.<br><br>0 = Corresponding interrupt channel is disabled.<br>1 = Corresponding interrupt channel is enabled. |

### AIC Interrupt Mask Register (High) (AIC_IMRH)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **AIC_IMRH** | AIC_BA+0x12C | R | AIC Interrupt Mask Register (High) | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | **IM61** | **IM60** | **IM59** | **IM58** | **IM57** | **IM56** |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **IM55** | **IM54** | **IM53** | **IM52** | **IM51** | **IM50** | **IM49** | **IM48** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **IM47** | **IM46** | **IM45** | **IM44** | **IM43** | **IM42** | **IM41** | **IM40** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **IM39** | **IM38** | **IM37** | **IM36** | **IM35** | **IM34** | **IM33** | **IM32** |

| Bits | Description | |
|---|---|---|
| [31:30] | **Reserved** | Reserved. |
| [29:0] | **IM x** | **Interrupt Mask**<br>This bit determines whether the corresponding interrupt channel is enabled or disabled. Every interrupt channel can be active no matter whether it is enabled or disabled. If an interrupt channel is enabled, it does not definitely mean it is active. Every interrupt channel can be authorized by the AIC only when it is both active and enabled.<br><br>0 = Corresponding interrupt channel is disabled.<br>1 = Corresponding interrupt channel is enabled. |

### AIC Mask Enable Command Register (AIC_MECR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| AIC_MECR | AIC_BA+0x130 | W | AIC Mask Enable Command Register | Undefined |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MEC31 | MEC30 | MEC29 | MEC28 | MEC27 | MEC26 | MEC25 | MEC24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MEC23 | MEC22 | MEC21 | MEC20 | MEC19 | MEC18 | MEC17 | MEC16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MEC15 | MEC14 | MEC13 | MEC12 | MEC11 | MEC10 | MEC9 | MEC8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MEC7 | MEC6 | MEC5 | MEC4 | MEC3 | MEC2 | MEC1 | Reserved |

| Bits | Description | |
|---|---|---|
| [31:1] | **MEC x** | **Mask Enable Command**<br>0 = No effect.<br>1 = Enables the corresponding interrupt channel. |

## AIC Mask Enable Command Register (High) (AIC_MECRH)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| AIC_MECRH | AIC_BA+0x134 | W | AIC Mask Enable Command Register (High) | Undefined |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | MEC61 | MEC60 | MEC59 | MEC58 | MEC57 | MEC56 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MEC55 | MEC54 | MEC53 | MEC52 | MEC51 | MEC50 | MEC49 | MEC48 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MEC47 | MEC46 | MEC45 | MEC44 | MEC43 | MEC42 | MEC41 | MEC40 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MEC39 | MEC38 | MEC37 | MEC36 | MEC35 | MEC34 | MEC33 | MEC32 |

| Bits | Description | |
|---|---|---|
| [31:30] | **Reserved** | Reserved. |
| [29:0] | **MEC x** | **Mask Enable Command**<br>0 = No effect.<br>1 = Enables the corresponding interrupt channel. |

### AIC Mask Disable Command Register (AIC_MDCR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| AIC_MDCR | AIC_BA+0x138 | W | AIC Mask Disable Command Register | Undefined |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MDC31 | MDC30 | MDC29 | MDC28 | MDC27 | MDC26 | MDC25 | MDC24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MDC23 | MDC22 | MDC21 | MDC20 | MDC19 | MDC18 | MDC17 | MDC16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MDC15 | MDC14 | MDC13 | MDC12 | MDC11 | MDC10 | MDC9 | MDC8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MDC7 | MDC6 | MDC5 | MDC4 | MDC3 | MDC2 | MDC1 | Reserved |

| Bits | Description | |
|---|---|---|
| [31:1] | **MDC x** | **Mask Disable Command**<br>0 = No effect.<br>1 = Disables the corresponding interrupt channel. |

## AIC Mask Disable Command Register (High) (AIC_MDCRH)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **AIC_MDCRH** | AIC_BA+0x13C | W | AIC Mask Disable Command Register (High) | Undefined |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | **MDC61** | **MDC60** | **MDC59** | **MDC58** | **MDC57** | **MDC56** |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **MDC55** | **MDC54** | **MDC53** | **MDC52** | **MDC51** | **MDC50** | **MDC49** | **MDC48** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **MDC47** | **MDC46** | **MDC45** | **MDC44** | **MDC43** | **MDC42** | **MDC41** | **MDC40** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **MDC39** | **MDC38** | **MDC37** | **MDC36** | **MDC35** | **MDC34** | **MDC33** | **MDC32** |

| Bits | Description | |
|---|---|---|
| [31:30] | **Reserved** | Reserved. |
| [29:0] | **MDC x** | **Mask Disable Command**<br>0 = No effect.<br>1 = Disables the corresponding interrupt channel. |

### AIC Source Set Command Register (AIC_SSCR)

When chip is under debugging or verification, software can activate any interrupt channel by setting the corresponding bit in this register. This feature is useful in hardware verification or software debugging.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **AIC_SSCR** | AIC_BA+0x140 | W | AIC Source Set Command Register | Undefined |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **SSC31** | **SSC30** | **SSC29** | **SSC28** | **SSC27** | **SSC26** | **SSC25** | **SSC24** |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **SSC23** | **SSC22** | **SSC21** | **SSC20** | **SSC19** | **SSC18** | **SSC17** | **SSC16** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **SSC15** | **SSC14** | **SSC13** | **SSC12** | **SSC11** | **SSC10** | **SSC9** | **SSC8** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **SSC7** | **SSC6** | **SSC5** | **SSC4** | **SSC3** | **SSC2** | **SSC1** | **Reserved** |

| Bits | Description | |
|---|---|---|
| [31:1] | **SSC x** | **Source Set Command**<br>0 = No effect.<br>1 = Activates the corresponding interrupt channel. |

### AIC Source Set Command Register (High) (AIC_SSCRH)

When chip is under debugging or verification, software can activate any interrupt channel by setting the corresponding bit in this register. This feature is useful in hardware verification or software debugging.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **AIC_SSCRH** | AIC_BA+0x144 | W | AIC Source Set Command Register (High) | Undefined |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | **SSC61** | **SSC60** | **SSC59** | **SSC58** | **SSC57** | **SSC56** |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **SSC55** | **SSC54** | **SSC53** | **SSC52** | **SSC51** | **SSC50** | **SSC49** | **SSC48** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **SSC47** | **SSC46** | **SSC45** | **SSC44** | **SSC43** | **SSC42** | **SSC41** | **SSC40** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **SSC39** | **SSC38** | **SSC37** | **SSC36** | **SSC35** | **SSC34** | **SSC33** | **SSC32** |

| Bits | Description | |
|---|---|---|
| [31:30] | **Reserved** | Reserved. |
| [29:0] | **SSC x** | **Source Set Command**<br>0 = No effect.<br>1 = Activates the corresponding interrupt channel. |

### AIC Source Clear Command Register (AIC_SCCR)

When the chip is under debugging or verification, software can deactivate any interrupt channel by setting the corresponding bit in this register. This feature is useful in hardware verification or software debugging.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **AIC_SCCR** | AIC_BA+0x148 | W | AIC Source Clear Command Register | Undefined |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **SCC31** | **SCC30** | **SCC29** | **SCC28** | **SCC27** | **SCC26** | **SCC25** | **SCC24** |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **SCC23** | **SCC22** | **SCC21** | **SCC20** | **SCC19** | **SCC18** | **SCC17** | **SCC16** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **SCC15** | **SCC14** | **SCC13** | **SCC12** | **SCC11** | **SCC10** | **SCC9** | **SCC8** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **SCC7** | **SCC6** | **SCC5** | **SCC4** | **SCC3** | **SCC2** | **SCC1** | **Reserved** |

| Bits | Description | |
|---|---|---|
| [31:1] | **SCC x** | **Source Clear Command**<br>0 = No effect.<br>1 = Deactivates the corresponding interrupt channels. |

### AIC Source Clear Command Register (High) (AIC_SCCRH)

When the chip is under debugging or verification, software can deactivate any interrupt channel by setting the corresponding bit in this register. This feature is useful in hardware verification or software debugging.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **AIC_SCCRH** | AIC_BA+0x14C | W | AIC Source Clear Command Register (High) | Undefined |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | **SCC61** | **SCC60** | **SCC59** | **SCC58** | **SCC57** | **SCC56** |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **SCC55** | **SCC54** | **SCC53** | **SCC52** | **SCC51** | **SCC50** | **SCC49** | **SCC48** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **SCC47** | **SCC46** | **SCC45** | **SCC44** | **SCC43** | **SCC42** | **SCC41** | **SCC40** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **SCC39** | **SCC38** | **SCC37** | **SCC36** | **SCC35** | **SCC34** | **SCC33** | **SCC32** |

| Bits | Description | |
|---|---|---|
| [31:30] | **Reserved** | Reserved. |
| [29:0] | **SCC x** | **Source Clear Command**<br>0 = No effect.<br>1 = Deactivates the corresponding interrupt channels. |

### AIC End of Service Command Register (AIC_EOSCR)

This register is used by the interrupt service routine to indicate that it is completely served. Thus, the interrupt handler can write any value to this register to indicate the end of its interrupt service.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **AIC_EOSCR** | AIC_BA+0x150 | W | AIC End of Service Command 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** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:0] | **EOS** | **End of Service**<br>Write this register indicating the interrupt service routine is completely served.<br>The AIC_EOSCR is a write only register and the value read from this register is undefined. |

## 5.5 SDRAM Interface Controller (SDIC)

### 5.5.1 Overview

The SDRAM Controller support SDR, DDR, Low-Power DDR and DDR2 type SDRAM. The memory device size type can be from 16M bit and up to 1G bits. Only 16-bit data bus width is supported. The total system memory size can be from 2M bytes and up to 256M bytes for different SDRAM configuration.

The SDRAM controller interface to three isolated AHB. All these AHB masters can access the memory independent. Except the memory access, the masters of AHB also could access the SDRAM control registers.

For performance and function issue, the SDRAM controller also supports the proprietary Enhanced-AHB. The EAHB add the down-count address mode, byte-enable signal and explicit burst access number. The explicit access number function is reached by modify the HBURST signal to EHBURST and it represent the access number. The maximum EAHB access number is 16. The SDRAM controller also builds a BIST module to test the external memory device.

An internal arbiter is used to schedule the access from the masters and the BIST request, the BIST request with the highest priority and the then the AHB3 master, AHB2 master and AHB1 master.

The SDRAM controller uses 3 pipe queues to improve the SDRAM command and data bus efficiency. The request in queue0 is the SDRAM active data access request. Simultaneous, the requests in queue1 can request the controller to issue the ACTIVE or PRECHARGE command to reduce the access latency for the later command. The queue1 also can issue the READ or WRITE command to close the SDRAM command when advance pipe queue

The SDRAM refresh rate is programmable. The Refresh and Power-on control module generate the refresh request signal and SDRAM power on sequence. The SDRAM controller also supports software reset, SDRAM self refresh and auto power down function.

### 5.5.2 Features

- Support DDR, DDR2 and LPDDR SDRAM
- Clock speed up to 150 MHz
- Support 16-bit data bus width
- Support two chip selects
- Support total memory size up to 256M bytes (each chip select for 128M bytes)

### 5.5.3 Block Diagram

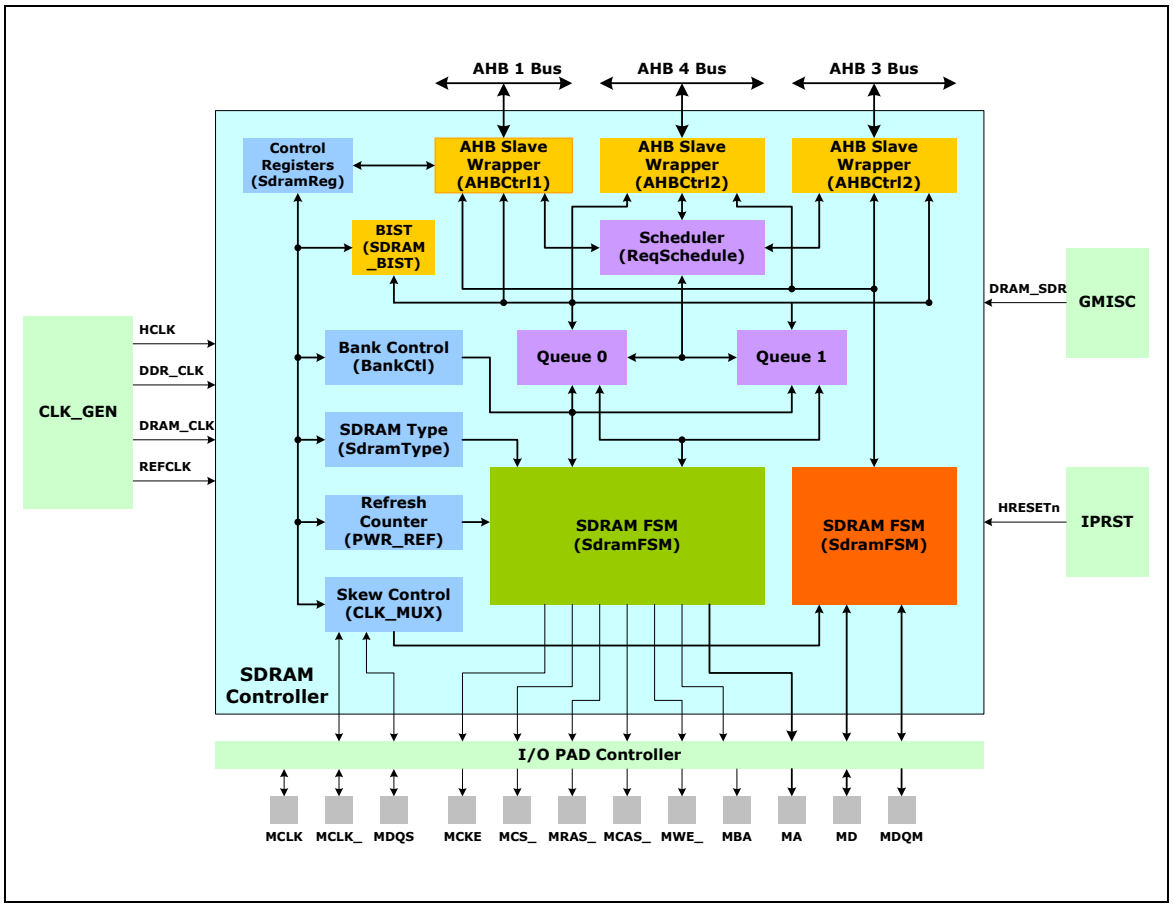



Figure 5.5-1 SDRAM Controller Block Diagram

### 5.5.4 Basic Configuration

Before using SDRAM, please refer to SDRAM Power-Up Sequence section to initial SDRAM. After completing the SDRAM power-up sequence, the SDRAM could be access correctly.

### 5.5.5 Functional Description

#### 5.5.5.1 *SDRAM Control Timing*

The SDIC supports programmable CAS Latency and Refresh Rate control. It also can control the SDRAM to enter self-refresh mode to reduce the power consumption in power-down mode.

The SDIC provides the fixed sequential burst type and burst length is 4. In addition, SDIC implements some programmable controls for the SDRAM operations:

- Configurable SDRAM Type to support DDR, DDR2 and LPDDR SDRAM.
- Configurable SDRAM Size to support 16Mbits, 64Mbits, 128Mbits, 256Mbits, 512Mbits and 1Gbits SDRAM with 16 bits data width.
- Configurable SDRAM Timing to adjust tWR, tRP, tRCD, tRAS, tRFC, tXSR, tRC, tRRD and tWTR timings.
- Configurable SDRAM Read Latency from 2 to 4 clocks.
- Configurable SDRAM Refresh timing for Auto Refresh or power save mode Self Refresh.

5.5.5.2 *SDRAM Power-Up Sequence*

Before the SDRAM can be accessed for after power on, or when exiting deep power-down mode, an SDRAM device must be initialized by software to progress an initialization sequence.

Because the DDR, DDR2 and LPDDR SDRAM require different initialization sequences and different parameters, the sequence is driven by software manually by using the registers SDIC_CMD, SDIC_MR, SDIC_EMR, SDIC_EMR2 and SDIC_EMR3.

**DDR initialization sequence**

1. Wait for 200us after power up.
2. Set the SDRAM type to DDR. This is accomplished by writing 10 to SD_TYPE (SDIC_OPMCTL[6:5]).
3. Set the SDRAM controller in initialization state. This is accomplished by writing 1 to InitState (SDIC_CMD[0]).
4. Set the CKE_H (SDIC_CMD[1]) to be 1 to force the CKE at high state.
5. Apply a PRECHARGE ALL command. This is accomplished by writing 1 to PALL_CMD (SDIC_CMD[2]). The PALL_CMD bit will auto clear after the PRECHARGE command completed.
6. Apply a MRS (Mode Register Set) command to EMR (Extended Mode Register) to enable DLL. This is accomplished by writing appropriate value to the register SDIC_EMR (SDRAM EXTEND MODE Register).
7. Apply a MRS (Mode Register Set) command to MR (Mode Register) with A8 high to set DDR SDRAM in normal operation with resetting the DLL. This is accomplished by writing appropriate value with bit [8] high to the register SDIC_MR (SDRAM MODE Register).
8. Apply a PRECHARGE ALL command. This is accomplished by writing 1 to PALL_CMD (SDIC_CMD[2]). The PALL_CMD bit will auto clear after the PRECHARGE command completed.
9. Apply two or more AUTOREFRESH commands. This is accomplished by writing 1 to REF_CMD (SDIC_CMD[3]) twice or more. The REF_CMD is auto cleared after SDRAM controller completes each CAS-BEFORE-RAS refresh command.
10. Apply a MRS (Mode Register Set) command to MR (Mode Register) with A8 low to set DDR SDRAM in normal operation without resetting the DLL. This is accomplished by writing appropriate value with bit [8] low to the register SDIC_MR (SDRAM MODE Register).
11. Apply 200 dummy clocks to meet minimum latency delay between MRS and normal operation command (ACTIVE, READ, WRITE …). This is accomplished by inserting a period of delay.
12. SDRAM initialization sequence completed and SDRAM controller exit initialization state and enter normal operating mode. This is accomplished by writing 0 to both InitState (SDIC_CMD[0]) and CKE_H (SDIC_CMD[1]).

**DDR2 initialization sequence**

1. Wait for 200us after power up.
2. Apply NOP or DESELECT commands for a minimum 400 ns.
3. Set the SDRAM type is DDR. This is accomplished by writing 11 to SD_TYPE (SDIC_OPMCTL[6:5]).

4. Set the SDRAM controller in initialization state. This is accomplished by writing 1 to InitState (SDIC_CMD[0]).
5. Set the CKE_H (SDIC_CMD[1]) to be 1 to force the CKE at high state.
6. Apply a PRECHARGE ALL command. This is accomplished by writing 1 to PALL_CMD (SDIC_CMD[2]). The PALL_CMD bit will auto clear after the PRECHARGE command completed.
7. Apply a MRS (Mode Register Set) command to EMR2 (Extended Mode Register 2). This is accomplished by writing appropriate value to the register SDIC_EMR2 (SDRAM EXTEND MODE Register 2).
8. Apply a MRS (Mode Register Set) command to EMR3 (Extended Mode Register 3). This is accomplished by writing appropriate value to the register SDIC_EMR3 (SDRAM EXTEND MODE Register 3).
9. Apply a MRS (Mode Register Set) command to EMR (Extended Mode Register) to enable DLL. This is accomplished by writing appropriate value to the register SDIC_EMR (SDRAM EXTEND MODE Register).
10. Apply a MRS (Mode Register Set) command to MR (Mode Register) with A8 high to set DDR SDRAM in normal operation with resetting the DLL. This is accomplished by writing appropriate value with bit [8] high to the register SDIC_MR (SDRAM MODE Register).
11. Apply a PRECHARGE ALL command. This is accomplished by writing 1 to PALL_CMD (SDIC_CMD[2]). The PALL_CMD bit will auto clear after the PRECHARGE command completed.
12. Apply two or more AUTOREFRESH commands. This is accomplished by writing 1 to REF_CMD (SDIC_CMD[3]) twice or more. The REF_CMD is auto cleared after SDRAM controller completes each CAS-BEFORE-RAS refresh command.
13. Apply a MRS (Mode Register Set) command to MR (Mode Register) with A8 low to set DDR SDRAM in normal operation without resetting the DLL. This is accomplished by writing appropriate value with bit [8] low to the register SDIC_MR (SDRAM MODE Register).
14. Apply a MRS (Mode Register Set) command to EMR (Extended Mode Register) to enable OCD default state. This is accomplished by writing appropriate value with 3is accomplished by to the register SDIC_EMR (SDRAM EXTEND MODE Register).
15. Apply a MRS (Mode Register Set) command to EMR (Extended Mode Register) to enable OCD exit state. This is accomplished by writing appropriate value with 3is accomplished by to the register SDIC_EMR (SDRAM EXTEND MODE Register).
16. Apply 200 dummy clocks to meet minimum latency delay between MRS and normal operation command (ACTIVE, READ, WRITE…). This is accomplished by inserting a period of delay..
17. SDRAM initialization sequence completed and SDRAM controller exit initialization state and enter normal operating mode. This is accomplished by writing 0 to both InitState (SDIC_CMD[0]) and CKE_H (SDIC_CMD[1]).

#### 5.5.5.3 *System Memory Address and SDRAM Address Mapping*

The table shown below indicates how the 32-bit system memory address be mapped to SDRAM address. All the SDRAM devices listed below are 16-bit data bus width.

**For DDR SDRAM**

| Type | R X C | R/C | BA1 | BA0 | A13 | A12 | A11 | A10 | A09 | A08 | A07 | A06 | A05 | A04 | A03 | A02 | A01 | A00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 128M 8Mx16 | 12x9 | R | 11 | 10 | | | 23 | 12 | 13 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 |
| | | C | | | | | | AP | | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 256M 16Mx16 | 13x9 | R | 11 | 10 | | 24 | 23 | 12 | 13 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 |
| | | C | | | | | | AP | | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 512M 32Mx16 | 13x10 | R | 12 | 11 | | 25 | 23 | 24 | 13 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 |
| | | C | | | | | | AP | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 1G 64Mx16 | 14x10 | R | 12 | 11 | 26 | 25 | 23 | 24 | 13 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 |
| | | C | | | | | | AP | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

**For DDR2 SDRAM**

| Type | R X C | R/C | BA2 | BA1 | BA0 | A12 | A11 | A10 | A09 | A08 | A07 | A06 | A05 | A04 | A03 | A02 | A01 | A00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 128M 8Mx16 | 12x9 | R | | 11 | 10 | | 23 | 12 | 13 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 |
| | | C | | | | | | AP | | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 256M 16Mx16 | 13x9 | R | | 11 | 10 | 24 | 23 | 12 | 13 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 |
| | | C | | | | | | AP | | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 512M 32Mx16 | 13x10 | R | | 12 | 11 | 25 | 23 | 24 | 13 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 |
| | | C | | | | | | AP | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 1G 64Mx16 | 13x10 | R | 13 | 12 | 11 | 26 | 23 | 25 | 24 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 |
| | | C | | | | | | AP | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

**Note:** The AHB bus address HADDR prefixes have been omitted on the following tables.

A13 ~ A00 are the Address pins of the SDRAM interface.

BA2, BA1 and BA0 are the Bank Selected Signal of SDRAM.

### 5.5.6 Register Map

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDIC Base Address:**<br>**SDIC_BA = 0xB000_1800** | | | | |
| **SDIC_OPMCTL** | SDIC_BA + 0x000 | R/W | SDRAM Controller Operation Mode Control Register | 0x0003_04x6 |
| **SDIC_CMD** | SDIC_BA + 0x004 | R/W | SDRAM Command Register | 0x0000_0021 |
| **SDIC_REFCTL** | SDIC_BA + 0x008 | R/W | SDRAM Controller Refresh Control Register | 0x0000_80FF |
| **SDIC_SIZE0** | SDIC_BA + 0x010 | R/W | SDRAM 0 Size Register | 0x0000_000X |
| **SDIC_SIZE1** | SDIC_BA + 0x014 | R/W | SDRAM 1 Size Register | 0x1000_0000 |
| **SDIC_MR** | SDIC_BA + 0x018 | R/W | SDRAM Mode Register | 0x0000_0032 |
| **SDIC_EMR** | SDIC_BA + 0x01C | R/W | SDRAM Extended Mode Register | 0x0000_4000 |
| **SDIC_EMR2** | SDIC_BA + 0x020 | R/W | SDRAM Extended Mode Register 2 | 0x0000_8000 |
| **SDIC_EMR3** | SDIC_BA + 0x024 | R/W | SDRAM Extended Mode Register 3 | 0x0000_C000 |
| **SDIC_TIME** | SDIC_BA + 0x028 | R/W | SDRAM Timing Control Register | 0x2BDE_9649 |
| **SDIC_DQSODS** | SDIC_BA + 0x030 | R/W | DQS Output Delay Selection Register | 0x0000_1010 |
| **SDIC_CKDQSDS** | SDIC_BA + 0x034 | R/W | Clock and DQS Delay Selection Register | 0x0044_4400 |
| **SDIC_DAENSEL** | SDIC_BA + 0x038 | R/W | Data Latch Enable Selection Register | 0x0000_0000 |

### 5.5.7 Register Description

### SDRAM Controller Operation Mode Control Register (SDIC_OPMCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDIC_OPMCTL | SDIC_BA + 0x000 | R/W | SDRAM Controller Operation Mode Control Register | 0x0003_04x6 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | RD2WR_CTL | OEDelay | LowFreq | PreActBnk | AutoPDn |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | RDBUFTH | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | SD_TYPE | | PchMode | OPMode | MCLKMode | DRAM_EN | Reserved |

| Bits | Description | |
|---|---|---|
| [31:19] | **Reserved** | Reserved. |
| [20] | **RD2WR_CTL** | **Read-to-write Turn Around Control**<br>This bit is to insert one more turn around cycle between memory read and memory write access to SDRA'M device.<br>0 = Default turn-around cycle is used.<br>1 = One more turn-around cycle is inserted between memory read and write access. |
| [19] | **OEDelay** | **Output Enable Delay Half MCLK**<br>This bit control the data output enable signal. If set high, the data output enable will be turned off half MCLK earlier.<br>0 = Default data output enable timing.<br>1 = Turn off data output enable half MCLK earlier. |
| [18] | **LowFreq** | **Low Frequency Mode**<br>For low power DDR (LPDDR) SDRAM, the valid read data outputted by LPDDR SDRAM is not ready at clock edge. If this bit is enabled, the SDRAM controller will sampled read data based on the following timing:<br>If CL is 2, the read data output latency will be 2*tCK+tAC.<br>If CL is 3, the read data output latency will be tCK+tAC.<br>CL: CAS Latency.<br>tCK: Clock cycle time for LPDDR SDRAM.<br>tAC: Data output latency from clock for LPDDR SDRAM.<br>This bit only takes effect when the SD_TYPE is selected in DDR or DDR2 SDRAM.<br>0 = SDRAM controller sampled read data based on the DDR/DDR2 standard. (Default)<br>1 = SDRAM controller sampled read data based on the LPDDR standard. |

<table>
<tr><td>[17]</td><td>PreActBnk</td><td>Pre-active Bank<br>If this bit is enabled, the SDRAM controller will open request bank early to get better performance. It means maybe more than one bank active and consumes more power.<br>There are several bus requests in this chip and the SDRAM controller checks all these requests simultaneous. If request in queue access bank is different with current bank, the SDRAM controller will open the new bank early to reduce the access latency to get better performance.<br>The mode takes effect for Close-Page mode (OPMode is 0) only. In Open-Page mode, SDRAM controller always opens bank early.<br><br>0 = Disable Pre-Active-Bank mode.<br>1 = Enable Pre-Active-Bank mode. (Default)</td></tr>
<tr><td>[16]</td><td>AutoPDn</td><td>Auto Power Down Mode<br>If this bit is enabled, the SDRAM controller will make SDRAM to enter power down mode (CKE low) automatically while the memory request is stop. Otherwise, the SDRAM is in IDLE state (CKE = high).<br><br>0 = Disable auto power down mode.<br>1 = Enable auto power down mode. (Default)</td></tr>
<tr><td>[15:11]</td><td>Reserved</td><td>Reserved.</td></tr>
<tr><td>[10:8]</td><td>RDBUFTH</td><td>The AHB Read SDRAM Read Buffer Threshold Control<br>Due to the SDRAM working clock may be higher than the AHB BUS clock, the SDRAM controller contains a read buffer for each AHB interface and they are used for data pre-read. The controller read the data to buffer full or till AHB read request end. When the data buffer full, the controller stop the read request and to service another AHB request. The RDBUFTH is used to control the buffer threshold level for the SDRAM re-start the memory request. This function can minimize the redundant memory read.<br><br>
<table>
<tr><td>[10:8]</td><td>RDBUFTH</td></tr>
<tr><td>0,0,0</td><td>Reserved</td></tr>
<tr><td>others</td><td>Re-start memory read when the data buffer remain data level is equal to RDBUFTH</td></tr>
</table></td></tr>
<tr><td>[7]</td><td>Reserved</td><td>Reserved.</td></tr>
<tr><td>[6:5]</td><td>SD_TYPE</td><td>SDRAM Type<br>This file indicates which type of SDRAM is used.<br>The reset value is decided by chip’s system power-on setting.<br><br>
<table>
<tr><td>SD_TYPE[1:0]</td><td>SDRAM TYPE</td></tr>
<tr><td>00</td><td>SDR SDRAM Type.  (Single Data Rate SDRAM)</td></tr>
<tr><td>01</td><td>LPDDR SDRAM Type.</td></tr>
<tr><td>10</td><td>DDR SDRAM Type. (Double-Data-Rate SDRAM)</td></tr>
<tr><td>11</td><td>DDR2 SDRAM Type.</td></tr>
</table></td></tr>
</table>

| | | |
|---|---|---|
| [4] | **PchMode** | **Auto Pre-charge Mode**<br>This bit controls if SDRAM controller will pre-charge all active banks while there is no new memory request.<br>The SDRAM power consumption increases with the active bank number. If no new memory access request, the active bank can be pre-charge to save power, but the SDRAM controller may lose some performance.<br>0 = The SDRAM controller keeps bank active.<br>1 = Pre-charge all bank if there is no new memory request. (Default)<br>**Note:** This bit only take effect while OPMode is high. |
| [3] | **OPMode** | **Open Page Mode**<br>This bit controls if the SDRAM controller will send pre-charge command to close the active bank page after SDRAM access.<br>If this bit doesn't be enabled, the SDRAM controller will pre-charge bank after each burst read or write cycle. This could make the SDRAM consume less power.<br>If set this bit high, the state machine will keep on the bank-active state until a page missed read/write request or at a period refresh request. This makes SDRAM controller to get better performance, but SDRAM will consume more power.<br>0 = Pre-charge after each read/write request. (Default)<br>1 = No auto pre-charge. The bank page keeps in active state after read/write. |
| [2] | **MCLKMode** | **MCLK Mode**<br>This bit controls the SDRAM clock (MCLK) is always enabled, or is enabled and disabled by SDRAM controller automatically.<br>0 = The MCLK is enabled and disabled by SDRAM controller automatically. The MCLK will keep low when the SDRAM is in the power down state.<br>1 = MCLK is always enabled. (Default) |
| [1] | **SDRAM_EN** | **SDRAM Controller Enable**<br>Set this bit 0 will disable the SDRAM controller function.<br>0 = Disable the SDRAM controller.<br>1 = Enable the SDRAM controller. (Default) |
| [0] | **Reserved** | Reserved. |

## SDRAM Command Register (SDIC_CMD)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDIC_CMD | SDIC_BA + 0x004 | R/W | SDRAM Command Register | 0x0000_0021 |

| 31 | 30 | 29 | 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 | | AutoExSelfRef | SELF_REF | REF_CMD | PALL_CMD | CKE_H | InitState |

| Bits | Description | |
|---|---|---|
| [31:6] | **Reserved** | Reserved. |
| [5] | **AutoExSelfRef** | **Auto Exit Self-refresh**<br>This controls if the SDRAM will exit self refresh mode automatically while the system interrupt occurred.<br>0 = Disable auto exit self-refresh function. The SDRAM keep in self-refresh state when the interrupt occur.<br>1 = Enable auto exit self-refresh function. The SDRAM will exit self-refresh state when the interrupt occur. (Default) |
| [4] | **SELF_REF** | **Self-refresh Command**<br>Set this bit high, the SDRAM controller will make SDRAM to enter self-refresh mode. SDRAM controller will not have response to any read, write or refresh request until this bit is cleared.<br>If the bit 5 (AutoExSelfRef) is set high, this bit will be cleared automatically when the system interrupt occurred.<br>0 = SDRAM in normal operation mode. (Default)<br>1 = Set the SDRAM enter the Self Refresh power saving state. |
| [3] | **REF_CMD** | **Auto Refresh Command**<br>Set this bit high the SDRAM controller will issue an auto refresh command to SDRAM. This bit will be cleared by SDRAM controller automatically after the auto refresh command is end.<br>0 = No operation. (Default)<br>1 = Issue an auto refresh command to the SDRAM. |
| [2] | **PALL_CMD** | **Pre-charge All Bank Command**<br>Set this bit high, the SDRAM controller will issue a pre-charge all bank command to the SDRAM. This bit will be cleared by SDRAM controller automatically after the pre-chare all bank command is end<br>0 = No operation. (Default)<br>1 = Issue a pre-charge all bank command to the SDRAM. |

| | | |
|---|---|---|
| [1] | **CKE_H** | **CKE High**<br>This bit indicates the CKE is controlled by SDRAM controller state machine or always keeps high.<br>0 = Set the CKE signal in normal state and controlled by the SDRAM controller state machine. (Default)<br>1 = Set the CKE signal keep in llerateed by. |
| [0] | **InitState** | **Initial State**<br>This bit indicates if the SDRAM is in the Initialize State. When the SDRAM is in the initialize state, SDRAM controller will not accept any SDRAM read or write request.<br>The logical state of the internal circuit of the SDRAM is undefined after power on. The SDRAM must be initialized to set the SDRAM into the right operation. The SDR SDRAM, LP SDRAM, DDR SDRAM and DDR2 SDRAM have different initialization sequence, and the users must set the right sequence to initialize the SDRAM.<br>This bit is default high and means the SDRAM is not initialized yet. After the initialization, user must set this bit low to set the SDRAM controller in correct mode.<br>0 = The SDRAM is in normal state.<br>1 = The SDRAM is in initialization state, the SDRAM initialization doesn't complete yet. (Default) |

## SDRAM Controller Refresh Control Register (SDIC_REFCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDIC_REFCTL** | SDIC_BA + 0x008 | R/W | SDRAM Controller Refresh Control Register | 0x0000_80FF |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [15] | **REF_EN** | **Refresh Period Counter Enable**<br>This bit controls if the refresh period counter is enabled.<br>If refresh period counter is disabled, the SDRAM controller would never issue auto-refresh command to SDRAM automatically. However, if refresh period counter is enabled, the SDRAM controller will issue auto-refresh command to SDRAM automatically once the refresh period counter is equal to REFRATE.<br>0 = Refresh period counter is disabled.<br>1 = Refresh period counter is enabled to trigger SDRAM controller to issue auto-refresh command to SDRAM periodically. (Default) |
| [14:0] | **REFRATE** | **Refresh Count Value**<br>This field defines the period for SDRAM controller to generate the auto-refresh command to SDRAM.<br>The SDRAM controller will issue an auto-refresh cycle to SDRAM automatically for every period programmed in the REFRAT field when the REF_EN bit of is set.<br>The refresh period is calculated as Period = REFRAT / fSCLK.<br>The fSCLK is the frequency of external crystal for chip. |

## SDRAM Size Register (SDIC_SIZE)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDIC_SIZE0** | SDIC_BA + 0x010 | R/W | SDRAM 0 Size Register | 0x0000_000X |
| **SDIC_SIZE1** | SDIC_BA + 0x014 | R/W | SDRAM 1 Size Register | 0x1000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:29] | **Reserved** | Reserved. |
| [28:21] | **BASADDR** | **Base Address**<br>This field defines the memory space where the SDRAM is mapped.<br>In this chip, the SDRAM could be mapped to address 0x0000_0000 ~ 0x1fff_ffff of system memory, and shadow address on 0x8000_0000 ~ 0x9fff_ffff of system memory.<br>The minimum supported SDRAM size is 2M bytes.<br>Based on the above criteria, the bit [28:21] is used to define the base address. For example, if [28:21] is set as 0x01, the address 0 of SDRAM memory will be mapped to 0x00200000 of system memory. |
| [20:4] | **Reserved** | Reserved. |
| [3] | **BUSWD** | **SDRAM Data Bus Width**<br>This bit defines if the data bus width of SDRAM is 16 bit or 32 bit. The DDR and DDR2 type SDRAM only support 16 bit data bus width and this bit will be 1'b0.<br>In this chip, SDRAM controller only support 16 bit SDRAM. So, this bit will be fixed at 1ype SDRAM only support 16 bit data bus width and t0 = 16bits SDRAM data BUS width (Default).<br>1 = Reserved.<br>**Note:** In register SDIC_SIZE1, this field is reserved. |

| | | |
|---|---|---|
| [2:0] | **DRAMSIZE** | **Size of SDRAM Device**<br>This field indicates the size of SDRAM device.<br>The default memory size is 2MB or 16MB depend on power on setting value. If the power on setting value indicates the SDRAM type is DDR/DDR2, the default size is 16MB (8Mx16). Otherwise, the default size is 2MB (1Mx16).<br>**Note:** In register SDIC_SIZE1, this field is reserved. |

| [2:0] | | | SIZE of SDRAM （Byte） |
|---|---|---|---|
| 0 | 0 | 0 | SDRAM disable |
| 0 | 0 | 1 | 2M |
| 0 | 1 | | 4M |
| 0 | 1 | 1 | 8M |
| 1 | 0 | 0 | 16M |
| 1 | 0 | 1 | 32M |
| 1 | 1 | 0 | 64M |
| 1 | 1 | 1 | 128M |

SDRAM TYPE (Byte) table

| **SDRAM SIZE** | **DDR SDRAM** | **DDR2 SDRAM** |
|---|---|---|
| 2MB | Reserved | Reserved |
| 4MB | Reserved | Reserved |
| 8MB | Reserved | Reserved |
| 16MB | 8Mx16 (128Mbits) | 8Mx16 (128Mbits) |
| 32MB | 16Mx16 (256Mbits) | 16Mx16 (256Mbits) |
| 64MB | 32Mx16 (512Mbits) | 32Mx16 (512Mbits) |
| 128MB | 64Mx16 (1Gbits) | 64Mx16 (1Gbits) |

## SDRAM Mode Register (SDIC_MR)

The SDRAM mode registers is used to configure the Mode Register of SDRAM device. This Mode Register value will be applied to both SDRAM 0 and SDRAM 1 devices.

Write this register, the SDRAM controller will generate a Load Mode Register (LMR) command to the SDRAM device.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDIC_MR | SDIC_BA + 0x018 | R/W | SDRAM Mode Register | 0x0000_0032 |

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

| Bits | Description | |
|---|---|---|
| [31:14] | **Reserved** | Reserved. |
| [13:7] | **Configure** | **SDRAM Dependent Configuration**<br>The value of this field is SDRAM type dependent. The definition of bits in this field is different between SDR SDRAM, DDR SDRAM and DDR2 SDRAM. Please refer the SDRAM initial sequence and related SDRAM specification to know what value should be configured in this field. |

<table>
<tr>
<td>[6:4]</td>
<td>LATENCY</td>
<td>
CAS Latency<br>
This field defines the CAS latency parameter of external SDRAM device.<br>
In this chip, SDRAM controller doesn’t support the mode CAS latency is 2.5. Setting CAS latency to be 2.5 is inhibited.<br>
For DDR2 SDRAM, SDRAM controller only support CAS latency is 3 or 4. Setting CAS latency to be 5 or 6 is inhibited.<br>
<table>
<tr><th colspan="3">LATENCY</th><th>SDR</th><th>DDR</th><th>DDR400</th><th>DDR2</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Reserved</td><td>Reserved</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Reserved</td><td>Reserved</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>2</td><td>2</td><td>2</td><td>Reserved</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>3</td><td>3</td><td>Reserved</td><td>3</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>4</td><td>Reserved</td><td>Reserved</td><td>4</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Reserved</td><td>Reserved</td><td>Reserved</td><td>5 (Inhibit)</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Reserved</td><td>2.5 (Inhibit)</td><td>2.5 (Inhibit)</td><td>6 (Inhibit)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Reserved</td><td>Reserved</td><td>Reserved</td><td>Reserved</td></tr>
</table>
The CAS latency setting listed above is for reference. Before configuring SDRAM CAS latency, it5 or 6 is inhibited.irm the CAS latency value supported by SDRAM device.
</td>
</tr>
<tr>
<td>[3]</td>
<td>BrstType</td>
<td>
Burst Type<br>
This bit indicates the burst type of SDRAM device is sequential or interleaved.<br>
In this chip, the SDRAM controller only support sequential burst type and this bit will be fixed at 1 6 is0 = Sequential burst type. (Default)<br>
1 = Reserved.
</td>
</tr>
</table>

| | | |
|---|---|---|
| [2:0] | **BrstLength** | **Burst Length**<br>This field defines the burst length of external SDRAM device.<br>SDRAM controller only supports the burst length 4. Setting burst length to be other value is inhibited. |

| Burst Length | | | SDR | DDR | DDR2 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 (Inhibit) | Reserved | Reserved |
| 0 | 0 | 1 | 2 (Inhibit) | 2 (Inhibit) | Reserved |
| 0 | 1 | 0 | 4 | 4 | 4 |
| 0 | 1 | 1 | 8 (Inhibit) | 8 (Inhibit) | 8 (Inhibit) |
| 1 | 0 | 0 | Reserved | Reserved | Reserved |
| 1 | 0 | 1 | Reserved | Reserved | Reserved |
| 1 | 1 | 0 | Reserved | Reserved | Reserved |
| 1 | 1 | 1 | Full Page | Reserved | Reserved |

## SDRAM Extended Mode Register (SDIC_EMR)

The SDRAM Extended Mode Register is used to configure SDRAM Extend Mode Register. This Extended Mode Register value will be applied to both SDRAM 0 and SDRAM 1 devices.

Write this register, the SDRAM controller will generate a Load Mode Register (LMR) command to the SDRAM. This Extended Mode Register is only used for DDR and DDR2 SDRAM.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDIC_EMR | SDIC_BA + 0x01C | R/W | SDRAM Extended Mode Register | 0x0000_4000 |

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

| Bits | Description | |
|---|---|---|
| [31:14] | **Reserved** | Reserved. |
| [13:2] | **Configure** | **SDRAM Dependent Configuration**<br>The value of this field is SDRAM type dependent. The definition of bits in this field is different between DDR SDRAM and DDR2 SDRAM. Please refer the SDRAM initial sequence and related SDRAM specification to know what value should be configured in this field. |
| [1] | **DrvStrength** | **Output Drive Strength**<br>This bit sets the SDRAM output drive strength.<br>0 = Normal drive strength.<br>1 = Reduced drive strength. |
| [0] | **DLLEN** | **DLL Enable**<br>This bit is to enable or disable the DLL of SDRAM device.<br>0 = Enable DLL of SDRAM device.<br>1 = Disable DLL of SDRAM device. |

### SDRAM Extended Mode Register 2 (SDIC_EMR2)

The SDRAM Extended Mode Register 2 is used to configure SDRAM Extend Mode Register 2. This Extended Mode Register 2 value will be applied to both SDRAM 0 and SDRAM 1 devices.

Write this register, the SDRAM controller will generate a Load Mode Register (LMR) command to the SDRAM. This Extended Mode Register is only used for DDR2 SDRAM.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDIC_EMR2 | SDIC_BA + 0x020 | R/W | SDRAM Extended Mode Register 2 | 0x0000_8000 |

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

| Bits | Description | |
|---|---|---|
| [31:14] | **Reserved** | Reserved. |
| [13:0] | **Configure** | **SDRAM Dependent Configuration**<br>The value of this field is SDRAM type dependent and only available for DDR2 SDRAM.<br>Please refer the SDRAM initial sequence and related SDRAM specification to know what value should be configured in this field. |

## SDRAM Extended Mode Register 3 (SDIC_EMR3)

The SDRAM Extended Mode Register 3 is used to configure SDRAM Extend Mode Register 3. This Extended Mode Register 3 value will be applied to both SDRAM 0 and SDRAM 1 devices.

Write this register, the SDRAM controller will generate a Load Mode Register (LMR) command to the SDRAM. This Extended Mode Register is only used for DDR2 SDRAM.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDIC_EMR3** | SDIC_BA + 0x024 | R/W | SDRAM Extended Mode Register 3 | 0x0000_C000 |

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

| Bits | Description | |
|---|---|---|
| [31:18] | **Reserved** | Reserved. |
| [17:15] | **MR_DEF** | **Mode Register Definition**<br>For Extended Mode Register 3, this field is fixed at 3AM.th S |
| [14:0] | **Configure** | **SDRAM Dependent Configuration**<br>The value of this field is SDRAM type dependent and only available for DDR2 SDRAM.<br>Please refer the SDRAM initial sequence and related SDRAM specification to know what value should be configured in this field. |

## SDRAM Timing Control Register (SDIC_TIME)

This timing control register defines some SDRAM timing parameters that should be followed during SDRAM access. These timing parameters are SDRAM dependent. Refer SDRAM devicend related SDRAM specification to know what value should be c

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDIC_TIME | SDIC_BA + 0x028 | R/W | SDRAM Timing Control Register | 0x2BDE_9649 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | tWTR | | tRRD | | tRC | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| tRC | | tXSR | | | | | tRFC |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| tRFC | | | | tRAS | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| tRCD | | | tRP | | | tWR | |

| Bits | Description | |
|---|---|---|
| [31] | **Reserved** | Reserved. |
| [30:29] | **tWTR** | **Internal Write to Read Command Delay**<br>This timing defines the minimum delay latency from last write data to next new valid READ command and only takes effect while SDRAM type is DDR or DDR2.<br>tWTR = tHCLK * (tWTR+1).<br><br>HCLK: It's the operating clock of SDRAM controller. |
| [28:27] | **tRRD** | **Active Bank a to Active Bank B Command Delay**<br>This timing defines the minimum delay latency between SDRAM bank a ACTIVE command to SDRAM bank B ACTIVE command.<br>tRRD = tHCLK * (tRRD+1).<br><br>HCLK: It's the operating clock of SDRAM controller. |
| [26:22] | **tRC** | **Active to Active Command Delay**<br>This timing defines the minimum delay latency between two ACTIVE commands.<br>tRC = tHCLK * (tRC+1).<br><br>HCLK: It's the operating clock of SDRAM controller. |
| [21:17] | **tXSR** | **Exit SELF REFRESH to ACTIVE Command Delay**<br>This timing defines the minimum delay latency from SDRAM exiting self refresh mode to next valid ACTIVE command.<br>tXSR = tHCLK * (tXSR+1).<br><br>HCLK: It's the operating clock of SDRAM controller. |

| | | |
|---|---|---|
| [16:12] | **tRFC** | **AUTO REFRESH Period**<br>This timing defines the minimum delay latency from AUTO-REFRESH command to any other command.<br>tRFC = tHCLK * (tRFC+1).<br><br>HCLK: It's the operating clock of SDRAM controller. |
| [11:8] | **tRAS** | **ACTIVE to PRECHARGE Command Delay**<br>This timing defines the minimum delay latency from a valid ACTIVE command to PRECHARGE command.<br>tRAS = tHCLK * (tRAS+1).<br><br>HCLK: It's the operating clock of SDRAM controller. |
| [7:5] | **tRCD** | **Active to READ or WRITE Delay**<br>This timing defines the minimum delay latency from a ACTIVE command to READ or WRITE command.<br>tRCD = tHCLK * (tRCD+1).<br><br>HCLK: It's the operating clock of SDRAM controller. |
| [4:2] | **tRP** | **PRECHARGE Command Period**<br>This timing defines the minimum delay latency from PRECHARGE command to any other command.<br>tRP = tHCLK * (tRP+1).<br><br>HCLK: It's the operating clock of SDRAM controller. |
| [1:0] | **tWR** | **WRITE Recovery Time**<br>This timing defines the minimum delay latency from last valid write data to PRECHARGE command.<br>tWR = tHCLK * (tWR+1).<br><br>HCLK: It's the operating clock of SDRAM controller. |

## DQS Output Delay Selection Register (SDIC_DQSODS)

This register controls the DQS output delay and source selection circuit for DQS0 and DQS1 output generation. This control register only takes effect while SDRAM type is DDR or DDR2. The function equivalent circuit for DQS output delay selection is shown below. There are two same circuits in the SDRAM controller. One is for DQS0 generation and the other is for DQS1 generation.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDIC_DQSODS | SDIC_BA + 0x030 | R/W | DQS Output Delay Selection Register | 0x0000_1010 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | DQSInvEn | DQS1_ODS | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | DQS0_ODS | | | | |

| Bits | Description | |
|---|---|---|
| [31:14] | **Reserved** | Reserved. |
| [13] | **DQSInvEn** | **DQS Invert Enable**<br>This bit controls if the clock DRAM_CLK or DDR_CLK/2 is inverted for DQS0 output generation while DQS0_ODS [3:0] is 4'b0000. This control bit takes the same effect for DQS1 output generation while DQS1_ODS [11:8] is 4'b0000.<br><br>0 = DRAM_CLK and DDR_CLK/2 is not inverted for DQS0/DQS1 output generation. (Default)<br>1 = DRAM_CLK and DDR_CLK/2 is inverted for DQS0/DQS1 output generation. |

<table>
<tr><td>[12:8]</td><td>DQS1_ODS</td><td>DQS1 Output Delay Selection<br>This field controls the DQS1 output delay value and source selection circuit for DQS1 output generation. The brief circuit for this function is listed below.<br><br>
<table>
<tr><td>DQS1_ODS</td><td>DQS1 Output</td></tr>
<tr><td>5’b0_0000</td><td>The DQS1 is generated from DRAM_CLK.</td></tr>
<tr><td>5’b0_0001<br>~<br>5’b0_1111</td><td>The DQS1 is generated from DRAM_CLK with a delay value. This delay value is controlled by the following equation:<br>DQS1 delay = DQS1_ODS[11:8] * DelayCLKMUX.<br><br>DelayCLKMUX: It’s the gate delay of a CLKMUX gate.</td></tr>
<tr><td>5’b1_0000</td><td>The DQS1 is generated from DDR_CLK/2.</td></tr>
<tr><td>5’b1_0001<br>~<br>5’b1_1111</td><td>The DQS1 is generated from DDR_CLK/2 with a delay value. This delay value is controlled by the following equation:<br>DQS1 delay = DQS1_ODS[11:8] * DelayCLKMUX.<br><br>DelayCLKMUX: It’s the gate delay of a CLKMUX gate.</td></tr>
</table></td></tr>
<tr><td>[7:5]</td><td>Reserved</td><td>Reserved.</td></tr>
<tr><td>[4:0]</td><td>DQS0_ODS</td><td>DQS0 Output Delay Selection<br>This field controls the DQS0 output delay value and source selection circuit for DQS0 output generation. The brief circuit for this function is listed below.<br><br>
<table>
<tr><td>DQS0_ODS</td><td>DQS0 Output</td></tr>
<tr><td>5’b0_0000</td><td>The DQS0 is generated from DRAM_CLK.</td></tr>
<tr><td>5’b0_0001<br>~<br>5’b0_1111</td><td>The DQS0 is generated from DRAM_CLK with a delay value. This delay value is controlled by the following equation:<br>DQS1 delay = DQS0_ODS[3:0] * DelayCLKMUX.<br><br>DelayCLKMUX: It’s the gate delay of a CLKMUX gate.</td></tr>
<tr><td>5’b1_0000</td><td>The DQS0 is generated from DDR_CLK/2.</td></tr>
<tr><td>5’b1_0001<br>~<br>5’b1_1111</td><td>The DQS1 is generated from DDR_CLK/2 with a delay value. This delay value is controlled by the following equation:<br>DQS1 delay = DQS0_ODS[3:0] * DelayCLKMUX.<br><br>DelayCLKMUX: It’s the gate delay of a CLKMUX gate.</td></tr>
</table></td></tr>
</table>

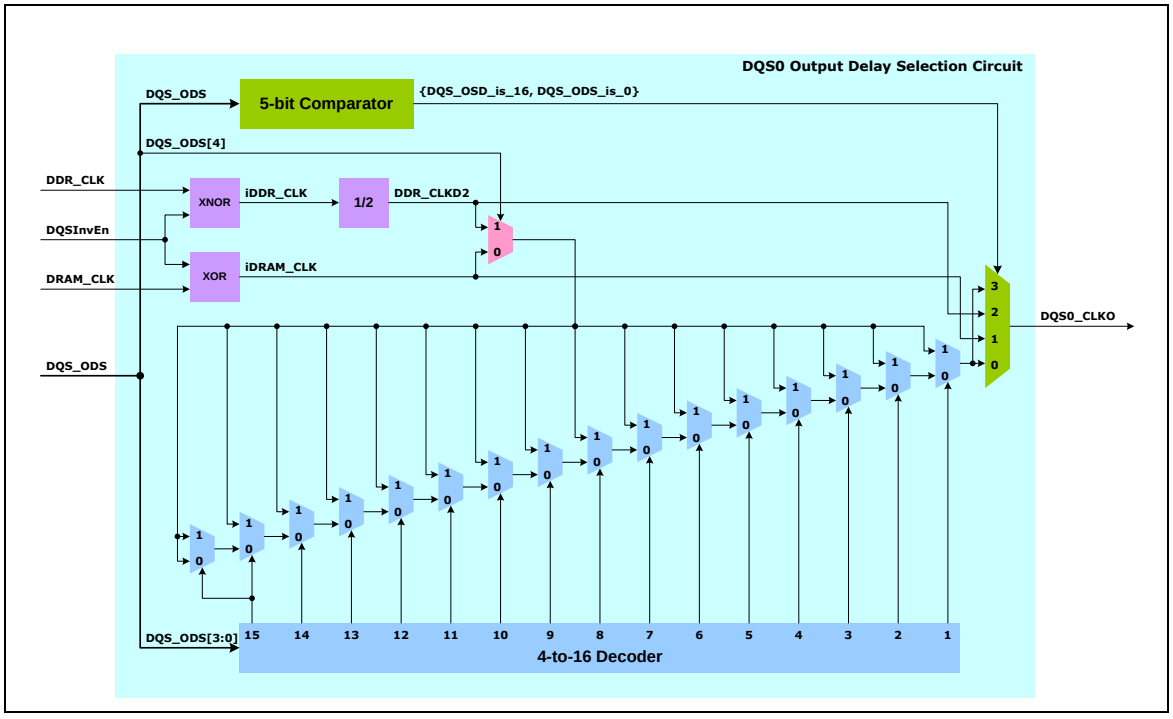



Figure 5.5-2 Clock Delay Circuit

## Clock and DQS Delay Selection Register (SDIC_CKDQSDS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDIC_CKDQSDS | SDIC_BA + 0x034 | R/W | Clock and DQS Delay Selection Register | 0x0044_4400 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DQS1_DS1 | | | | DQS1_DS0 | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| DQS0_DS1 | | | | DQS0_DS0 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DCLK_DS | | | | DCLKSrcSel | MCLK_ODS | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:20] | **DQS1_DS1** | **DQS1 Input Delay Selection 1**<br>This field controls the DQS1 input delay selection circuit to generate a clock signal DQS11_CLKIn. DQS11_CLKIn is used to sample the data bits [15:12] outputted by SDRAM device. This field only takes effect while the SDRAM type is DDR or DDR2.<br>This delay value is controlled by the following equation:<br>DQS11_CLKIn delay = DQS1_DS1 * DelayCLKMUX.<br><br>DelayCLKMUX: It's the gate delay of a CLKMUX gate. |
| [19:16] | **DQS1_DS0** | **DQS1 Input Delay Selection 0**<br>This field controls the DQS1 input delay selection circuit to generate a clock signal DQS10_CLKIn. DQS10_CLKIn is used to sample the data bits [11:8] outputted by SDRAM device. This field only takes effect while the SDRAM type is DDR or DDR2.<br>This delay value is controlled by the following equation:<br>DQS10_CLKIn delay = DQS1_DS0 * DelayCLKMUX.<br><br>DelayCLKMUX: It's the gate delay of a CLKMUX gate. |
| [15:12] | **DQS0_DS1** | **DQS0 Input Delay Selection 1**<br>This field controls the DQS0 input delay selection circuit to generate a clock signal DQS01_CLKIn. DQS01_CLKIn is used to sample the data bits [7:4] outputted by SDRAM device. This field only takes effect while the SDRAM type is DDR or DDR2.<br>This delay value is controlled by the following equation:<br>DQS01_CLKIn delay = DQS0_DS1 * DelayCLKMUX.<br><br>DelayCLKMUX: It's the gate delay of a CLKMUX gate. |

| | | |
|---|---|---|
| [11:8] | **DQS0_DS0** | **DQS0 Input Delay Selection 0**<br>This field controls the DQS0 input delay selection circuit to generate a clock signal DQS00_CLKIn. DQS00_CLKIn is used to sample the data bits [3:0] outputted by SDRAM device. This field only takes effect while the SDRAM type is DDR or DDR2.<br>This delay value is controlled by the following equation:<br>DQS00_CLKIn delay = DQS0_DS0 * DelayCLKMUX.<br><br>DelayCLKMUX: It's the gate delay of a CLKMUX gate. |
| [7:5] | **DCLK_DS** | **Data Clock Delay Selection**<br>This field controls the delay selection circuit to generate a clock signal DataCLK. If SDRAM type is DDR or DDR2, the DataCLK is used to sample the data registered by {DQS11_CLKIn, DQS10_CLKIn, DQS01_CLKIn, DQS00_CLKIn}. Or, the DataCLK is used to sample the data outputted by SDRAM device.<br>This control field only takes effect while DCLKSrcSel is set low.<br>The delay value is controlled by the following equation:<br>DataCLK delay = DCLK_DS * DelayCLKMUX.<br><br>DelayCLKMUX: It's the gate delay of a CLKMUX gate. |
| [4] | **DCLKSrcSel** | **Data Clock Source Selection**<br>This bit controls if the DataCLK source is from HCLK or MCLK (from SDRAM clock MCLK I/O buffer).<br><br>0 = DataCLK is from MCLK. (Default)<br>1 = DataCLK is from HCLK. |
| [3:0] | **MCLK_ODS** | **MCLK Output Delay Selection**<br>This field controls the delay selection circuit for SDRAM clock MCLK generation.<br>The delay value is controlled by the following equation:<br>MCLK delay = MCLK_ODS * DelayCLKMUX.<br><br>DelayCLKMUX: It's the gate delay of a CLKMUX gate. |

## Data Latch Enable Selection Register (SDIC_DAENSEL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDIC_DAENSEL | SDIC_BA + 0x038 | R/W | Data Latch Enable 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 |
| DALATDLY | Resrved | | | DALATDS | | | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7] | **DALATDLY** | **Data Latch Delay 1 MCLK Enable**<br>0 = Data latch delay 1 MCLK Disabled.<br>1 = Data latch delay 1 MCLK Enabled. |
| [6:4] | **Reserved** | Reserved. |
| [3:0] | **DALATDS** | **Data Latch Enable Delay Selection**<br>This field controls the delay selection circuit for enable to latch the data from SDRAM.<br>The delay value is controlled by the following equation:<br>Data Latch Enable Delay = DALATDS * DelayCLKMUX.<br><br>DelayCLKMUX: It's the gate delay of a CLKMUX gate. |

## 5.6 MTP Controller (MTP)

### 5.6.1 Overview

The MTP (Multi-Time Programmable) controller performs an easy way to use and program the 256-bit Key for IP Security Engine. There is a MTP EPROM in this chip, and it can be programmed 15 times. User can program 256-bit key and 8-bit user defined fields each time. The 256-bit key is program-only, and only can be used by IP Security Engine. User can use the 8-bit user defined field for special purpose. The MTP also supports a LOCK function to protect the content of programmed key and user defined field.

### 5.6.2 Features

- Support MTP EPROM programming
  - 256-bit Key and 8-bit user defined data field
  - Up to 15 times programming
- Support LOCK function

### 5.6.3 Block Diagram

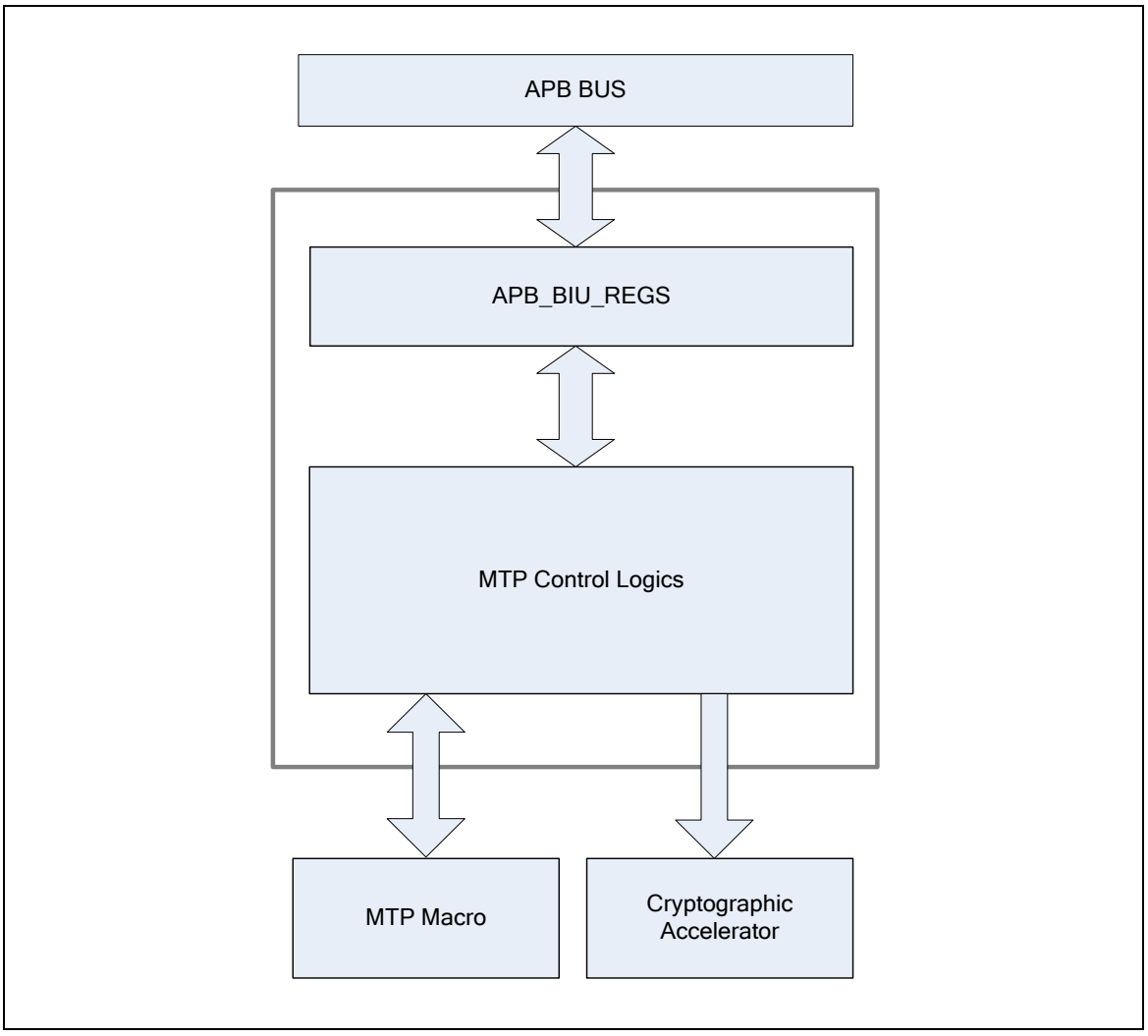



Figure 5.6-1 MTP Controller Block Diagram

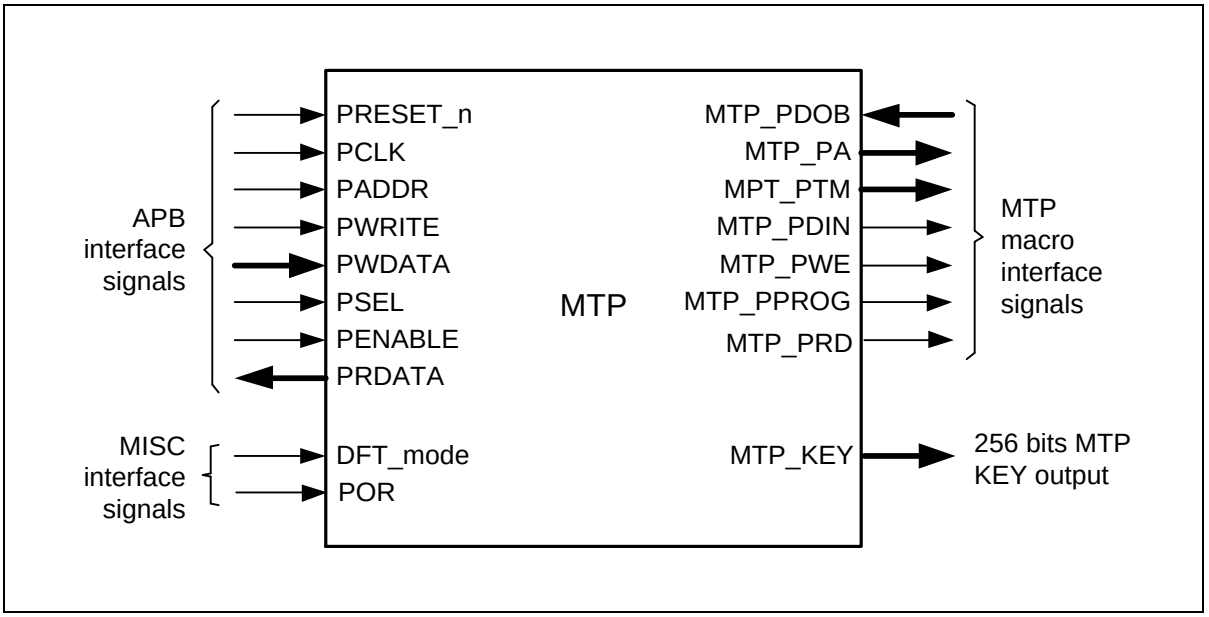



Figure 5.6-2 MTP INTERFACE PIN DESCRIPTION

### 5.6.4 Basic Configuration

Before using MTP, it's necessary to enable clock of MTP. Set MTP (CLK_PCLKEN1[26]) high to enable MTP's clock.

### 5.6.5 Functional Description

MTP provide the software control MTP macro and provide Key to the Cryptographic Accelerator. User can follow the recommended programing flow, enable MTP or write key to MTP or lock the MTP.

### 5.6.6 Software Programming Flow

- Enable MTP
  - Enable MTP IP clock (in CLK_APB)
  - Write 0x59, 0x16, 0x88 to **MTP_REGLCTL, check REGLCTL(MTP_REGLCTL[0]) show 0x1.**
  - Write 0x1 to **MTP_KEYEN** to enable MTP function.
  - Read **MTP_STATUS ,** check until **MTPEN** (**MTP_STATUS[0]**) is set
  - If **NONPRG** (**MTP_STATUS[2]**) is set, it means MTP enabled and there's no any key programmed. (**PRGCNT** is 0)
  - If **KEYVALID** (**MTP_STATUS[1]**) is set, it means the latest key programmed to MTP is enabled. (**PRGCNT** is not 0)
- Program MTP
  - Enable MTP (refer to the Enable MTP flow)
  - Write 0x2 to **MTP_CTL.**
  - Write 0x60AE to **MTP_PCYCLE (**if PCLK=75Mhz, MTP_PCYCLE=24750**)**
  - Write the 256 bits key to **MTP_KEY0 ~ MTP_KEY7**
  - Write 8 bits user defined data to **MTP_USERDATA.**
  - Write 0x1 to **PSTART (MTP_PSTART[0]).**
  - Check until **PSTART** goes back to 0.
  - If **PRGFAIL** (**MTP_STATUS[4]**) is set, program operation failed. Otherwise, program operation successes.
- Lock MTP
  - Enable MTP (refer to the Enable MTP flow)
  - Write 0x3 to **MTP_CTL**.
  - Write 0x60AE to **MTP_PCYCLE (**if PCLK=75Mhz, MTP_PCYCLE=24750**).**
  - Write 0x1 to **PSTART (MTP_PSTART[0]).**
  - Check until **PSTART** goes back to 0
  - Enable MTP (refer to the Enable MTP flow).
  - If **MTPEN** (**MTP_STATUS[0]**), **KEYVALID** (**MTP_STATUS[1]**), and **LOCKED** (**MTP_STATUS[3]**) are all set, LOCK operation successes. Otherwise, LOCK operation failed.

### 5.6.7 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **MTP Base Address:**<br>**MTP_BA = 0xB800_C000** | | | | |
| **MTP_KEYEN** | MTP_BA+0x000 | R/W | MTP Key Enable Register | 0x0000_0000 |
| **MTP_USERDATA** | MTP_BA+0x00c | R/W | MTP User Defined Data Register | 0x0000_0000 |
| **MTP_KEY0** | MTP_BA+0x010 | W | MTP KEY Value 0 Register | 0x0000_0000 |
| **MTP_KEY1** | MTP_BA+0x014 | W | MTP KEY Value 1 Register | 0x0000_0000 |
| **MTP_KEY2** | MTP_BA+0x018 | W | MTP KEY Value 2 Register | 0x0000_0000 |
| **MTP_KEY3** | MTP_BA+0x01c | W | MTP KEY Value 3 Register | 0x0000_0000 |
| **MTP_KEY4** | MTP_BA+0x020 | W | MTP KEY Value 4 Register | 0x0000_0000 |
| **MTP_KEY5** | MTP_BA+0x024 | W | MTP KEY Value 5 Register | 0x0000_0000 |
| **MTP_KEY6** | MTP_BA+0x028 | W | MTP KEY Value 6 Register | 0x0000_0000 |
| **MTP_KEY7** | MTP_BA+0x02c | W | MTP KEY Value 7 Register | 0x0000_0000 |
| **MTP_PCYCLE** | MTP_BA+0x030 | R/W | MTP Program Cycle Control Register | 0x0000_60AE |
| **MTP_CTL** | MTP_BA+0x034 | R/W | MTP Control Register | 0x0000_0000 |
| **MTP_PSTART** | MTP_BA+0x038 | R/W | MTP Program Start Register | 0x0000_0000 |
| **MTP_STATUS** | MTP_BA+0x040 | R | MTP Status Register | 0x0000_0000 |
| **MTP_REGLCTL** | MTP_BA+0x050 | R/W | MTP Register Write-Protection Control Register | 0x0000_0000 |

### 5.6.8 Register Description

### MTP KEYEN Register (MTP_KEYEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **MTP_KEYEN** | MTP_BA+0x000 | R/W | MTP Key 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** | | | | | | | **KEYEN** |

| Bits | Description | |
|---|---|---|
| [31:1] | **Reserved** | Reserved. |
| [0] | **KEYEN** | **MTP Key Enable**<br>0 = MTP Key .<br>1 = Enable MTP Key . Check MTPEN in MTP_STATUS |

## MTP User Defined Data Register (MTP_USERDATA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| MTP_USERDATA | MTP_BA+0x00c | R/W | MTP User Defined 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 |
| UDAT | | | | | | | |

| Bits | Description | |
|---|---|---|
| [7:0] | UDAT | **MTP User Defined Data Register**<br>8 bits in MTP for software use.<br>When MTP in program key mode, write this register to program user defined data into MTP.<br>S/W can read this register after enable MTP. |

### MTP Key Value Registers (MTP_KEY0 ~ MTP_KEY7)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **MTP_KEY0** | MTP_BA+0x010 | W | MTP KEY Value 0 Register | 0x0000_0000 |
| **MTP_KEY1** | MTP_BA+0x014 | W | MTP KEY Value 1 Register | 0x0000_0000 |
| **MTP_KEY2** | MTP_BA+0x018 | W | MTP KEY Value 2 Register | 0x0000_0000 |
| **MTP_KEY3** | MTP_BA+0x01c | W | MTP KEY Value 3 Register | 0x0000_0000 |
| **MTP_KEY4** | MTP_BA+0x020 | W | MTP KEY Value 4 Register | 0x0000_0000 |
| **MTP_KEY5** | MTP_BA+0x024 | W | MTP KEY Value 5 Register | 0x0000_0000 |
| **MTP_KEY6** | MTP_BA+0x028 | W | MTP KEY Value 6 Register | 0x0000_0000 |
| **MTP_KEY7** | MTP_BA+0x02c | W | MTP KEY Value 7 Register | 0x0000_0000 |

| Bits | Description | |
|---|---|---|
| [31:0] | **KEY** | **MTP KEY VALUE Register (Write-only)**<br>Write this register to program Key in MTP. |

## MTP Program Cycle Register (MTP_PCYCLE)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **MTP_PCYCLE** | MTP_BA+0x030 | R/W | MTP Program Cycle Control Register | 0x0000_60AE |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **PCYCLE** | **MTP CYCLE Register**<br>**Set the cycle counts to meet 330us for write MTP.**<br>Example: PCLK = 75MHz (13.3333ns).<br>MTP_PCYCLE = 330000/10 = 24750.<br><br>Note: Before program/lock MTP, must configure this register to meet timing. |

### MTP Mode Control Register (MTP_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **MTP_CTL** | MTP_BA+0x034 | R/W | MTP 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** | | | | | | **MODE** | |

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [1:0] | **MODE** | **MTP Mode**<br>2'b00 = MTP Idle Mode.<br>2'b01 = Reserved.<br>2'b10 = MTP Program Key Mode.<br>2'b11 = MTP Lock Key Mode. |

### MTP Program Start Register (MTP_PSTART)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **MTP_PSTART** | MTP_BA+0x038 | R/W | MTP Program 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** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | | | | | **PSTART** |

| Bits | Description | |
|---|---|---|
| [31:1] | **Reserved** | Reserved. |
| [0] | **PSTART** | **MTP Program Start Register**<br>Write 1 to start MTP Program/LOCK.<br>**Note:** this bit will auto clear after MTP program/lock finish. |

## MTP Status Register (MTP_STATUS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| MTP_STATUS | MTP_BA+0x040 | R | MTP Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | BUSY |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | PRGCNT | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | PRGFAIL | LOCKED | NONPRG | KEYVALID | MTPEN |

| Bits | Description | |
|---|---|---|
| [31:25] | **Reserved** | Reserved. |
| [24] | **BUSY** | **MTP Busy Status**<br>0 = MTP engine is idle.<br>1 = MTP engine is busy. |
| [19:16] | **PRGCNT** | **MTP Program Counts**<br>This register show the program times of MTP after read MTP.<br>Maximum program count is 15. |
| [15:5] | **Reserved** | Reserved. |
| [4] | **PRGFAIL** | **MTP Program Fail Status**<br>0 = MTP program without fail.<br>1 = MTP program fail. |
| [3] | **LOCKED** | **MTP Locked Status**<br>0 = MTP is programmable if PRGCNT < 15.<br>1 = MTP locked. Can't program key. |
| [2] | **NONPRG** | **MTP Non-program Status**<br>0 = MTP has ever been programmed.<br>1 = MTP is still not programmed. (program counts is 0) |
| [1] | **KEYVALID** | **MTP KEY Status**<br>0 = There's no Key in MTP or MTP is not enabled.<br>1 = The latest key in MTP is valid. |
| [0] | **MTPEN** | **MTP Enable Status**<br>0 = MTP is not enabled.<br>1 = MTP is enabled. |

### MTP Register Write-Protection Control Register (MTP_REGLCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **MTP_REGLCTL** | MTP_BA+0x050 | R/W | MTP Register Write-Protection 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[7:1]** | | | | | | | **REGLCTL[0]**<br>**REGPRTDIS** |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **REGLCTL** | **Register Write-protection Code (Write Only)**<br>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. |
| [0] | **REGPRTDIS** | **Register Write-protection Disable Indicator (Read Only)**<br>0 = Write-protection Enabled for writing protected registers. Any write to the protected register is ignored.<br>1 = Write-protection Disabled for writing protected registers. |

## 5.7 External Bus Interface (EBI)

### 5.7.1 Overview

This chip supports External Bus Interface (EBI), which controls the access to the external memory (SRAM) and External I/O devices. The EBI has up to 5 chip select signals to select different devices with 10-bit address bus. It supports 8-bit and 16-bit external data bus width for each bank.

### 5.7.2 Features

- Support SRAM and external I/O devices.
- Support 8/16-bit data bus width.
- Support 80 and 68 mode interface signals.
- Support up to 5 chip selects for SRAM and external I/O devices.
- Support programmable access cycle.
- Support four 32-bit write buffers.

### 5.7.3 Block Diagram

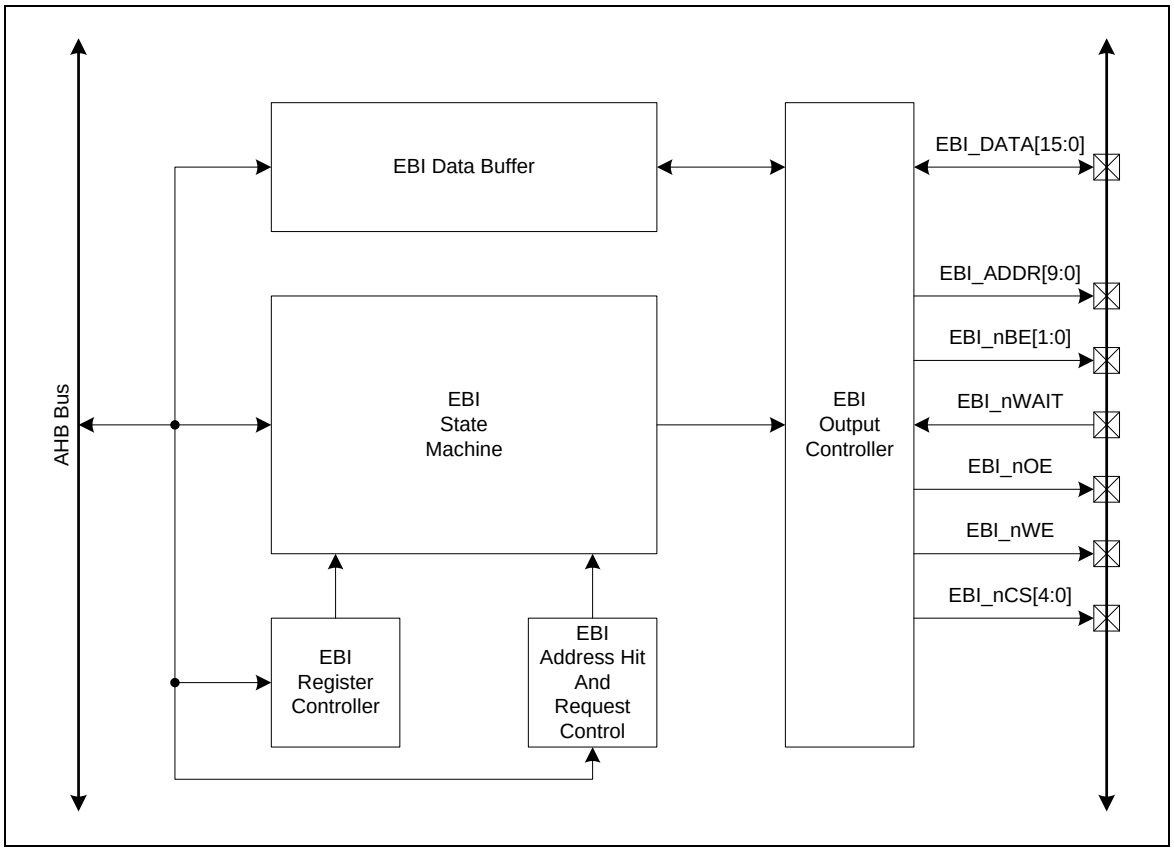



Figure 5.7-1 EBI Block Diagram

### 5.7.4 Basic Configuration

Before using External Bus Interface, it's necessary to configure related pins as the EBI function and enable EBI's clock.

For EBI related pin configuration, please refer to the register SYS_MFP_GPDH, SYS_MFP_GPHL, SYS_MFP_GPHH, SYS_MFP_GPIL and SYS_MFP_GPIH to know how to configure related pins as the EBI function.

To enable EBI's clock for operation, please set EBI (CLK_HCLKEN[9]) high.

### 5.7.5 Functional Description

#### 5.7.5.1 *EBI Memory Space*

In this chip, two system memory spaces, 1st memory space is from 0x20000000 to 0x2FFFFFFF and the other memory space is from 0xA0000000 to 0xAFFFFFFF, are defined to EBI. By programming BASADDR (EBI_BNKCTLx[31:19], x is 0, 1, 2, 3 and 4) appropriately, user could map external device to these two system memory spaces. When system request address hit the EBI's memory space, the corresponding EBI chip select assert and EBI state machine operates.

### 5.7.6 Register Map

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EBI Base Address:**<br>**EBI_BA = 0xB000_1000** | | | | |
| **EBI_CTL** | EBI_BA+0x000 | R/W | EBI Control Register | 0x0001_0001 |
| **EBI_BNKCTL0** | EBI_BA+0x018 | R/W | External Bus Bank 0 Control Register | 0x0000_0000 |
| **EBI_BNKCTL1** | EBI_BA+0x01C | R/W | External Bus Bank 1 Control Register | 0x0000_0000 |
| **EBI_BNKCTL2** | EBI_BA+0x020 | R/W | External Bus Bank 2 Control Register | 0x0000_0000 |
| **EBI_BNKCTL3** | EBI_BA+0x024 | R/W | External Bus Bank 3 Control Register | 0x0000_0000 |
| **EBI_BNKCTL4** | EBI_BA+0x028 | R/W | External Bus Bank 4 Control Register | 0x0000_0000 |

### 5.7.7 Register Description

## EBI Control Register (EBI_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EBI_CTL** | EBI_BA+0x000 | R/W | EBI Control Register | 0x0001_0001 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | **EXBE4** | **EXBE3** | **EXBE2** | **EXBE1** | **EXBE0** |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **M68E4** | **M68E3** | **M68E2** | **M68E1** | **M68E0** | **Reserved** | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | | | **WAITVT** | | **LITTLE** |

| Bits | Description | |
|---|---|---|
| [31:29] | **Reserved** | Reserved. |
| [28] | **EXBE4** | **External Bus Bank 4 Byte Enable**<br>This bit and M68E4 (EBI_CTL[23]) defines how the pins EBI_nBE1, EBI_nBE0 and EBI_nWE are used when external bus bank 4 accessed.<br>Please refer to the table shown below for detail information. |

| EXBE4 | M68E4 | Description |
|---|---|---|
| 0 | 0 | 80-type interface. Pin EBI_nBE1 and EBI_nBE0 used as byte write strobe signal. |
| 1 | 0 | 80-type interface. Pin EBI_nBE1 and EBI_nBE0 used as byte enable signals while EBI_nWE used as write strobe signal to external device. |
| 0 | 1 | 68-type interface. EBI_nCS4 pin is the enable signal, EBI_nWE used as read/write strobe signal |
| 1 | 1 | Reserved |

<table>
<tr><td>[27]</td><td>EXBE3</td><td>
External Bus Bank 3 Byte Enable<br>
This bit and M68E3(EBI_CTL[22]) defines how the pins EBI_nBE1, EBI_nBE0 and EBI_nWE are used when external bus bank 3 accessed.<br>
Please refer to the table shown below for detail information.
<table>
<tr><td>EXBE3</td><td>M68E3</td><td>Description</td></tr>
<tr><td>0</td><td>0</td><td>80-type interface. Pin EBI_nBE1 and EBI_nBE0 used as byte write strobe signal.</td></tr>
<tr><td>1</td><td>0</td><td>80-type interface. Pin EBI_nBE1 and EBI_nBE0 used as byte enable signals while EBI_nWE used as write strobe signal to external device.</td></tr>
<tr><td>0</td><td>1</td><td>68-type interface. EBI_nCS3 pin is the enable signal, EBI_nWE used as read/write strobe signal</td></tr>
<tr><td>1</td><td>1</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>[26]</td><td>EXBE2</td><td>
External Bus Bank 2 Byte Enable<br>
This bit and M68E2(EBI_CTL[21]) defines how the pins EBI_nBE1, EBI_nBE0 and EBI_nWE are used when external bus bank 2 accessed.<br>
Please refer to the table shown below for detail information.
<table>
<tr><td>EXBE2</td><td>M68E2</td><td>Description</td></tr>
<tr><td>0</td><td>0</td><td>80-type interface. Pin EBI_nBE1 and EBI_nBE0 used as byte write strobe signal.</td></tr>
<tr><td>1</td><td>0</td><td>80-type interface. Pin EBI_nBE1 and EBI_nBE0 used as byte enable signals while EBI_nWE used as write strobe signal to external device.</td></tr>
<tr><td>0</td><td>1</td><td>68-type interface. EBI_nCS2 pin is the enable signal, EBI_nWE used as read/write strobe signal</td></tr>
<tr><td>1</td><td>1</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>[25]</td><td>EXBE1</td><td>
External Bus Bank 1 Byte Enable<br>
This bit and M68E1(EBI_CTL[20]) defines how the pins EBI_nBE1, EBI_nBE0 and EBI_nWE are used when external bus bank 1 accessed.<br>
Please refer to the table shown below for detail information.
<table>
<tr><td>EXBE1</td><td>M68E1</td><td>Description</td></tr>
<tr><td>0</td><td>0</td><td>80-type interface. Pin EBI_nBE1 and EBI_nBE0 used as byte write strobe signal.</td></tr>
<tr><td>1</td><td>0</td><td>80-type interface. Pin EBI_nBE1 and EBI_nBE0 used as byte enable signals while EBI_nWE used as write strobe signal to external device.</td></tr>
<tr><td>0</td><td>1</td><td>68-type interface. EBI_nCS1 pin is the enable signal, EBI_nWE used as read/write strobe signal</td></tr>
<tr><td>1</td><td>1</td><td>Reserved</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>[24]</td><td>EXBE0</td><td>External Bus Bank 0 Byte Enable<br>This bit and M68E0(EBI_CTL[19]) defines how the pins EBI_nBE1, EBI_nBE0 and EBI_nWE are used when external bus bank 0 accessed.<br>Please refer to the table shown below for detail information.<br><table><tr><td>EXBE0</td><td>M68E0</td><td>Description</td></tr><tr><td>0</td><td>0</td><td>80-type interface. Pin EBI_nBE1 and EBI_nBE0 used as byte write strobe signal.</td></tr><tr><td>1</td><td>0</td><td>80-type interface. Pin EBI_nBE1 and EBI_nBE0 used as byte enable signals while EBI_nWE used as write strobe signal to external device.</td></tr><tr><td>0</td><td>1</td><td>68-type interface. EBI_nCS0 pin is the enable signal, EBI_nWE used as read/write strobe signal</td></tr><tr><td>1</td><td>1</td><td>Reserved</td></tr></table></td></tr>
<tr><td>[23]</td><td>M68E4</td><td>External Bus Bank 4 M68 Mode Enable<br>This bit and EXBE4 (EBI_CTL[28]) defines how the pins EBI_nBE1, EBI_nBE0 and EBI_nWE are used when external bus bank 4 accessed.<br>Please refer to the description of EXBE4 (EBI_CTL[28]) for detail information.</td></tr>
<tr><td>[22]</td><td>M68E3</td><td>External Bus Bank 3 M68 Mode Enable<br>This bit and EXBE3 (EBI_CTL[27]) defines how the pins EBI_nBE1, EBI_nBE0 and EBI_nWE are used when external bus bank 3 accessed.<br>Please refer to the description of EXBE3 (EBI_CTL[27]) for detail information.</td></tr>
<tr><td>[21]</td><td>M68E2</td><td>External Bus Bank 2 M68 Mode Enable<br>This bit and EXBE2 (EBI_CTL[26]) defines how the pins EBI_nBE1, EBI_nBE0 and EBI_nWE are used when external bus bank 2 accessed.<br>Please refer to the description of EXBE3 (EBI_CTL[26]) for detail information.</td></tr>
<tr><td>[20]</td><td>M68E1</td><td>External Bus Bank 1 M68 Mode Enable<br>This bit and EXBE1 (EBI_CTL[25]) defines how the pins EBI_nBE1, EBI_nBE0 and EBI_nWE are used when external bus bank 1 accessed.<br>Please refer to the description of EXBE1 (EBI_CTL[25]) for detail information.</td></tr>
<tr><td>[19]</td><td>M68E0</td><td>External Bus Bank 0 M68 Mode Enable<br>This bit and EXBE0 (EBI_CTL[24]) defines how the pins EBI_nBE1, EBI_nBE0 and EBI_nWE are used when external bus bank 0 accessed.<br>Please refer to the description of EXBE0 (EBI_CTL[24]) for detail information.</td></tr>
<tr><td>[18:3]</td><td>Reserved</td><td>Reserved.</td></tr>
<tr><td>[2:1]</td><td>WAITVT</td><td>Valid Time of EBI_NWAIT Signal<br>This bit recognizes the EBI_nWAIT signal at the next “nth” MCLK rising edge after the nOE or nWBE active cycle. WAITVT bits determine the n.<br>00 = 1 MCLK cycle.<br>01 = 2 MCLK cycle.<br>10 = 3 MCLK cycle.<br>11 = 4 MCLK cycle.</td></tr>
<tr><td>[0]</td><td>LITTLE</td><td>Little Endian Mode<br>This bit is read only and always read as 1.</td></tr>
</table>

## External Bus Bank Control Register (EBI_BNKCTL0 – EBI_BNKCTL4)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EBI_BNKCTL0** | EBI_BA+0x018 | R/W | External Bus Bank 0 Control Register | 0x0000_0000 |
| **EBI_BNKCTL1** | EBI_BA+0x01C | R/W | External Bus Bank 1 Control Register | 0x0000_0000 |
| **EBI_BNKCTL2** | EBI_BA+0x020 | R/W | External Bus Bank 2 Control Register | 0x0000_0000 |
| **EBI_BNKCTL3** | EBI_BA+0x024 | R/W | External Bus Bank 3 Control Register | 0x0000_0000 |
| **EBI_BNKCTL4** | EBI_BA+0x028 | R/W | External Bus Bank 4 Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| BASADDR | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BASADDR | | | | | SIZE | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| ADRS | tACC | | | | tCOH | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| tACS | | | tCOS | | | DBWD | |

| Bits | Description | |
|---|---|---|
| [31:19] | **BASADDR** | **Base Address Pointer of External I/O Bank 0~4**<br>The start address of each external I/O bank is calculated as “BASADDR” base pointer << 18. Each external I/O bank base address pointer together with the “SIZE” bits constitutes the whole address range of each external I/O bank. |
| [18:16] | **SIZE** | **The Size of the External I/O Bank 0~4**<br>000 = 256 kB.<br>Others = Reserved. |
| [15] | **ADRS** | **Address Bus Alignment for External I/O Bank 0~4**<br>When ADRS is set, EBI bus is alignment to byte address format, and ignores DBWD [1:0] setting. |

| | | |
|---|---|---|
| [14:11] | **tACC** | **Access Cycles (NOE or NWE Active Time) for External I/O Bank 0~4**<br>0000 = Reserved.<br>0001 = Access Cycles (nOE or nWE active time) is 1 MCLK.<br>0010 = Access Cycles (nOE or nWE active time) is 2 MCLK.<br>0011 = Access Cycles (nOE or nWE active time) is 3 MCLK.<br>0100 = Access Cycles (nOE or nWE active time) is 4 MCLK.<br>0101 = Access Cycles (nOE or nWE active time) is 5 MCLK.<br>0110 = Access Cycles (nOE or nWE active time) is 6 MCLK.<br>0111 = Access Cycles (nOE or nWE active time) is 7 MCLK.<br>1000 = Access Cycles (nOE or nWE active time) is 9 MCLK.<br>1001 = Access Cycles (nOE or nWE active time) is 11 MCLK.<br>1010 = Access Cycles (nOE or nWE active time) is 13 MCLK.<br>1011 = Access Cycles (nOE or nWE active time) is 15 MCLK.<br>1100 = Access Cycles (nOE or nWE active time) is 17 MCLK.<br>1101 = Access Cycles (nOE or nWE active time) is 19 MCLK.<br>1110 = Access Cycles (nOE or nWE active time) is 21 MCLK.<br>1111 = Access Cycles (nOE or nWE active time) is 23 MCLK. |
| [10:8] | **tCOH** | **Chip Selection Hold-on Time on NOE or NWBE for External I/O Bank 0~4**<br>000 = Chip Selection Hold-On Time on nOE or nWBE is 0 MCLK.<br>001 = Chip Selection Hold-On Time on nOE or nWBE is 1 MCLK.<br>010 = Chip Selection Hold-On Time on nOE or nWBE is 2 MCLK.<br>011 = Chip Selection Hold-On Time on nOE or nWBE is 3 MCLK.<br>100 = Chip Selection Hold-On Time on nOE or nWBE is 4 MCLK.<br>101 = Chip Selection Hold-On Time on nOE or nWBE is 5 MCLK.<br>110 = Chip Selection Hold-On Time on nOE or nWBE is 6 MCLK.<br>111 = Chip Selection Hold-On Time on nOE or nWBE is 7 MCLK. |
| [7:5] | **tACS** | **Address Set-up Before NECS for External I/O Bank 0~4**<br>000 = Address Set-up Before nECS is 0 MCLK.<br>001 = Address Set-up Before nECS is 1 MCLK.<br>010 = Address Set-up Before nECS is 2 MCLK.<br>011 = Address Set-up Before nECS is 3 MCLK.<br>100 = Address Set-up Before nECS is 4 MCLK.<br>101 = Address Set-up Before nECS is 5 MCLK.<br>110 = Address Set-up Before nECS is 6 MCLK.<br>111 = Address Set-up Before nECS is 7 MCLK. |
| [4:2] | **tCOS** | **Chip Selection Set-up Time on NOE or NWBE for External I/O Bank 0~4**<br>When the bank is configured, the access to its bank stretches chip selection time before the nOE or new signal is activated.<br>000 = Chip selection set-up time on nOE or nWBE is 0 MCLK.<br>001 = Chip selection set-up time on nOE or nWBE is 1 MCLK.<br>010 = Chip selection set-up time on nOE or nWBE is 2 MCLK.<br>011 = Chip selection set-up time on nOE or nWBE is 3 MCLK.<br>100 = Chip selection set-up time on nOE or nWBE is 4 MCLK.<br>101 = Chip selection set-up time on nOE or nWBE is 5 MCLK.<br>110 = Chip selection set-up time on nOE or nWBE is 6 MCLK.<br>111 = Chip selection set-up time on nOE or nWBE is 7 MCLK. |

| | | |
|---|---|---|
| [1:0] | **DBWD** | **Programmable Data Bus Width for External I/O Bank 0~4**<br>00 = Disable bus.<br>01 = 8-bit.<br>10 = 16-bit.<br>11 = Reserved. |

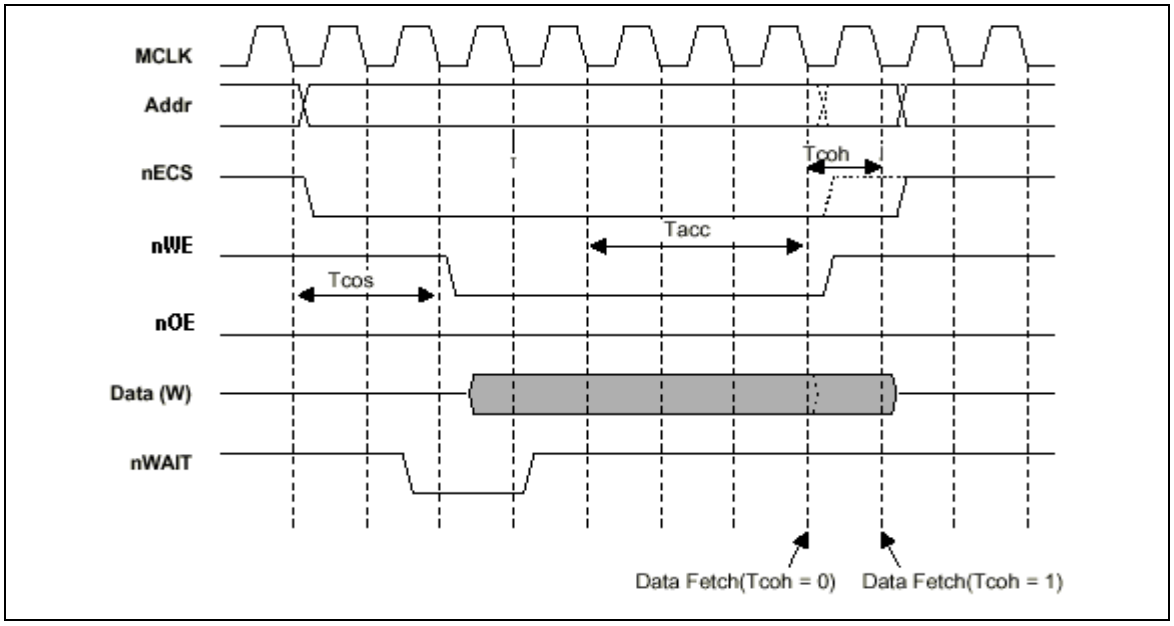



Figure 5.7-2 External I/O Write operation timing

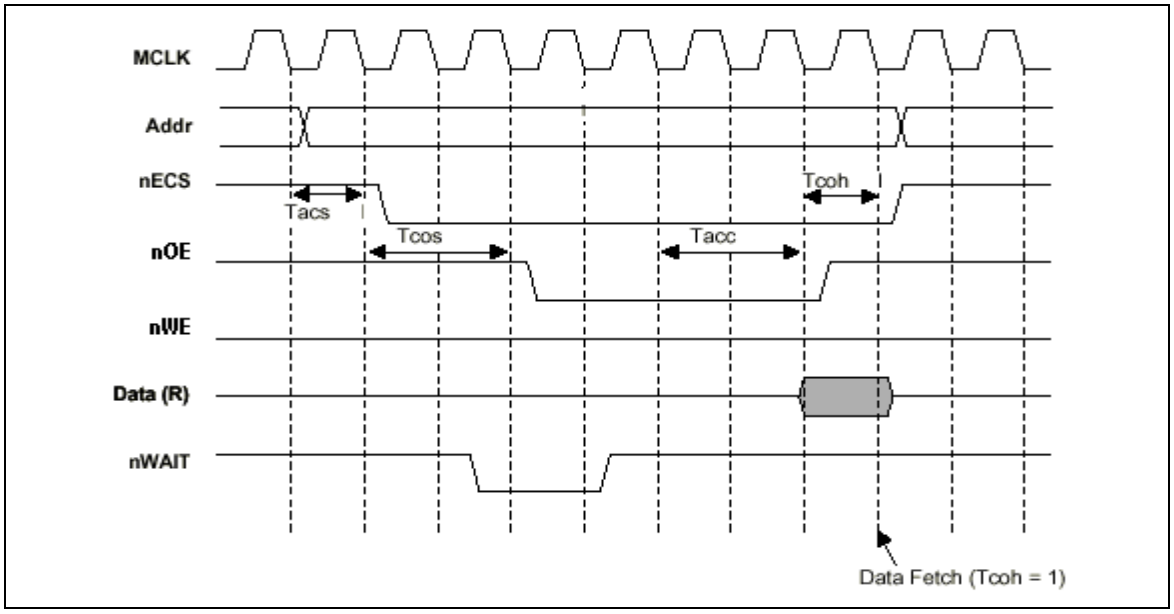



Figure 5.7-3 External I/O Read operation timing

## 5.8 General Purpose I/O (GPIO)

### 5.8.1 Overview

The NUC970 series have up to 148 General-Purpose I/O (GPIO) pins and can be shared with other function pins depending on the chip configuration. These 148 pins are arranged in 10 ports named as PA, PB, PC, PD, PE, PF, PG, PH, PI and PJ. PA, PB, PD, PE, PF, PG, PH and PI have 16 pins on port, PC has 15 pins on port and PJ has 5 pins on port. Each of the 148 I/O pins is independent and can be easily configured by user to meet various system configurations and design requirements. After reset, all 148 I/O pins are configured in General-Purpose I/O Input mode.

When any of the 148 I/O pins used as a General-Purpose I/O, its I/O type can be configured by user individually as Input or Output mode. In Input mode, the input buffer type could be selected as CMOS input buffer or Schmitt trigger input buffer. Each I/O pin also equips a pull-up resistor (45 kΩ ~ 82 kΩ) and a pull-down resistor (37 kΩ ~ 91 kΩ). The enable of pull-up/pull-down resistor is controllable.

### 5.8.2 Features

- Support input and output mode.
- Support CMOS and Schmitt trigger input buffer.
- Support controllable pull-up and pull-down resistor.
- Support both edge and level interrupt.
- Support de-bounce circuit to filter the noise.

### 5.8.3 Block Diagram

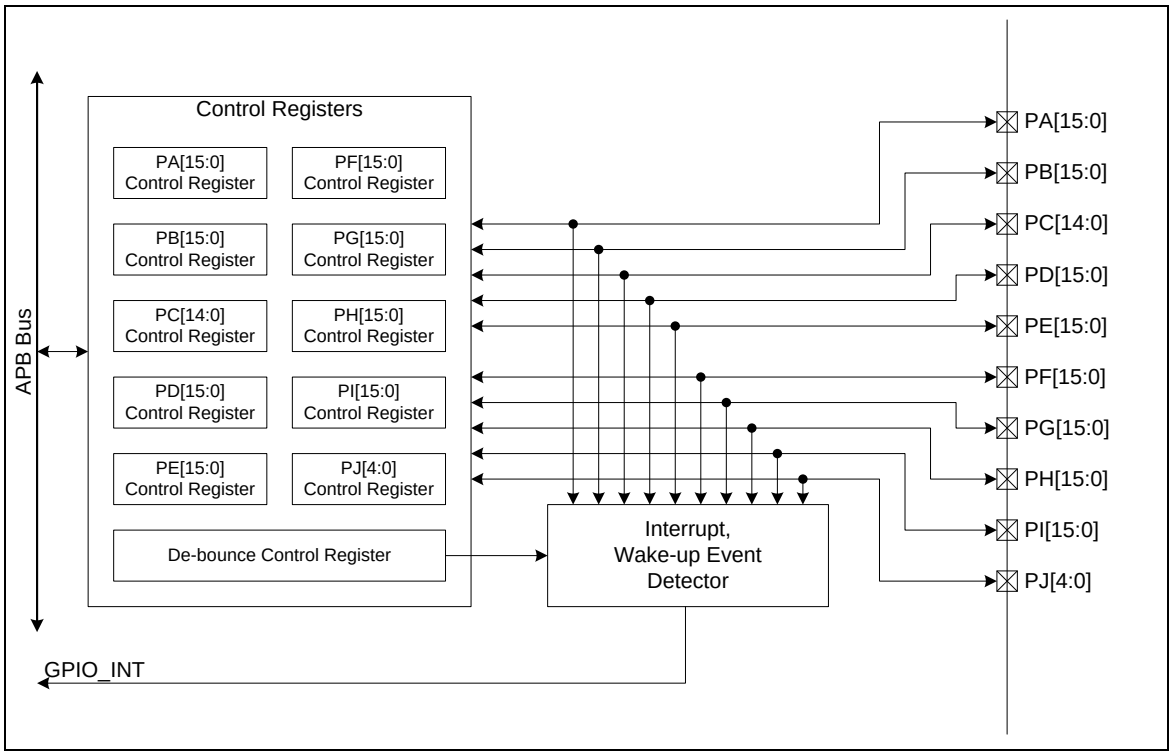



Figure 5.8-1 GPIO Block Diagram

### 5.8.4 Basic Configuration

Before using GPIO, it's necessary to enable clock of GPIO. Set GPIO (CLK_PCLKEN0[4]) high to enable GPIO's clock.

To configure pin Px.n as a General-Purpose I/O, set the corresponding field of register SYS_GPA_MFPL, SYS_GPA_MFPH, SYS_GPB_MFPL, SYS_GPB_MFPH, SYS_GPC_MFPL, SYS_GPC_MFPH, SYS_GPD_MFPL, SYS_GPD_MFPH, SYS_GPE_MFPL, SYS_GPE_MFPH, SYS_GPF_MFPL, SYS_GPF_MFPH, SYS_GPG_MFPL, SYS_GPG_MFPH, SYS_GPH_MFPL, SYS_GPH_MFPH, SYS_GPI_MFPL, SYS_GPI_MFPH and SYS_GPJ_MFPL to 0.

For example, if user want to configure pin PA.0 as a General-Purpose I/O, it's necessary to set MFP_GPA0 (SYS_GPA_MFPL[3:0]) to 0.

### 5.8.5 Functional Description

#### 5.8.5.1 *Input mode*

Set OUTEN (GPIOx_DIR[n]) to 0 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 DATAIN (GPIOx_DATAIN[n]) value reflects the status of the corresponding port pins.

Input buffer type is selectable in this mode. By setting ICEN (GPIOx_ICEN[n]) high to enable CMOS type input buffer while by setting ISEN (GPIOx_ISEN[n]) high to enable Schmitt trigger type input buffer. By setting both ICEN (GPIOx_ICEN[n]) and ISEN (GPIOx_ISEN[n]) low input buffer disabled. Setting both ICEN (GPIOx_ICEN[n]) and ISEN (GPIOx_ISEN[n]) high is forbidden.

5.8.5.2 *Output mode*

Set OUTEN (GPIOx_DIR[n]) to 1 as the Px.n pin is in output mode and the I/O pin reflects the value written in DATAOUT (GPIOx_DATAOUT[n]).

5.8.5.3 *GPIO Interrupt*

Each GPIO pin can be set as chip interrupt source by setting correlative IMD (GPIOx_IMD[n]), IREN (GPIOx_IREN[n]) and IFEN (GPIOx_IFEN[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. For edge trigger condition, user can enable input signal de-bounce function to prevent unexpected interrupt happened which caused by noise. The de-bounce clock sampling cycle period can be set through DBCLKSEL (GPIO_DBNCECON[3:0]) register.

### 5.8.6 Register Map

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GPIO Base Address:**<br>**GPIO_BA = 0xB800_3000** | | | | |
| **GPIOA_DIR** | GPIO_BA+0x000 | R/W | GPIO Port A Direction Control Register | 0x0000_0000 |
| **GPIOA_DATAOUT** | GPIO_BA+0x004 | R/W | GPIO Port A Data Output Register | 0x0000_0000 |
| **GPIOA_DATAIN** | GPIO_BA+0x008 | R | GPIO Port A Data Input Register | 0x0000_xxxx |
| **GPIOA_IMD** | GPIO_BA+0x00C | R/W | GPIO Port A Interrupt Mode Register | 0x0000_0000 |
| **GPIOA_IREN** | GPIO_BA+0x010 | R/W | GPIO Port A Interrupt Rising-Edge or Level-High Enable Register | 0x0000_0000 |
| **GPIOA_IFEN** | GPIO_BA+0x014 | R/W | GPIO Port A Interrupt Falling-Edge or Level-Low Enable Register | 0x0000_0000 |
| **GPIOA_ISR** | GPIO_BA+0x018 | R/W | GPIO Port A Interrupt Status Register | 0x0000_0000 |
| **GPIOA_DBEN** | GPIO_BA+0x01C | R/W | GPIO Port A De-bounce Enable Register | 0x0000_0000 |
| **GPIOA_PUEN** | GPIO_BA+0x020 | R/W | GPIO Port A Pull-Up Enable Register | 0x0000_0000 |
| **GPIOA_PDEN** | GPIO_BA+0x024 | R/W | GPIO Port A Pull-Down Enable Register | 0x0000_0000 |
| **GPIOA_ICEN** | GPIO_BA+0x028 | R/W | GPIO Port A CMOS Input Enable Register | 0x0000_FFFF |
| **GPIOA_ISEN** | GPIO_BA+0x02C | R/W | GPIO Port A Schmitt-Trigger Input Enable Register | 0x0000_0000 |
| **GPIOB_DIR** | GPIO_BA+0x040 | R/W | GPIO Port B Direction Control Register | 0x0000_0000 |
| **GPIOB_DATAOUT** | GPIO_BA+0x044 | R/W | GPIO Port B Data Output Register | 0x0000_0000 |
| **GPIOB_DATAIN** | GPIO_BA+0x048 | R | GPIO Port B Data Input Register | 0x0000_xxxx |
| **GPIOB_IMD** | GPIO_BA+0x04C | R/W | GPIO Port B Interrupt Mode Register | 0x0000_0000 |
| **GPIOB_IREN** | GPIO_BA+0x050 | R/W | GPIO Port B Interrupt Rising-Edge or Level-High Enable Register | 0x0000_0000 |
| **GPIOB_IFEN** | GPIO_BA+0x054 | R/W | GPIO Port B Interrupt Falling-Edge or Level-Low Enable Register | 0x0000_0000 |
| **GPIOB_ISR** | GPIO_BA+0x058 | R/W | GPIO Port B Interrupt Status Register | 0x0000_0000 |
| **GPIOB_DBEN** | GPIO_BA+0x05C | R/W | GPIO Port B De-bounce Enable Register | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **GPIOB_PUEN** | GPIO_BA+0x060 | R/W | GPIO Port B Pull-Up Enable Register | 0x0000_0000 |
| **GPIOB_PDEN** | GPIO_BA+0x064 | R/W | GPIO Port B Pull-Down Enable Register | 0x0000_0000 |
| **GPIOB_ICEN** | GPIO_BA+0x068 | R/W | GPIO Port B CMOS Input Enable Register | 0x0000_FFFF |
| **GPIOB_ISEN** | GPIO_BA+0x06C | R/W | GPIO Port B Schmitt-Trigger Input Enable Register | 0x0000_0000 |
| **GPIOC_DIR** | GPIO_BA+0x080 | R/W | GPIO Port C Direction Control Register | 0x0000_0000 |
| **GPIOC_DATAOUT** | GPIO_BA+0x084 | R/W | GPIO Port C Data Output Register | 0x0000_0000 |
| **GPIOC_DATAIN** | GPIO_BA+0x088 | R | GPIO Port C Data Input Register | 0x0000_xxxx |
| **GPIOC_IMD** | GPIO_BA+0x08C | R/W | GPIO Port C Interrupt Mode Register | 0x0000_0000 |
| **GPIOC_IREN** | GPIO_BA+0x090 | R/W | GPIO Port C Interrupt Rising-Edge or Level-High Enable Register | 0x0000_0000 |
| **GPIOC_IFEN** | GPIO_BA+0x094 | R/W | GPIO Port C Interrupt Falling-Edge or Level-Low Enable Register | 0x0000_0000 |
| **GPIOC_ISR** | GPIO_BA+0x098 | R/W | GPIO Port C Interrupt Status Register | 0x0000_0000 |
| **GPIOC_DBEN** | GPIO_BA+0x09C | R/W | GPIO Port C De-bounce Enable Register | 0x0000_0000 |
| **GPIOC_PUEN** | GPIO_BA+0x0A0 | R/W | GPIO Port C Pull-Up Enable Register | 0x0000_0000 |
| **GPIOC_PDEN** | GPIO_BA+0x0A4 | R/W | GPIO Port C Pull-Down Enable Register | 0x0000_0000 |
| **GPIOC_ICEN** | GPIO_BA+0x0A8 | R/W | GPIO Port C CMOS Input Enable Register | 0x0000_EFFF |
| **GPIOC_ISEN** | GPIO_BA+0x0AC | R/W | GPIO Port C Schmitt-Trigger Input Enable Register | 0x0000_0000 |
| **GPIOD_DIR** | GPIO_BA+0x0C0 | R/W | GPIO Port D Direction Control Register | 0x0000_0000 |
| **GPIOD_DATAOUT** | GPIO_BA+0x0C4 | R/W | GPIO Port D Data Output Register | 0x0000_0000 |
| **GPIOD_DATAIN** | GPIO_BA+0x0C8 | R | GPIO Port D Data Input Register | 0x0000_xxxx |
| **GPIOD_IMD** | GPIO_BA+0x0CC | R/W | GPIO Port D Interrupt Mode Register | 0x0000_0000 |
| **GPIOD_IREN** | GPIO_BA+0x0D0 | R/W | GPIO Port D Interrupt Rising-Edge or Level-High Enable Register | 0x0000_0000 |
| **GPIOD_IFEN** | GPIO_BA+0x0D4 | R/W | GPIO Port D Interrupt Falling-Edge or Level-Low Enable Register | 0x0000_0000 |
| **GPIOD_ISR** | GPIO_BA+0x0D8 | R/W | GPIO Port D Interrupt Status Register | 0x0000_0000 |
| **GPIOD_DBEN** | GPIO_BA+0x0DC | R/W | GPIO Port D De-bounce Enable Register | 0x0000_0000 |
| **GPIOD_PUEN** | GPIO_BA+0x0E0 | R/W | GPIO Port D Pull-Up Enable Register | 0x0000_0000 |
| **GPIOD_PDEN** | GPIO_BA+0x0E4 | R/W | GPIO Port D Pull-Down Enable Register | 0x0000_0000 |
| **GPIOD_ICEN** | GPIO_BA+0x0E8 | R/W | GPIO Port D CMOS Input Enable Register | 0x0000_FFFF |
| **GPIOD_ISEN** | GPIO_BA+0x0EC | R/W | GPIO Port D Schmitt-Trigger Input Enable Register | 0x0000_0000 |
| **GPIOE_DIR** | GPIO_BA+0x100 | R/W | GPIO Port E Direction Control Register | 0x0000_0000 |
| **GPIOE_DATAOUT** | GPIO_BA+0x104 | R/W | GPIO Port E Data Output Register | 0x0000_0000 |
| **GPIOE_DATAIN** | GPIO_BA+0x108 | R | GPIO Port E Data Input Register | 0x0000_xxxx |
| **GPIOE_IMD** | GPIO_BA+0x10C | R/W | GPIO Port E Interrupt Mode Register | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **GPIOE_IREN** | GPIO_BA+0x110 | R/W | GPIO Port E Interrupt Rising-Edge or Level-High Enable Register | 0x0000_0000 |
| **GPIOE_IFEN** | GPIO_BA+0x114 | R/W | GPIO Port E Interrupt Falling-Edge or Level-Low Enable Register | 0x0000_0000 |
| **GPIOE_ISR** | GPIO_BA+0x118 | R/W | GPIO Port E Interrupt Status Register | 0x0000_0000 |
| **GPIOE_DBEN** | GPIO_BA+0x11C | R/W | GPIO Port E De-bounce Enable Register | 0x0000_0000 |
| **GPIOE_PUEN** | GPIO_BA+0x120 | R/W | GPIO Port E Pull-Up Enable Register | 0x0000_0000 |
| **GPIOE_PDEN** | GPIO_BA+0x124 | R/W | GPIO Port E Pull-Down Enable Register | 0x0000_0000 |
| **GPIOE_ICEN** | GPIO_BA+0x128 | R/W | GPIO Port E CMOS Input Enable Register | 0x0000_FFFF |
| **GPIOE_ISEN** | GPIO_BA+0x12C | R/W | GPIO Port E Schmitt-Trigger Input Enable Register | 0x0000_0000 |
| **GPIOF_DIR** | GPIO_BA+0x140 | R/W | GPIO Port F Direction Control Register | 0x0000_0000 |
| **GPIOF_DATAOUT** | GPIO_BA+0x144 | R/W | GPIO Port F Data Output Register | 0x0000_0000 |
| **GPIOF_DATAIN** | GPIO_BA+0x148 | R | GPIO Port F Data Input Register | 0x0000_xxxx |
| **GPIOF_IMD** | GPIO_BA+0x14C | R/W | GPIO Port F Interrupt Mode Register | 0x0000_0000 |
| **GPIOF_IREN** | GPIO_BA+0x150 | R/W | GPIO Port F Interrupt Rising-Edge or Level-High Enable Register | 0x0000_0000 |
| **GPIOF_IFEN** | GPIO_BA+0x154 | R/W | GPIO Port F Interrupt Falling-Edge or Level-Low Enable Register | 0x0000_0000 |
| **GPIOF_ISR** | GPIO_BA+0x158 | R/W | GPIO Port F Interrupt Status Register | 0x0000_0000 |
| **GPIOF_DBEN** | GPIO_BA+0x15C | R/W | GPIO Port F De-bounce Enable Register | 0x0000_0000 |
| **GPIOF_PUEN** | GPIO_BA+0x160 | R/W | GPIO Port F Pull-Up Enable Register | 0x0000_0000 |
| **GPIOF_PDEN** | GPIO_BA+0x164 | R/W | GPIO Port F Pull-Down Enable Register | 0x0000_0000 |
| **GPIOF_ICEN** | GPIO_BA+0x168 | R/W | GPIO Port F CMOS Input Enable Register | 0x0000_FFFF |
| **GPIOF_ISEN** | GPIO_BA+0x16C | R/W | GPIO Port F Schmitt-Trigger Input Enable Register | 0x0000_0000 |
| **GPIOG_DIR** | GPIO_BA+0x180 | R/W | GPIO Port G Direction Control Register | 0x0000_0000 |
| **GPIOG_DATAOUT** | GPIO_BA+0x184 | R/W | GPIO Port G Data Output Register | 0x0000_0000 |
| **GPIOG_DATAIN** | GPIO_BA+0x188 | R | GPIO Port G Data Input Register | 0x0000_xxxx |
| **GPIOG_IMD** | GPIO_BA+0x18C | R/W | GPIO Port G Interrupt Mode Register | 0x0000_0000 |
| **GPIOG_IREN** | GPIO_BA+0x190 | R/W | GPIO Port G Interrupt Rising-Edge or Level-High Enable Register | 0x0000_0000 |
| **GPIOG_IFEN** | GPIO_BA+0x194 | R/W | GPIO Port G Interrupt Falling-Edge or Level-Low Enable Register | 0x0000_0000 |
| **GPIOG_ISR** | GPIO_BA+0x198 | R/W | GPIO Port G Interrupt Status Register | 0x0000_0000 |
| **GPIOG_DBEN** | GPIO_BA+0x19C | R/W | GPIO Port G De-bounce Enable Register | 0x0000_0000 |
| **GPIOG_PUEN** | GPIO_BA+0x1A0 | R/W | GPIO Port G Pull-Up Enable Register | 0x0000_0000 |
| **GPIOG_PDEN** | GPIO_BA+0x1A4 | R/W | GPIO Port G Pull-Down Enable Register | 0x0000_0000 |
| **GPIOG_ICEN** | GPIO_BA+0x1A8 | R/W | GPIO Port G CMOS Input Enable Register | 0x0000_FFFF |

| | | | | |
|---|---|---|---|---|
| **GPIOG_ISEN** | GPIO_BA+0x1AC | R/W | GPIO Port G Schmitt-Trigger Input Enable Register | 0x0000_0000 |
| **GPIOH_DIR** | GPIO_BA+0x1C0 | R/W | GPIO Port H Direction Control Register | 0x0000_0000 |
| **GPIOH_DATAOUT** | GPIO_BA+0x1C4 | R/W | GPIO Port H Data Output Register | 0x0000_0000 |
| **GPIOH_DATAIN** | GPIO_BA+0x1C8 | R | GPIO Port H Data Input Register | 0x0000_xxxx |
| **GPIOH_IMD** | GPIO_BA+0x1CC | R/W | GPIO Port H Interrupt Mode Register | 0x0000_0000 |
| **GPIOH_IREN** | GPIO_BA+0x1D0 | R/W | GPIO Port H Interrupt Rising-Edge or Level-High Enable Register | 0x0000_0000 |
| **GPIOH_IFEN** | GPIO_BA+0x1D4 | R/W | GPIO Port H Interrupt Falling-Edge or Level-Low Enable Register | 0x0000_0000 |
| **GPIOH_ISR** | GPIO_BA+0x1D8 | R/W | GPIO Port H Interrupt Status Register | 0x0000_0000 |
| **GPIOH_DBEN** | GPIO_BA+0x1DC | R/W | GPIO Port H De-bounce Enable Register | 0x0000_0000 |
| **GPIOH_PUEN** | GPIO_BA+0x1E0 | R/W | GPIO Port H Pull-Up Enable Register | 0x0000_0000 |
| **GPIOH_PDEN** | GPIO_BA+0x1E4 | R/W | GPIO Port H Pull-Down Enable Register | 0x0000_0000 |
| **GPIOH_ICEN** | GPIO_BA+0x1E8 | R/W | GPIO Port H CMOS Input Enable Register | 0x0000_FFFF |
| **GPIOH_ISEN** | GPIO_BA+0x1EC | R/W | GPIO Port H Schmitt-Trigger Input Enable Register | 0x0000_0000 |
| **GPIOI_DIR** | GPIO_BA+0x200 | R/W | GPIO Port I Direction Control Register | 0x0000_0000 |
| **GPIOI_DATAOUT** | GPIO_BA+0x204 | R/W | GPIO Port I Data Output Register | 0x0000_0000 |
| **GPIOI_DATAIN** | GPIO_BA+0x208 | R | GPIO Port I Data Input Register | 0x0000_xxxx |
| **GPIOI_IMD** | GPIO_BA+0x20C | R/W | GPIO Port I Interrupt Mode Register | 0x0000_0000 |
| **GPIOI_IREN** | GPIO_BA+0x210 | R/W | GPIO Port I Interrupt Rising-Edge or Level-High Enable Register | 0x0000_0000 |
| **GPIOI_IFEN** | GPIO_BA+0x214 | R/W | GPIO Port I Interrupt Falling-Edge or Level-Low Enable Register | 0x0000_0000 |
| **GPIOI_ISR** | GPIO_BA+0x218 | R/W | GPIO Port I Interrupt Status Register | 0x0000_0000 |
| **GPIOI_DBEN** | GPIO_BA+0x21C | R/W | GPIO Port I De-bounce Enable Register | 0x0000_0000 |
| **GPIOI_PUEN** | GPIO_BA+0x220 | R/W | GPIO Port I Pull-Up Enable Register | 0x0000_0000 |
| **GPIOI_PDEN** | GPIO_BA+0x224 | R/W | GPIO Port I Pull-Down Enable Register | 0x0000_0000 |
| **GPIOI_ICEN** | GPIO_BA+0x228 | R/W | GPIO Port I CMOS Input Enable Register | 0x0000_FFFF |
| **GPIOI_ISEN** | GPIO_BA+0x22C | R/W | GPIO Port I Schmitt-Trigger Input Enable Register | 0x0000_0000 |
| **GPIOJ_DIR** | GPIO_BA+0x240 | R/W | GPIO Port J Direction Control Register | 0x0000_0000 |
| **GPIOJ_DATAOUT** | GPIO_BA+0x244 | R/W | GPIO Port J Data Output Register | 0x0000_0000 |
| **GPIOJ_DATAIN** | GPIO_BA+0x248 | R | GPIO Port J Data Input Register | 0x0000_00xx |
| **GPIOJ_IMD** | GPIO_BA+0x24C | R/W | GPIO Port J Interrupt Mode Register | 0x0000_0000 |
| **GPIOJ_IREN** | GPIO_BA+0x250 | R/W | GPIO Port J Interrupt Rising-Edge or Level-High Enable Register | 0x0000_0000 |
| **GPIOJ_IFEN** | GPIO_BA+0x254 | R/W | GPIO Port J Interrupt Falling-Edge or Level-Low Enable Register | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **GPIOJ_ISR** | GPIO_BA+0x258 | R/W | GPIO Port J Interrupt Status Register | 0x0000_0000 |
| **GPIOJ_DBEN** | GPIO_BA+0x25C | R/W | GPIO Port J De-bounce Enable Register | 0x0000_0000 |
| **GPIOJ_PUEN** | GPIO_BA+0x260 | R/W | GPIO Port J Pull-Up Enable Register | 0x0000_0000 |
| **GPIOJ_PDEN** | GPIO_BA+0x264 | R/W | GPIO Port J Pull-Down Enable Register | 0x0000_0000 |
| **GPIOJ_ICEN** | GPIO_BA+0x268 | R/W | GPIO Port J CMOS Input Enable Register | 0x0000_001F |
| **GPIOJ_ISEN** | GPIO_BA+0x26C | R/W | GPIO Port J Schmitt-Trigger Input Enable Register | 0x0000_0000 |
| **GPIO_DBNCECON** | GPIO_BA+0x3F0 | R/W | GPIO Debounce Control Register | 0x0000_0020 |
| **GPIO_ISR** | GPIO_BA+0x3FC | R | GPIO Port Interrupt Status Register | 0x0000_0000 |

### 5.8.7 Register Description

### GPIO Port A-J Direction Control Register (GPIOx_DIR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GPIOA_DIR** | GPIO_BA+0x000 | R/W | GPIO Port A Direction Control Register | 0x0000_0000 |
| **GPIOB_DIR** | GPIO_BA+0x040 | R/W | GPIO Port B Direction Control Register | 0x0000_0000 |
| **GPIOC_DIR** | GPIO_BA+0x080 | R/W | GPIO Port C Direction Control Register | 0x0000_0000 |
| **GPIOD_DIR** | GPIO_BA+0x0C0 | R/W | GPIO Port D Direction Control Register | 0x0000_0000 |
| **GPIOE_DIR** | GPIO_BA+0x100 | R/W | GPIO Port E Direction Control Register | 0x0000_0000 |
| **GPIOF_DIR** | GPIO_BA+0x140 | R/W | GPIO Port F Direction Control Register | 0x0000_0000 |
| **GPIOG_DIR** | GPIO_BA+0x180 | R/W | GPIO Port G Direction Control Register | 0x0000_0000 |
| **GPIOH_DIR** | GPIO_BA+0x1C0 | R/W | GPIO Port H Direction Control Register | 0x0000_0000 |
| **GPIOI_DIR** | GPIO_BA+0x200 | R/W | GPIO Port I Direction Control Register | 0x0000_0000 |
| **GPIOJ_DIR** | GPIO_BA+0x240 | R/W | GPIO Port J Direction 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** |
| **OUTEN** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **OUTEN** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **OUTEN** | **GPIO Output Enable**<br>This filed defines the direction of the pin that configured as a General-Purpose I/O.<br><br>0: GPIO is in input mode<br>1: GPIO is in output mode<br>**Note1:** For GPIOC, the OUTEN[15] are reserved.<br>**Note2:** For GPIOJ, the OUTEN[15:5] are reserved. |

## GPIO Port A-J Data Output Register (GPIOx_DATAOUT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GPIOA_DATAOUT** | GPIO_BA+0x004 | R/W | GPIO Port A Data Output Register | 0x0000_0000 |
| **GPIOB_DATAOUT** | GPIO_BA+0x044 | R/W | GPIO Port B Data Output Register | 0x0000_0000 |
| **GPIOC_DATAOUT** | GPIO_BA+0x084 | R/W | GPIO Port C Data Output Register | 0x0000_0000 |
| **GPIOD_DATAOUT** | GPIO_BA+0x0C4 | R/W | GPIO Port D Data Output Register | 0x0000_0000 |
| **GPIOE_DATAOUT** | GPIO_BA+0x104 | R/W | GPIO Port E Data Output Register | 0x0000_0000 |
| **GPIOF_DATAOUT** | GPIO_BA+0x144 | R/W | GPIO Port F Data Output Register | 0x0000_0000 |
| **GPIOG_DATAOUT** | GPIO_BA+0x184 | R/W | GPIO Port G Data Output Register | 0x0000_0000 |
| **GPIOH_DATAOUT** | GPIO_BA+0x1C4 | R/W | GPIO Port H Data Output Register | 0x0000_0000 |
| **GPIOI_DATAOUT** | GPIO_BA+0x204 | R/W | GPIO Port I Data Output Register | 0x0000_0000 |
| **GPIOJ_DATAOUT** | GPIO_BA+0x244 | R/W | GPIO Port J 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** |
| **DATAOUT** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **DATAOUT** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **DATAOUT** | **GPIO Data Output**<br>This field defines the output value of a pin when it's configured as a General-Purpose I/O and it's OUTEN (GPIOx_DIR[n]) set as 1.<br><br>0: GPIO data output is 0<br>1: GPIO data output is 1<br>**Note1:** For GPIOC, the DATAOUT[15] are reserved.<br>**Note2:** For GPIOJ, the DATAOUT[15:5] are reserved. |

## GPIO Port A-J Data Input Register (GPIOx_DATAIN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GPIOA_DATAIN** | GPIO_BA+0x008 | R | GPIO Port A Data Input Register | 0x0000_xxxx |
| **GPIOB_DATAIN** | GPIO_BA+0x048 | R | GPIO Port B Data Input Register | 0x0000_xxxx |
| **GPIOC_DATAIN** | GPIO_BA+0x088 | R | GPIO Port C Data Input Register | 0x0000_xxxx |
| **GPIOD_DATAIN** | GPIO_BA+0x0C8 | R | GPIO Port D Data Input Register | 0x0000_xxxx |
| **GPIOE_DATAIN** | GPIO_BA+0x108 | R | GPIO Port E Data Input Register | 0x0000_xxxx |
| **GPIOF_DATAIN** | GPIO_BA+0x148 | R | GPIO Port F Data Input Register | 0x0000_xxxx |
| **GPIOG_DATAIN** | GPIO_BA+0x188 | R | GPIO Port G Data Input Register | 0x0000_xxxx |
| **GPIOH_DATAIN** | GPIO_BA+0x1C8 | R | GPIO Port H Data Input Register | 0x0000_xxxx |
| **GPIOI_DATAIN** | GPIO_BA+0x208 | R | GPIO Port I Data Input Register | 0x0000_xxxx |
| **GPIOJ_DATAIN** | GPIO_BA+0x248 | R | GPIO Port J Data Input 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** |
| **DATAIN** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **DATAIN** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **DATAIN** | **GPIO Data Input**<br>This field reflects the value of the pin that configured as a General-Purpose I/O.<br>If the pin is not configured as a General-Purpose I/O, the value of this field is dependent on the setting of GPIOLBEN (SYS_MISCFCR[12]). If the pin is not configured as a General-Purpose I/O and GPIOLBEN (SYS_MISCFCR[12]) is low, this field will always read as 0. If the pin is not configured as a General-Purpose I/O and GPIOLBEN (SYS_MISCFCR[12]) is high, this field reflect the value of the pin, too.<br>0: GPIO data input is 0<br>1: GPIO data input is 1<br>**Note1:** For GPIOC, the DATAIN[15] are reserved.<br>**Note2:** For GPIOJ, the DATAIN[15:5] are reserved. |

## GPIO Port A-J Interrupt Mode Register (GPIOx_IMD)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GPIOA_IMD** | GPIO_BA+0x00C | R/W | GPIO Port A Interrupt Mode Register | 0x0000_0000 |
| **GPIOB_IMD** | GPIO_BA+0x04C | R/W | GPIO Port B Interrupt Mode Register | 0x0000_0000 |
| **GPIOC_IMD** | GPIO_BA+0x08C | R/W | GPIO Port C Interrupt Mode Register | 0x0000_0000 |
| **GPIOD_IMD** | GPIO_BA+0x0CC | R/W | GPIO Port D Interrupt Mode Register | 0x0000_0000 |
| **GPIOE_IMD** | GPIO_BA+0x10C | R/W | GPIO Port E Interrupt Mode Register | 0x0000_0000 |
| **GPIOF_IMD** | GPIO_BA+0x14C | R/W | GPIO Port F Interrupt Mode Register | 0x0000_0000 |
| **GPIOG_IMD** | GPIO_BA+0x18C | R/W | GPIO Port G Interrupt Mode Register | 0x0000_0000 |
| **GPIOH_IMD** | GPIO_BA+0x1CC | R/W | GPIO Port H Interrupt Mode Register | 0x0000_0000 |
| **GPIOI_IMD** | GPIO_BA+0x20C | R/W | GPIO Port I Interrupt Mode Register | 0x0000_0000 |
| **GPIOJ_IMD** | GPIO_BA+0x24C | R/W | GPIO Port J Interrupt Mode 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** |
| **IMD** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **IMD** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **IMD** | **GPIO Interrupt Mode**<br>Each of General-Purpose I/O could be used as an interrupt source. This field defines the interrupt is an edge trigger interrupt or level trigger interrupt.<br><br>0: Edge trigger interrupt<br>1: Level trigger interrupt<br>**Note1:** For GPIOC, the IMD[15] are reserved.<br>**Note2:** For GPIOJ, the IMD[15:5] are reserved. |

### GPIO Port A-J Interrupt Rising-Edge Or Level-High Enable Register (GPIOx_IREN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GPIOA_IREN** | GPIO_BA+0x010 | R/W | GPIO Port A Interrupt Rising-Edge or Level-High Enable Register | 0x0000_0000 |
| **GPIOB_IREN** | GPIO_BA+0x050 | R/W | GPIO Port B Interrupt Rising-Edge or Level-High Enable Register | 0x0000_0000 |
| **GPIOC_IREN** | GPIO_BA+0x090 | R/W | GPIO Port C Interrupt Rising-Edge or Level-High Enable Register | 0x0000_0000 |
| **GPIOD_IREN** | GPIO_BA+0x0D0 | R/W | GPIO Port D Interrupt Rising-Edge or Level-High Enable Register | 0x0000_0000 |
| **GPIOE_IREN** | GPIO_BA+0x110 | R/W | GPIO Port E Interrupt Rising-Edge or Level-High Enable Register | 0x0000_0000 |
| **GPIOF_IREN** | GPIO_BA+0x150 | R/W | GPIO Port F Interrupt Rising-Edge or Level-High Enable Register | 0x0000_0000 |
| **GPIOG_IREN** | GPIO_BA+0x190 | R/W | GPIO Port G Interrupt Rising-Edge or Level-High Enable Register | 0x0000_0000 |
| **GPIOH_IREN** | GPIO_BA+0x1D0 | R/W | GPIO Port H Interrupt Rising-Edge or Level-High Enable Register | 0x0000_0000 |
| **GPIOI_IREN** | GPIO_BA+0x210 | R/W | GPIO Port I Interrupt Rising-Edge or Level-High Enable Register | 0x0000_0000 |
| **GPIOJ_IREN** | GPIO_BA+0x250 | R/W | GPIO Port J Interrupt Rising-Edge or Level-High 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** |
| **IREN** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **IREN** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [15:0] | **IREN** | **GPIO Interrupt Rising-edge or Level-high Enable**<br>This field controls the enable of rising-edge or level-high detection of a General-Purpose I/O.<br>When this bit is high and corresponding bit of GPIOx_IMD is low, rising-edge detection enabled. When this bit is high and corresponding bit of GPIOx_IMD is high, level-high detection enabled.<br>0: Disable rising-edge or level-high interrupt<br>1: Enable rising-edge or level-high interrupt<br>**Note1:** For GPIOC, the IREN[15] are reserved.<br>**Note2:** For GPIOJ, the IREN[15:5] are reserved. |

### GPIO Port A-J Interrupt Falling-Edge Or Level-Low Enable Register (GPIOx_IFEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GPIOA_IFEN** | GPIO_BA+0x014 | R/W | GPIO Port A Interrupt Falling-Edge or Level-Low Enable Register | 0x0000_0000 |
| **GPIOB_IFEN** | GPIO_BA+0x054 | R/W | GPIO Port B Interrupt Falling-Edge or Level-Low Enable Register | 0x0000_0000 |
| **GPIOC_IFEN** | GPIO_BA+0x094 | R/W | GPIO Port C Interrupt Falling-Edge or Level-Low Enable Register | 0x0000_0000 |
| **GPIOD_IFEN** | GPIO_BA+0x0D4 | R/W | GPIO Port D Interrupt Falling-Edge or Level-Low Enable Register | 0x0000_0000 |
| **GPIOE_IFEN** | GPIO_BA+0x114 | R/W | GPIO Port E Interrupt Falling-Edge or Level-Low Enable Register | 0x0000_0000 |
| **GPIOF_IFEN** | GPIO_BA+0x154 | R/W | GPIO Port F Interrupt Falling-Edge or Level-Low Enable Register | 0x0000_0000 |
| **GPIOG_IFEN** | GPIO_BA+0x194 | R/W | GPIO Port G Interrupt Falling-Edge or Level-Low Enable Register | 0x0000_0000 |
| **GPIOH_IFEN** | GPIO_BA+0x1D4 | R/W | GPIO Port H Interrupt Falling-Edge or Level-Low Enable Register | 0x0000_0000 |
| **GPIOI_IFEN** | GPIO_BA+0x214 | R/W | GPIO Port I Interrupt Falling-Edge or Level-Low Enable Register | 0x0000_0000 |
| **GPIOJ_IFEN** | GPIO_BA+0x254 | R/W | GPIO Port J Interrupt Falling-Edge or Level-Low 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** |
| **IFEN** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **IFEN** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [15:0] | **IFEN** | **GPIO Interrupt Falling-edge or Level-low Enable**<br>This field controls the enable of falling-edge or level-low detection of a General-Purpose I/O.<br>When this bit is high and corresponding bit of GPIOx_IMD is low, falling-edge detection enabled. When this bit is high and corresponding bit of GPIOx_IMD is high, level-low detection enabled.<br>0: Disable falling-edge or level-low interrupt<br>1: Enable falling-edge or level-low interrupt<br>**Note1:** For GPIOC, the IFEN[15] are reserved.<br>**Note2:** For GPIOJ, the IFEN[15:5] are reserved. |

## GPIO Port A-J Interrupt Status Register (GPIOx_ISR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GPIOA_ISR** | GPIO_BA+0x018 | R/W | GPIO Port A Interrupt Status Register | 0x0000_0000 |
| **GPIOB_ISR** | GPIO_BA+0x058 | R/W | GPIO Port B Interrupt Status Register | 0x0000_0000 |
| **GPIOC_ISR** | GPIO_BA+0x098 | R/W | GPIO Port C Interrupt Status Register | 0x0000_0000 |
| **GPIOD_ISR** | GPIO_BA+0x0D8 | R/W | GPIO Port D Interrupt Status Register | 0x0000_0000 |
| **GPIOE_ISR** | GPIO_BA+0x118 | R/W | GPIO Port E Interrupt Status Register | 0x0000_0000 |
| **GPIOF_ISR** | GPIO_BA+0x158 | R/W | GPIO Port F Interrupt Status Register | 0x0000_0000 |
| **GPIOG_ISR** | GPIO_BA+0x198 | R/W | GPIO Port G Interrupt Status Register | 0x0000_0000 |
| **GPIOH_ISR** | GPIO_BA+0x1D8 | R/W | GPIO Port H Interrupt Status Register | 0x0000_0000 |
| **GPIOI_ISR** | GPIO_BA+0x218 | R/W | GPIO Port I Interrupt Status Register | 0x0000_0000 |
| **GPIOJ_ISR** | GPIO_BA+0x258 | R/W | GPIO Port J 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** |
| **ISR** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ISR** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **ISR** | **GPIO Interrupt Status**<br>This field keeps the interrupt detection status of the pin. When any bit is high, GPIO would trigger an interrupt to CPU.<br>Write 1 to clear this bit to zero.<br>0: No interrupt<br>1: Interrupt trigger<br>**Note1:** For GPIOC, the ISR[15] are reserved.<br>**Note2:** For GPIOJ, the ISR[15:5] are reserved. |

### GPIO Port A-J De-bounce Enable Register (GPIOx_DBEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GPIOA_DBEN** | GPIO_BA+0x01C | R/W | GPIO Port A De-bounce Enable Register | 0x0000_0000 |
| **GPIOB_DBEN** | GPIO_BA+0x05C | R/W | GPIO Port B De-bounce Enable Register | 0x0000_0000 |
| **GPIOC_DBEN** | GPIO_BA+0x09C | R/W | GPIO Port C De-bounce Enable Register | 0x0000_0000 |
| **GPIOD_DBEN** | GPIO_BA+0x0DC | R/W | GPIO Port D De-bounce Enable Register | 0x0000_0000 |
| **GPIOE_DBEN** | GPIO_BA+0x11C | R/W | GPIO Port E De-bounce Enable Register | 0x0000_0000 |
| **GPIOF_DBEN** | GPIO_BA+0x15C | R/W | GPIO Port F De-bounce Enable Register | 0x0000_0000 |
| **GPIOG_DBEN** | GPIO_BA+0x19C | R/W | GPIO Port G De-bounce Enable Register | 0x0000_0000 |
| **GPIOH_DBEN** | GPIO_BA+0x1DC | R/W | GPIO Port H De-bounce Enable Register | 0x0000_0000 |
| **GPIOI_DBEN** | GPIO_BA+0x21C | R/W | GPIO Port I De-bounce Enable Register | 0x0000_0000 |
| **GPIOJ_DBEN** | GPIO_BA+0x25C | R/W | GPIO Port J De-bounce 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** |
| **DBEN** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **DBEN** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **DBEN** | **GPIO De-bounce Enable**<br>This field controls the enable of de-bounce function of the pin configured as a General-Purpose I/O.<br><br>0: Disable de-bounce function<br>1: Enable de-bounce function<br>**Note1:** For GPIOC, the DBEN[15] are reserved.<br>**Note2:** For GPIOJ, the DBEN[15:5] are reserved. |

## GPIO Port A-J Pull-Up Enable Register (GPIOx_PUEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GPIOA_PUEN** | GPIO_BA+0x020 | R/W | GPIO Port A Pull-Up Enable Register | 0x0000_0000 |
| **GPIOB_PUEN** | GPIO_BA+0x060 | R/W | GPIO Port B Pull-Up Enable Register | 0x0000_0000 |
| **GPIOC_PUEN** | GPIO_BA+0x0A0 | R/W | GPIO Port C Pull-Up Enable Register | 0x0000_0000 |
| **GPIOD_PUEN** | GPIO_BA+0x0E0 | R/W | GPIO Port D Pull-Up Enable Register | 0x0000_0000 |
| **GPIOE_PUEN** | GPIO_BA+0x120 | R/W | GPIO Port E Pull-Up Enable Register | 0x0000_0000 |
| **GPIOF_PUEN** | GPIO_BA+0x160 | R/W | GPIO Port F Pull-Up Enable Register | 0x0000_0000 |
| **GPIOG_PUEN** | GPIO_BA+0x1A0 | R/W | GPIO Port G Pull-Up Enable Register | 0x0000_0000 |
| **GPIOH_PUEN** | GPIO_BA+0x1E0 | R/W | GPIO Port H Pull-Up Enable Register | 0x0000_0000 |
| **GPIOI_PUEN** | GPIO_BA+0x220 | R/W | GPIO Port I Pull-Up Enable Register | 0x0000_0000 |
| **GPIOJ_PUEN** | GPIO_BA+0x260 | R/W | GPIO Port J Pull-Up 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** |
| **PUEN** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **PUEN** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **PUEN** | **GPIO Pull-up Enable**<br>This field controls the pull-up resistor enable of the pin that can be configured as a General-Purpose I/O.<br>This control bit always takes effect no matter the pin configured as a General-Purpose I/O or not.<br>0: Disable pull-up function<br>1: Enable pull-up function<br>**Note1:** For GPIOC, the PUEN[15] are reserved.<br>**Note2:** For GPIOJ, the PUEN[15:5] are reserved. |

## GPIO Port A-J Pull-Down Enable Register (GPIOx_PDEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GPIOA_PDEN** | GPIO_BA+0x024 | R/W | GPIO Port A Pull-Down Enable Register | 0x0000_0000 |
| **GPIOB_PDEN** | GPIO_BA+0x064 | R/W | GPIO Port B Pull-Down Enable Register | 0x0000_0000 |
| **GPIOC_PDEN** | GPIO_BA+0x0A4 | R/W | GPIO Port C Pull-Down Enable Register | 0x0000_0000 |
| **GPIOD_PDEN** | GPIO_BA+0x0E4 | R/W | GPIO Port D Pull-Down Enable Register | 0x0000_0000 |
| **GPIOE_PDEN** | GPIO_BA+0x124 | R/W | GPIO Port E Pull-Down Enable Register | 0x0000_0000 |
| **GPIOF_PDEN** | GPIO_BA+0x164 | R/W | GPIO Port F Pull-Down Enable Register | 0x0000_0000 |
| **GPIOG_PDEN** | GPIO_BA+0x1A4 | R/W | GPIO Port G Pull-Down Enable Register | 0x0000_0000 |
| **GPIOH_PDEN** | GPIO_BA+0x1E4 | R/W | GPIO Port H Pull-Down Enable Register | 0x0000_0000 |
| **GPIOI_PDEN** | GPIO_BA+0x224 | R/W | GPIO Port I Pull-Down Enable Register | 0x0000_0000 |
| **GPIOJ_PDEN** | GPIO_BA+0x264 | R/W | GPIO Port J Pull-Down 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** |
| **PDEN** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **PDEN** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **PDEN** | **GPIO Pull-down Enable**<br>This field controls the pull-down resistor enable of the pin that can be configured as a General-Purpose I/O.<br>This control bit always takes effect no matter the pin configured as a General-Purpose I/O or not.<br>0: Disable pull-down function<br>1: Enable pull-down function<br>**Note1:** For GPIOC, the PDEN[15] are reserved.<br>**Note2:** For GPIOE, the PDEN[15:5] are reserved. |

## GPIO Port A-J CMOS Input Enable Register (GPIOx_ICEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GPIOA_ICEN** | GPIO_BA+0x028 | R/W | GPIO Port A CMOS Input Enable Register | 0x0000_FFFF |
| **GPIOB_ICEN** | GPIO_BA+0x068 | R/W | GPIO Port B CMOS Input Enable Register | 0x0000_FFFF |
| **GPIOC_ICEN** | GPIO_BA+0x0A8 | R/W | GPIO Port C CMOS Input Enable Register | 0x0000_EFFF |
| **GPIOD_ICEN** | GPIO_BA+0x0E8 | R/W | GPIO Port D CMOS Input Enable Register | 0x0000_FFFF |
| **GPIOE_ICEN** | GPIO_BA+0x128 | R/W | GPIO Port E CMOS Input Enable Register | 0x0000_FFFF |
| **GPIOF_ICEN** | GPIO_BA+0x168 | R/W | GPIO Port F CMOS Input Enable Register | 0x0000_FFFF |
| **GPIOG_ICEN** | GPIO_BA+0x1A8 | R/W | GPIO Port G CMOS Input Enable Register | 0x0000_FFFF |
| **GPIOH_ICEN** | GPIO_BA+0x1E8 | R/W | GPIO Port H CMOS Input Enable Register | 0x0000_FFFF |
| **GPIOI_ICEN** | GPIO_BA+0x228 | R/W | GPIO Port I CMOS Input Enable Register | 0x0000_FFFF |
| **GPIOJ_ICEN** | GPIO_BA+0x268 | R/W | GPIO Port J CMOS Input Enable Register | 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** |
| **ICEN** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ICEN** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **ICEN** | **GPIO CMOS Input Enable**<br>This fields controls the CMOS input buffer enable of the pin that can be configured as a General-Purpose I/O.<br>This control bit always takes effect no matter the pin configured as a General-Purpose I/O or not.<br>0: Disable CMOS input path<br>1: Enable CMOS input path<br>**Note1:** For GPIOC, the ICEN[15] are reserved.<br>**Note2:** For GPIOJ, the ICEN[15:5] are reserved. |

## GPIO Port A-J Schmitt-Trigger Input Enable Register (GPIOx_ISEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GPIOA_ISEN** | GPIO_BA+0x02C | R/W | GPIO Port A Schmitt-Trigger Input Enable Register | 0x0000_0000 |
| **GPIOB_ISEN** | GPIO_BA+0x06C | R/W | GPIO Port B Schmitt-Trigger Input Enable Register | 0x0000_0000 |
| **GPIOC_ISEN** | GPIO_BA+0x0AC | R/W | GPIO Port C Schmitt-Trigger Input Enable Register | 0x0000_0000 |
| **GPIOD_ISEN** | GPIO_BA+0x0EC | R/W | GPIO Port D Schmitt-Trigger Input Enable Register | 0x0000_0000 |
| **GPIOE_ISEN** | GPIO_BA+0x12C | R/W | GPIO Port E Schmitt-Trigger Input Enable Register | 0x0000_0000 |
| **GPIOF_ISEN** | GPIO_BA+0x16C | R/W | GPIO Port F Schmitt-Trigger Input Enable Register | 0x0000_0000 |
| **GPIOG_ISEN** | GPIO_BA+0x1AC | R/W | GPIO Port G Schmitt-Trigger Input Enable Register | 0x0000_0000 |
| **GPIOH_ISEN** | GPIO_BA+0x1EC | R/W | GPIO Port H Schmitt-Trigger Input Enable Register | 0x0000_0000 |
| **GPIOI_ISEN** | GPIO_BA+0x22C | R/W | GPIO Port I Schmitt-Trigger Input Enable Register | 0x0000_0000 |
| **GPIOJ_ISEN** | GPIO_BA+0x26C | R/W | GPIO Port J Schmitt-Trigger 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** |
| **ISEN** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ISEN** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [15:0] | **ISEN** | **GPIO Schmitt Input Enable**<br>This fields controls the Schmitt trigger input buffer enable of the pin that can be configured as a General-Purpose I/O.<br>This control bit always takes effect no matter the pin configured as a General-Purpose I/O or not.<br><br>0: Disable Schmitt input path<br>1: Enable Schmitt input path<br>**Note1:** For GPIOC, the ISEN[15] are reserved.<br>**Note2:** For GPIOJ, the ISEN[15:5] are reserved. |

### GPIO Debounce Control Register (GPIO_DBNCECON)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIO_DBNCECON | GPIO_BA+0x3F0 | R/W | GPIO Debounce 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 | | ICLK_ON | Reserved | DBCLKSEL | | | |

| Bits | Description | |
|---|---|---|
| [31:6] | **Reserved** | Reserved. |
| [5] | **ICLK_ON** | **Interrupt Clock on Mode**<br>This bit controls the interrupt detection clock enable of the pin that can be configured as a General-Purpose I/O.<br>When this bit is high, interrupt detection clock of the pin that can be configured as a General-Purpose I/O enabled permanently.<br>When this bit is low, interrupt detection clock enable is dependent on the setting of GPIOx_IMD. When this bit is low and GPIOx_IMD[n] is high, interrupt detection clock of that pin enabled. Otherwise, the interrupt detection clock disabled.<br>0: Clock Disabled if the GPIOx_IMD[n] interrupt is disabled.<br>1: Interrupt generated circuit clock always Enabled. |
| [4] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [3:0] | **DBCLKSEL** | **De-bounce Sampling Cycle Selection**<br>0000 = Sample interrupt input once per 1 clocks.<br>0001 = Sample interrupt input once per 2 clocks.<br>0010 = Sample interrupt input once per 4 clocks.<br>0011 = Sample interrupt input once per 8 clocks.<br>0100 = Sample interrupt input once per 16 clocks.<br>0101 = Sample interrupt input once per 32 clocks.<br>0110 = Sample interrupt input once per 64 clocks.<br>0111 = Sample interrupt input once per 128 clocks.<br>1000 = Sample interrupt input once per 256 clocks.<br>1001 = Sample interrupt input once per 2*256 clocks.<br>1010 = Sample interrupt input once per 4*256 clocks.<br>1011 = Sample interrupt input once per 8*256 clocks.<br>1100 = Sample interrupt input once per 16*256 clocks.<br>1101 = Sample interrupt input once per 32*256 clocks.<br>1110 = Sample interrupt input once per 64*256 clocks.<br>1111 = Sample interrupt input once per 128*256 clocks. |

## GPIO Port Interrupt Status Register (GPIO_ISR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIO_ISR | GPIO_BA+0x3FC | R | GPIO Port 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 | | | | | | GPIOJINT | GPIOIINT |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIOHINT | GPIOGINT | GPIOFINT | GPIOEINT | GPIODINT | GPIOCINT | GPIOBINT | GPIOAINT |

| Bits | Description | |
|---|---|---|
| [31:10] | **Reserved** | Reserved. |
| [9] | **GPIOJINT** | **GPIO Port J Interrupt Status**<br>This bit indicates if the GPIO interrupt triggered by pin of GPIO port J.<br><br>0: GPIO port J did not trigger GPIO interrupt<br>1: GPIO port J trigger GPIO interrupt |
| [8] | **GPIOIINT** | **GPIO Port I Interrupt Status**<br>This bit indicates if the GPIO interrupt triggered by pin of GPIO port I.<br><br>0: GPIO port I did not trigger GPIO interrupt<br>1: GPIO port I trigger GPIO interrupt |
| [7] | **GPIOHINT** | **GPIO Port H Interrupt Status**<br>This bit indicates if the GPIO interrupt triggered by pin of GPIO port H.<br><br>0: GPIO port H did not trigger GPIO interrupt<br>1: GPIO port H trigger GPIO interrupt |
| [6] | **GPIOGINT** | **GPIO Port G Interrupt Status**<br>This bit indicates if the GPIO interrupt triggered by pin of GPIO port G.<br><br>0: GPIO port G did not trigger GPIO interrupt<br>1: GPIO port G trigger GPIO interrupt |

| | | |
|---|---|---|
| [5] | **GPIOFINT** | **GPIO Port F Interrupt Status**<br>This bit indicates if the GPIO interrupt triggered by pin of GPIO port F.<br><br>0: GPIO port F did not trigger GPIO interrupt<br>1: GPIO port F trigger GPIO interrupt |
| [4] | **GPIOEINT** | **GPIO Port E Interrupt Status**<br>This bit indicates if the GPIO interrupt triggered by pin of GPIO port E.<br><br>0: GPIO port E did not trigger GPIO interrupt<br>1: GPIO port E trigger GPIO interrupt |
| [3] | **GPIODINT** | **GPIO Port D Interrupt Status**<br>This bit indicates if the GPIO interrupt triggered by pin of GPIO port D.<br><br>0: GPIO port D did not trigger GPIO interrupt<br>1: GPIO port D trigger GPIO interrupt |
| [2] | **GPIOCINT** | **GPIO Port C Interrupt Status**<br>This bit indicates if the GPIO interrupt triggered by pin of GPIO port C.<br><br>0: GPIO port C did not trigger GPIO interrupt<br>1: GPIO port C  trigger GPIO interrupt |
| [1] | **GPIOBINT** | **GPIO Port B Interrupt Status**<br>This bit indicates if the GPIO interrupt triggered by pin of GPIO port B.<br><br>0: GPIO port B did not trigger GPIO interrupt<br>1: GPIO port B trigger GPIO interrupt |
| [0] | **GPIOAINT** | **GPIO Port a Interrupt Status**<br>This bit indicates if the GPIO interrupt triggered by pin of GPIO port A.<br><br>0: GPIO port A did not trigger GPIO interrupt<br>1: GPIO port A trigger GPIO interrupt |

## 5.9 General DMA Controller (GDMA)

### 5.9.1 Overview

The chip has a two-channel general DMA controller with or without descriptor fetch operation, called the GDMA. The two-channel GDMA performs the memory-to-memory data transfers without the CPU intervention:

The on-chip GDMA can be started by the software. Software can also be used to restart the GDMA operation after it has been stopped. The CPU can recognize the completion of a GDMA operation by software polling or when it receives an internal GDMA interrupt. The GDMA controller can increment source or destination address, decrement them as well, and conduct 8-bit (byte), 16-bit (half-word), or 32-bit (word) data transfers.

### 5.9.2 Features

- AMBA AHB compliant
- Descriptor and Non-Descriptor based function
- Supports 8-data burst mode to boost performance
- Provides support for external GDMA device
- Demand mode speeds up external GDMA operations

### 5.9.3 Block Diagram

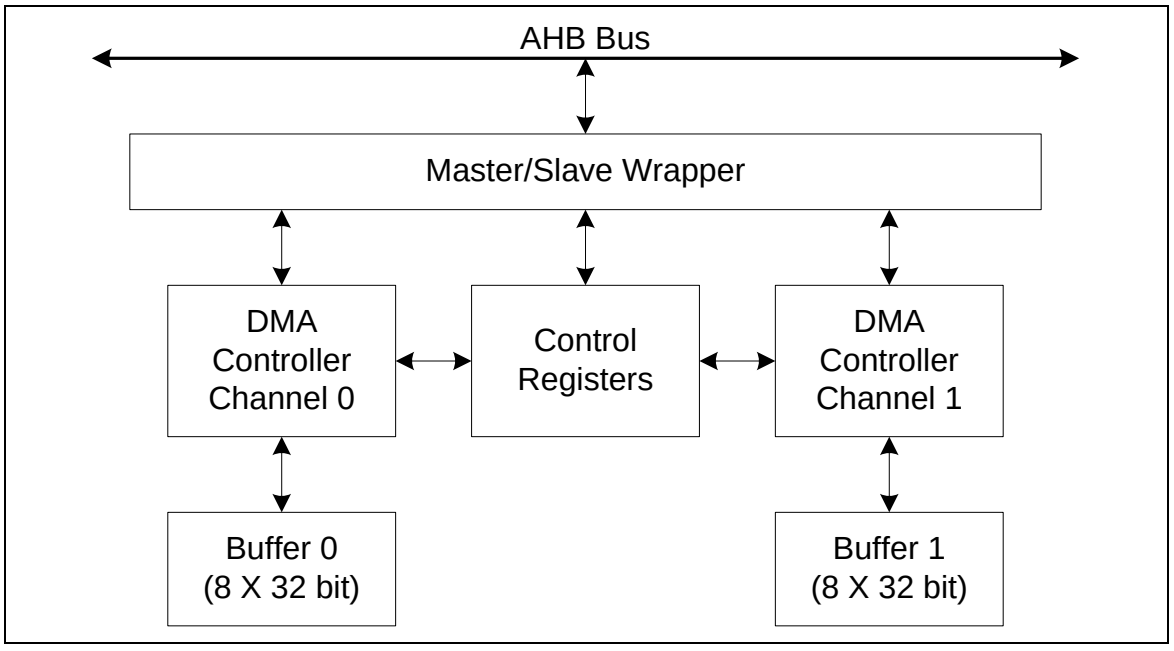



Figure 5.9-1 GDMA Block Diagram

### 5.9.4 Basic Configuration

Before using GDMA, it's necessary to enable clock of GDMA. Set GDMA (CLK_HCLKEN[12]) high to enable GDMA's clock.

### 5.9.5 Functional Description

#### 5.9.5.1 *Non-Descriptor Mode*

The GDMA directly transfers data between source and destination. The GDMA starts to transfer data after it receives service requests from software. When the entire data have been transferred completely, the GDMA becomes idle. Nevertheless, if another transfer is needed, then the GDMA must be programmed again. There are three transfer modes:

#### 5.9.5.2 *Single Mode*

Single mode requires a GDMA request for each data transfer. A GDMA request causes one byte, one half-word, or one word to transfer if the 4-data burst mode is disabled, or four times of transfer width is the 4-data burst mode is enabled.

#### 5.9.5.3 *Block Mode*

The assertion of a single GDMA request causes all of the data to be transferred in a single operation. The GDMA transfer is completed when the current transfer count register reaches zero.

#### 5.9.5.4 *Descriptor Mode*

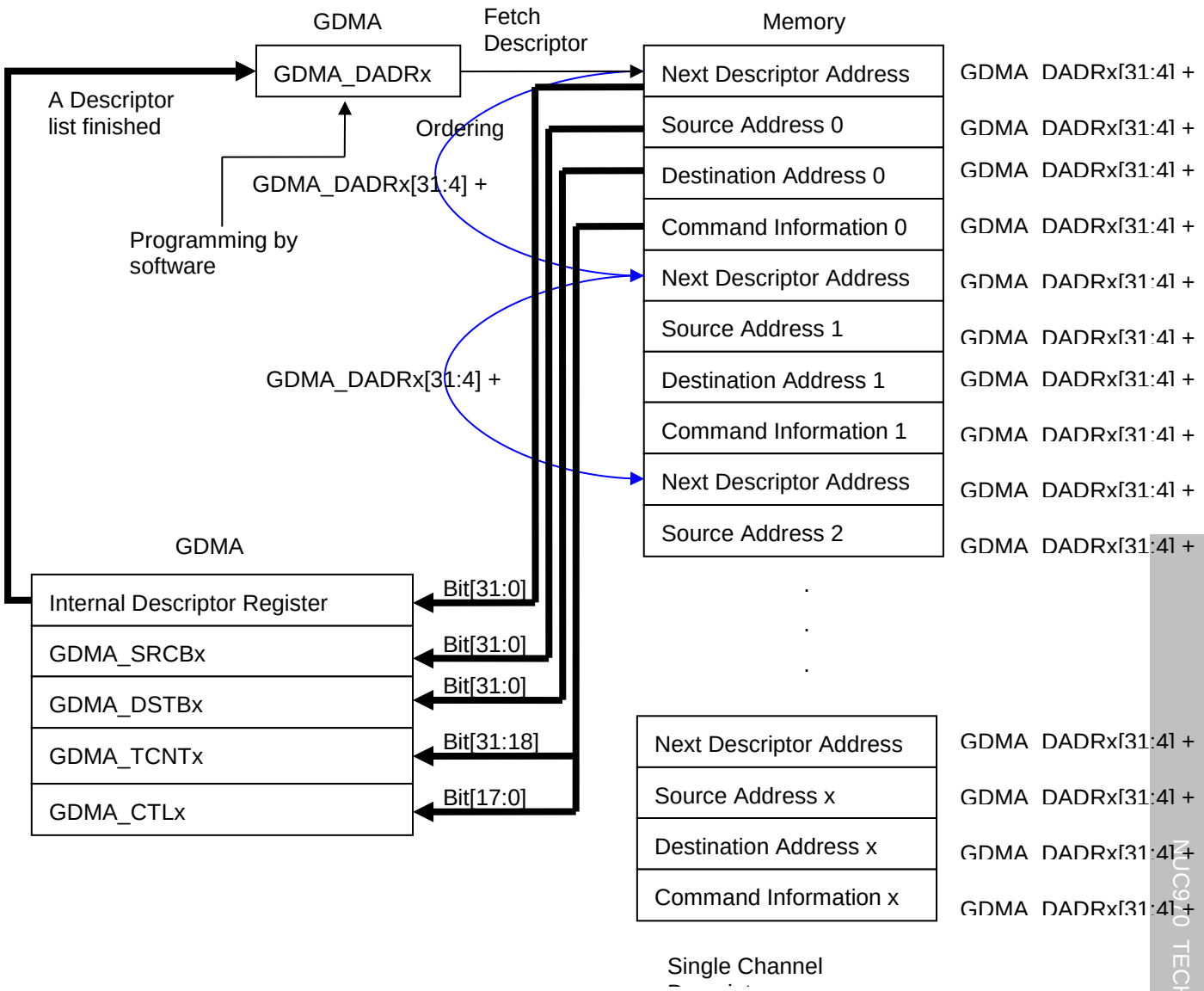
The descriptor-fetch function works when RUN (GDMA_DADRx[3]) is set and NON_DSPTR (GDMA_DADRx[2]) is cleared in Descriptor Register(GDMA_DADRx) and the GDMA_CTLx bit setting as following table. The Non-descriptor-fetch function works when software triggers the SOFTREQ (GDMA_CTLx[16]) and the GDMAEN (GDMA_CTLx[0]). Software can also be used to restart the GDMA operation after it has been stopped. The CPU can recognize the completion of a GDMA operation by software polling or when it receives an internal GDMA interrupt. The GDMA controller can increase source or destination address, decrease them as well, and conduct 8-bit (byte), 16-bit (half-word), or 32-bit (word) data transfers.

Operation Mode relevant to bit enable

| Mode | Enable Bit |
|---|---|
| Non-Descriptor Mode with SW Enable | GDMA_CTLx : gdmaen[0] softreq[16] gdmams[3:2] |
| Non-Descriptor Mode with I/O Enable | GDMA_CTLx : gdmaen[0] gdmams[3:2] |
| Descriptor Mode with SW Enable | GDMA_DADRx : run[3] non-dsptrmode[2];<br>GDMA_CTLx in Descriptor List : gdmaen[0] gdmams[3:2] |
| Descriptor Mode with I/O Enable | GDMA_DADRx : run[3] non-dsptrmode[2];<br>GDMA_CTLx in Descriptor List : gdmaen[0] gdmams[3:2] |

5.9.5.5 *Descriptor Fetch Function*

The Illustration of Descriptor list fetches:




Descriptor-based function (GDMA_DADRx[NON_DSPTRMODE] = 0) operate in the following condition:

5.9.5.6 *Memory to Memory*

1. Software can write a value 0x04 to current GDMA_DADRx register to reset the register and disable Descriptor based function first.
2. Then software can program the bits of [Descriptor Address], [RUN], [NON_DSPTRMODE] and [ORDEN] to the GDMA_DADRx register to enable Descriptor based function. (The

Descriptor can only work when the [RUN][3] is set and [NON_DSPTRMODE][2] bit is cleared properly.)

3. After sets current GDMA_DADRx register, the GDMA will fetch four-word information from memory immediately which contains the next Descriptor address, Source Address, Destination Address and Command information. (Command information consists of control and counter registers)

**NOTE:** GDMA will read the descriptor list from memory such the diagram above and write back to GDMA internal register (next GDMA_DADRx), GDMA_SRCBx, GDMA_DSTBx, GDMA_CTLx and GDMA_TCNTx registers. The most important one of write back is command information, which will separate some bits of command information into control and counter registers respectively. The first fourteen bits of the MSB of the Command information in Descriptor list will be written back to GDMA_TCNTx register, and the others bits of the Command information will be written back to GDMA_CTLx register. The control register part of the Command information will update the GDMA_CTLx register during every descriptor fetch. The allocation of command information is described at GDMA Register Description.

The Allocation of Command Information in Descriptor List:

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GDMA_TCNTx[13:6] ( Command Info[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GDMA_TCNTx[5:0] ( Command Info[23:18] | | | | | | BLOCK | SOFTREQ |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | Reserved | TWS | | Reserved | D_INTS | Reserved | Reserved |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SAFIX | DAFIX | SADIR | DADIR | GDMAMS | | BME | GDMAEN |

4. GDMA will depend on the information to request a bus ownership and start the data transfer when GDMA has gotten a bus grant from the arbiter, otherwise, it will wait until get bus grant. The data transfer direction is dependent on the Control register.
5. The GDMA transfers data and releases bus at every burst transfer. The GDMA will stop transfer for current descriptor when the counter is decreased to zero. The current GDMA_DADRx will be updated by next GDMA_DADRx at end of each descriptor transfer.
6. The GDMA is running consecutively unless the next GDMA_DADRx[RUN] bit is zero or interrupt status bit of GDMA_INTCS register is cleared. The CPU can recognize the completion of a GDMA descriptor fetch operation by polling the current GDMA_DADRx[NON_DSPTRMODE] bit or set the GDMA_CTLx[D_INTS] to receive a interrupt from GDMA.(**Note:** The recommendation is the [NON_DSPTRMODE] bit in list is set at the same time)
7. When an error occurs in the descriptor operation, GDMA will clear [RUN] bit and stop channel operation immediately. Software can reset the channel, and sets the current GDMA_DADRx[RUN] register to start again.

#### 5.9.5.7 *Ordering function in Descriptor fetch mode*

This function determines the source of next descriptor address. If [ORDEN] is set, the GDMA controller fetches the next descriptor from current GDMA_DADRx[Descriptor Address] + 16 bytes.

If this bit is cleared, GDMA fetches the next descriptor from the current GDMA_DADRx[Descriptor Address] .

GDMA_DADRx[ORDEN] is only relevant to descriptor-fetch function (GDMA_DADRx[NON_DSPTRMODE] = 0).

#### 5.9.5.8 *Channel Reset*

The Channel reset is turned on when the bit-0 of GDMA_DADRx is set. This function will clear all status and stop the descriptor based function relative to individual channel. The GDMA_DADRx register value is 0x05h when reset bit is set.

#### 5.9.5.9 *Non-Descriptor Fetch Function*

The non-descriptor-fetch function will take place when current GDMA_DADRx[NON_DSPTRMODE] is set and the GDMA_DADRx register will have no any intention for the GDMA controller.

The default value of GDMA_DADRx is 0x04. Software can clear GDMA_DADRx with value 0x04 as well. In this mode, software should write a valid source address to the GDMA_SRCBx register, a destination address to the GDMA_DSTBx register, and a transfer count to the GDMA_TCNTx register. Next, the GDMA_CTLx of [gdmaen] and [softreq] bits must be set. A non-descriptor fetch is performed when bus granted. After transferring a number of bytes or words correspond with burst mode or not, the channel either waits for the next request or continues with the data transfer until the GDMA_CTCNTx reaches zero. When GDMA_CTCNTx reaches zero, the channel stops operation.

When an error occurs during the GDMA operation, the channel stops unless software clears the error condition and sets the GDMA_CTLx of [gdmaen] and [softreq] bits field to start again.

### 5.9.6 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 |
|---|---|---|---|---|
| **GDMA Base Address:**<br>**GDMA_BA = 0xB000_4000** | | | | |
| **GDMA_CTL0** | GDMA_BA+0x000 | R/W | Channel 0 Control Register | 0x0000_0000 |
| **GDMA_SRCBA0** | GDMA_BA+0x004 | R/W | Channel 0 Source Base Address Register | 0x0000_0000 |
| **GDMA_DSTBA0** | GDMA_BA+0x008 | R/W | Channel 0 Destination Base Address Register | 0x0000_0000 |
| **GDMA_TCNT0** | GDMA_BA+0x00C | R/W | Channel 0 Transfer Count Register | 0x0000_0000 |
| **GDMA_CSRCA0** | GDMA_BA+0x010 | R | Channel 0 Current Source Address Register | 0x0000_0000 |
| **GDMA_CDSTA0** | GDMA_BA+0x014 | R | Channel 0 Current Destination Address Register | 0x0000_0000 |
| **GDMA_CTCNT0** | GDMA_BA+0x018 | R | Channel 0 Current Transfer Count Register | 0x0000_0000 |
| **GDMA_DADR0** | GDMA_BA+0x01C | R/W | Channel 0 Descriptor Address Register | 0x0000_0004 |
| **GDMA_CTL1** | GDMA_BA+0x020 | R/W | Channel 1 Control Register | 0x0000_0000 |
| **GDMA_SRCBA1** | GDMA_BA+0x024 | R/W | Channel 1 Source Base Address Register | 0x0000_0000 |
| **GDMA_DSTBA1** | GDMA_BA+0x028 | R/W | Channel 1 Destination Base Address Register | 0x0000_0000 |
| **GDMA_TCNT1** | GDMA_BA+0x02C | R/W | Channel 1 Transfer Count Register | 0x0000_0000 |
| **GDMA_CSRCA1** | GDMA_BA+0x030 | R | Channel 1 Current Source Address Register | 0x0000_0000 |
| **GDMA_CDSTA1** | GDMA_BA+0x034 | R | Channel 1 Current Destination Address Register | 0x0000_0000 |
| **GDMA_CTCNT1** | GDMA_BA+0x038 | R | Channel 1 Current Transfer Count Register | 0x0000_0000 |
| **GDMA_DADR1** | GDMA_BA+0x03C | R/W | Channel 1 Descriptor Address Register | 0x0000_0004 |
| **GDMA_BUFFER0** | GDMA_BA+0x080 | R | GDMA Internal Buffer Word 0 Register | 0x0000_0000 |
| **GDMA_BUFFER1** | GDMA_BA+0x084 | R | GDMA Internal Buffer Word 1 Register | 0x0000_0000 |
| **GDMA_BUFFER2** | GDMA_BA+0x088 | R | GDMA Internal Buffer Word 2 Register | 0x0000_0000 |
| **GDMA_BUFFER3** | GDMA_BA+0x08C | R | GDMA Internal Buffer Word 3 Register | 0x0000_0000 |
| **GDMA_BUFFER4** | GDMA_BA+0x090 | R | GDMA Internal Buffer Word 4 Register | 0x0000_0000 |
| **GDMA_BUFFER5** | GDMA_BA+0x094 | R | GDMA Internal Buffer Word 5 Register | 0x0000_0000 |
| **GDMA_BUFFER6** | GDMA_BA+0x098 | R | GDMA Internal Buffer Word 6 Register | 0x0000_0000 |
| **GDMA_BUFFER7** | GDMA_BA+0x09C | R | GDMA Internal Buffer Word 7 Register | 0x0000_0000 |
| **GDMA_INTS** | GDMA_BA+0x0A0 | R/W | GDMA Interrupt Control and Status Register | 0x0000_0000 |

### 5.9.7 Register Description

### Channel 0/1 Control Register (GDMA_CTL0, GDMA_CTL1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GDMA_CTL0** | GDMA_BA+0x000 | R/W | Channel 0 Control Register | 0x0000_0000 |
| **GDMA_CTL1** | GDMA_BA+0x020 | R/W | Channel 1 Control Register | 0x0000_0000 |

The control registers has two formats for descriptor fetch and non-descriptor fetch function respectively. The functionality of each control bit is described in following table.

1. Non-Descriptor fetches Mode

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | SABNDERR | DABNDERR | Reserved | AUTOIEN | Reserved | BLOCK | SOFTREQ |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | TWS | | SBMS | Reserved | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SAFIX | DAFIX | SADIR | DADIR | GDMAMS | | BME | GDMAEN |

2. Descriptor fetches Mode

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | SABNDERR | DABNDERR | Reserved | | | BLOCK | SOFTREQ |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | TWS | | Reserved | D_INTS | Reserved | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SAFIX | DAFIX | SADIR | DADIR | GDMAMS | | BME | GDMAEN |

**NOTE:**

The [SABNDERR], [DABNDERR] and [GDMAERR] can be read at descriptor fetch mode.

Regardless of GDMA operate in descriptor mode or non-descriptor mode, when transfer width is 16-bit (half word) and the address with decrement function enable for starting source address or destination address or both are used should set the least two bit of addresses is 0xF.

Control Register of Non-Descriptor fetches Mode:

| Bits | Description | |
|---|---|---|
| [31:23] | **Reserved** | Reserved. |
| [22] | **SABNDERR** | **Source Address Boundary Alignment Error Flag**<br>If TWS [13:12]=10, GDMA_SRCB [1:0] should be 00.<br>If TWS [13:12]=01, GDMA_SRCB [0] should be 0.<br>Except the SADIR function enabled.<br>The address boundary alignment should be depended on TWS [13:12].<br>0 = the GDMA_SRCB is on the boundary alignment.<br>1 = the GDMA_SRCB not on the boundary alignment.<br>The SABNDERR register bits just can be read only. |
| [21] | **DABNDERR** | **Destination Address Boundary Alignment Error Flag**<br>If TWS [13:12]=10, GDMA_DSTB [1:0] should be 00.<br>If TWS [13:12]=01, GDMA_DSTB [0] should be 0.<br>Except the SADIR function enabled.<br>The address boundary alignment should be depended on TWS [13:12].<br>0 = the GDMA_DSTB is on the boundary alignment.<br>1 = the GDMA_DSTB not on the boundary alignment.<br>The DABNDERR register bits just can be read only. |
| [20] | **Reserved** | Reserved. |
| [19] | **AUTOIEN** | **Auto Initialization Enable**<br>0 = Disables auto initialization.<br>1 = Enables auto initialization, the GDMA_CSRC0/1,GDMA_CDST0/1,and GDMA_CTCNT0/1 registers are updated by the GDMA_SRC0/1, GDMA_DST0/1, and GDMA_TCNT0/1 registers automatically when transfer is complete.<br>GDMA will start another transfer when SOFTREQ set again. |
| [18] | **Reserved** | Reserved. |
| [17] | **BLOCK** | **Bus Lock**<br>0 = Unlocks the bus during the period of transfer.<br>1 = locks the bus during the period of transfer. |
| [16] | **SOFTREQ** | **Software Triggered GDMA Request**<br>Software can request the GDMA transfer service by setting this bit to 1. This bit is automatically cleared by hardware when the transfer is completed. This bit is available only while GDMAMS [3:2] register bits are set on software mode (memory to memory and memory to I/O). |
| [15:14] | **Reserved** | Reserved. |
| [13:12] | **TWS** | **Transfer Width Select**<br>00 = One byte (8 bits) is transferred for every GDMA operation.<br>01 = One half-word (16 bits) is transferred for every GDMA operation.<br>10 = One word (32 bits) is transferred for every GDMA operation.<br>11 = Reserved.<br>The GDMA_SCRB and GDMA_DSTB should be alignment under the TWS selection |

| | | |
|---|---|---|
| [11] | **SBMS** | **Single/Block Mode Select**<br>0 = Selects single mode. It requires an external GDMA request for every incurring GDMA operation.<br>1 = Selects block mode. It requires a single external GDMA request during the atomic GDMA operation. An atomic GDMA operation is defined as the sequence of GDMA operations until the transfer count register reaches zero. |
| [10:8] | **Reserved** | Reserved. |
| [7] | **SAFIX** | **Source Address Fixed**<br>0 = Source address is changed during the GDMA operation.<br>1 = Do not change the destination address during the GDMA operation. This feature can be used when data were transferred from a single source to multiple destinations. |
| [6] | **DAFIX** | **Destination Address Fixed**<br>0 = Destination address is changed during the GDMA operation.<br>1 = Do not change the destination address during the GDMA operation. This feature can be used when data were transferred from multiple sources to a single destination. |
| [5] | **DADIR** | **Source Address Direction**<br>0 = Source address is incremented successively.<br>1 = Source address is decremented successively. |
| [4] | **DADIR** | **Destination Address Direction**<br>0 = Destination address is incremented successively.<br>1 = Destination address is decremented successively. |
| [3:2] | **GDMAMS** | **GDMA Mode Select**<br>00 = Software mode (memory-to-memory).<br>01 = Reserved.<br>10 = Reserved.<br>11 = Reserved. |
| [1] | **BME** | **Burst Mode Enable**<br>0 = Disables the 8-data burst mode.<br>1 = Enables the 8-data burst mode.<br>FF there are 8 words to be transferred, and BME [1]=1, the GDMA_TCNTx should be 0x01;.<br>However, if BME [1]=0, the GDMA_TCNTx should be 0x08. |
| [0] | **GDMAEN** | **GDMA Enable**<br>0 = Disables the GDMA operation.<br>1 = Enables the GDMA operation; this bit will be clear automatically when the transfer is complete on AUTOIEN [19] register bit is on Disable mode.<br>**Note:**<br>when operate in Non-Descriptor mode, this bit determine the Memory-to Memory, Memory-to-I/O and I/O-to-Memory operation or not.<br>When operate in Descriptor mode, this bit is determined in descriptor list.<br>**Note:** Channel reset will clear this bit. |

Descriptor fetches mode of Control Register:

| Bits | Description | |
|---|---|---|
| [31:23] | **Reserved** | Reserved. |
| [22] | **SABNDERR** | **Source Address Boundary Alignment Error Flag**<br>If TWS [13:12]=10, GDMA_SRCB [1:0] should be 00.<br>If TWS [13:12]=01, GDMA_SRCB [0] should be 0.<br>Except the SADIR function enabled.<br>The address boundary alignment should be depended on TWS [13:12].<br>0 = the GDMA_SRCB is on the boundary alignment.<br>1 = the GDMA_SRCB not on the boundary alignment.<br>The SABNDERR register bits just can be read only. |
| [21] | **DABNDERR** | **Destination Address Boundary Alignment Error Flag**<br>If TWS [13:12]=10, GDMA_DSTB [1:0] should be 00.<br>If TWS [13:12]=01, GDMA_DSTB [0] should be 0.<br>Except the DADIR function enabled.<br>The address boundary alignment should be depended on TWS [13:12].<br>0 = the GDMA_DSTB is on the boundary alignment.<br>1 = the GDMA_DSTB not on the boundary alignment.<br>The DABNDERR register bits just can be read only. |
| [20:18] | **Reserved** | Reserved. |
| [17] | **BLOCK** | **Bus Lock**<br>0 = Unlocks the bus during the period of transfer.<br>1 = locks the bus during the period of transfer. |
| [16] | **SOFTREQ** | **Software Triggered GDMA Request**<br>Software can request the GDMA transfer service by setting this bit to 1. This bit is automatically cleared by hardware when the transfer is completed. This bit is available only while GDMAMS [3:2] register bits are set on software mode (memory to memory and memory to I/O). |
| [15:14] | **Reserved** | Reserved. |
| [13:12] | **TWS** | **Transfer Width Select**<br>00 = One byte (8 bits) is transferred for every GDMA operation.<br>01 = One half-word (16 bits) is transferred for every GDMA operation.<br>10 = One word (32 bits) is transferred for every GDMA operation.<br>11 = Reserved.<br>The GDMA_SCRB and GDMA_DSTB should be alignment under the TWS selection |
| [11] | **Reserved** | Reserved. |
| [10] | **D_INTS** | **Descriptor Fetch Mode Interrupt Select**<br>0 = The interrupt will take place at every end of descriptor fetch transfer.<br>1 = The interrupt only take place at the last descriptor fetch transfer.<br>**NOTE:** this bit is only available in descriptor mode and lists intention. |
| [9:8] | **Reserved** | Reserved. |
| [7] | **SAFIX** | **Source Address Fixed**<br>0 = Source address is changed during the GDMA operation.<br>1 = Do not change the source address during the GDMA operation. This feature can be used when data were transferred from a single source to multiple destinations. |

| | | |
|---|---|---|
| [6] | **DAFIX** | **Destination Address Fixed**<br>0 = Destination address is changed during the GDMA operation.<br>1 = Do not change the destination address during the GDMA operation. This feature can be used when data were transferred from multiple sources to a single destination. |
| [5] | **SADIR** | **Source Address Direction**<br>0 = Source address is incremented successively.<br>1 = Source address is decremented successively. |
| [4] | **DADIR** | **Destination Address Direction**<br>0 = Destination address is incremented successively.<br>1 = Destination address is decremented successively. |
| [3:2] | **GDMAMS** | **GDMA Mode Select**<br>00 = Software mode (Memory-to-Memory).<br>01 = Reserved.<br>10 = Reserved.<br>11 = Reserved. |
| [1] | **BME** | **Burst Mode Enable**<br>0 = Disables the 8-data burst mode.<br>1 = Enables the 8-data burst mode.<br>FF there are 8 words to be transferred, and BME [1]=1, the GDMA_TCNT should be 0x01;.<br>However, if BME [1]=0, the GDMA_TCNT should be 0x08. |
| [0] | **GDMAEN** | **GDMA Enable**<br>0 = Disables the GDMA operation.<br>1 = Enables the GDMA operation; this bit will be clear automatically when the transfer is complete on AUTOIEN [19] register bit is on Disable mode.<br>when operate in Non-Descriptor mode, this bit determine the Memory-to-Memory, Memory-to-I/O and I/O-to-Memory operation or not.<br>When operate in Descriptor mode, this bit determines the I/O-to-Memory operation or not.<br>Channel reset will clear this bit. |

### Channel 0/1 Source Base Address Register (GDMA_SRCBA0, GDMA_SRCBA1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GDMA_SRCBA0** | GDMA_BA+0x004 | R/W | Channel 0 Source Base Address Register | 0x0000_0000 |
| **GDMA_SRCBA1** | GDMA_BA+0x024 | R/W | Channel 1 Source Base Address Register | 0x0000_0000 |

| 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** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:0] | **BADDR** | **Source Base Address**<br>The GDMA channel starts reading its data from the source address as defined in this source base address register. |

### Channel 0/1 Destination Base Address Register (GDMA_DSTBA0, GDMA_DSTBA1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GDMA_DSTBA0** | GDMA_BA+0x008 | R/W | Channel 0 Destination Base Address Register | 0x0000_0000 |
| **GDMA_DSTBA1** | GDMA_BA+0x028 | R/W | Channel 1 Destination Base Address Register | 0x0000_0000 |

| 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** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:0] | **BADDR** | **Destination Base Address**<br>The GDMA channel starts writing its data to the destination address as defined in this destination base address register. During a block transfer, the GDMA determines successive destination addresses by adding to or subtracting from the destination base address. |

### Channel 0/1 Transfer Count Register (GDMA_TCNT0, GDMA_TCNT1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GDMA_TCNT0** | GDMA_BA+0x00C | R/W | Channel 0 Transfer Count Register | 0x0000_0000 |
| **GDMA_TCNT1** | GDMA_BA+0x02C | R/W | Channel 1 Transfer Count Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **COUNT** | **Transfer Count**<br>Non-Descriptor Mode:24-bit TFR_CNT [23:0]<br>The TFR_CNT represents the required number of GDMA transfers. The maximum transfer count is 16M –1.<br><br>Descriptor Mode: 14-bit TFR_CNT [13:0]<br>The TFR_CNT represents the required number of GDMA transfers. The maximum transfer count is 16K –1. |

## Channel 0/1 Current Source Address Register (GDMA_CSRCA0, GDMA_CSRCA1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GDMA_CSRCA0** | GDMA_BA+0x010 | R | Channel 0 Current Source Address Register | 0x0000_0000 |
| **GDMA_CSRCA1** | GDMA_BA+0x030 | R | Channel 1 Current Source Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **CADDR** | **Current Source Address**<br>The CURRENT_SRC_ADDR indicates the source address where the GDMA transfer is just occurring. During a block transfer, the GDMA determines the successive source addresses by adding to or subtracting from the source base address. Depending on the settings you make to the control register, the current source address will remain the same or will be incremented or decremented. |

## Channel 0/1 Current Destination Address Register (GDMA_CDSTA0, GDMA_CDSTA1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GDMA_CDSTA0** | GDMA_BA+0x014 | R | Channel 0 Current Destination Address Register | 0x0000_0000 |
| **GDMA_CDSTA1** | GDMA_BA+0x034 | R | Channel 1 Current Destination Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **CADDR** | **Current Destination Address**<br>The CURRENT_DST_ADDR indicates the destination address where the GDMA transfer is just occurring. During a block transfer, the GDMA determines the successive destination addresses by adding to or subtracting from the destination base address. Depending on the settings you make to the control register, the current destination address will remain the same or will be incremented or decremented. |

### Channel 0/1 Current Transfer Count Register (GDMA_CTCNT0, GDMA_CTCNT1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GDMA_CTCNT0** | GDMA_BA+0x018 | R | Channel 0 Current Transfer Count Register | 0x0000_0000 |
| **GDMA_CTCNT1** | GDMA_BA+0x038 | R | Channel 1 Current Transfer Count Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [23:0] | **CCNT** | **Current Transfer Count**<br>The Current transfer count register indicates the number of transfer being performed.<br>Non-Descriptor Mode: 24-bit CURENT_TFR_CNT [23:0]<br>Descriptor Mode: 14-bit CURENT_TFR_CNT [13:0] |

## Channel 0/1 Descriptor Register (GDMA_DADR0/1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GDMA_DADR0** | GDMA_BA+0x01C | R/W | Channel 0 Descriptor Address Register | 0x0000_0004 |
| **GDMA_DADR1** | GDMA_BA+0x03C | R/W | Channel 1 Descriptor Address Register | 0x0000_0004 |

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

| Bits | Description | |
|---|---|---|
| [31:4] | **ADDR** | **Descriptor Address**<br>Contains address of next descriptor. |
| [3] | **RUN** | **Run**<br>The RUN bit can be cleared during descriptor data transfer, and set RUN bit to starts the stopped channel under [Descriptor Address] and [Non-DSPTRMODE] bits are set properly. When RUN bit is cleared and the NON_DSPTRMODE bit is set that non-descriptor fetch occurs whether a valid descriptor address is written to register GDMA_DADRx or not. This bit will reset automatically when each descriptor transfer stopped or the bit in descriptor list is zero. The Descriptor interrupt is determined by bit-10 of the GDMA_CTLx Register.<br>0 = Stops the channel.<br>1 = Starts the channel.<br>**Note:** must co-operate to [NON_DSPTRMODE] to start the channel with Descriptor fetch function. |
| [2] | **NON_DSPTR** | **Non-descriptor-fetch**<br>When NON_DSPTRMODE is set, the channel is considered as a channel with no descriptors. In this mode, the GDMA does not initiate descriptor fetching and software can program the SCRBx, DSTBx,CTRx and TCNTx registers to transfer data until the TCNTx reaches zero. The GDMA_DADRx register is not used in non-descriptor mode. If NON_DSPTRMDOE is cleared under [RUN] and [Descriptor Address] are set properly, GDMA controller initiates descriptor-fetching. The descriptor fetch transfer stops when the counter for the current transfer reaches zero, [RUN] bit is cleared and [NON_DSPTRMODE] is set base on the bits of the descriptor list.<br>0 = Descriptor-fetch transfer.<br>1 = NON-descriptor-fetch transfer.<br>**Note:** this bit = 1 will disable Descriptor function regardless of the RUN bit is 1 or not. |
| [1] | **ORDEN** | **Enable Ordering Execution for Descriptor List**<br>The GDMA_DADRx[ORDEN] determine which the next descriptor address will be fetched. If [ORDEN] is set, the GDMA controller fetches the next descriptor from Current GDMA_DADRx[Descriptor Address] + 16 bytes. |

| | | |
|---|---|---|
| | | If this bit is cleared, GDMA fetches the next descriptor address from the current GDMA_DADRx[Descriptor Address] register.<br>GDMA_DADRx[ORDEN] is relevant only for descriptor-fetch function (GDMA_DADRx[NON_DSPTRMODE] = 0).<br>0 = Disable descriptor ordering. Fetch the next descriptor from register GDMA_DDADRx[Descriptor Address].<br>1 = Enable descriptor ordering. |
| [0] | **RESET** | **Reset Channel**<br>0 = Disable channel reset.<br>1 = Enable channel status reset and disable descriptor based function. |

### GDMA Internal Buffer Word Register (GDMA_BUFFER)

Softeware can set the [17-16] bit of GDMA_INTCS to select channels and watch the value which has read from memory.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GDMA_BUFFER0** | GDMA_BA+0x080 | R | GDMA Internal Buffer Word 0 Register | 0x0000_0000 |
| **GDMA_BUFFER1** | GDMA_BA+0x084 | R | GDMA Internal Buffer Word 1 Register | 0x0000_0000 |
| **GDMA_BUFFER2** | GDMA_BA+0x088 | R | GDMA Internal Buffer Word 2 Register | 0x0000_0000 |
| **GDMA_BUFFER3** | GDMA_BA+0x08C | R | GDMA Internal Buffer Word 3 Register | 0x0000_0000 |
| **GDMA_BUFFER4** | GDMA_BA+0x090 | R | GDMA Internal Buffer Word 4 Register | 0x0000_0000 |
| **GDMA_BUFFER5** | GDMA_BA+0x094 | R | GDMA Internal Buffer Word 5 Register | 0x0000_0000 |
| **GDMA_BUFFER6** | GDMA_BA+0x098 | R | GDMA Internal Buffer Word 6 Register | 0x0000_0000 |
| **GDMA_BUFFER7** | GDMA_BA+0x09C | R | GDMA Internal Buffer Word 7 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** | **Internal Buffer Register**<br>Each channel has its own internal buffer from Word 0 to Word 7. The [17-16] bit of GDMA_INTCS will determine the values of channels mapping to GDMA_INTBUF0~7.<br>**NOTE:** The GDMA_INTBUF0~7 are available when burst mode used, otherwise, only the GDMA_INTBUF0 available. |

## GDMA Interrupt Control and Status Register (GDMA_INTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GDMA_INTS** | GDMA_BA+0x0A0 | R/W | GDMA Interrupt Control and Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | **BUF_RD_SEL** | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | **TERR1F** | **TC1F** | **TERR0F** | **TC0F** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | | **TERR1EN** | **TC1EN** | **TERR0EN** | **TC0EN** |

| Bits | Description | |
|---|---|---|
| [31:18] | **Reserved** | Reserved. |
| [17:16] | **BUF_RD_SEL** | **Internal Buffer Read Select**<br>00 = Read Internal Buffer for Channel 0.<br>01 = Read Internal Buffer for Channel 1.<br>10 = RESERVED.<br>11 = RESERVED. |
| [15:12] | **Reserved** | Reserved. |
| [11] | **TERR1F** | **Channel 1 Transfer Error**<br>O = No error occurs.<br>1 = Hardware sets this bit on a GDMA transfer failure.<br>This bit will be cleared when write logic 1.<br>Transfer error will generate GDMA interrupt |
| [10] | **TC1F** | **Channel 1 Terminal Count**<br>0 = Channel does not expire.<br>1 = Channel expires; this bit is set only by GDMA hardware, and clear by software to write logic 1.<br>TC1 is the GDMA interrupt flag. TC1 or GDMATERR1 will generate interrupt |
| [9] | **TERR0F** | **Channel 0 Transfer Error**<br>O = No error occurs.<br>1 = Hardware sets this bit on a GDMA transfer failure.<br>This bit will be cleared when write logic 1.<br>Transfer error will generate GDMA interrupt |

| | | |
|---|---|---|
| [8] | **TC0F** | **Channel 0 Terminal Count**<br>0 = Channel does not expire.<br>1 = Channel expires; this bit is set only by GDMA hardware, and clear by software to write logic 1.<br>TC0 is the GDMA interrupt flag. TC0 or GDMATERR0 will generate interrupt |
| [7:4] | **Reserved** | Reserved. |
| [3] | **TEER1EN** | **Channel 1 Interrupt Enable for Transfer Error**<br>0 = Disable Interrupt.<br>1 = Enable Interrupt. |
| [2] | **TC1EN** | **Channel 1 Interrupt Enable for Terminal Count**<br>0 = Disable Interrupt.<br>1 = Enable Interrupt. |
| [1] | **TEER0EN** | **Channel 0 Interrupt Enable for Transfer Error**<br>0 = Disable Interrupt.<br>1 = Enable Interrupt. |
| [0] | **TC0EN** | **Channel 0 Interrupt Enable for Terminal Count**<br>0 = Disable Interrupt.<br>1 = Enable Interrupt. |

## 5.10 Timer Controller (TMR)

### 5.10.1 Overview

The general timer controller includes five channels, TIMER0, TIMER1, TIMER2, TIMER3, and TIMER4, which allow user to easily implement a counting scheme or timing control for applications. The timer possesses features such as adjustable resolution, and programmable counting period. The timer can generate an interrupt signal upon timeout, or provide the current value of count during operation.

### 5.10.2 Features

- Independent Clock Source for each Timer channel (TMRx_CLK, x= 0, 1, 2, 3, 4).
- Five channels with a 24-bit up counter and an interrupt request each.
- Internal 8-bit pre-scale counter.
- Internal 24-bit up counter is readable through Timer Data Register, TDR (TMR_DR[23:0]).
- Supports One-shot, Periodic, and Continuous operation mode.
- Time-out period = (Period of timer clock input) * (8-bit pre-scale counter + 1) * (24-bit TCMP setting value).
- Maximum counting time = $\frac{1}{12\ \text{MHz}} \times 256 \times (2^{24} - 1)$, if TMRx_CLK = 12 MHz.

### 5.10.3 Block Diagram

Each timer is equipped with an 8-bit pre-scale counter, a 24-bit up-counter, a 24-bit compare register and an interrupt request signal.

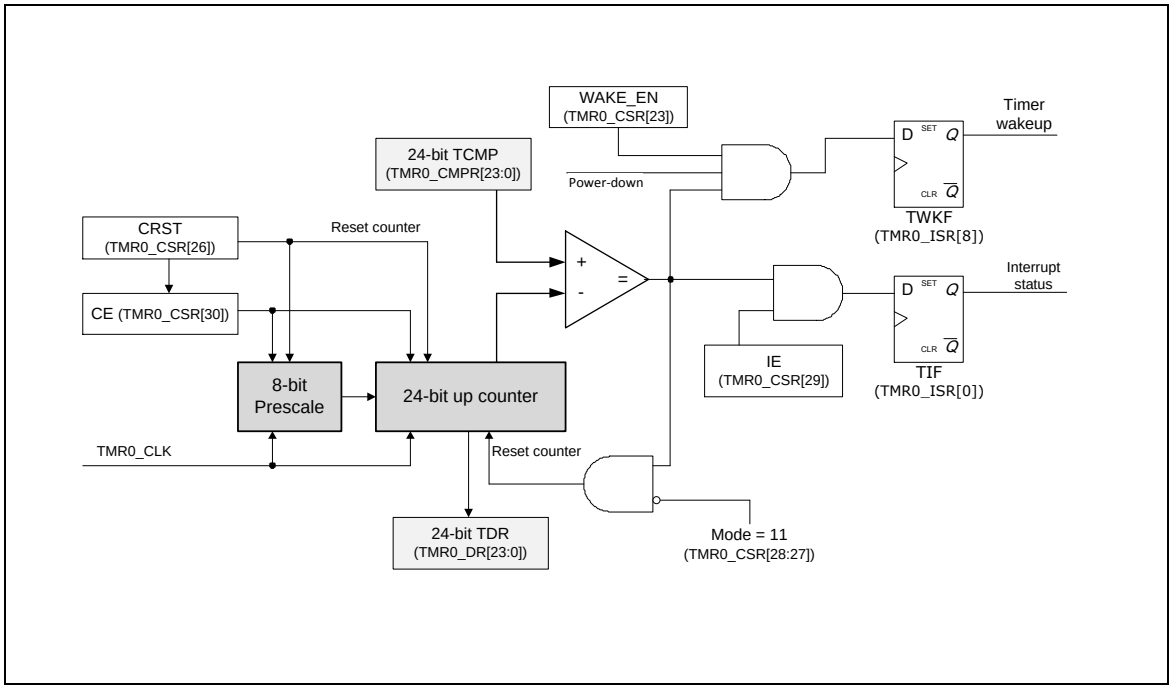



Figure 5.10-1 Timer Controller Block Diagram

### 5.10.4 Basic Configuration

Before using Timer, it's necessary to enable clock of Timer. Set TIMER0 (CLK_PCLKEN0[8]), TIMER1 (CLK_PCLKEN0[9]), TIMER2 (CLK_PCLKEN0[10]), TIMER3 (CLK_PCLKEN0[11]), TIMER4 (CLK_PCLKEN0[12]) high to enable clock of each Timer.

### 5.10.5 Functional Description

Timer controller provides One-shot, Periodic, and Continuous operation modes. Each operating function mode is shown as follows:

5.10.5.1 *One-Shot Mode*

If the timer is operated in One-shot mode when MODE (TMRx_CSR[28:27]) is 0x0 and the timer counter enable bit CE (TMRx_CSR[30]) is set to 1, the timer counter starts up counting. Once the timer counter value TDR (TMRx_DR[23:0]) reaches timer compare register TCMP (TMRx_CMPR[23:0]) value, the timer interrupt signal will be asserted. If the timer interrupt signal is asserted and the timer interrupt enable bit IE (TMRx_CSR[29]) is set to 1, the timer interrupt flag TIF (TMRx_ISR[0]) will be asserted by hardware and then software can write 1 into TIF (TMRx_ISR[0]) bit to clear it.

In this operating mode, once the timer counter value TDR (TMRx_DR[23:0]) reaches timer compare register TCMP (TMRx_CMPR[23:0]) value will set the timer interrupt flag TIF (TMR_ISR[0]) to 1, then timer counting operation stops and the timer counter value TDR (TMRx_DR[23:0]) goes back to

counting initial value then timer counter enable bit CE (TMRx_CSR[30]) is cleared to 0 by timer controller automatically. That is to say, timer operates timer counting and compares with TCMP (TMRx_CMPR[23:0]) value function only one time after programming the timer compare register TCMP (TMRx_CMPR[23:0]) value and timer counter enable bit CE (TMRx_CSR[30]) is set to 1. So, this operating mode is called One-shot mode.

#### 5.10.5.2 *Periodic Mode*

If the timer is operated in Periodic mode (MODE (TMRx_CSR[28:27]) is 0x1) and timer counter enable bit CE (TMRx_CSR[30]) is set to 1, the timer counter starts up counting. Once the timer counter value (TMRx_DR) reaches timer compare register (TMRx_CMPR) value, the interrupt signal will be asserted then timer interrupt flag TIF (TMRx_ISR[0]) will set to 1 if timer interrupt enable bit IE=0 (TMRx_CSR[29]=1). If IE (TMRx_CSR[29]) is set to 0, the timer interrupt flag TIF (TMRx_ISR[0]) will not be set to 1.

In this operating mode, once the timer counter value TDR (TMRx_DR[23:0]) reaches timer compare register TCMP (TMRx_CMPR[23:0]) value and IE (TMRx_CSR[29]) set to 1, timer interrupt flag TIF (TMRx_ISR[0]) will set to 1, the timer counter value TDR (TMRx_DR[23:0]) goes back to counting initial value and timer counter enable bit CE (TMRx_CSR[30]) is kept at 1 (counting enable Periodically) and timer counter operates up counting again. If timer interrupt flag TIF (TMRx_ISR[0]) is cleared by software, once the timer counter value TDR (TMRx_DR[23:0]) reaches timer compare register TCMP (TMRx_CMPR[23:0]) value again, TIF (TMRx_ISR[0]) will set to 1 also. That is to say, timer operates timer counting and compares with TCMP (TMRx_CMPR[23:0]) value function periodically. The timer counting operation does not stop until the timer counter enable bit CE (TMRx_CSR[30]) is set to 0. The interrupt signal is also generated periodically. So, this operating mode is called Periodic mode.

#### 5.10.5.3 *Continuous Mode*

If the timer is operated in Continuous mode (MODE (TMRx_CSR[28:27]) is 0x3) and timer counter enable bit CE (TMRx_CSR[30]) is set to 1, the timer counter starts up counting. The 24-bit timer counter in Continuous mode will keep counting until the counting value reaches to the maximum value (0xFFFFFF), then timer counter value TDR (TMRx_DR[23:0]) will be reset to 0x0 and keep up counting while CE=1 (TMRx_CSR[30]=1).

Once the timer counter value TDR (TMRx_DR[23:0]) is equal to the timer compare register TCMP (TMRx_CMPR[23:0]) value and timer interrupt enable bit IE (TMRx_CSR[29]) is set to 1, the timer interrupt flag TIF (TMRx_ISR[0]) will set to 1. If IE (TMRx_CSR[29]) is set to 0, TIF (TMRx_ISR[0]) will not be asserted. In Continuous mode, changing the value of compare register TCMP (TMRx_CMPR[23:0]) will not affect the current value of TDR (TMRx_DR[23:0]), but it will clear timer counter value TDR (TMRx_DR[23:0]) to 0x0 in other operation modes. User can change the compare register TCMP (TMRx_CMPR[23:0]) at any time and the timer counter will keep up counting continuously to generate the new interrupt event. So, this operating mode is called Continuous mode.

### 5.10.6 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **TMR Base Address:**<br>**TMR0_BA = 0xB800_1000**<br>**TMR1_BA = 0xB800_1010**<br>**TMR2_BA = 0xB800_1020**<br>**TMR3_BA = 0xB800_1030**<br>**TMR4_BA = 0xB800_1040** | | | | |
| **TMR0_CSR** | TMR0_BA+0x000 | R/W | Timer Control and Status Register 0 | 0x0000_0005 |
| **TMR0_CMPR** | TMR0_BA+0x004 | R/W | Timer Compare Register 0 | 0x0000_0000 |
| **TMR0_DR** | TMR0_BA+0x008 | R | Timer Data Register 0 | 0x0000_0000 |
| **TMR1_CSR** | TMR1_BA+0x000 | R/W | Timer Control and Status Register 1 | 0x0000_0005 |
| **TMR1_CMPR** | TMR1_BA+0x004 | R/W | Timer Compare Register 1 | 0x0000_0000 |
| **TMR1_DR** | TMR1_BA+0x008 | R | Timer Data Register 1 | 0x0000_0000 |
| **TMR2_CSR** | TMR2_BA+0x000 | R/W | Timer Control and Status Register 2 | 0x0000_0005 |
| **TMR2_CMPR** | TMR2_BA+0x004 | R/W | Timer Compare Register 2 | 0x0000_0000 |
| **TMR2_DR** | TMR2_BA+0x008 | R | Timer Data Register 2 | 0x0000_0000 |
| **TMR3_CSR** | TMR3_BA+0x000 | R/W | Timer Control and Status Register 3 | 0x0000_0005 |
| **TMR3_CMPR** | TMR3_BA+0x004 | R/W | Timer Compare Register 3 | 0x0000_0000 |
| **TMR3_DR** | TMR3_BA+0x008 | R | Timer Data Register 3 | 0x0000_0000 |
| **TMR4_CSR** | TMR4_BA+0x000 | R/W | Timer Control and Status Register 4 | 0x0000_0005 |
| **TMR4_CMPR** | TMR4_BA+0x004 | R/W | Timer Compare Register 4 | 0x0000_0000 |
| **TMR4_DR** | TMR4_BA+0x008 | R | Timer Data Register 4 | 0x0000_0000 |
| **TMR_ISR** | TMR0_BA+0x060 | R/W | Timer Interrupt Status Register | 0x0000_0000 |

### 5.10.7 Register Description

**Timer Control and Status Register (TMR_CSR)**

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **TMR0_CSR** | TMR0_BA+0x000 | R/W | Timer Control and Status Register 0 | 0x0000_0005 |
| **TMR1_CSR** | TMR1_BA+0x000 | R/W | Timer Control and Status Register 1 | 0x0000_0005 |
| **TMR2_CSR** | TMR2_BA+0x000 | R/W | Timer Control and Status Register 2 | 0x0000_0005 |
| **TMR3_CSR** | TMR3_BA+0x000 | R/W | Timer Control and Status Register 3 | 0x0000_0005 |
| **TMR4_CSR** | TMR4_BA+0x000 | R/W | Timer Control and Status Register 4 | 0x0000_0005 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | CE | IE | MODE | | CRST | CACT | 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 |
| PRESCALE | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31] | **Reserved** | Reserved. |
| [30] | **CE** | **Counter Enable**<br>0 = Stops counting.<br>1 = Starts counting. |
| [29] | **IE** | **Interrupt Enable**<br>0 = Disables timer interrupt.<br>1 = Enables timer interrupt. If timer interrupt is enabled, the timer asserts its interrupt flag when the associated counter decrements to zero. |
| [28:27] | **MODE** | **Timer Operating Mode**<br>00 = One-shot mode. The associated interrupt signal is generated once (if IE is enabled) and CE is automatically cleared then.<br>01 = Periodic mode. The associated interrupt signal is generated periodically (if IE is enabled).<br>10 = Reserved.<br>11 = Continuous mode. The associated interrupt signal is generated continuously (if IE is enabled). |
| [26] | **CRST** | **Counter Reset**<br>Set this bit will reset the TIMER counter, and also force CE to 0.<br>0 = No effect.<br>1 = Reset Timer's prescale counter, internal 24-bit counter and CE. |

| | | |
|---|---|---|
| | | **Note:** This bit will be cleared by hardware automatically. |
| [25] | **CACT** | **Timer Is in Active**<br>This bit indicates the counter status of timer.<br>0 = Timer is not active.<br>1 = Timer is in active. |
| [24:8] | **Reserved** | Reserved. |
| [7:0] | **PRESCALE** | **Clock Pre-scale Divide Count**<br>Clock input is divided by PRESCALE + 1 before it is fed to the counter (here PRESCALE is considered as a decimal number). If PRESCALE = 0, then there is no scaling. |

### Timer Compare Registers (TMR_CMPR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **TMR0_CMPR** | TMR0_BA+0x004 | R/W | Timer Compare Register 0 | 0x0000_0000 |
| **TMR1_CMPR** | TMR1_BA+0x004 | R/W | Timer Compare Register 1 | 0x0000_0000 |
| **TMR2_CMPR** | TMR2_BA+0x004 | R/W | Timer Compare Register 2 | 0x0000_0000 |
| **TMR3_CMPR** | TMR3_BA+0x004 | R/W | Timer Compare Register 3 | 0x0000_0000 |
| **TMR4_CMPR** | TMR4_BA+0x004 | R/W | Timer Compare Register 4 | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [30:24] | **Reserved** | Reserved. |
| [23:0] | **TCMP** | **Timer Compared Value**<br>This is a 24-bit value representing the final count. Timer counter will be cleared to zero whenever the counter is incremented to this setting value.<br>**NOTE:**<br>Never write 0x0 in TCMP, or the core will run into unknown state.<br>Besides the Timer operates in Continuous mode (TMR_CSR[28:27] = 0x3), if software write a new value into this register, the Timer counting value (TMR_DR[23:0]) will be cleared and then restart a new counting with this new compared value even though CE (TMRx_CSR[30]) is 0. |

### Timer Data Registers (TMR_DR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **TMR0_DR** | TMR0_BA+0x008 | R | Timer Data Register 0 | 0x0000_0000 |
| **TMR1_DR** | TMR1_BA+0x008 | R | Timer Data Register 1 | 0x0000_0000 |
| **TMR2_DR** | TMR2_BA+0x008 | R | Timer Data Register 2 | 0x0000_0000 |
| **TMR3_DR** | TMR3_BA+0x008 | R | Timer Data Register 3 | 0x0000_0000 |
| **TMR4_DR** | TMR4_BA+0x008 | R | Timer Data Register 4 | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **TDR** | **Timer Data Register**<br>The current count is registered in this 24-bit value. |

### Timer Interrupt Status Registers (TMR_ISR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **TMR_ISR** | TMR0_BA+0x060 | R/W | Timer 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** | | | **TIF4** | **TIF3** | **TIF2** | **TIF1** | **TIF0** |

| Bits | Description | |
|---|---|---|
| [31:5] | **Reserved** | Reserved. |
| [4] | **TIF4** | **Timer Interrupt Flag 4**<br>0 = It indicates that the timer 4 does not count up to TCMP (TMR4_CMPR[23:0]) value yet. Software can reset this bit after the timer interrupt 4 had occurred.<br>1 = It indicates that the counter of timer 4 is incremented to corresponding TCMP (TMR4_CMPR[23:0]) setting value.<br>**NOTE:** This bit is read only, but can be cleared by writing 1 to this bit. |
| [3] | **TIF3** | **Timer Interrupt Flag 3**<br>0 = It indicates that the timer 3 does not count up to TCMP (TMR3_CMPR[23:0]) value yet. Software can reset this bit after the timer interrupt 3 had occurred.<br>1 = It indicates that the counter of timer 3 is incremented to corresponding TCMP (TMR3_CMPR[23:0]) setting value.<br>**NOTE:** This bit is read only, but can be cleared by writing 1 to this bit. |
| [2] | **TIF2** | **Timer Interrupt Flag 2**<br>0 = It indicates that the timer 2 does not count up to TCMP (TMR2_CMPR[23:0]) value yet. Software can reset this bit after the timer interrupt 2 had occurred.<br>1 = It indicates that the counter of timer 2 is incremented to corresponding TCMP (TMR2_CMPR[23:0]) setting value.<br>**NOTE:** This bit is read only, but can be cleared by writing 1 to this bit. |
| [1] | **TIF1** | **Timer Interrupt Flag 1**<br>0 = It indicates that the timer 1 does not count up to TCMP (TMR1_CMPR[23:0]) value yet. Software can reset this bit after the timer interrupt 1 had occurred.<br>1 = It indicates that the counter of timer 1 is incremented to corresponding TCMP (TMR1_CMPR[23:0]) setting value.<br>**NOTE:** This bit is read only, but can be cleared by writing 1 to this bit. |

| | | |
|---|---|---|
| [0] | **TIF0** | **Timer Interrupt Flag 0**<br>0 = It indicates that the timer 0 does not count up to TCMP (TMR0_CMPR[23:0]) value yet. Software can reset this bit after the timer interrupt 0 had occurred.<br>1 = It indicates that the counter of timer 0 is incremented to corresponding TCMP (TMR0_CMPR[23:0]) setting value.<br>**NOTE:** This bit is read only, but can be cleared by writing 1 to this bit. |

## 5.11 Enhance Timer Controller (ETMR)

### 5.11.1 Overview

This chip is equipped with four enhance timer modules including ETIMER0, ETIMER1, ETIMER2 and ETIMER3, which allow user to easily implement a counting scheme or timing control for applications. The timer can perform functions like frequency measurement, interval measurement, clock generation, delay timing, and so on. The timer can generate an interrupt signal upon timeout, or provide the current value of count during operation.

### 5.11.2 Features

- Independent Clock Enable Control for each Timer (ECLKetmr0, ECLKetmr1, ECLKetmr2 and ECLKetmr3)
- Time-out period = (Period of timer clock input) * (8-bit pre-scale counter + 1) * (24-bit TCMP)
- Counting cycle time = (1 / ECLKetmr) * (2^8) * (2^24)
- Internal 8-bit pre-scale counter
- Internal 24-bit up counter is readable through TDR (Timer Data Register)
- Supports One-shot, Periodic, Output Toggle and Continuous Counting Operation mode
- Supports external pin capture for interval measurement
- Supports external pin capture for timer counter reset

### 5.11.3 Block Diagram

Each timer is equipped with an 8-bit pre-scale counter, a 24-bit up-counter, a 24-bit compare register and an interrupt request signal. Refer to Figure 5.11-1 for the timer controller block diagram. There are four options of clock sources for each timer.

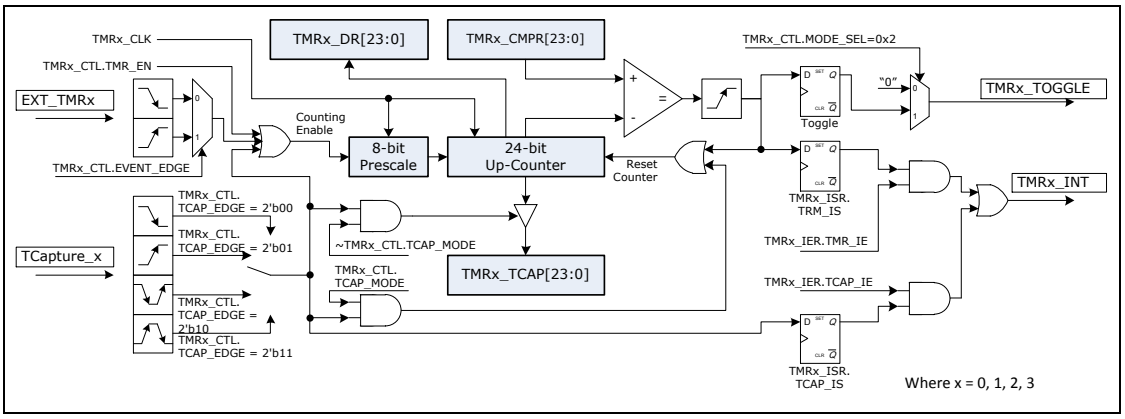



Figure 5.11-1 Enhance Timer Controller Block Diagram

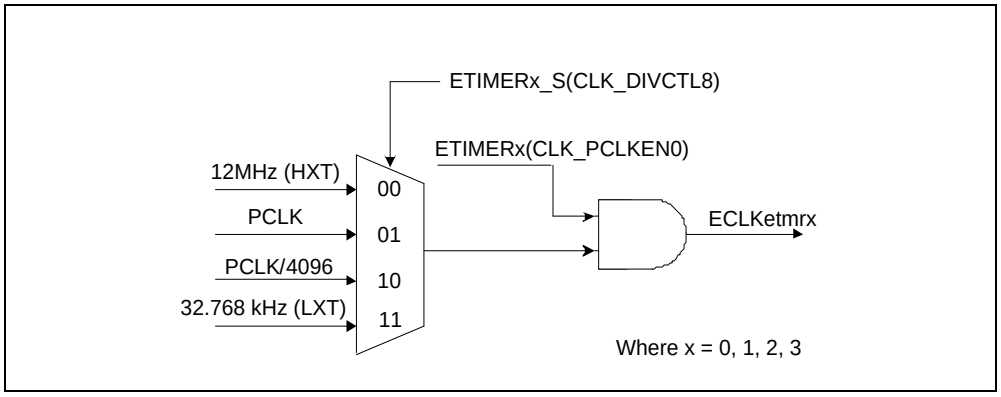



Figure 5.11-2 Enhance Timer Clock Controller Diagram

### 5.11.4 Basic Configuration

When using Enhance Timer, it's necessary to configure related pins as the ETMR function and enable ETMR's clock.

To use ETMR's toggle out or external pin capture functionality, choose a pin for toggle out or capture in is necessary. Please refer to the register SYS_MFP_GPBL, SYS_MFP_GPCL, SYS_MFP_GPCH, and SYS_MFP_GPFH to know how to configure related pins as the ETMR's toggle out or capture in function.

Set ETIMER0 (CLK_PCLKEN0[4]), ETIMER1 (CLK_PCLKEN0[5]), ETIMER2 (CLK_PCLKEN0[6]), and ETIMER3 (CLK_PCLKEN0[7]) high to enable clock of ETMR0, ETMR1, ETMR2 and ETMR3 respectively.

### 5.11.5 Functional Description

Timer controller provides One-shot, Period, Toggle and Continuous Counting operation modes. The external pin capture function is also provided for interval measurement or reset timer counter. Each operating function mode is shown as follows:

#### 5.11.5.1 *One-Shot Mode*

If the timer is operated in One-shot mode (MODE_SEL[1:0] is 00) and ETMR_EN (ETMRn_CTL[0] timer counter enable bit) is set to 1, the timer counter starts up counting. Once the timer counter value (ETMRn_DR value) reaches timer compare register (ETMRn_CMPR) value, the ETMR_IS (ETMRn_ISR[0] timer interrupt status) will set to 1. If ETMR_IE (ETMRn_IER[0] timer interrupt enable bit) is set to 1 then the interrupt signal is generated and sent to AIC to inform CPU for indicating that the timer counting overflow happens. If ETMR_IE (ETMRn_IER[0] timer interrupt enable bit) is set to 0, no interrupt signal is generated.

In this operating mode, once the timer counter value (ETMRn_DR value) reaches timer compare register (ETMRn_CMPR) value, ETMR_IS (ETMRn_ISR[0] timer interrupt status) will set to 1, timer counting operation stops and the timer counter value (ETMRn_DR value) goes back to counting initial value then ETMR_EN (ETMRn_CTL[0] timer counter enable bit) is cleared to 0 by timer controller automatically. That is to say, timer operates timer counting and compares with ETMRn_CMPR value function only one time after programming the timer compare register (ETMRn_CMPR) value and ETMR_EN (ETMRn_CTL[0] timer counter enable bit) is set to 1. So, this operating mode is called One-Shot mode.

#### 5.11.5.2 *Periodic Mode*

If the timer is operated in Periodic mode (MODE_SEL[1:0] is 01) and ETMR_EN (ETMRn_CTL[0] timer counter enable bit) is set to 1, the timer counter starts up counting. Once the timer counter value (ETMRn_DR value) reaches timer compare register (ETMRn_CMPR) value, the ETMR_IS (ETMRn_ISR[0] timer interrupt status) will set to 1. If ETMR_IE (ETMRn_IER[0] timer interrupt enable bit) is set to 1 then the interrupt signal is generated and sent to AIC to inform CPU for indicating that the timer counting overflow happens. If ETMR_IE (ETMRn_IER[0] timer interrupt enable bit) is set to 0, no interrupt signal is generated.

In this operating mode, once the timer counter value (ETMRn_DR value) reaches timer compare register (ETMRn_CMPR) value, ETMR_IS (ETMRn_ISR[0] timer interrupt status) will set to 1, the timer counter value (ETMRn_DR value) goes back to counting initial value and ETMR_EN (ETMRn_CTL[0] timer counter enable bit) is kept at 1 (counting enable continuously) and timer counter operates up counting again. If ETMR_IS (ETMRn_ISR[0] timer interrupt status) is cleared by software, once the timer counter value (ETMRn_DR value) reaches timer compare register (ETMRn_CMPR) value again, ETMR_IS (ETMRn_ISR[0] timer interrupt status) will set to 1 also. That is to say, timer operates timer counting and compares with ETMRn_CMPR value function periodically. The timer counting operation does not stop until the ETMR_EN (ETMRn_CTL[0] timer counter enable bit) is set to 0. The interrupt signal is also generated periodically. So, this operating mode is called Periodic mode.

#### 5.11.5.3 *Toggle Mode*

If the timer is operated in Toggle mode (MODE_SEL[1:0] is 10) and ETMR_EN (ETMRn_CTL[0] timer counter enable bit) is set to 1, the timer counter starts up counting. Once the timer counter value (ETMRn_DR value) reaches timer compare register (ETMRn_CMPR) value, the ETMR_IS (ETMRn_ISR[0] timer interrupt status) will set to 1. If ETMR_IE (ETMRn_IER[0] timer interrupt enable bit) is set to 1 then the interrupt signal is generated and sent to AIC to inform CPU for indicating that the timer counting overflow happens. If ETMR_IE (ETMRn_IER[0] timer interrupt enable bit) is set to 0, no interrupt signal is generated.

In this operating mode, once the timer counter value (ETMRn_DR value) reaches timer compare register (ETMRn_CMPR) value, ETMR_IS (ETMRn_ISR[0] timer interrupt status) and toggle out signal will set to 1, the timer counter value (ETMRn_DR value) goes back to counting initial value and ETMR_EN (ETMRn_CTL[0] timer counter enable bit) is still kept at 1 (counting enable continuously), and timer counter operates up counting again. when the timer counter value (ETMRn_DR value) reaches timer compare register value again, toggle out signal is set to 0, and ETMR_IS (ETMRn_ISR[0] timer interrupt status) will set to 1 also. The timer counting operation does not stop until the ETMR_EN (ETMRn_CTL[0] timer counter enable bit) is set to 0. Thus, the toggle output signal changes back and forth with 50% duty cycle. So, this operating mode is called Toggle mode.

#### 5.11.5.4 *Continuous Counting Mode*

If the timer is operated in Continuous Counting mode (MODE_SEL[1:0] is 11) and ETMR_EN (ETMRn_CTL[0] timer counter enable bit) is set to 1, the timer counter starts up counting. Once the timer counter value (ETMRn_DR value) reaches timer compare register (ETMRn_CMPR) value, the ETMR_IS (ETMRn_ISR[0] timer interrupt status) will set to 1. If ETMR_IE (ETMRn_IER[0] timer counter enable bit) is set to 1 then the interrupt signal is generated and sent to AIC to inform CPU for indicating that the timer counting overflow happens. If ETMR_IE (ETMRn_IER[0] timer counter enable bit) is set to 0, no interrupt signal is generated.

In this operating mode, once the timer counter value (ETMRn_DR value) reaches timer compare register (ETMRn_CMPR) value, ETMR_IS (ETMRn_ISR[0] timer interrupt status) will set to 1 and ETMR_EN (ETMRn_CTL[0] timer counter enable bit) is kept at 1 (counting enable continuously) and timer counter continuous counting without reload the timer counter value (ETMRn_DR value) to counting initial value. User can change different timer compare register (ETMRn_CMPR) value immediately without disabling timer counter and restarting timer counter counting.

For example, the timer compare register (ETMRn_CMPR) value is set as 80, first. (The timer compare register (ETMRn_CMPR) should be less than $2^{24}$ and be greater than 1). Once the timer counter value (ETMRn_DR value) reaches to 80, ETMR_IS (ETMRn_ISR[0] timer interrupt status) will set to 1 and ETMR_EN (ETMRn_CTL[0] timer counter enable bit) is still kept at 1 (counting enable continuously). Next, user clears the ETMR_IS (ETMRn_ISR[0] timer interrupt status) and reprograms timer compare register (ETMRn_CMPR) value as 200, then ETMR_IS (ETMRn_ISR[0] timer interrupt status) will set to 1 again when timer counter value (ETMRn_DR value) reaches to 200. At last, user clears ETMR_IS (ETMRn_ISR[0] timer interrupt status) and reprograms timer compare register (ETMRn_CMPR) value as 500, then ETMR_IS (ETMRn_ISR[0] timer interrupt status) will set to 1 again when timer counter value (ETMRn_DR value) reaches to 500. In this mode, when the timer counter value (ETMRn_DR value) continues counting up to $2^{24}$ -1, then recount up from 0 continuously. The timer counter value (ETMRn_DR value) is always keeping up counting even if ETMR_IS (ETMRn_ISR[0] timer interrupt status) is 1. Therefore, this operation mode is called as Continuous Counting mode.

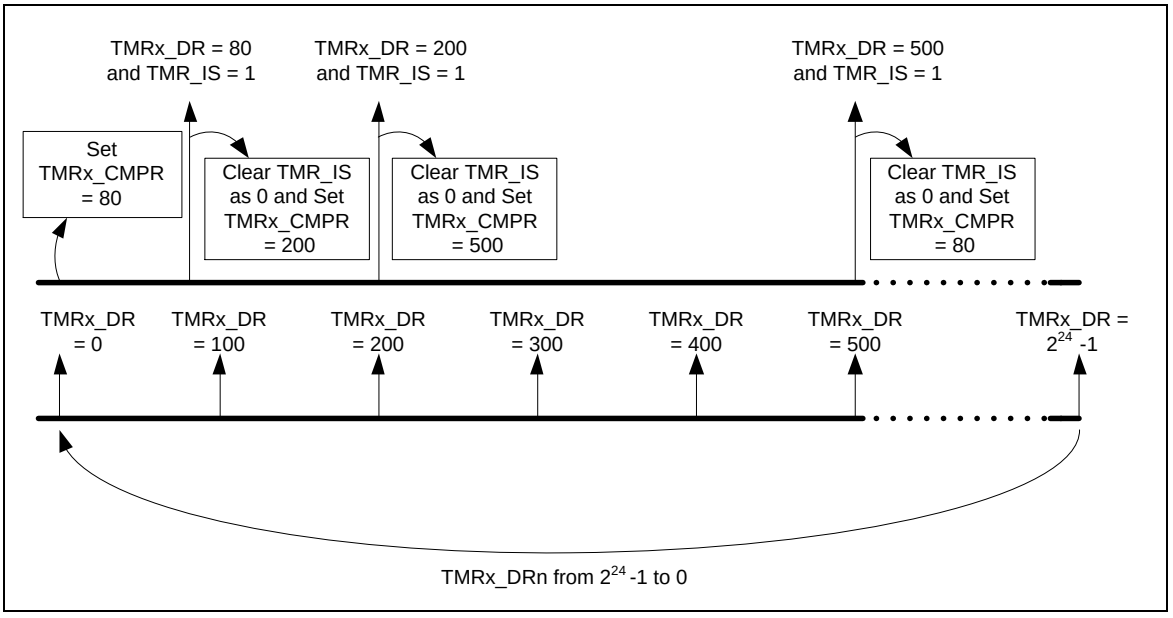



Figure 5.11-3 Timer Clock Controller Diagram

5.11.5.5 *Timer Counter Capture/Reset Function*

In this mode, Timer will monitor the transition of external pin to save the 24-bit counter value or reset the 24-bit counter.

If TCAP_MODE is 0, the transition on external pin is used as timer counter capture function. In this mode, if CAP_CNT_MOD is 0, the free-counting mode, 24-bit up-counting timer will keep counting continuously. And when the transition of external pin matches the TCAP_EDGE setting, the value of 24-bit up-counting timer will be saved into register ETMRn_TCAP. If CAP_CNT_MOD is 1, the trigger-counting mode, 24-bit up-counting timer will keep its value at zero. Once the transition of external pin matches the 1st transition of TCAP_EDGE setting, the 24-bit up-counting timer will start counting. And then if the transition of external pin matches the 2nd transition of TCAP_EDGE setting, the 24-bit up-counting timer will stop counting. And its value will be saved into register ETMRn_TCAP.

If TCAP_MODE is 1, the transition on external pin is used as timer counter reset function. In this mode, once the transition of external pin matches the TCAP_EDGE setting, the 24-bit up-counting timer will be reset.

To detect the transition of external pin, the timer circuit implements the de-bounce circuit for external pin. Based on the result of de-bounce circuit and external pin, the rising-edge or falling-

edge could be detected. The reset value of de-bounce circuit is “0” and the de-bounce circuit would only active when both TCAP_DEB_EN and TCAP_EN are enabled. So, if the external pin level is “1” and TCAP_EDGE is set to detect rising-edge of external pin, then after de-bounce circuit active (TCAP_DEB_EN is “1” and TCAP_EN is “1”), a false rising-edge would be detected. This would result in the incorrect capture data (ETMRn_TCAP) while 1st time the TCAP_IS is set. To avoid this incorrect capture data to affect the capture application, discard this 1st capture data is necessary and recommended.

### 5.11.6 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ETMR Base Address:**<br>**ETMR0_BA = 0x4001_0000**<br>**ETMR1_BA = 0x4001_0100**<br>**ETMR2_BA = 0x4011_0000**<br>**ETMR3_BA = 0x4011_0100** | | | | |
| **ETMRn_CTL**<br>**n=0,1,2,3** | ETMRn_BA+0x000 | R/W | Enhance Timer n Control Register | 0x0000_0000 |
| **ETMRn_PRECNT**<br>**n=0,1,2,3** | ETMRn_BA+0x004 | R/W | Enhance Timer n Pre-Scale Counter Register | 0x0000_0000 |
| **ETMRn_CMPR**<br>**n=0,1,2,3** | ETMRn_BA+0x008 | R/W | Enhance Timer n Compare Register | 0x0000_0000 |
| **ETMRn_IER**<br>**n=0,1,2,3** | ETMRn_BA+0x00C | R/W | Enhance Timer n Interrupt Enable Register | 0x0000_0000 |
| **ETMRn_ISR**<br>**n=0,1,2,3** | ETMRn_BA+0x010 | R/W | Enhance Timer n Interrupt Status Register | 0x0000_0000 |
| **ETMRn_DR**<br>**n=0,1,2,3** | ETMRn_BA+0x014 | R | Enhance Timer n Data Register | 0x0000_0000 |
| **ETMRn_TCAP**<br>**n=0,1,2,3** | ETMRn_BA+0x018 | R | Enhance Timer n Capture Data Register | 0x0000_0000 |

### 5.11.7 Register Description

## Enhance Timer n Control Register (ETMRn_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ETMRn_CTL n=0,1,2,3** | ETMRn_BA+0x000 | R/W | Enhance Timer n Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | **TCAP_DEB_EN** | **Reserved** | **CAP_CNT_MOD** | **TCAP_EDGE** | | **TCAP_MODE** | **TCAP_EN** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ETMR_ACT** | **Reserved** | **MODE_SEL** | | **DBGACK_EN** | **WAKE_EN** | **SW_RST** | **ETMR_EN** |

| Bits | Description | |
|---|---|---|
| [31:23] | **Reserved** | Reserved. |
| [22] | **CAP_DEB_EN** | **Tcapture Pin De-bounce Enable**<br>When CAP_DEB_EN is set, the Tcapture pin de-bounce circuit will be enabled to eliminate the bouncing of the signal.<br>In de-bounce circuit the Tcapture pin signal will be sampled 4 times by ECLKetmr.<br>0 = De-bounce circuit Disabled.<br>1 = De-bounce circuit Enabled.<br>**Note:** When TCAP_EN is enabled, enable this bit is recommended. And, while TCAP_EN is disabled, disable this bit is recommended to save power consumption. |
| [21] | **Reserved** | Reserved. |
| [20] | **CAP_CNT_MOD** | **Timer Capture Counting Mode Selection**<br>This bit indicates the behavior of 24-bit up-counting timer while TCAP_EN is set to high.<br>If this bit is 0, the free-counting mode, the behavior of 24-bit up-counting timer is defined by MODE_SEL field. When TCAP_EN is set, TCAP_MODE is 0, and the transition of Tcapture pin matches the TCAP_EDGE setting, the value of 24-bit up-counting timer will be saved into register ETMRn_TCAPn.<br>If this bit is 1, Trigger-counting mode, 24-bit up-counting timer will be not counting and keep its value at zero. When TCAP_EN is set, TCAP_MODE is 0, and once the transition of external pin matc$^{he}$s the 1$^{st}$ transition of TCAP_EDGE setting, the 24-bit up-counting timer will start counting. And then if the transition of external pin matches the 2$^{nd}$ transition of TCAP_EDGE setting, the 24-bit up-counting timer will stop counting. And its value will be saved into register ETMRn_TCAPn.<br>0 = Capture with free-counting timer mode.<br>1 = Capture with trigger-counting timer mode.<br>**Note:** For ETMRn+1_CTL, if INTR_TRG_EN is set, the CAP_CNT_MOD will be forced to high, the capture with Trigger-counting Timer mode. |

<table>
<tr><th>Bits</th><th colspan="2">Description</th></tr>
<tr><td>[19:18]</td><td>TCAP_EDGE</td><td>
Tcapture Pin Edge Detect Selection<br>
This field defines that active transition of Tcapture pin is for timer counter reset function or for timer capture function.<br>
For timer counter reset function and free-counting mode of timer capture function, the configurations are:
<table>
<tr><th>TCAP_EDGE</th><th>Tcapture Pin Edge Detect</th></tr>
<tr><td>00</td><td>A falling edge (1 to 0 transition) on Tcapture pin is an active transition.</td></tr>
<tr><td>01</td><td>A rising edge (0 to 1 transition) on Tcapture pin is an active transition.</td></tr>
<tr><td>10</td><td>Both falling edge (1 to 0 transition) and rising edge (0 to 1 transition) on Tcapture pin are active transitions.</td></tr>
<tr><td>11</td><td>Both falling edge (1 to 0 transition) and rising edge (0 to 1 transition) on Tcapture pin are active transitions.</td></tr>
</table>
For trigger-counting mode of timer capture function, the configurations are:
<table>
<tr><th>TCAP_EDGE</th><th>Tcapture Pin Edge Detect</th></tr>
<tr><td>00</td><td>1st falling edge on Tcapture pin triggers 24-bit timer to start counting, while 2nd falling edge triggers 24-bit timer to stop countin g.</td></tr>
<tr><td>01</td><td>1st rising edge on Tcapture pin triggers 24-bit timer to start counting, while 2nd rising edge triggers 24-bit timer to stop counting.</td></tr>
<tr><td>10</td><td>Falling edge on Tcapture pin triggers 24-bit timer to start counting, while rising edge triggers 24-bit timer to stop counting.</td></tr>
<tr><td>11</td><td>Rising edge on Tcapture pin triggers 24-bit timer to start counting, while falling edge triggers 24-bit timer to stop counting.</td></tr>
</table>
Note: For ETMRn+1_CTL, if INTR_TRG_EN is set, the TCAP_EDGE will be forced to 11.
</td></tr>
<tr><td>[17]</td><td>TCAP_MODE</td><td>
Tcapture Pin Function Mode Selection<br>
This bit indicates if the transition on Tcapture pin is used as timer counter reset function or timer capture function.<br>
0 = The transition on Tcapture pin is used as timer capture function.<br>
1 = The transition on Tcapture pin is used as timer counter reset function.<br>
Note: For ETMRn+1_CTL, if INTR_TRG_EN is set, the TCAP_MODE will be forced to low.
</td></tr>
<tr><td>[16]</td><td>TCAP_EN</td><td>
Tcapture Pin Functional Enable<br>
This bit controls if the transition on Tcapture pin could be used as timer counter reset function or timer capture function.<br>
0 = The transition on Tcapture pin is ignored.<br>
1 = The transition on Tcapture pin will result in the capture or reset of 24-bit timer counter.<br>
Note: For ETMRn_CTL, if INTR_TRG_EN is set, the TCAP_EN will be forced to low and the Tcapture pin transition is ignored.<br>
Note: For ETMRn+1_CTL, if INTR_TRG_EN is set, the TCAP_EN will be forced to high.
</td></tr>
<tr><td>[15:8]</td><td>Reserved</td><td>Reserved.</td></tr>
</table>

| Bits | Description | |
|---|---|---|
| [7] | **ETMR_ACT** | **Timer Active Status Bit (Read Only)**<br>This bit indicates the timer counter status of timer.<br>0 = Timer is not active.<br>1 = Timer is in active. |
| [6] | **Reserved** | Reserved. |
| [5:4] | **MODE_SEL** | **Timer Operating Mode Select** |

| MODE_SEL | Timer Operating Mode |
|---|---|
| 00 | **The timer is operating in the one-shot mode.**<br>In this mode, the associated interrupt signal is generated (if ETMR_IER [ETMR_IE] is enabled) once the value of 24-bit up counter equals the ETMRn_CMPR. And ETMR_CTL [ETMR_EN] is automatically cleared by hardware. |
| 01 | **The timer is operating in the periodic mode.**<br>In this mode, the associated interrupt signal is generated periodically (if ETMR_IER [ETMR_IE] is enabled) while the value of 24-bit up counter equals the ETMRn_CMPR. After that, the 24-bit counter will be reset and starts counting from zero again. |
| 10 | **The timer is operating in the periodic mode with output toggling.**<br>In this mode, the associated interrupt signal is generated periodically (if ETMR_IER [ETMR_IE] is enabled) while the value of 24-bit up counter equals the ETMRn_CMPR. After that, the 24-bit counter will be reset and starts counting from zero again.<br>At the same time, timer controller will also toggle the output pin ETMRn_TOG_OUT to its inverse level (from low to high or from high to low).<br>**Note:** The default level of ETMRn_TOG_OUT after reset is low. |
| 11 | **The timer is operating in continuous counting mode.**<br>In this mode, the associated interrupt signal is generated when ETMR_DR = ETMR_CMPR (if ETMR_IER [ETMR_IE] is enabled). However, the 24-bit up-counter counts continuously without reset. |

| | | |
|---|---|---|
| [3] | **DBGACK_EN** | **ICE Debug Mode Acknowledge Ineffective Enable**<br>0 = ICE debug mode acknowledgement effects TIMER counting and TIMER counter will be held while ICE debug mode acknowledged.<br>1 = ICE debug mode acknowledgement is ineffective and TIMER counter will keep going no matter ICE debug mode acknowledged or not. |
| [2] | **WAKE_EN** | **Wake-up Enable**<br>When WAKE_EN is set and the ETMR_IS or TCAP_IS is set, the timer controller will generate a wake-up trigger event to CPU.<br>0 = Wake-up trigger event Disabled.<br>1 = Wake-up trigger event Enabled. |

| Bits | Description | |
|---|---|---|
| [1] | **SW_RST** | **Software Reset**<br>Set this bit will reset the timer counter, pre-scale counter and also force ETMR_CTL [ETMR_EN] to 0.<br>0 = No effect.<br>1 = Reset Timer's pre-scale counter, internal 24-bit up-counter and ETMR_CTL [ETMR_EN] bit.<br>**Note:** This bit will be auto cleared and takes at least 3 ECLKetmr clock cycles. |
| [0] | **ETMR_EN** | **Timer Counter Enable Bit**<br>0 = Stops/Suspends counting.<br>1 = Starts counting.<br>**Note1:** Set ETMR_EN to 1 enables 24-bit counter keeps up counting from the last stop counting value.<br>**Note2:** This bit is auto-cleared by hardware in one-shot mode (MODE_SEL [5:4] = 00) once the value of 24-bit up counter equals the ETMRn_CMPR. |

### Enhance Timer n Pre-Scale Counter Register (ETMRn_PRECNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ETMRn_PRECNT**<br>**n=0,1,2,3** | ETMRn_BA+0x004 | R/W | Enhance Timer n Pre-Scale 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** |
| **PRESCALE_CNT** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **PRESCALE_CNT** | **Pre-scale Counter**<br>Clock input is divided by PRESCALE_CNT + 1 before it is fed to the counter. If PRESCALE_CNT =0, then there is no scaling. |

### Enhance Timer n Compare Register (ETMRn_CMPR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ETMRn_CMPR**<br>**n=0,1,2,3** | ETMRn_BA+0x008 | R/W | Enhance Timer n Compare Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **ETMR_CMP** | **Timer Compared Value**<br>ETMR_CMP is a 24-bit compared register. When the internal 24-bit up-counter counts and its value is equal to ETMR_CMP value, a Timer Interrupt is requested if the timer interrupt is enabled with ETMR_IER [ETMR_IE] is enabled. The ETMR_CMP value defines the timer counting cycle time.<br>Time-out period = (Period of timer clock input) * (8-bit PRESCALE_CNT + 1) * (24-bit ETMR_CMP).<br>**Note1:** Never write 0 or 1 in ETMR_CMP, or the core will run into unknown state.<br>**Note2:** No matter ETMR_CTL [ETMR_EN] is 0 or 1, whenever software write a new value into this register, TIMER will restart counting using this new value and abort previous count. |

### Enhance Timer n Interrupt Enable Register (ETMRn_IER)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ETMRn_IER**<br>**n=0,1,2,3** | ETMRn_BA+0x00C | R/W | Enhance Timer n 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** | | | | | | **TCAP_IE** | **ETMR_IE** |

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [1] | **TCAP_IE** | **Timer Capture Function Interrupt Enable**<br>0 = Timer External Pin Function Interrupt Disabled.<br>1 = Timer External Pin Function Interrupt Enabled.<br>**Note:** If timer external pin function interrupt is enabled, the timer asserts its interrupt signal when the TCAP_EN is set and the transition of external pin matches the TCAP_EDGE setting |
| [0] | **ETMR_IE** | **Timer Interrupt Enable**<br>0 = Timer Interrupt Disabled.<br>1 = Timer Interrupt Enabled.<br>**Note:** If timer interrupt is enabled, the timer asserts its interrupt signal when the associated counter is equal to ETMR_CMPR. |

## Enhance Timer n Interrupt Status Register (ETMRn_ISR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ETMRn_ISR n=0,1,2,3** | ETMRn_BA+0x010 | R/W | Enhance Timer n 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** | | **NCAP_DET_STS** | **ETMR_WAKE_STS** | **Reserved** | | **TCAP_IS** | **ETMR_IS** |

| Bits | Description | |
|---|---|---|
| [31:6] | **Reserved** | Reserved. |
| [5] | **NCAP_DET_STS** | **New Capture Detected Status**<br>This status is to indicate there is a new incoming capture event detected before CPU clearing the TCAP_IS status.<br>If the above condition occurred, the Timer will keep register ETMRn_CAP unchanged and drop the new capture value.<br>This bit is also cleared to 0 while TCAP_IS is cleared.<br>0 = New incoming capture event didn't detect before CPU clearing TCAP_IS status.<br>1 = New incoming capture event detected before CPU clearing TCAP_IS status. |
| [4] | **ETMR_WAKE_STS** | **Timer Wake-up Status**<br>If timer causes CPU wakes up from power-down mode, this bit will be set to high. It must be cleared by software with a write 1 to this bit.<br>0 = Timer does not cause system wake-up.<br>1 = Wakes system up from power-down mode by Timer timeout. |
| [3:2] | **Reserved** | Reserved. |
| [1] | **TCAP_IS** | **Timer Capture Function Interrupt Status**<br>This bit indicates the external pin function interrupt status of Timer.<br>This bit is set by hardware when TCAP_EN is set high, and the transition of external pin matches the TCAP_EDGE setting. Write 1 to clear this bit to zero.<br>If this bit is active and TCAP_IE is enabled, Timer will trigger an interrupt to CPU. |
| [0] | **ETMR_IS** | **Timer Interrupt Status**<br>This bit indicates the interrupt status of Timer.<br>This bit is set by hardware when the up counting value of internal 24-bit counter matches the timer compared value (ETMR_CMPR). Write 1 to clear this bit to 0.<br>If this bit is active and ETMR_IE is enabled, Timer will trigger an interrupt to CPU. |

### Enhance Timer n Data Register (ETMRn_DR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ETMRn_DR n=0,1,2,3** | ETMRn_BA+0x014 | R | Enhance Timer n Data Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **TDR** | **Timer Data Register**<br>User can read this register for internal 24-bit timer up-counter value. |

### Enhance Timer n Capture Data Register (ETMRn_TCAP)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ETMRn_TCAP n=0,1,2,3** | ETMRn_BA+0x018 | R | Enhance Timer n Capture Data Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **CAP** | **Timer Capture Data Register**<br>When TCAP_EN is set, TCAP_MODE is 0, and the transition of external pin matches the TCAP_EDGE setting, the value of 24-bit up-counting timer will be saved into register ETMRn_TCAP. User can read this register for the counter value. |

## 5.12 Pulse Width Modulation (PWM)

### 5.12.1 Overview

This chip has one PWM controller, and it has 4 independent PWM outputs, CH0~CH3, or as 2 complementary PWM pairs, (CH0, CH1), (CH2, CH3) with 2 programmable dead-zone generators. Each PWM pair has one Prescaler, one clock divider, two clock selectors, two 16-bit PWM counters, two 16-bit comparators, and one Dead-Zone generator. They are all driven by APB system clock (PCLK) in chip. Each PWM channel can be used as a timer and issue interrupt independently.

Two channels PWM Timers in one pair share the same prescaler. The Clock divider provides each PWM channel with 5 divided clock sources (1, 1/2, 1/4, 1/8, 1/16). Each channel receives its own clock signal from clock divider which receives clock from 8-bit prescaler. The 16-bit down-counter in each channel receive clock signal from clock selector and can be used to handle one PWM period. The 16-bit comparator compares PWM counter value with threshold value in register CMR (PWM_CMR[15:0]) loaded previously to generate PWM duty cycle. The clock signal from clock divider is called PWM clock. Dead-Zone generator utilize PWM clock as clock source. Once Dead-Zone generator is enabled, two outputs of the corresponding PWM channel pair will be replaced by the output of Dead-Zone generator. The Dead-Zone generator is used to control off-chip power device.

To prevent PWM driving output pin with unsteady waveform, 16-bit down-counter and 16-bit comparator are implemented with double buffering feature. User can feel free to write data to counter buffer register and comparator buffer register without generating glitch. When 16-bit down-counter reaches zero, the interrupt request is generated to inform CPU that time is up. When counter reaches zero, if counter is set as periodic mode, it is reloaded automatically and start to generate next cycle. User can set PWM counter as one-shot mode instead of periodic mode. If counter is set as one-shot mode, counter will stop and generate one interrupt request when it reaches zero. The value of comparator is used for pulse width modulation. The counter control logic changes the output level when down-counter value matches the value of compare register.

### 5.12.2 Features

- 4 PWM channels with a 16-bit down counter and an interrupt each
- 2 complementary PWM pairs, (CH0, CH1), (CH2, CH3), with a programmable dead-zone generator each
- Internal 8-bit prescaler and a clock divider for each PWM paired channel
- Independent clock source selection for each PWM channel
- Internal 16-bit down counter and 16-bit comparator for each independent PWM channel
- PWM down-counter supports One-shot or Periodic mode

### 5.12.3 Block Diagram

The following figure describes the architecture of one PWM pair.

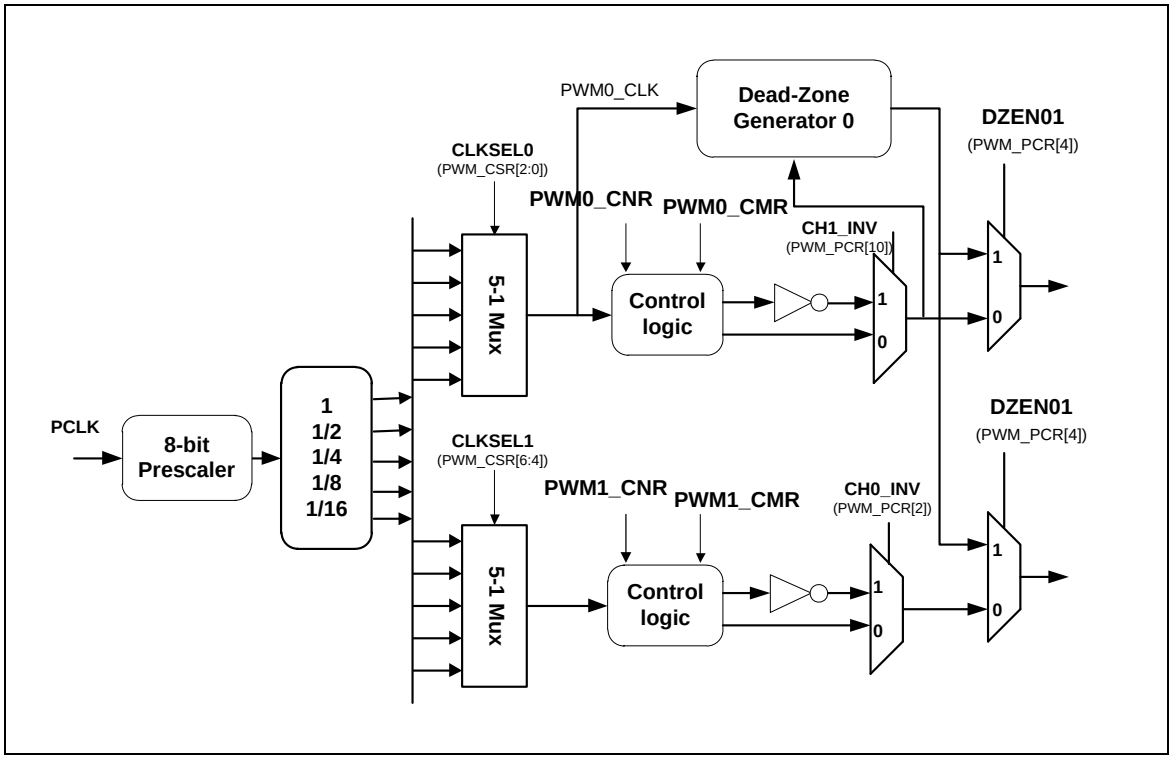



Figure 5.12-1 Two channels of PWM in one pair

### 5.12.4 Basic Configuration

The PWM pin functions are configured in PA_H_MFP, PB_L_MFP, PC_H_MFP, PD_H_MFP, and PH_L_MFP registers. The clock enable of PWM function is configured in PCLKEN1[27].

### 5.12.5 Functional Description

5.12.5.1 *PWM Timer Operation*

The PWM period and duty control are decided by register CNR (PWM_CNR[15:0]) and CMR (PWM_CMR[15:0]). The PWM-timer timing operation is shown in **Error! Reference source not found.**. The pulse width modulation follows the formula below and the legend of PWM-Timer Comparator is shown in Figure 5.12-2.

PWM frequency = PCLK/(prescale+1)*(clock divider)/(CNR+1); depending on selected PWM channel.

Duty ratio = (CMR+1)/(CNR+1).

CMR >= CNR: PWM output is always high.

CMR < CNR: PWM low width = (CNR - CMR) unit1; PWM high width = (CMR+1) unit.

If CMR = 0: PWM low width = (CNR) unit; PWM high width = 1 unit.

**Note:** 1. Unit = one PWM clock cycle.

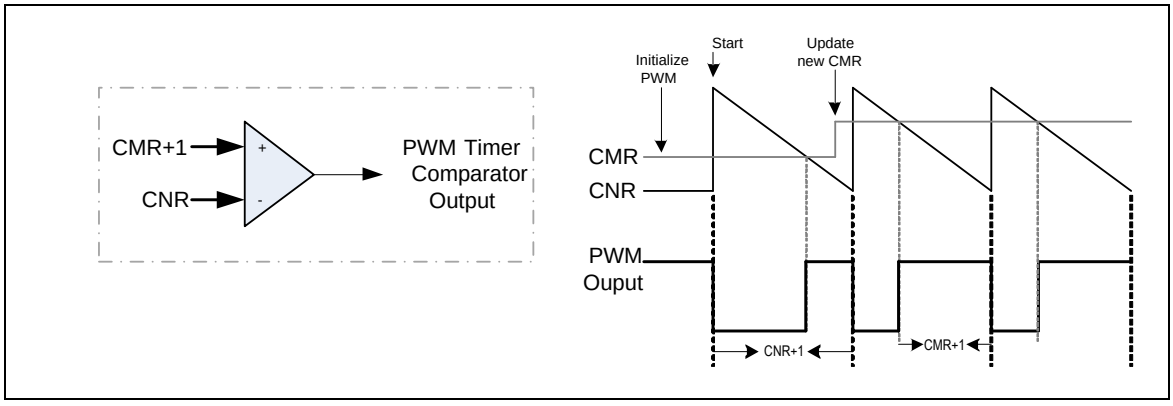



Figure 5.12-2 Legend of Internal Comparator Output of PWM-Timer

5.12.5.2 *PWM Double Buffering, Periodic and One-shot Operation*

The PWM timers have double buffering function; the reload value is updated at the start of next period without affecting current timer operation. The PWM counter value can be written into CNR (PWM_CNR[15:0]).

The bit CH0MOD (PWM_PCR[3]) defines PWM operation in Periodic or One-shot mode If CH0MOD (PWM_CTL[3]) is set to one (periodic mode), the controller loads CNR (PWM_CNR[15:0]) to PWM counter when PWM counter reaches zero. If CNR (PWM_CNR[15:0]) is set to zero, PWM counter will be halt when PWM counter counts to zero.

In one-shot mode (CH0MOD=0; PWM_PCR[3]=0), the corresponding channel will output only one cycle of duty waveform and then PWM counter will be stopped if no further corresponding duty register updated. When PWM counter is running, updating corresponding duty register will engage the next cycle of duty waveform.

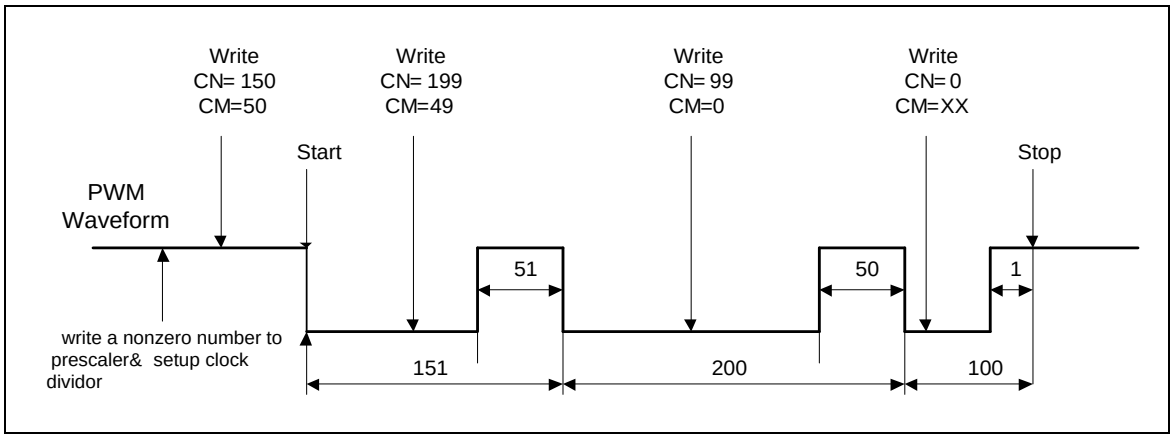



Figure 5.12-3 PWM Double Buffer Illustration

5.12.5.3 *Modulate Duty Ratio*

The double buffering function allows CMR (PWM_CMR[15:0]) to be written at any point in current

cycle. The loaded value will take effect from next cycle.

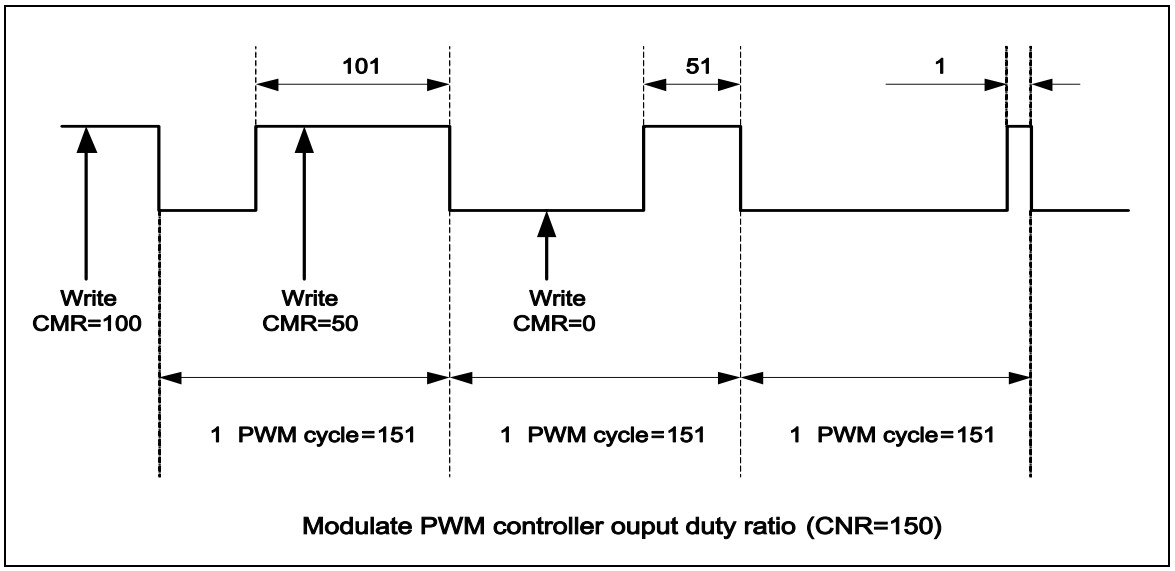



Figure 5.12-4 PWM Controller Output Duty Ratio

#### 5.12.5.4 *Dead-Zone Generator*

PWM implements Dead Zone generator. They are built for power device protection. This function generates a programmable time gap called “Dead-Zone” to delay PWM rising output, and it is in order to prevent damage for the power switch devices that connected to the PWM output pins. User can program Dead-Zone counter to determine the Dead Zone interval of channel 0, 1 pair with DZL01 (PWM_PPR[23:16]). The Dead Zone period of channel 0, 1 pair can be calculated by (PWM_CLK period x (DZL01 + 1)) and the enable bit is DZEN01 (PWM_PCR[4]).

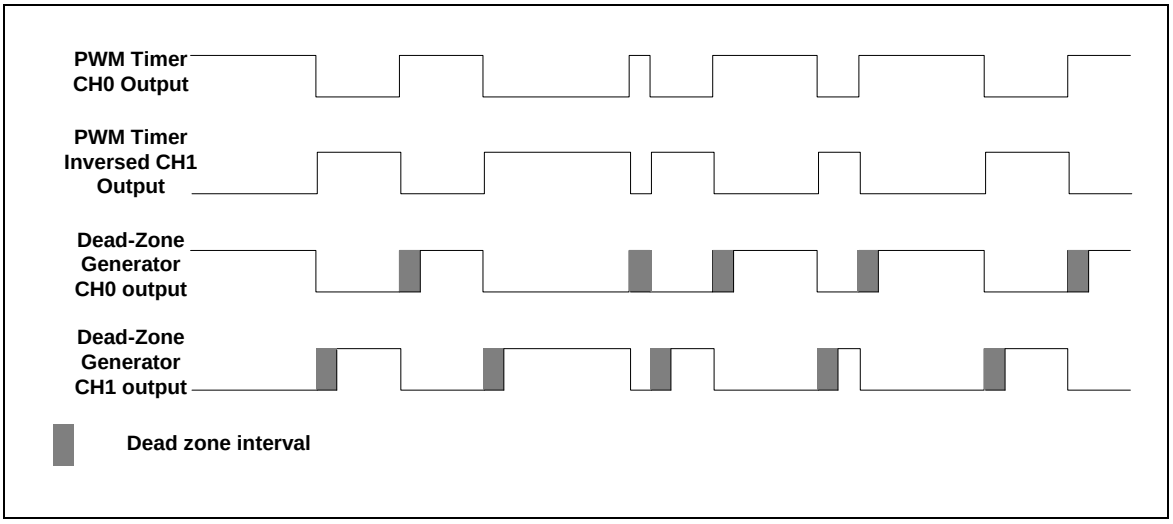



Figure 5.12-5 Paired PWM Output with Dead Zone Generation Operation

#### 5.12.5.5 *PWM-Timer Start Procedure*

Take PWM channel 0 for example, and the following procedure is for starting a PWM drive.

Setup clock selector CLKSEL0 (PWM_CSR[2:0])

Setup prescaler PRESCALE01 (PWM_PPR[7:0])

Setup inverter on/off, dead zone generator on/off, periodic/one-shot mode and Stop PWM-timer (PWM_PCR)

Setup interrupt enable register PIER0 (PWM_PIER[0])

Setup the corresponding GPI/O pins to PWM function

Setup PWM comparator register CMR (PWM_CMR[15:0]) and PWM counter register CNR (PWM_CNR[15:0]) for setting PWM period and duty length

Enable PWM down-counter start running (Set CH0EN = 1 (PWM_PCR[0]))

The procedure mentioned above may be set up not in the order and PWM Timer can still work fine

#### 5.12.5.6 *PWM-Timer Stop Procedure*

Take PWM channel 0 for example.

**Method 1:** Set 16-bit down counter CNR (PWM_CNR[15:0]) as 0, and monitor data register PDR (PWM_PDR[15:0]). When PDR (PWM_PDR[15:0]) reaches to 0, disable PWM Timer by setting CH0EN = 0 (PWM_PCR[0] = 0). (Recommended)

**Method 2:** Set 16-bit down counter CNR (PWM_CNR[15:0]) as 0. When interrupt request happen, disable PWM Timer by setting CH0EN = 0 (PWM_PCR[0] = 0). (Recommended)

**Method 3:** Disable PWM Timer by setting CH0EN = 0 (PWM_PCR[0] = 0). (Not recommended)

### 5.12.6 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM Base Address:**<br>**PWM_BA = 0xB800_7000** | | | | |
| **PWM_PPR** | PWM_BA+0x000 | R/W | PWM Pre-scale Register | 0x0000_0000 |
| **PWM_CSR** | PWM_BA+0x004 | R/W | PWM Clock Select Register | 0x0000_0000 |
| **PWM_PCR** | PWM_BA+0x008 | R/W | PWM Control Register | 0x0000_0000 |
| **PWM0_CNR** | PWM_BA+0x00C | R/W | PWM Counter Register 0 | 0x0000_0000 |
| **PWM0_CMR** | PWM_BA+0x010 | R/W | PWM Comparator Register 0 | 0x0000_0000 |
| **PWM0_PDR** | PWM_BA+0x014 | R | PWM Data Register 0 | 0x0000_0000 |
| **PWM1_CNR** | PWM_BA+0x018 | R/W | PWM Counter Register 1 | 0x0000_0000 |
| **PWM1_CMR** | PWM_BA+0x01C | R/W | PWM Comparator Register 1 | 0x0000_0000 |
| **PWM1_PDR** | PWM_BA+0x020 | R | PWM Data Register 1 | 0x0000_0000 |
| **PWM2_CNR** | PWM_BA+0x024 | R/W | PWM Counter Register 2 | 0x0000_0000 |
| **PWM2_CMR** | PWM_BA+0x028 | R/W | PWM Comparator Register 2 | 0x0000_0000 |
| **PWM2_PDR** | PWM_BA+0x02C | R | PWM Data Register 2 | 0x0000_0000 |
| **PWM3_CNR** | PWM_BA+0x030 | R/W | PWM Counter Register 3 | 0x0000_0000 |
| **PWM3_CMR** | PWM_BA+0x034 | R/W | PWM Comparator Register 3 | 0x0000_0000 |
| **PWM3_PDR** | PWM_BA+0x038 | R | PWM Data Register 3 | 0x0000_0000 |
| **PWM_PIER** | PWM_BA+0x03C | R/W | PWM Timer Interrupt Enable Register | 0x0000_0000 |
| **PWM_PIIR** | PWM_BA+0x040 | R/W | PWM Timer Interrupt Indication Register | 0x0000_0000 |

### 5.12.7 Register Description

### PWM Pre-scale Register (PWM_PPR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM_PPR** | PWM_BA+0x000 | R/W | PWM Pre-scale Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **DZL23** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **DZL01** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **PRESCALE23** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **PRESCALE01** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **DZL23** | **Dead Zone Length Register 1**<br>These 8 bits determine the dead-zone length of channel 2, 3 pair.<br>The unit time of dead zone length is received from clock selector 1. |
| [23:16] | **DZL01** | **Dead Zone Length Register 0**<br>These 8 bits determine the dead-zone length of channel 0, 1 pair.<br>The unit time of dead zone length is received from clock selector 0. |
| [15:8] | **PRESCALE23** | **Prescale Register for Channel 2 & 3**<br>Prescale output clock frequency = PCLK / (PRESCALE23 + 1).<br>If PPR=0, then the prescale output clock will be stopped. |
| [7:0] | **PRESCALE01** | **Prescale Register for Channel 0 & 1**<br>Prescale output clock frequency = PCLK / (PRESCALE01 + 1).<br>If PPR=0, then the prescale output clock will be stopped. |

### PWM Clock Select Register (PWM_CSR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM_CSR** | PWM_BA+0x004 | R/W | PWM Clock 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** | **CLKSEL3** | | | **Reserved** | **CLKSEL2** | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | **CLKSEL1** | | | **Reserved** | **CLKSEL0** | | |

| Bits | Description | |
|---|---|---|
| [31:15] | **Reserved** | Reserved. |
| [14:12] | **CLKSEL3** | **Channel 3 Clock Source Selection**<br>Select PWM clock source for PWM timer channel 3<br>000 = Prescale output divided by 2.<br>001 = Prescale output divided by 4.<br>010 = Prescale output divided by 8.<br>011 = Prescale output divided by 16.<br>100 = Prescale output divided by 1. |
| [11] | **Reserved** | Reserved. |
| [10:8] | **CLKSEL2** | **Channel 2 Clock Source Selection**<br>Select PWM clock source for PWM timer channel 2<br>(Table is the same as CH3) |
| [7] | **Reserved** | Reserved. |
| [6:4] | **CLKSEL1** | **Channel 1 Clock Source Selection**<br>Select PWM clock source for PWM timer channel 1<br>(Table is the same as CH3) |
| [3] | **Reserved** | Reserved. |
| [2:0] | **CLKSEL0** | **Channel 0 Clock Source Selection**<br>Select PWM clock source for PWM timer channel 0<br>(Table is the same as CH3) |

## PWM Control Register (PWM_PCR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| PWM_PCR | PWM_BA+0x008 | R/W | PWM Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | CH3MOD | CH3INV | Reserved | CH3EN |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CH2MOD | CH2INV | Reserved | CH2EN | CH1MOD | CH1INV | Reserved | CH1EN |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | DZEN23 | DZEN01 | CH0MOD | CH0INV | Reserved | CH0EN |

| Bits | Description | |
|---|---|---|
| [31:20] | **Reserved** | Reserved. |
| [19] | **CH3MOD** | **Channel 3 Periodic/One-shot Mode**<br>0: PWM counter of channel 3 operates as One-Shot Mode<br>1: PWM counter of channel 3 operates as Periodic Mode |
| [18] | **CH3INV** | **Channel 3 Inverter Switch**<br>0: Inverter OFF. The output polarity of PWM channel 3 will be kept as usual<br>1: Inverter ON. The output polarity of PWM channel 3 will be inverted |
| [17] | **Reserved** | Reserved. |
| [16] | **CH3EN** | **Channel 3 Enable/Disable**<br>0: Disable the output of PWM channel 3<br>1: Enable the output of PWM channel 3 |
| [15] | **CH2MOD** | **Channel 2 Periodic/One-shot Mode**<br>0: PWM counter of channel 2 operates as One-Shot Mode<br>1: PWM counter of channel 2 operates as Periodic Mode |
| [14] | **CH2INV** | **Channel 2 Inverter Switch**<br>0: Inverter OFF. The output polarity of PWM channel 2 will be kept as usual<br>1: Inverter ON. The output polarity of PWM channel 2 will be inverted |
| [13] | **Reserved** | Reserved. |
| [12] | **CH2EN** | **Channel 2 Enable/Disable**<br>0: Disable the output of PWM channel 2<br>1: Enable the output of PWM channel 2 |
| [11] | **CH1MOD** | **Channel 1 Periodic/One-shot Mode**<br>0: PWM counter of channel 1 operates as One-Shot Mode<br>1: PWM counter of channel 1 operates as Periodic Mode |

| | | |
|---|---|---|
| [10] | **CH1INV** | **Channel 1 Inverter Switch**<br>0: Inverter OFF. The output polarity of PWM channel 1 will be kept as usual<br>1: Inverter ON. The output polarity of PWM channel 1 will be inverted |
| [9] | **Reserved** | Reserved. |
| [8] | **CH1EN** | **Channel 1 Enable/Disable**<br>0: Disable the output of PWM channel 1<br>1: Enable the output of PWM channel 1 |
| [7:6] | **Reserved** | Reserved. |
| [5] | **DZEN23** | **Dead-zone Generator 1 Enable/Disable**<br>0: Disable the Dead-Zone output of PWM channel 2, 3<br>1: Enable the Dead-Zone output of PWM channel 2, 3 |
| [4] | **DZEN01** | **Dead-zone Generator 0 Enable/Disable**<br>0: Disable the Dead-Zone output of PWM channel 0, 1<br>1: Enable the Dead-Zone output of PWM channel 0, 1 |
| [3] | **CH0MOD** | **Channel 0 Periodic/One-shot Mode**<br>0: PWM counter of channel 0 operates as One-Shot Mode<br>1: PWM counter of channel 0 operates as Periodic Mode |
| [2] | **CH0INV** | **Channel 0 Inverter Switch**<br>0: Inverter OFF. The output polarity of PWM channel 0 will be kept as usual<br>1: Inverter ON. The output polarity of PWM channel 0 will be inverted |
| [1] | **Reserved** | Reserved. |
| [0] | **CH0EN** | **Channel 0 Enable/Disable**<br>0: Disable the output of PWM channel 0<br>1: Enable the output of PWM channel 0 |

### PWM Counter Register (PWM_CNR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM0_CNR** | PWM_BA+0x00C | R/W | PWM Counter Register 0 | 0x0000_0000 |
| **PWM1_CNR** | PWM_BA+0x018 | R/W | PWM Counter Register 1 | 0x0000_0000 |
| **PWM2_CNR** | PWM_BA+0x024 | R/W | PWM Counter Register 2 | 0x0000_0000 |
| **PWM3_CNR** | PWM_BA+0x030 | R/W | PWM Counter Register 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** |
| **CNR[15:8]** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **CNR[7:0]** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **CNR** | **PWM Counter Reload Value**<br>The PWM period = CNR + 1, and if CNR is set to zero, PWM down-counting will be stopped.<br>**Note:** Software can write a value to CNR at any time, and it will take effect in next PWM period. |

### PWM Comparator Register (PWM_CMR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM0_CMR** | PWM_BA+0x010 | R/W | PWM Comparator Register 0 | 0x0000_0000 |
| **PWM1_CMR** | PWM_BA+0x01C | R/W | PWM Comparator Register 1 | 0x0000_0000 |
| **PWM2_CMR** | PWM_BA+0x028 | R/W | PWM Comparator Register 2 | 0x0000_0000 |
| **PWM3_CMR** | PWM_BA+0x034 | R/W | PWM Comparator Register 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** |
| **CMR[15:8]** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **CMR[7:0]** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **CMR** | **PWM Comparator Register**<br>PWM duty length follows the description below:<br>CMR >= CNR : PWM output is always high.<br>CMR < CNR : PWM output high for (CMR + 1) unit<br>CMR = 0 : PWM output high for 1 unit.<br>(Unit : 1 PWM clock cycle) |

### PWM Data Register (PWM_PDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM0_PDR** | PWM_BA+0x014 | R | PWM Data Register 0 | 0x0000_0000 |
| **PWM1_PDR** | PWM_BA+0x020 | R | PWM Data Register 1 | 0x0000_0000 |
| **PWM2_PDR** | PWM_BA+0x02C | R | PWM Data Register 2 | 0x0000_0000 |
| **PWM3_PDR** | PWM_BA+0x038 | R | PWM Data Register 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** |
| **PDR** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **PDR** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **PDR** | **PWM Data Register**<br>It indicates the current value of the PWM down-counter. |

## PWM Timer Interrupt Enable Register (PWM_PIER)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM_PIER** | PWM_BA+0x03C | R/W | PWM Timer 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** | | | | **PIER3** | **PIER2** | **PIER1** | **PIER0** |

| Bits | Description | |
|---|---|---|
| [31:4] | **Reserved** | Reserved. |
| [3] | **PIER3** | **PWM Timer Channel 3 Interrupt Enable**<br>0: Disable the PWM interrupt function of channel 3<br>1: Enable the PWM interrupt function of channel 3 |
| [2] | **PIER2** | **PWM Timer Channel 2 Interrupt Enable**<br>0: Disable the PWM interrupt function of channel 2<br>1: Enable the PWM interrupt function of channel 2 |
| [1] | **PIER1** | **PWM Timer Channel 1 Interrupt Enable**<br>0: Disable the PWM interrupt function of channel 1<br>1: Enable the PWM interrupt function of channel 1 |
| [0] | **PIER0** | **PWM Timer Channel 0 Interrupt Enable**<br>0: Disable the PWM interrupt function of channel 0<br>1: Enable the PWM interrupt function of channel 0 |

### PWM Timer Interrupt Indication Register (PWM_PIIR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM_PIIR** | PWM_BA+0x040 | R/W | PWM Timer Interrupt Indication 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** | | | | **PIIR3** | **PIIR2** | **PIIR1** | **PIIR0** |

| Bits | Description | |
|---|---|---|
| [31:4] | **Reserved** | Reserved. |
| [3] | **PIIR3** | **PWM Timer Channel 3 Interrupt Flag**<br>This flag is set by hardware when PWM3 down-counter reaches zero, software can clear this bit by writing a one into this bit. |
| [2] | **PIIR2** | **PWM Timer Channel 2 Interrupt Flag**<br>This flag is set by hardware when PWM2 down-counter reaches zero, software can clear this bit by writing a one into this bit. |
| [1] | **PIIR1** | **PWM Timer Channel 1 Interrupt Flag**<br>This flag is set by hardware when PWM1 down-counter reaches zero, software can clear this bit by writing a one into this bit. |
| [0] | **PIIR0** | **PWM Timer Channel 0 Interrupt Flag**<br>This flag is set by hardware when PWM0 down-counter reaches zero, software can clear this bit by writing a one into this bit. |

## 5.13 Watchdog Timer (WDT)

### 5.13.1 Overview

The purpose of Watchdog Timer (WDT) is 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.

### 5.13.2 Features

- 18-bit free running up counter for WDT time-out interval
- Selectable time-out interval ($2^4$ ~ $2^{18}$) and the time-out interval is 0.48828125 ms ~ 8 s if WDT_CLK = 32.768 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 WDTON in PWRON register
- Supports WDT time-out wake-up function only if WDT clock source is selected as 32 kHz

### 5.13.3 Block Diagram

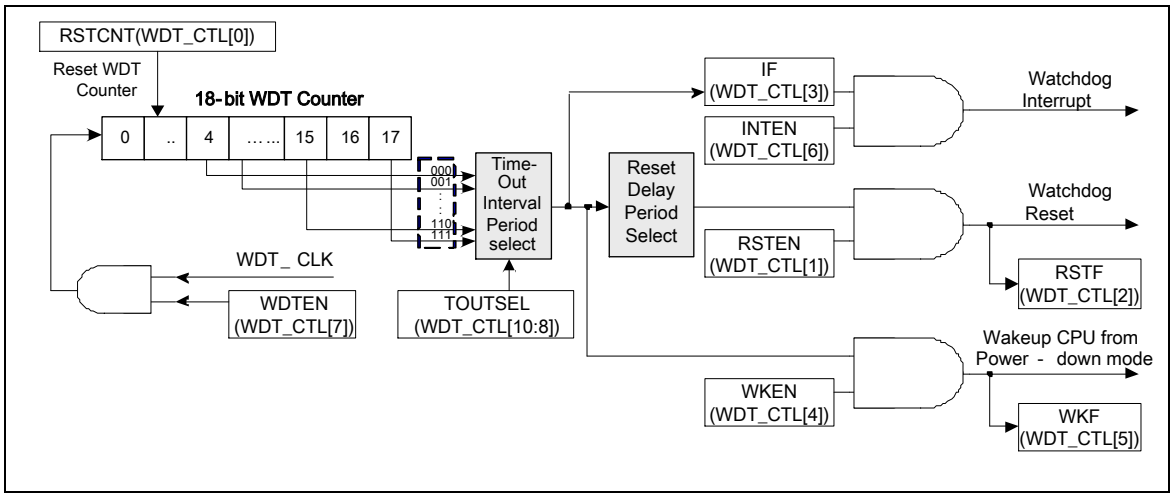



Figure 5.13-1 Watchdog Timer Block Diagram

**Note1:** WDT resets CPU and lasts 63 WDT_CLK.

**Note2:** If user intends to use WDT to wake-up Idle/Power-down mode, it is recommended that CPU clock source is set the same as WDT clock source before CPU enters Power-Down mode.

**Note3:** The WDT reset delay period can be selected as 3/18/130/1026 WDT_CLK.

### 5.13.4 Basic Configuration

The WDT clock control and block diagram are shown as follows.

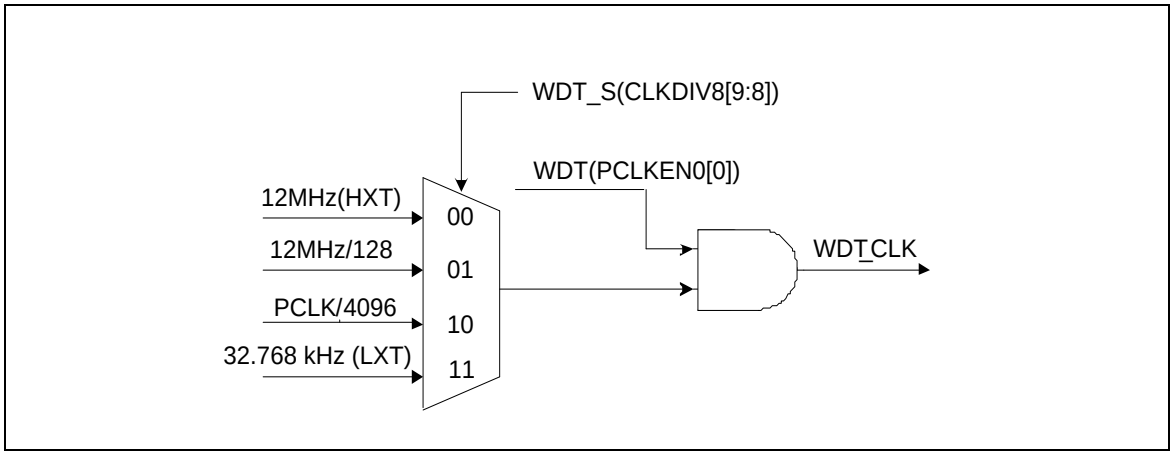



Figure 5.13-2 Watchdog Timer Clock Control

The WDT peripheral clock is enabled in WDT (PCLKEN0[0]) and clock source can be selected in WDT_S (CLKDIV8[9:8]).

WDT controller also can be forced enabled and active in 32 kHz after chip powered on or reset while WDTON is configure to 1 in PWRON register.

### 5.13.5 Functional Description

The WDT includes an 18-bit free running up counter with programmable time-out intervals. Table 5.13-1 shows the WDT time-out interval period selection and Table 5.13-2 shows the WDT reset period timing.

#### 5.13.5.1 *WDT Time-out Interrupt*

Setting WDTEN (WDTCR[7]) to 1 will enable the WDT function and the WDT counter to start counting up. There are eight time-out interval period can be selected by setting TOUTSEL (WDTCR[10:8]). When the WDT up counter reaches the TOUTSEL (WDTCR[10:8]) settings, WDT time-out interrupt will occur then WDT time-out interrupt flag IF (WDT_CTL[3]) will be set to 1 immediately.

#### 5.13.5.2 *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 set RSTCNT (WDT_CTL[0]) to reset the 18-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. Refer to Figure 5.13-3, $T_{RST}$ 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 reset 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.

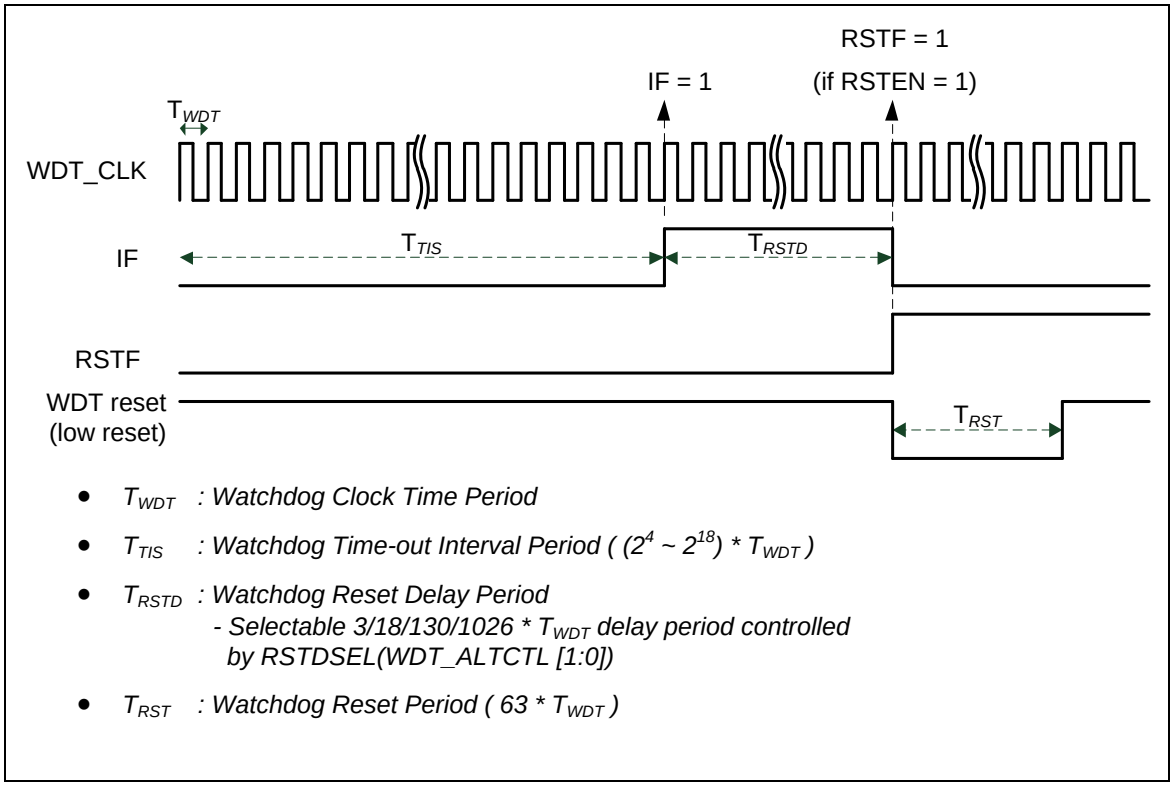
| TOUTSEL | Time-Out Interval Period<br>$T_{TIS}$ | Reset Delay Period ($T_{RSTD}$) |
|---|---|---|
| 000 | $2^4 * T_{WDT}$ | (3/18/130/1026) * $T_{WDT}$ |
| 001 | $2^6 * T_{WDT}$ | (3/18/130/1026) * $T_{WDT}$ |

| | | |
|---|---|---|
| 010 | $2^8 * T_{WDT}$ | (3/18/130/1026) * $T_{WDT}$ |
| 011 | $2^{10} * T_{WDT}$ | (3/18/130/1026) * $T_{WDT}$ |
| 100 | $2^{12} * T_{WDT}$ | (3/18/130/1026) * $T_{WDT}$ |
| 101 | $2^{14} * T_{WDT}$ | (3/18/130/1026) * $T_{WDT}$ |
| 110 | $2^{16} * T_{WDT}$ | (3/18/130/1026) * $T_{WDT}$ |
| 111 | $2^{18} * T_{WDT}$ | (3/18/130/1026) * $T_{WDT}$ |

Table 5.13-1 Watchdog Time-out Interval Period Selection

| WDT_S (CLK_DIVCTL8 [9:8]) | Watchdog Reset Period ($T_{RST}$) |
|---|---|
| 00 | 5.3 us |
| 01 | 682 us |
| 10 | 87.3 ms |
| 11 | 1.95 ms |

Table 5.13-2 Watchdog Reset Period Selection




- $T_{WDT}$ : *Watchdog Clock Time Period*
- $T_{TIS}$ : *Watchdog Time-out Interval Period ( ($2^4$ ~ $2^{18}$) * $T_{WDT}$ )*
- $T_{RSTD}$ : *Watchdog Reset Delay Period*
  *- Selectable 3/18/130/1026 * $T_{WDT}$ delay period controlled by RSTDSEL(WDT_ALTCTL [1:0])*
- $T_{RST}$ : *Watchdog Reset Period ( 63 * $T_{WDT}$ )*

Figure 5.13-3 Watchdog Timer Time-out Interval and Reset Period Timing

5.13.5.3 *WDT Wake-up*

If WDT clock source is selected to 32 kHz, system can be waken-up from Power-down mode

while WDT time-out interrupt signal is generated and WKEN (WDT_CTL[4]) enabled. Notice that user should set XTAL_EN (CLK_PMCON[0]) to enable crystal clock source before system entries power down mode because the system peripheral clock are disabled when system is power down mode. In the meanwhile, the WDT (SYS_WKUPSSR[28]) will set to 1 automatically, user can check WDT (SYS_WKUPSSR[28]) status by software to recognize the system has been waken-up by WDT time-out interrupt or not.

### 5.13.6 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **WDT Base Address:**<br>**WDT_BA = 0xB800_1800** | | | | |
| **WDT_CTL** | WDT_BA+0x00 | R/W | WDT Control Register | 0x0000_0700 |
| **WDT_ALTCTL** | WDT_BA+0x04 | R/W | WDT Alternative Control Register | 0x0000_0000 |

### 5.13.7 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_0700 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **ICEDEBUG** | **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** | **Reserved** | **WKEN** | **IF** | **RSTF** | **RSTEN** | **RSTCNT** |

| Bits | Description | |
|---|---|---|
| [31] | **ICEDEBUG** | **ICE Debug Mode Acknowledge Disable Control (Write Protect)**<br>0 = ICE debug mode acknowledgement affects WDT counting.<br>WDT up counter will be held while CPU is held by ICE.<br>1 = ICE debug mode acknowledgement Disabled.<br>WDT up counter will keep going no matter CPU is held by ICE or not.<br>**Note:** This bit is write protected. Refer to the REGWRPROT register. |
| [30:11] | **Reserved** | Reserved. |
| [10:8] | **TOUTSEL** | **WDT Time-out Interval Selection (Write Protect)**<br>These three bits select the time-out interval period for the WDT.<br>000 = $2^4$ *$T_{WDT}$.<br>001 = $2^6$ *$T_{WDT}$.<br>010 = $2^8$ *$T_{WDT}$.<br>011 = $2^{10}$ *$T_{WDT}$.<br>100 = $2^{12}$ *$T_{WDT}$.<br>101 = $2^{14}$ *$T_{WDT}$.<br>110 = $2^{16}$ *$T_{WDT}$.<br>111 = $2^{18}$ *$T_{WDT}$.<br>**Note:** This bit is write protected. Refer to the REGWRPROT register. |
| [7] | **WDTEN** | **WDT Enable Control (Read Only)**<br>Please refer to 0 about how to enable WDT function.<br>0 = WDT Disabled (This action will reset the internal up counter value).<br>1 = WDT Enabled. |

| | | |
|---|---|---|
| [6] | **INTEN** | **WDT Time-out Interrupt Enable Control (Write Protect)**<br>If this bit is enabled, the WDT time-out interrupt signal is generated and inform to CPU.<br>0 = WDT time-out interrupt Disabled.<br>1 = WDT time-out interrupt Enabled.<br>**Note:** This bit is write protected. Refer to the REGWRPROT register. |
| [5] | **Reserved** | Reserved. |
| [4] | **WKEN** | **WDT Time-out Wake-up Function Control (Write Protect)**<br>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.<br>0 = Wake-up trigger event Disabled if WDT time-out interrupt signal generated.<br>1 = Wake-up trigger event Enabled if WDT time-out interrupt signal generated.<br>**Note1:** This bit is write protected. Refer to the REGWRPROT register.<br>**Note2:** Chip can be woken-up by WDT time-out interrupt signal generated only if WDT clock source is selected to 32 kHz oscillator. |
| [3] | **IF** | **WDT Time-out Interrupt Flag**<br>This bit will set to 1 while WDT up counter value reaches the selected WDT time-out interval<br>0 = WDT time-out interrupt did not occur.<br>1 = WDT time-out interrupt occurred.<br>**Note:** This bit is cleared by writing 1 to it. |
| [2] | **RSTF** | **WDT Time-out Reset Flag**<br>This bit indicates the system has been reset by WDT time-out reset or not.<br>0 = WDT time-out reset did not occur.<br>1 = WDT time-out reset occurred.<br>**Note:** This bit is cleared by writing 1 to it. |
| [1] | **RSTEN** | **WDT Time-out Reset Enable Control (Read Only)**<br>This bit high indicates that WDT time-out reset function enabled If the WDT up counter value has not been cleared after the specific WDT reset delay period expires.<br>Please refer to 0 about how to enable WDT Time-Out Reset function.<br>0 = WDT time-out reset function Disabled.<br>1 = WDT time-out reset function Enabled. |
| [0] | **RSTCNT** | **Reset WDT Up Counter (Write Protect)**<br>0 = No effect.<br>1 = Reset the internal 18-bit WDT up counter value.<br>**Note1:** This bit is write protected. Refer to the REGWRPROT register.<br>**Note2:** This bit will be automatically cleared by hardware. |

### 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** | | | | | | **WTRDSEL** | |

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [1:0] | **RSTDSEL** | **WDT Reset Delay Selection (Write Protect)**<br>When WDT time-out happened, user has a time named WDT Reset Delay Period to clear WDT counter by setting RSTCNT (WDT_CTL[0]) to prevent WDT time-out reset happened. User can select a suitable setting of RSTDSEL for different WDT Reset Delay Period.<br>00 = WDT Reset Delay Period is 1026 * WDT_CLK.<br>01 = WDT Reset Delay Period is 130 * WDT_CLK.<br>10 = WDT Reset Delay Period is 18 * WDT_CLK.<br>11 = WDT Reset Delay Period is 3 * WDT_CLK.<br>**Note1:** This bit is write protected. Refer to the REGWRPROT register.<br>**Note2:** This register will be reset to 0 if WDT time-out reset happened. |

## 5.14 Windowed Watchdog Timer (WWDT)

### 5.14.1 Overview

The Window Watchdog Timer (WWDT) is used to perform a system reset within a specified window period to prevent software run to uncontrollable status by any unpredictable condition.

### 5.14.2 Features

- 6-bit down counter value (CNTDAT) and 6-bit compare value (CMPDAT) to make the WWDT time-out window period flexible.
- Supports 4-bit value (PSCSEL) to programmable maximum 11-bit prescale counter period of WWDT counter.

### 5.14.3 Block Diagram

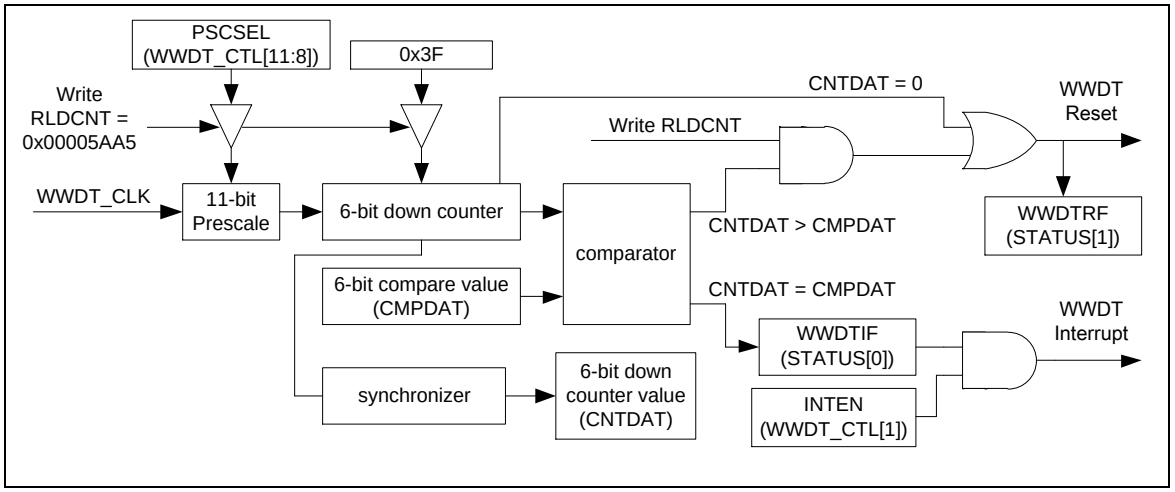



Figure 5.14-1 WWDT Block Diagram

### 5.14.4 Basic Configuration

The WWDT clock control and block diagram are shown as follows.

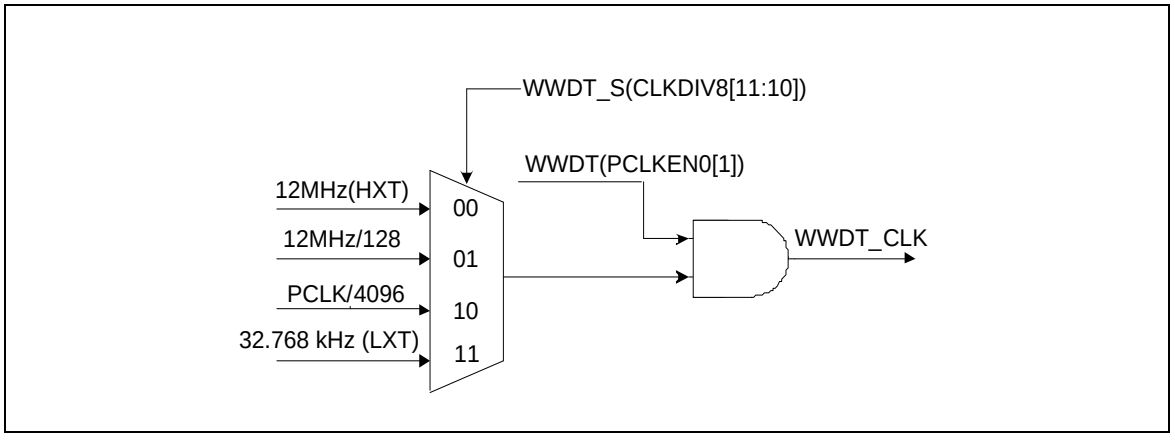



Figure 5.14-2 WWDT Clock Control

The WWDT peripheral clock is enabled in WWDT (PCLKEN[1]) and clock source can be selected in WWDT_S[1:0] (CLKDIV8[11:10]).

### 5.14.5 Function 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 4096 (PCLK/4096), external 12 MHz oscillator or internal 32 kHz oscillator 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 the following table.

| PSCSEL | Prescaler Value | Max. Time-Out Period | Max. Time-Out Interval (WWDT_CLK=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 * $T_{WWDT}$ | 1.024 s |
| 1100 | 768 | 768 * 64 * $T_{WWDT}$ | 1.536 s |
| 1101 | 1024 | 1024 * 64 * $T_{WWDT}$ | 2.048 s |
| 1110 | 1536 | 1536 * 64 * $T_{WWDT}$ | 3.072 s |
| 1111 | 2048 | 2048 * 64 * $T_{WWDT}$ | 4.096 s |

Table 5.14-1 Window Watchdog Prescaler Value Selection

#### 5.14.5.1 *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[0]), 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.

#### 5.14.5.2 *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.

5.14.5.3 *WWDT Reset System*

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 info system reset. If 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 immediately to cause chip reset also.

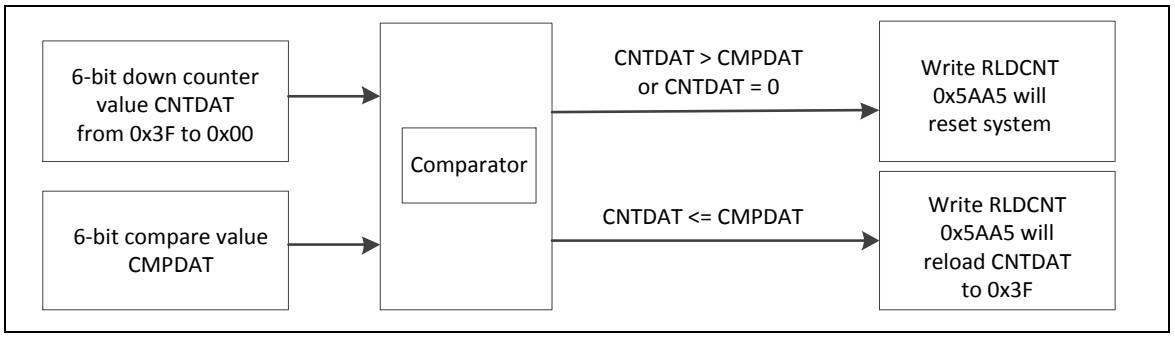



Figure 5.14-3 WWDT Reset and Reload Behavior

#### 5.14.5.4 *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. Notice that if user set 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.

| PSCSEL | Prescale Value | Valid CMPDAT Value |
|---|---|---|
| 0000 | 1 | 0x3 ~ 0x3F |
| 0001 | 2 | 0x2 ~ 0x3F |
| Others | Others | 0x0 ~ 0x3F |

Table 5.14-2 CMPDA Setting Limitation

### 5.14.6 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **WWDT Base Address:**<br>**WWDT_BA = 0xB800_1900** | | | | |
| **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 |

### 5.14.7 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 |
|---|---|---|---|---|---|---|---|
| **WWDT_RLDCNT[31:24]** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **WWDT_RLDCNT[23:16]** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **WWDT_RLDCNT[15:8]** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **WWDT_RLDCNT[7:0]** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:0] | **WWDT_RLDCNT** | **WWDT Reload Counter Register**<br>Writing 0x00005AA5 to this register will reload the WWDT counter value to 0x3F.<br>**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 generate immediately. |

## 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 write 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 Control**<br>0 = ICE debug mode acknowledgement effects WWDT counting.<br>WWDT down counter will be held while CPU is held by ICE.<br>1 = ICE debug mode acknowledgement Disabled.<br>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**<br>Set this register to adjust the valid reload window.<br>**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 immediately. |
| [15:12] | **Reserved** | Reserved. |
| [11:8] | **PSCSEL** | **WWDT Counter Prescale Period Selection**<br>0000 = Pre-scale is 1; Max time-out period is 1 * 64 * $T_{WWDT}$.<br>0001 = Pre-scale is 2; Max time-out period is 2 * 64 * $T_{WWDT}$.<br>0010 = Pre-scale is 4; Max time-out period is 4 * 64 * $T_{WWDT}$.<br>0011 = Pre-scale is 8; Max time-out period is 8 * 64 * $T_{WWDT}$.<br>0100 = Pre-scale is 16; Max time-out period is 16 * 64 * $T_{WWDT}$.<br>0101 = Pre-scale is 32; Max time-out period is 32 * 64 * $T_{WWDT}$.<br>0110 = Pre-scale is 64; Max time-out period is 64 * 64 * $T_{WWDT}$.<br>0111 = Pre-scale is 128; Max time-out period is 128 * 64 * $T_{WWDT}$.<br>1000 = Pre-scale is 192; Max time-out period is 192 * 64 * $T_{WWDT}$.<br>1001 = Pre-scale is 256; Max time-out period is 256 * 64 * $T_{WWDT}$.<br>1010 = Pre-scale is 384; Max time-out period is 384 * 64 * $T_{WWDT}$. |

| | | |
|---|---|---|
| | | 1011 = Pre-scale is 512; Max time-out period is 512 * 64 * $T_{WWDT}$.<br>1100 = Pre-scale is 768; Max time-out period is 768 * 64 * $T_{WWDT}$.<br>1101 = Pre-scale is 1024; Max time-out period is 1024 * 64 * $T_{WWDT}$.<br>1110 = Pre-scale is 1536; Max time-out period is 1536 * 64 * $T_{WWDT}$.<br>1111 = Pre-scale is 2048; Max time-out period is 2048 * 64 * $T_{WWDT}$. |
| [7:2] | **Reserved** | Reserved. |
| [1] | **INTEN** | **WWDT Interrupt Enable Control Bit**<br>If this bit is enabled, the WWDT counter compare match interrupt signal is generated and inform to CPU.<br>0 = WWDT counter compare match interrupt Disabled.<br>1 = WWDT counter compare match interrupt Enabled. |
| [0] | **WWDTEN** | **WWDT Enable Control Bit**<br>Set this bit to enable WWDT counter counting.<br>0 = WWDT counter is stopped.<br>1 = WWDT counter is starting 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**<br>This bit indicates the system has been reset by WWDT time-out reset or not.<br>0 = WWDT time-out reset did not occur.<br>1 = WWDT time-out reset occurred.<br>**Note:** This bit is cleared by writing 1 to it. |
| [0] | **WWDTIF** | **WWDT Compare Match Interrupt Flag**<br>This bit indicates the interrupt flag status of WWDT while WWDT counter value matches CMPDAT (WWDT_CTL[21:16]).<br>0 = No effect.<br>1 = WWDT counter value matches CMPDAT.<br>**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**<br>CNTDAT will be updated continuously to monitor 6-bit WWDT down counter value. |

## 5.15 Real Time Clock (RTC)

### 5.15.1 Overview

The Real Time Clock (RTC) controller provides the real time clock and calendar information. The clock source of RTC controller is from an external 32.768 kHz low-speed crystal which connected at pins X32_IN and X32_OUT (refer to pin Description). The RTC controller provides the real time clock (hour, minute, second) in RTC_TIME (RTC Time Loading Register) as well as calendar information (year, month, day) in RTC_CAL (RTC Calendar Loading Register). It also offers RTC alarm function that user can preset the alarm time in RTC_TALM (RTC Time Alarm Register) and alarm calendar in RTC_CALM (RTC Calendar Alarm Register). The data format of RTC time and calendar message are all expressed in BCD (Binary Coded Decimal) format.

The RTC controller supports periodic RTC Time Tick and Alarm Match interrupts. The periodic RTC 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 which are selected by RTC_TICK (RTC_TICK[2:0] Time Tick Register). When real time and calendar message in RTC_TIME and RTC_CAL are equal to alarm time and calendar settings in RTC_TALM and RTC_CALM, the ALMIF (RTC_INTSTS [0] RTC Alarm Interrupt Flag) is set to 1 and the RTC alarm interrupt signal is generated if the ALMIEN (RTC_INTEN [0] Alarm Interrupt Enable) is enabled.

Both RTC Time Tick and Alarm Match interrupt signal can cause chip to wake-up from Idle or Power-down mode if the corresponding interrupt enable bit (ALMIEN or TICKIEN) is set to 1 before chip enters Idle or Power-down mode.

Real Time Clock (RTC) block can operate with independent power supply (RTC_VDD) while the system power is off.

### 5.15.2 Features

- Supports real time counter and calendar counter for RTC time and calendar check.
- Supports time (hour, minute, second) and calendar (year, month, day) alarm and alarm mask settings.
- Selectable 12-hour or 24-hour time scale.
- Supports Leap Year indication.
- Supports Day of the Week counter.
- Supports frequency compensation mechanism for 32.768 kHz clock source.
- 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 chip wake-up from Idle or Power-down mode while alarm or relative alarm interrupt is generated.
- Supports 64 bytes spare registers to store user's important information.
- Supports power on/off control mechanism to control system core power.

### 5.15.3 Block Diagram

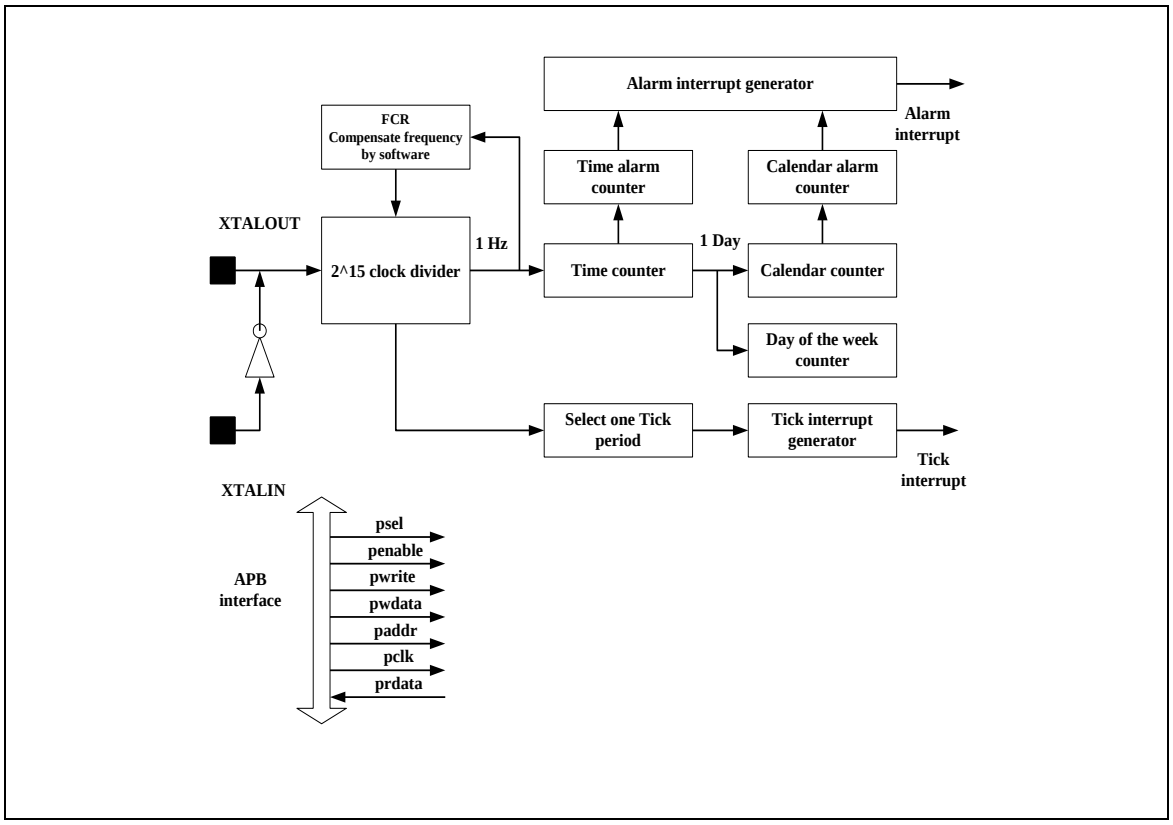



Figure 5.15-1 RTC Functional Block Diagram

### 5.15.4 Basic Configuration

Before using RTC, it's necessary to set RTC (CLK_PCLKEN[2]) high to enable clock for RTC control register access.

By setting MFP_GPH4 (SYS_MFP_GPHL[19:16]) and MFP_GPI3 (SYS_MFP_GPIL[15:12]) properly, RTC will output tick signal to pin PH.4 or PI.4. Please refer to register SYS_MFP_GPHL and SYS_MFP_GPIL for detail information.

### 5.15.5 Functional Description

#### 5.15.5.1 *RTC Initiation*

When RTC block is power on, programmer has to write a number (0xa5eb1357) to RTC_INIT to reset all logic. RTC_INIT act as hardware reset circuit. Once RTC_INIT has been set as 0xa5eb1357, user cannot reload any other value.

#### 5.15.5.2 *RTC write enable*

Register RTC_RWEN bit 15~0 is RTC read /write password. It is used to avoid signal interference from system during system power off. RTC_RWEN bit 15~0 has to be set as 0xa965 before user want to write new data into all registers besides RTC_INIT. If user set RTC_RWEN as 0xa965, RWENF will

be raised high. Then user can feel free to write data into register. RWENF will keep high for a short period (about 24ms) and it will be pull low by internal state machine automatically.

#### 5.15.5.3 *Frequency Compensation*

The RTC_FREQADJ allows software control digital compensation of a 32.768 KHz crystal oscillator. User can utilize a frequency counter to measure RTC tick output in pin PH.4 and PI.3 during manufacture, and store the value in Flash memory for retrieval when the product is first power on.

#### 5.15.5.4 *Time and Calendar counter*

RTC_TIME and RTC_CAL are used to read the time and calendar. RTC_TALM and RTC_CALM are used as alarm. They are all BCD counters.

#### 5.15.5.5 *12/24 hour Time scale selection*

The 12/24 hour time scale selection decided by 24HEN (RTC_TIMEFMT[0]).

#### 5.15.5.6 *Day of the week counter*

Count from Sunday to Saturday

#### 5.15.5.7 *Tick Time interrupt*

RTC block use a counter to calibrate the tick time count value. When the value in counter reaches zero, RTC will issue an interrupt.

#### 5.15.5.8 *RTC register property*

When system power is off but RTC power is on, data stored in RTC registers will not be lost except RTC_TSSR, RTC_INTEN and RTC_INTSTS. Because of difference between RTC clock and system clock, every time user write new data to any one register, the register will be updated until 2 RTC clock later (60us).

In addition, user must be aware that RTC block does not check whether loaded data is out of bounds. RTC does not check rationality between RTC_WEEKDAY and RTC_CLR either.

**Note:**

RTC_TALM, RTC_CALM, RTC_TIME and RTC_CAL are all BCD counter, but RTC_FREQADJ is not a BCD counter.

Programmer must be aware that the RTC block does not check whether the loaded value is reasonable. For example, Load RTC_CAL as 201a (year), 13 (month), 00 (day), or RTC_CAL does not match with RTC_WEEKDAY, etc.

In RTC_TIME and RTC_TALM, only 2 BCD digits are used to express “year”. We assume 2 BCD digits of XY denote 20XY, but not 19XY or 21XY.

### 5.15.6 Register Map

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC Base Address:**<br>**RTC_BA = 0xB800_4000** | | | | |
| **RTC_INIT** | RTC_BA+0x000 | R/W | RTC Initiation Register | 0x0000_0000 |
| **RTC_RWEN** | RTC_BA+0x004 | R/W | RTC Access Enable Register | 0x0000_0000 |
| **RTC_FREQADJ** | RTC_BA+0x008 | R/W | RTC Frequency Compensation Register | 0x0000_0700 |
| **RTC_TIME** | RTC_BA+0x00C | R/W | RTC Time Counter Register | 0x0000_0000 |
| **RTC_CAL** | RTC_BA+0x010 | R/W | RTC Calendar Counter Register | 0x0005_0101 |
| **RTC_TIMEFMT** | RTC_BA+0x014 | R/W | RTC Time Format Selection Register | 0x0000_0001 |
| **RTC_WEEKDAY** | RTC_BA+0x018 | R/W | RTC Day of the Week Register | 0x0000_0006 |
| **RTC_TALM** | RTC_BA+0x01C | R/W | RTC Time Alarm Register | 0x0000_0000 |
| **RTC_CALM** | RTC_BA+0x020 | R/W | RTC Calendar Alarm Register | 0x0000_0000 |
| **RTC_LEAPYEAR** | RTC_BA+0x024 | R | RTC Leap Year Indicator Register | 0x0000_0000 |
| **RTC_INTEN** | RTC_BA+0x028 | R/W | RTC Interrupt Enable Register | 0x0000_0000 |
| **RTC_INTSTS** | RTC_BA+0x02C | R/W | RTC Interrupt Status Register | 0x0000_0000 |
| **RTC_TICK** | RTC_BA+0x030 | R/W | RTC Time Tick Register | 0x0000_0000 |
| **RTC_PWRCTL** | RTC_BA+0x034 | R/W | RTC Power Control Register | 0x0000_7000 |
| **RTC_PWRCNT** | RTC_BA+0x038 | R | RTC Power Control Counter Register | 0x0000_0000 |
| **RTC_SPR0** | RTC_BA+0x040 | R/W | RTC Spare Register 0 | 0x0000_0000 |
| **RTC_SPR1** | RTC_BA+0x044 | R/W | RTC Spare Register 1 | 0x0000_0000 |
| **RTC_SPR2** | RTC_BA+0x048 | R/W | RTC Spare Register 2 | 0x0000_0000 |
| **RTC_SPR3** | RTC_BA+0x04C | R/W | RTC Spare Register 3 | 0x0000_0000 |
| **RTC_SPR4** | RTC_BA+0x050 | R/W | RTC Spare Register 4 | 0x0000_0000 |
| **RTC_SPR5** | RTC_BA+0x054 | R/W | RTC Spare Register 5 | 0x0000_0000 |
| **RTC_SPR6** | RTC_BA+0x058 | R/W | RTC Spare Register 6 | 0x0000_0000 |
| **RTC_SPR7** | RTC_BA+0x05C | R/W | RTC Spare Register 7 | 0x0000_0000 |
| **RTC_SPR8** | RTC_BA+0x060 | R/W | RTC Spare Register 8 | 0x0000_0000 |
| **RTC_SPR9** | RTC_BA+0x064 | R/W | RTC Spare Register 9 | 0x0000_0000 |
| **RTC_SPR10** | RTC_BA+0x068 | R/W | RTC Spare Register 10 | 0x0000_0000 |
| **RTC_SPR11** | RTC_BA+0x06C | R/W | RTC Spare Register 11 | 0x0000_0000 |
| **RTC_SPR12** | RTC_BA+0x070 | R/W | RTC Spare Register 12 | 0x0000_0000 |
| **RTC_SPR13** | RTC_BA+0x074 | R/W | RTC Spare Register 13 | 0x0000_0000 |
| **RTC_SPR14** | RTC_BA+0x078 | R/W | RTC Spare Register 14 | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **RTC_SPR15** | RTC_BA+0x07C | R/W | RTC Spare Register 15 | 0x0000_0000 |

### 5.15.7 Register Description

### RTC Initiation Register (RTC_INIT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_INIT** | RTC_BA+0x000 | 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** | | | | | | | **INIT/Active** |

| Bits | Description | |
|---|---|---|
| [31:1] | **INIT** | **RTC Initiation**<br>After RTC block is powered on, RTC is at reset state. User has to write a number (0x a5eb1357) to INIT to make RTC leaving reset state. Once the INIT is written as 0xa5eb1357, the RTC will be in un-reset state permanently.<br>The INIT is a write-only field and read value will be always “0”. |
| [0] | **INIT/Active** | **RTC Active Status (Read Only)**<br>0 = RTC is at reset state.<br>1 = RTC is at normal active state. |

### RTC Access Enable Register (RTC_RWEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_RWEN** | RTC_BA+0x004 | R/W | RTC Access Enable Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:17] | **Reserved** | Reserved. |
| [16] | **RWENF** | **RTC Register Access Enable Flag (Read Only)**<br>0 = RTC register read/write Disabled.<br>1 = RTC register read/write Enabled.<br>**Note:** This bit will be set after RWENPASSWD (RTC_RWEN[15:0]) register is load a 0xA965, and be cleared automatically after 1024 RTC clock. |
| [15:0] | **RWENPASSWD** | **RTC Register Access Enable Password (Write Only)**<br>Writing 0xA965 to this field will enable RTC access and keep 1024 RTC clock. |

## RTC Frequency Compensation Register (RTC_FREQADJ)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_FREQADJ** | RTC_BA+0x008 | R/W | RTC Frequency Compensation Register | 0x0000_0700 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **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 | |
|---|---|---|
| [31:12] | **Reserved** | Reserved. |
| [11:8] | **INTEGER** | **Integer Part** |

| Integer part of detected value | RTC_FREQADJ[11:8] | Integer part of detected value | RTC_FREQADJ[11:8] |
|---|---|---|---|
| 32776 | 1111 | 32768 | 0111 |
| 32775 | 1110 | 32767 | 0110 |
| 32774 | 1101 | 32766 | 0101 |
| 32773 | 1100 | 32765 | 0100 |
| 32772 | 1011 | 32764 | 0011 |
| 32771 | 1010 | 32763 | 0010 |
| 32770 | 1001 | 32762 | 0001 |
| 32769 | 1000 | 32761 | 0000 |

| | | |
|---|---|---|
| [7:6] | **Reserved** | Reserved. |
| [5:0] | **FRACTION** | **Fraction Part**<br>Formula: FRACTION = (fraction part of detected value) X 60.<br>**Note:** Digit in FCR must be expressed as hexadecimal number. |

| | | |
|---|---|---|
| **Frequency Compensation** | **Example 1** | **Frequency Counter Measurement: 32773.65Hz**<br>**Integer Part: 32773 => RTC_FREQADJ[11:8] = 0xc**<br>**Fraction Part: 0.65 X 60 = 39(0x27) => RTC_FREQADJ[5:0]=0x27** |
| | Example 2 | Frequency counter measurement: 32765.27Hz<br>Integer part: 32765=> RTC_FREQADJ[11:8] = 0x4 |

| | | |
|---|---|---|
| | | Fraction part: 0.27 X 60 = 16.2(0x10) => RTC_FREQADJ[5:0] = 0x10 |

### RTC Time Counter Register (RTC_TIME)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| RTC_TIME | RTC_BA+0x00C | R/W | RTC Time Counter Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | TENHOUR | | HOUR | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | TENMINUTE | | | MINUTE | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | TENSECOND | | | SECOND | | | |

| Bits | Description | |
|---|---|---|
| [31:22] | **Reserved** | Reserved. |
| [21:20] | **TENHOUR** | 10 Hour Time Digit (0 ~ 2) |
| [19:16] | **HOUR** | 1 Hour Time Digit (0 ~ 9) |
| [15] | **Reserved** | Reserved. |
| [14:12] | **TENMINUTE** | 10 Min Time Digit (0 ~ 5) |
| [11:8] | **MINUTE** | 1 Min Time Digit (0 ~ 9) |
| [7] | **Reserved** | Reserved. |
| [6:4] | **TENSECOND** | 10 Sec Time Digit (0 ~ 5) |
| [3:0] | **SECOND** | 1 Sec Time Digit (0 ~ 9) |

**Note:** RTC_TIME is a BCD digit counter and RTC will not check loaded data.

### RTC Calendar Counter Register (RTC_CAL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_CAL** | RTC_BA+0x010 | R/W | RTC Calendar Counter Register | 0x0005_0101 |

| 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** | | | **TENMONTH** | **MONTH** | | | |
| **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] | **TENMONTH** | 10-Month Calendar Digit (0 ~ 1) |
| [11:8] | **MONTH** | 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:** RTC_CAL is a BCD digit counter and RTC will not check loaded data.

## RTC Time Format Selection Register (RTC_TIMEFMT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_TIMEFMT** | RTC_BA+0x014 | R/W | RTC Time Format Selection 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** | | | | | | | **24HEN** |

| Bits | Description | |
|---|---|---|
| [31:1] | **Reserved** | Reserved. |
| [0] | **24HEN** | **24-hour / 12-hour Mode Selection**<br>It indicate that TLR and TAR are in 24-hour mode or 12-hour mode<br>0 = 12-hour time format with am and pm indication selected.<br>1 = 24-hour time format selected. |

| 24-hour time scale | 12-hour time scale | 24-hour time scale | 12-hour time scale |
|---|---|---|---|
| 00 | 12(AM12) | 12 | 32(PM12) |
| 01 | 01(AM01) | 13 | 21(PM01) |
| 02 | 02(AM02) | 14 | 22(PM02) |
| 03 | 03(AM03) | 15 | 23(PM03) |
| 04 | 04(AM04) | 16 | 24(PM04) |
| 05 | 05(AM05) | 17 | 25(PM05) |
| 06 | 06(AM06) | 18 | 26(PM06) |
| 07 | 07(AM07) | 19 | 27(PM07) |
| 08 | 08(AM08) | 20 | 28(PM08) |
| 09 | 09(AM09) | 21 | 29(PM09) |
| 10 | 10(AM10) | 22 | 30(PM10) |
| 11 | 11(AM11) | 23 | 31(PM11) |

## RTC Day of the Week Register (RTC_WEEKDAY)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_WEEKDAY** | RTC_BA+0x018 | 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**<br>0x0 = Sunday.<br>0x1 = Monday.<br>0x2 = Tuesday.<br>0x3 = Wednesday.<br>0x4 = Thursday.<br>0x5 = Friday.<br>0x6 = Saturday.<br>Others = Reserved. |

## RTC Time Alarm Register (RTC_TALM)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| RTC_TALM | RTC_BA+0x01C | R/W | RTC Time Alarm Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | HRALM_MSK | MINALM_MSK | SECALM_MSK | Reserved | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | TENHOUR | | HOUR | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | TENMINUTE | | | MINUTE | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | TENSECOND | | | SECOND | | | |

| Bits | Description | |
|---|---|---|
| [31] | **Reserved** | Reserved. |
| [30] | **HRALM_MSK** | **Hour Alarm Mask**<br>This bit control if TENHOUR (RTC_TALM[21:20] and HOUR (RTC_TALM[19:16]) could trigger RTC timer alarm.<br>0 = TENHOUR (RTC_TALM[21:20] and HOUR (RTC_TALM[19:16]) could trigger RTC time alarm.<br>1 = TENHOUR (RTC_TALM[21:20] and HOUR (RTC_TALM[19:16]) couldn't trigger RTC time alarm. |
| [29] | **MINALM_MSK** | **Minute Alarm Mask**<br>This bit control if TENMINUTE (RTC_TALM[14:12] and MINUTE (RTC_TALM[11:8]) could trigger RTC timer alarm.<br>0 = TENMINUTE (RTC_TALM[14:12] and MINUTE (RTC_TALM[11:8]) could trigger RTC time alarm.<br>1 = TENMINUTE (RTC_TALM[14:12] and MINUTE (RTC_TALM[11:8]) couldn't trigger RTC time alarm. |
| [28] | **SECALM_MSK** | **Minute Alarm Mask**<br>This bit control if TENSECOND (RTC_TALM[6:4] and SECOND (RTC_TALM[3:0]) could trigger RTC timer alarm.<br>0 = TENSECOND (RTC_TALM[6:4] and SECOND (RTC_TALM[3:0]) could trigger RTC time alarm.<br>1 = TENSECOND (RTC_TALM[6:4] and SECOND (RTC_TALM[3:0]) couldn't trigger RTC time alarm. |
| [27:22] | **Reserved** | Reserved. |
| [21:20] | **TENHOUR** | 10 Hour Time Digit (0 ~ 2) |
| [19:16] | **HOUR** | 1 Hour Time Digit (0 ~ 9) |
| [15] | **Reserved** | Reserved. |
| [14:12] | **TENMINUTE** | 10 Min Time Digit (0 ~ 5) |

| | | |
|---|---|---|
| [11:8] | **MINUTE** | 1 Min Time Digit (0 ~ 9) |
| [7] | **Reserved** | Reserved. |
| [6:4] | **TENSECOND** | 10 Sec Time Digit (0 ~ 5) |
| [3:0] | **SECOND** | 1 Sec Time Digit (0 ~ 9) |

**Note1:** RTC_TALM is a BCD digit counter and RTC will not check loaded data.

**Note2:** Set all MSK bits high would disable calendar alarm functionality.

## RTC Calendar Alarm Register (RTC_CALM)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| RTC_CALM | RTC_BA+0x020 | R/W | RTC Calendar Alarm Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| WKDALM_MSK | YRALM_MSK | MONALM_MSK | DAYALM_MSK | Reserved | WEEKDAY | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TENYEAR | | | | YEAR | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | TENMONTH | MONTH | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | TENDAY | | DAY | | | |

| Bits | Description | |
|---|---|---|
| [31] | **WKDALM_MSK** | **Day of Week Alarm Mask**<br>This bit controls if WEEKDAY (RTC_CALM[26:24]) could trigger RTC timer alarm.<br>0 = WEEKDAY (RTC_CALM[26:24]) could trigger RTC time alarm.<br>1 = WEEKDAY (RTC_CALM[26:24]) couldn't trigger RTC time alarm. |
| [30] | **YRALM_MSK** | **Year Alarm Mask**<br>This bit controls if TENYEAR (RTC_CALM[23:20]) and YEAR (RTC_CALM[19:16]) could trigger RTC timer alarm.<br>0 = TENYEAR (RTC_CALM[23:20]) and YEAR (RTC_CALM[19:16]) could trigger RTC time alarm.<br>1 = TENYEAR (RTC_CALM[23:20]) and YEAR (RTC_CALM[19:16]) couldn't trigger RTC time alarm. |
| [29] | **MONALM_MSK** | **Month Alarm Mask**<br>This bit controls if TENMONTH (RTC_CALM[12]) and MONTH (RTC_CALM[11:8]) could trigger RTC timer alarm.<br>0 = TENMONTH (RTC_CALM[12]) and MONTH (RTC_CALM[11:8]) could trigger RTC time alarm.<br>1 = TENMONTH (RTC_CALM[12]) and MONTH (RTC_CALM[11:8]) couldn't trigger RTC time alarm. |
| [28] | **DAYALM_MSK** | **Day Alarm Mask**<br>This bit controls if TENDAY (RTC_CALM[5:4]) and DAY (RTC_CALM[3:0]) could trigger RTC timer alarm.<br>0 = TENDAY (RTC_CALM[5:4]) and DAY (RTC_CALM[3:0]) could trigger RTC time alarm.<br>1 = TENDAY (RTC_CALM[5:4]) and DAY (RTC_CALM[3:0]) couldn't trigger RTC time alarm. |
| [27] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [26:24] | **WEEKDAY** | **Day of the Week**<br>000 = Sunday.<br>001 = Monday.<br>010 = Tuesday.<br>011 = Wednesday.<br>100 = Thursday.<br>101 = Friday.<br>110 = Saturday.<br>111 = Reserved. |
| [23:20] | **TENYEAR** | 10-Year Calendar Digit (0 ~ 9) |
| [19:16] | **YEAR** | 1-Year Calendar Digit (0 ~ 9) |
| [15:13] | **Reserved** | Reserved. |
| [12] | **TENMINUTE** | 10-Month Calendar Digit (0 ~ 1) |
| [11:8] | **MINUTE** | 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) |

**Note1:** RTC_CALM is a BCD digit counter and RTC will not check loaded data.

**Note2:** Set all MSK bits high would disable calendar alarm functionality.

## RTC Leap Year Indicator Register (RTC_LEAPYEAR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_LEAPYEAR** | RTC_BA+0x024 | 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 Indicator (Read Only)**<br>0 = It indicates that this year is not a leap year.<br>1 = It indicates that this year is leap year. |

## RTC Interrupt Enable Register (RTC_INTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| RTC_INTEN | RTC_BA+0x028 | R/W | RTC 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 | | KEYPRESIEN | RELALMIEN | PWRSWIEN | WAKEUPIEN | TICKIEN | ALMIEN |

| Bits | Description | |
|---|---|---|
| [31:6] | **Reserved** | Reserved. |
| [5] | **KEYPRESIEN** | **Key Press Interrupt Enable**<br>0 = Key Press interrupt disable.<br>1 = RTC Relative Alarm interrupt enable. |
| [4] | **RELALMIEN** | **Relative Alarm Interrupt Enable**<br>0 = RTC Relative Alarm interrupt disable.<br>1 = RTC Relative Alarm interrupt enable. |
| [3] | **PWRSWIEN** | **Power Switch Interrupt Enable**<br>0 = RTC Power Switch interrupt disable.<br>1 = RTC Power Switch interrupt enable. |
| [2] | **WAKEUPIEN** | **Wakeup Interrupt Enable**<br>0 = RTC Power Down wakeup interrupt disable.<br>1 = RTC Power Down wakeup interrupt enable. |
| [1] | **TICKIEN** | **Tick Interrupt Enable**<br>0 = RTC Time Tick Interrupt and counter disable.<br>1 = RTC Time Tick Interrupt and counter enable. |
| [0] | **ALMIEN** | **Alarm Interrupt Enable**<br>0 = RTC Alarm Interrupt disable.<br>1 = RTC Alarm Interrupt enable. |

## RTC Interrupt Status Register (RTC_INTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| RTC_INTSTS | RTC_BA+0x02C | R/W | RTC Interrupt Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| REGWRBUSY | 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 | | KEYPRESINT | RELALMINT | PWRSWINT | WAKEUPINT | TICKINT | ALMINT |

| Bits | Description | |
|---|---|---|
| [31] | **REGWRBUSY** | **Register Write Operation Busy**<br>0 = The new register write operation is acceptable.<br>1 = The last write operation is in progress and new register write operation prohibited. |
| [30:6] | **Reserved** | Reserved. |
| [5] | **KEYPRESINT** | **Key Pressed for 1 Second Interrupt Status**<br>This bit high indicates the SYS_nWAKEUP keep in low (user pressed the key) for 1 second when SYS_PWREN is high (stage machine is in Power_On state). User can write 1 to clear this bit.<br>0 = SYS_nWAKEUP pin didn't keep in low for 1 second.<br>1 = SYS_nWAKEUP pin keep in low for 1 second. |
| [4] | **RELALMINT** | **Relative Alarm Interrupt Status**<br>This bit high indicates the related timer have counted down to zero. User can write 1 to clear this bit.<br>0 = Related timer didn't count down to zero.<br>1 = Related timer counts down to zero. |
| [3] | **PWRSWINT** | **Power Switch Interrupt Status**<br>This bit high indicates the SYS_nWAKEUP transition from high to low (user pressed the key) when SYS_PWREN is high (stage machine is in Power_On state).<br>When state machine is not in Power_On state, SYS_nWAKEUP transition from high to low wouldn't set this bit high.<br>User can write 1 to clear this bit.<br>0 = Pin SYS_nWAKEUP didn't transit from high to low or pin SYS_PWREN is not high..<br>1 = Pin SYS_nWAKEUP transit from high to low when pin SYS_PWREN is high. |

| | | |
|---|---|---|
| [2] | **WAKEUPINT** | **Wakeup Interrupt Status**<br>This bit indicates the RTC generates a wakeup event to wakeup system from power down mode. In RTC, the wakeup source includes the RTC alarm and RTC related alarm. User can write 1 to clear this bit.<br>0 = System power control pin SYS_PWREN state didn't change.<br>1 = System power control pin SYS_PWREN state changed from low to high. |
| [1] | **TICKINT** | **RTC Time Tick Interrupt Indication REGISTER**<br>This bit indicates the RTC timer tick value configured in RTC_TICK has reached. User can write 1 to clear this bit.<br>0 = RTC timer tick value configured in RTC_TICK didn't reach.<br>1 = RTC timer tick value configured in RTC_TICK has reached. |
| [0] | **ALMINT** | **RTC Alarm Interrupt Indication REGISTER**<br>This bit indicates the RTC_TIME and RTC_CAL counter have counted to value configured in RTC_TALM and RTC_CALM. User can write 1 to clear this bit.<br>0 = It indicates that alarm interrupt has never occurred.<br>1 = It indicates that RTC_TIME and RTC_CAL counter have counted to a specified time configured in RTC_TALM and RTC_CALM. |

## RTC Time Tick Register (RTC_TICK)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| RTC_TICK | RTC_BA+0x030 | 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 | | | | | TTR | | |

| Bits | Description | |
|---|---|---|
| [2:0] | TTR | **RTC Tick Time Interrupt Request Interval**<br>The TTR [2:0] is used to select tick time interrupt request interval. The period of tick time interrupt is as follow:<br>000 = 1 second.<br>001 = 1/2 second.<br>010 = 1/4 second.<br>011 = 1/8 second.<br>100 = 1/16 second.<br>101 = 1/32 second.<br>110 = 1/64 second.<br>111 = 1/128 second. |

## RTC Power Control Register (RTC_PWRCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| RTC_PWRCTL | RTC_BA+0x034 | R/W | RTC Power Control Register | 0x0000_7000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | ALARM_MODE | RELALM_TIME | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RELALM_TIME | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PWROFF_TIME | | | | PWRON_TIME | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PWR_KEY | Reserved | EDGE_TRIG | REL_ALARM_EN | ALARM_EN | HW_PCLR_EN | SW_PCLR | PWR_ON |

| Bits | Description | |
|---|---|---|
| [31:29] | **Reserved** | Reserved. |
| [28] | **ALARM_MODE** | **Alarm Function Mode Selection**<br>This bit controls if pin SYS_PWREN be forced to low when RTC Time and Date counter or relative alarm counter meets the alarm condition.<br>0 =Pin SYS_PWREN be forced to low by alarm function Disabled.<br>1 = Pin SYS_PWREN be forced to low by alarm function Enabled. |
| [27:16] | **RELALM_TIME** | **Relative Alarm Time**<br>This field defines the relative alarm time period by unit second. The maximum value is 12'd1800. |
| [15:12] | **PWROFF_TIME** | **Power Off Time**<br>This field defines the time from SYS_nWAKEUP pressed to SYS_PWREN low. The time unit is second. And, the actual power-off time is the PWROFF_TIME+4. |
| [11:8] | **PWRON_TIME** | **Power on Time**<br>This field defines the period that SYS_nWAKEUP pressed to SYS_PWREN high. The time unit is second. |
| [7] | **PWR_KEY** | **Pin SYS_NWAKEUP Status**<br>This bit reflects the SYS_nWAKEUP pin status. |
| [6] | **Reserved** | Reserved. |
| [5] | **EDGE_TRIG** | **SYS_NWAKEUP Pin Trigger Mode**<br>0 = Level trigger mode. SYS_PWREN is high when SYS_nWAKEUP pressed longer than PWRON_TIME.<br>1 = Edge trigger mode. SYS_PWREN is high when SYS_nWAKEUP pressed longer than PWRON_TIME and then released. |

| | | |
|---|---|---|
| [4] | **REL_ALARM_EN** | **Relative Alarm Function Enable**<br>Set this bit high would enable the relative alarm function. When the relative alarm condition met, RTC would set RELALMINT (RTC_INTSTS[4]) interrupt status to high.<br>0 = Relative alarm function Disabled.<br>1 = Relative alarm function Enabled. |
| [3] | **ALARM_EN** | **Alarm Function Enable**<br>Set this bit high would enable the alarm function. When the alarm condition met, RTC would set ALMINT (RTC_INTSTS[0]) interrupt status to high.<br>0 = Alarm function Disabled.<br>1 = Alarm function Enabled. |
| [2] | **HW_PCLR_EN** | **Hardware Power Clear Enable**<br>0 = H/W wouldn't take any active on SYS_PWREN pin.<br>1 = H/W set SYS_PWREN pin to low automatically after PWROFF_TIME expired. |
| [1] | **SW_PCLR** | **Software Power Clear Enable**<br>When SYS_nWAKEUP is low (user pressed the key), S/W could write this bit high to force SYS_PWREN to be low directly.<br>When user released the key (SYS_nWAKEUP is high), RTC would clear this bit to low automatically.<br>0 = No operation.<br>1 = Force SYS_PWREN to low. |
| [0] | **PWR_ON** | **Software Power ON Control**<br>When SYS_nWAKEUP is low (user pressed the key), S/W could write this bit high to force SYS_PWREN to be high directly.<br>0 = No operation.<br>1 = To keep SYS_PWREN in high state when SYS_nWAKEUP released. |

## RTC Power Control Counter Register (RTC_PWRCNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_PWRCNT** | RTC_BA+0x038 | R | RTC Power Control Counter Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:21] | **Reserved** | Reserved. |
| [20:16] | **PWR_OFF_CNT** | **Power Off Counter Current Value**<br>This field shows the current value of power off counter.<br>This field is read only and write to this field wouldn't take any effect. |
| [15:12] | **PWR_ON_CNT** | **Power on Counter Current Value**<br>This field shows the current value of power on counter.<br>This field is read only and write to this field wouldn't take any effect. |
| [11:0] | **REL_ALARM _CNT** | **Relative Alarm Counter Current Value**<br>This field shows the current value of relative alarm counter.<br>This field is read only and write to this field wouldn't take any effect. |

## RTC Spare Register (RTC_SPRn, n = 0, 1, .., 15)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_SPR0** | RTC_BA+0x040 | R/W | RTC Spare Register 0 | 0x0000_0000 |
| **RTC_SPR1** | RTC_BA+0x044 | R/W | RTC Spare Register 1 | 0x0000_0000 |
| **RTC_SPR2** | RTC_BA+0x048 | R/W | RTC Spare Register 2 | 0x0000_0000 |
| **RTC_SPR3** | RTC_BA+0x04C | R/W | RTC Spare Register 3 | 0x0000_0000 |
| **RTC_SPR4** | RTC_BA+0x050 | R/W | RTC Spare Register 4 | 0x0000_0000 |
| **RTC_SPR5** | RTC_BA+0x054 | R/W | RTC Spare Register 5 | 0x0000_0000 |
| **RTC_SPR6** | RTC_BA+0x058 | R/W | RTC Spare Register 6 | 0x0000_0000 |
| **RTC_SPR7** | RTC_BA+0x05C | R/W | RTC Spare Register 7 | 0x0000_0000 |
| **RTC_SPR8** | RTC_BA+0x060 | R/W | RTC Spare Register 8 | 0x0000_0000 |
| **RTC_SPR9** | RTC_BA+0x064 | R/W | RTC Spare Register 9 | 0x0000_0000 |
| **RTC_SPR10** | RTC_BA+0x068 | R/W | RTC Spare Register 10 | 0x0000_0000 |
| **RTC_SPR11** | RTC_BA+0x06C | R/W | RTC Spare Register 11 | 0x0000_0000 |
| **RTC_SPR12** | RTC_BA+0x070 | R/W | RTC Spare Register 12 | 0x0000_0000 |
| **RTC_SPR13** | RTC_BA+0x074 | R/W | RTC Spare Register 13 | 0x0000_0000 |
| **RTC_SPR14** | RTC_BA+0x078 | R/W | RTC Spare Register 14 | 0x0000_0000 |
| **RTC_SPR15** | RTC_BA+0x07C | R/W | RTC Spare Register 15 | 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** | **Data**<br>This register is used to keep information written by user. |

## 5.16 UART Interface Controller (UART)

This chip equips up to eleven channels of Universal Asynchronous Receiver/Transmitters (UART). UART1/2/4/6/8/10 supports High-speed UART and UART0/3/5/7/9 perform Normal Speed UART, besides, all the UART channels support flow control function. The UART controller also supports IrDA (SIR), LIN Master/Slave and RS-485 function modes.

### 5.16.1 Overview

The Universal Asynchronous Receiver/Transmitter (UART) 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 channel supports seven types of interrupts including (1). transmitter FIFO empty interrupt (INT_THRE), (2). receiver threshold level reaching interrupt (INT_RDA), (3). line status interrupt (parity error or framing error or break interrupt) (INT_RLS), (4). receiver buffer time-out interrupt (INT_TOUT), (5). MODEM/Wake-up status interrupt (INT_MODEM), (6). Buffer error interrupt (INT_BUF_ERR), and (7). LIN interrupt (INT_LIN).

The UART1/ 2/ 4/ 6/ 8/ 10 is built-in with a 64-byte transmitter FIFO (TX_FIFO) and a 64-byte receiver FIFO (RX_FIFO) that reduces the number of interrupts presented to the CPU and the UART0/ 3/ 5/ 7/ 9 are equipped 16-byte transmitter FIFO (TX_FIFO) and 16-byte receiver FIFO (RX_FIFO). The CPU can read the status of the UART at any time during the operation. The reported status information includes the type and condition of the transfer operations being performed by the UART, as well as 4 error conditions (parity error, framing error, break interrupt and buffer error) probably occur while receiving data.

The UART includes a programmable baud rate generator that is capable of dividing clock input by divisors to produce the serial clock that transmitter and receiver need.

All of the controllers support auto-flow control function that uses two low-level signals, /CTS (clear-to-send) and /RTS (request-to-send), to control the flow of data transfer between the UART and external devices (ex: Modem). When auto-flow is enabled, the UART is not allowed to receive data until the UART asserts /RTS to external device. When the number of bytes in the RX FIFO equals the value of RTS_TRI_LEV (UA_FCR [19:16]), the /RTS is de-asserted. The UART sends data out when UART controller detects /CTS is asserted from external device. If a valid asserted /CTS is not detected the UART controller will not send data out.

The UART controllers also provides Serial IrDA (SIR, Serial Infrared) function (The IrDA mode is selected by setting the (FUN_SEL(UA_FUN_SEL[2:0]) = 010) to select 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 (half duplex). 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 10ms transfer delay between transmission and reception. This delay feature must be implemented by software.

The alternate function of UART controllers is LIN (Local Interconnect Network) function. The LIN mode is selected by setting the (FUN_SEL(UA_FUN_SEL[2:0]) = 001) to select LIN mode. In LIN mode, one start bit and 8-bit data format with 1-bit stop bit are required in accordance with the LIN standard.

For the NUC970 series, another alternate function of UART controllers is RS-485 9-bit mode function, and direction control provided by RTS pin to implement the function by software. The RS-485 mode is selected by setting the (FUN_SEL(UA_FUN_SEL[2:0]) = 011) to select RS-485 function. The RS-485 driver control is implemented using the RTS control signal from an asynchronous serial port to enable the RS-485 driver. In RS-485 mode, many characteristics of the RX and TX are the same as UART.

### 5.16.2 Features

- Full duplex, asynchronous communications
- Separate receive / transmit 64/16 bytes entry FIFO for data payloads
- Supports hardware auto flow control/flow control function (CTS, RTS) and programmable RTS flow control trigger level
- Programmable receiver buffer trigger level
- Supports programmable baud-rate generator for each channel individually
- Supports CTS 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(UA_TOR[15:8]) register
- Supports break error, frame error, parity error and receive / transmit buffer overflow detect 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 LIN function mode
- Supports RS-485 function mode

### 5.16.3 Block Diagram

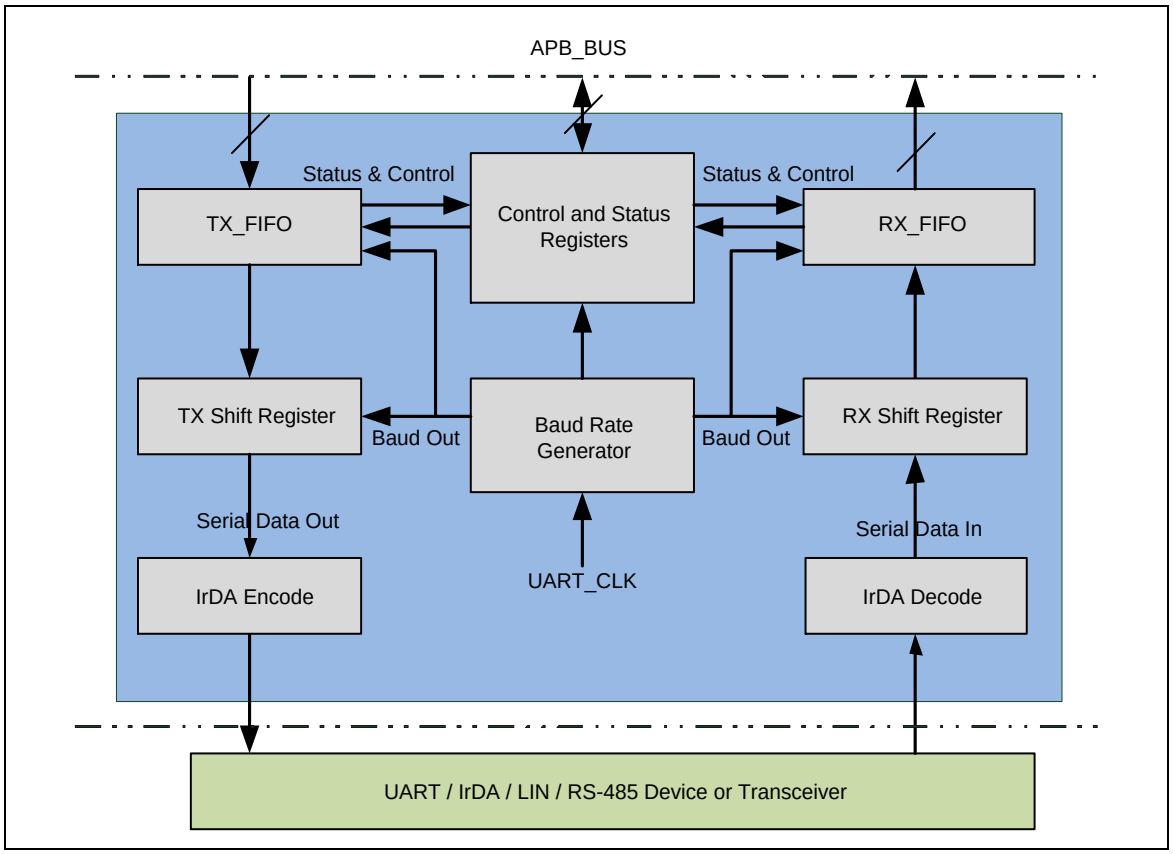



Figure 5.16-1 UART Block Diagram

TX_FIFO
The transmitter is buffered with a 64/16 byte FIFO to reduce the number of interrupts presented to the CPU.

RX_FIFO
The receiver is buffered with a 64/16 byte FIFO (plus three error bits per byte) to reduce the number of interrupts presented to the CPU.

TX shift Register
This block is the shifting the transmitting data out serially control block.

RX shift Register
This block is the shifting the receiving data in serially control block.

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 encode control block.

IrDA Decode
This block is IrDA decode control block.

Control and Status Register
This field is register set including the FIFO control registers (UA_FCR), FIFO status registers

(UA_FSR), and line control register (UA_LCR) for transmitter and receiver. The time-out control register (UA_TOR) identifies the condition of time-out interrupt. This register set also includes the interrupt enable register (UA_IER) and interrupt status register (UA_ISR) to enable or disable the responding interrupt and to identify the occurrence of the responding interrupt. There are seven types of interrupts, transmitter FIFO empty interrupt(INT_THRE), receiver threshold level reaching interrupt (INT_RDA), line status interrupt (parity error or framing error or break interrupt) (INT_RLS), time-out interrupt (INT_TOUT), MODEM/Wake-up status interrupt (INT_MODEM), Buffer error interrupt (INT_BUF_ERR) and LIN receiver break field detected interrupt (INT_LIN_RX_BREAK).

The following diagram demonstrates the auto-flow control block diagram. Refer to Auto-Flow Control section for detail description.

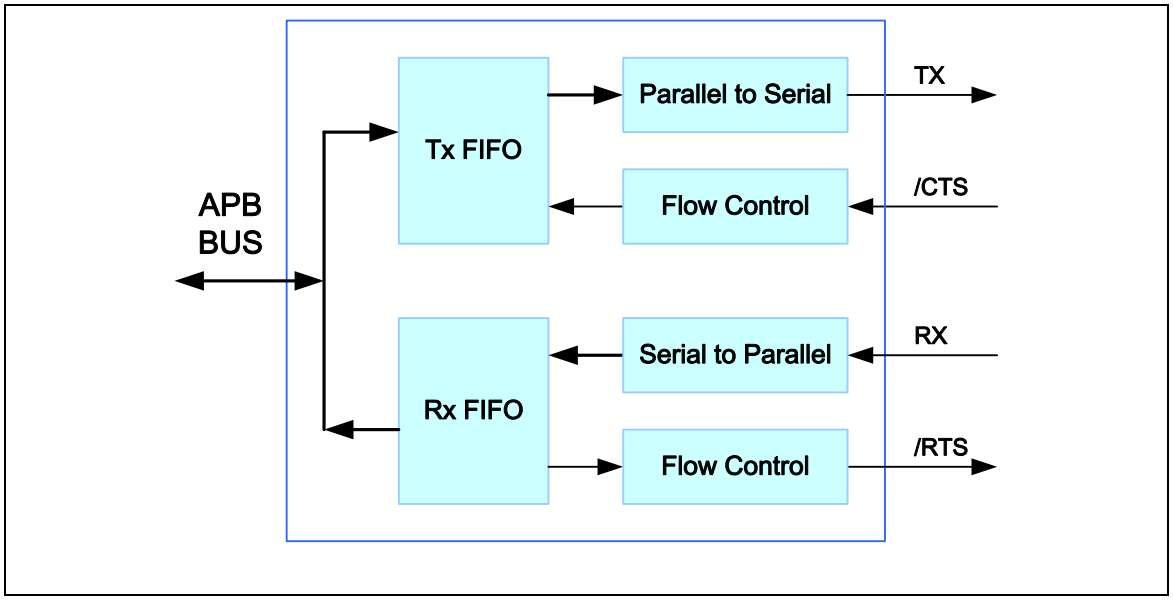



Figure 5.16-2 Auto Flow Control Block Diagram

### 5.16.4 Basic Configuration

Before using UART, it's necessary to configure related pins as the UART function and enable UART's clock.

For UART related pin configuration, please refer to the register SYS_MFP_GPBL, SYS_MFP_GPBH, SYS_MFP_GPCL, SYS_MFP_GPCH, SYS_MFP_GPDH, SYS_MFP_GPEL, SYS_MFP_GPEH, SYS_MFP_GPFH, SYS_MFP_GPGL, SYS_MFP_GPGH, SYS_MFP_GPHL, SYS_MFP_GPHH, SYS_MFP_GPIL and SYS_MFP_GPIH to know how to configure related pins as the UART function.

UARTs' clock controlled by CLK_PCLKEN0[26:16]. For example, set UART6 (CLK_PCLKEN0[22]) high to enable clock for UART6's operation.

### 5.16.5 Functional Description

#### 5.16.5.1 *Line Function Description*

The UART Controller supports fully programmable serial-interface characteristics by setting the UA_LCR register. Software can use the UA_LCR register to program the word length, stop bit and

parity bit. The following tables list the UART word and stop bit length settings and the UART parity bit settings.

| NSB (UA_LCR[2]) | WLS (UA_LCR[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 |

Figure 5.16-3 UART Line Control of Word and Stop Length Setting

| Parity Type | SPE (UA_LCR[5]) | EPE (UA_LCR[4]) | PBE (UA_LCR[3]) | Description |
|---|---|---|---|---|
| No Parity | x | x | 0 | No parity bit output. |
| Odd Parity | 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 | 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 Parity | 1 | 0 | 1 | Parity bit always logic 1.<br>Parity bit on the serial byte is set to “1” regardless of total number of “1’s” (even or odd counts). |
| Forced Space Parity | 1 | 1 | 1 | Parity bit always logic 0.<br>Parity bit on the serial byte is set to “0” regardless of total number of “1’s” (even or odd counts). |

Figure 5.16-4 UART Line Control of Parity Bit Setting

**Note:** User cannot change line controller setting when TE_FLAG(UA_FSR[28]) is not empty

5.16.5.2 *Baud Rate Generation*

The UART Controller includes a programmable baud rate generator capable of dividing clock input by dividers to produce the serial clock that transmitter and receiver need.

The baud rate equation is:

Baud Rate = UART_CLK / M * [BRD + 2]

where M and BRD are defined in Baud Rate Divider Register (UA_BAUD).

The following tables list the UART baud rate equations in the various conditions and UART baud rate parameter settings. There is no error for the baud rate results calculated through the baud rate parameter and register setting below. In IrDA function mode, the baud rate generator must be set in Mode 0.

| Mode | DIV_X_EN | DIV_X_ONE | DIVIDER X | BRD | Baud Rate Equation |
|---|---|---|---|---|---|
| 0 | Disable | 0 | Don't Care | A | UART_CLK / [16 * (A+2)] |
| 1 | Enable | 0 | B | A | UART_CLK / [(B+1) * (A+2)] , B must >= 8 |
| 2 | Enable | 1 | Don't care | A | UART_CLK / (A+2), A must >=9 |

Table 5.16-1 Baud Rate Equation Table

| System Clock = XT1_IN (12 MHz) | | | | | | |
|---|---|---|---|---|---|---|
| Baud Rate | Mode0 | | Mode1 | | Mode2 | |
| | Parameter | Register | Parameter | Register | Parameter | Register |
| 921600 | x | x | A=0,B=11 | 0x2B00_0000 | A=22 | 0x3000_0016 |
| 460800 | A=1 | 0x0000_0001 | A=1,B=15<br>A=2,B=11 | 0x2F00_0001<br>0x2B00_0002 | A=46 | 0x3000_002E |
| 230400 | A=4 | 0x0000_0004 | A=4,B=15<br>A=6,B=11 | 0x2F00_0004<br>0x2B00_0006 | A=94 | 0x3000_005E |
| 115200 | A=10 | 0x0000_000A | A=10,B=15<br>A=14,B=11 | 0x2F00_000A<br>0x2B00_000E | A=190 | 0x3000_00BE |
| 57600 | A=22 | 0x0000_0016 | A=22,B=15<br>A=30,B=11 | 0x2F00_0016<br>0x2B00_001E | A=382 | 0x3000_017E |
| 38400 | A=34 | 0x0000_0022 | A=62,B=8<br>A=46,B=11<br>A=34,B=15 | 0x2800_003E<br>0x2B00_002E<br>0x2F00_0022 | A=574 | 0x3000_023E |
| 19200 | A=70 | 0x0000_0046 | A=126,B=8<br>A=94,B=11<br>A=70,B=15 | 0x2800_007E<br>0x2B00_005E<br>0x2F00_0046 | A=1150 | 0x3000_047E |
| 9600 | A=142 | 0x0000_008E | A=254,B=8<br>A=190,B=11<br>A=142,B=15 | 0x2800_00FE<br>0x2B00_00BE<br>0x2F00_008E | A=2302 | 0x3000_08FE |
| 4800 | A=286 | 0x0000_011E | A=510,B=8<br>A=382,B=11<br>A=286,B=15 | 0x2800_01FE<br>0x2B00_017E<br>0x2F00_011E | A=4606 | 0x3000_11FE |

Table 5.16-2 Baud Rate Equation Table

### 5.16.5.3 *Controller FIFO Control and Status*

The UART controller is built-in with a 64/16-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 the 6 types of interrupts and condition of the transfer operations being performed by the UART, as well as 3 error conditions (parity error, framing error, break interrupt) probably occur while receiving data. This FIFO control and status also support all of UART, IrDA, and RS-485 function mode.

### 5.16.5.4 *Auto-Flow Control*

The UART controllers support auto-flow control function that uses two low-level signals, CTSn (clear-to-send) and RTSn (request-to-send) to control the flow of data transfer between the UART and external devices (ex: Modem). When auto-flow is enabled, the UART is not allowed to receive data until the UART asserts RTSn (RTSn high) to external device. When the number of bytes in the RX-FIFO equals the value of RTS_TRI_LEV (UART_TLCTL[13:12]), the RTSn is de-asserted. The UART sends data out when UART controller detects CTSn is asserted (CTSn high) from external device. If a valid asserted CTSn is not detected the UART controller will not send data out. The auto-flow function is implemented in High-speed UART only.

The following diagram demonstrates the auto-flow control block diagram.

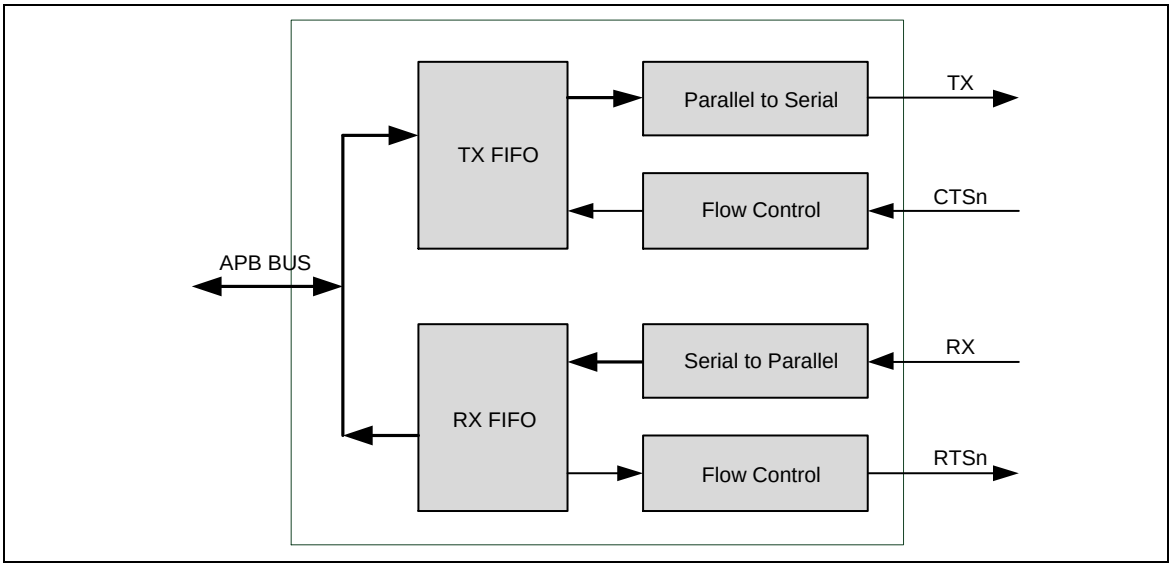



Figure 5.16-5 UART Auto-Flow Control Block Diagram

### 5.16.5.5 *Wake-Up Function*

The UART Controller supports wake-up system function. The wake-up source includes CTSn, RIn, DCD and DSR signals. All High-speed UART support CTSn signal, but only UART1 supports all full modem wake-up ports.

In power condition, when the WAKE_CTS_EN(UA_CTL[8]) is set and the toggle of CTSn pin can wake-up the system.

**Case 1:**

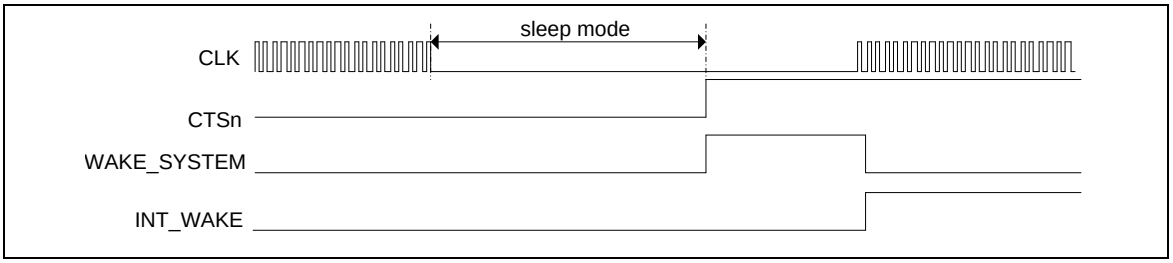



Figure 5.16-6 UART CTSn Wake-Up Case 1

**Case 2:**

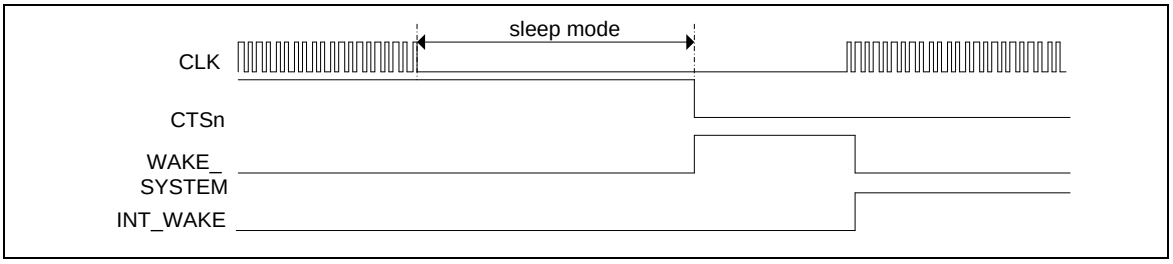



Figure 5.16-7 UART CTSn Wake-Up Case 2

5.16.5.6 *IrDA Mode*

The UART Controller provides Serial IrDA (SIR, Serial Infrared) transmit encoder and receive decoder function. The IrDA_EN(UART_FUN_SEL[2:0] = 010) bit are used to select 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 (half duplex). 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.

In IrDA Operation mode, the receive FIFO trigger level must be “1” by setting RFITL(UA_FCR[7:4]) = 0000 and the DIV_X_EN(UA_BAUD[29]) bit must be disabled in IrDA mode operation (Mode 1).

Baud Rate = Clock / (16 * BRD ), where BRD is Baud Rate Divider in BRD(UA_BAUD[15:0]).

The following diagram demonstrates the IrDA control block diagram.

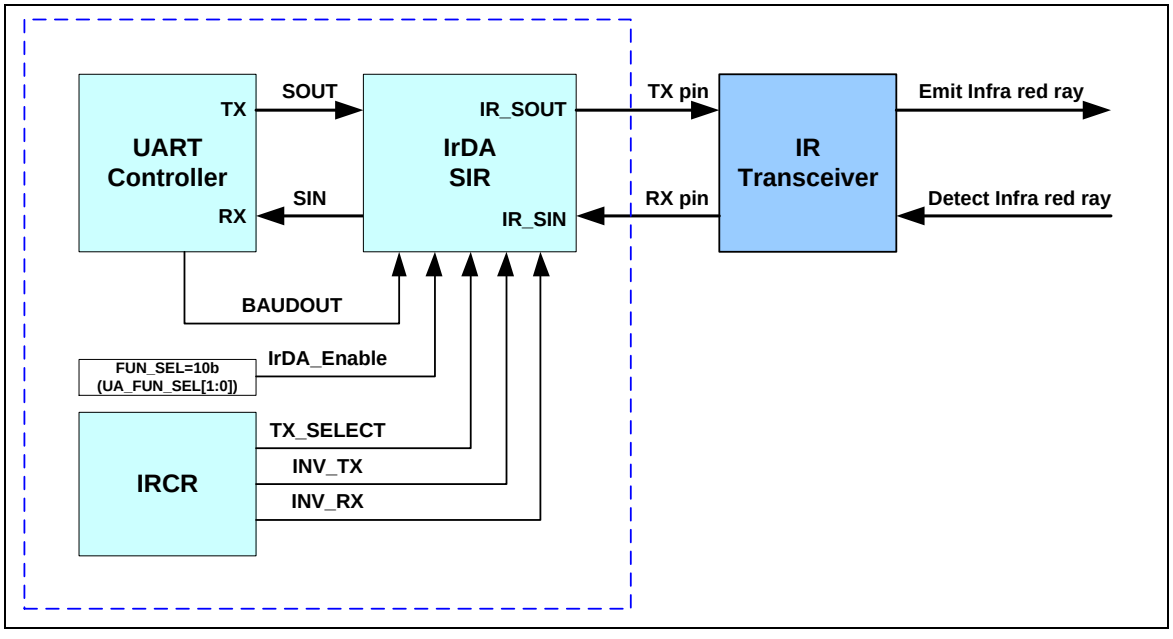



Figure 5.16-8 IrDA Block Diagram

### IrDA SIR Transmit Encoder

The IrDA SIR Transmit Encoder modulate Non-Return-to Zero (NRZ) transmit bit stream output from UART. The IrDA SIR physical layer specifies use of Return-to-Zero, Inverted (RZI) modulation scheme which represent 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. The decoder input is normally high in the idle state. (Because of this, INV_RX(UA_IRCR[6]) should be set as 1 by default)

A start bit is detected when the decoder input is LOW

**IrDA SIR Operation**

The IrDA SIR Encoder/Decoder provides functionality which converts between UART data stream and half duplex serial SIR interface. Refer to UA_IRCR register for detail description. The following diagram shows the IrDA encoder/decoder waveform.

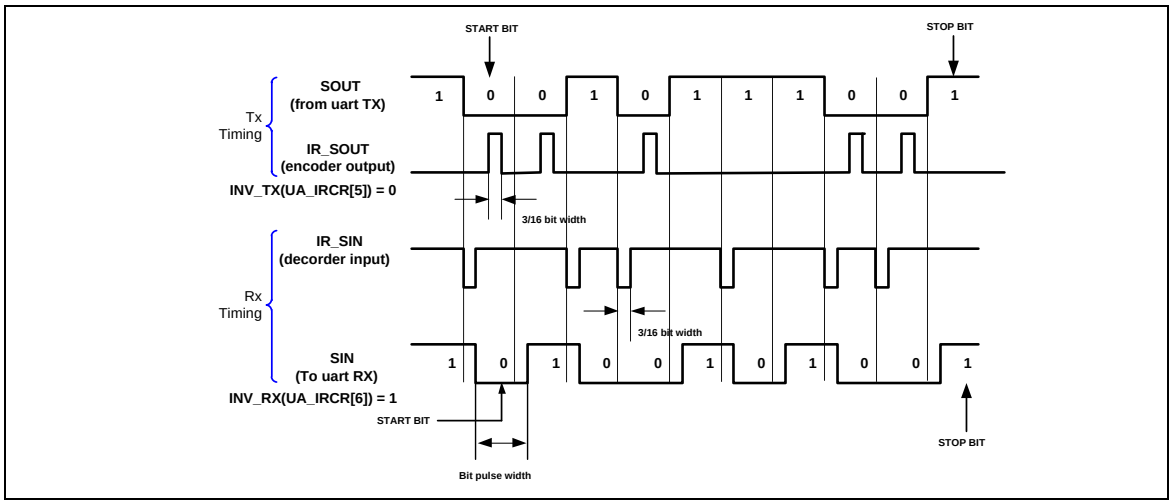



Figure 5.16-9 IrDA TX/RX Timing Diagram

5.16.5.7 *RS-485 Function Mode*

The UART supports RS-485 9-bit mode function. The RS-485 mode is selected by setting the FUN_SEL(UA_FUN_SEL[2:0]) to select RS-485 function. The RS-485 driver control is implemented using the RTS control signal from an asynchronous serial port to enable the RS-485 driver. In RS-485 mode, many characteristics of the RX and TX are same as UART.

In RS-485 mode, the bit 9 will be configured as address bit. The controller can configuration of it as an RS-485 addressable slave and the RS-485 master transmitter will identify an address character by setting the parity (9th bit) to 1.

For data characters, the bit 9 is set to “0”. Software can use UA_LCR register to control the 9-th bit (When the PBE(UA_LCR[3]), EPE(UA_LCR[4]) and SPE(UA_LCR[5]) 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 support three operation mode that is RS-485 Normal Multi-drop 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 UA_ALT_CSR register, and software can driving the transfer delay time between the last stop bit leaving the TX-FIFO and the de-assertion of by setting DLY(UA_TOR [15:8]).

Another alternate function of UART controllers is RS-485 9 bit mode function whose direction control can be controlled by RTSn pin or GPIO. The RS-485 function mode is selected by setting the FUN_SEL(UA_FUN_SEL[2:0]) to select RS-485 function. The RS-485 driver control is implemented by using the RTSn control signal from an asynchronous serial port to enable the RS-485 driver. In RS-485 mode, many characteristics of the RX and TX are same as UART.

**RS-485 Normal Multidrop Operation Mode (NMM)**

In RS-485 Normal Multi-drop operation mode, software must decide whether receiver will ignore data before an address byte is detected (bit 9 = “1”).

If software wants to receive any data before address byte detected, the flow is disable RX_DIS(UA_FCR [8]) then enable RS485_NMM(UA_ALT_CSR[8]) and the receiver will received any data. If an address byte is detected (bit9 =1), it will generator an interrupt to CPU and software can decide whether enable or disable receiver to accept the following data byte by setting RX_DIS.

When an address byte be detected (bit 9 = “1”) by hardware, the address byte data will be stored in the RX-FIFO. If the receiver is be enabled (RX_DIS(UA_FCR[8]) is low, all received byte data will be accepted and stored in the RX-FIFO, and if the receiver is disabled (RX_DIS(UA_FCR[8]) is high, all received byte data will be ignore until the next address byte be detected.

If software disable receiver by setting (RX_DIS(UA_FCR[8]) bit, when a next address byte be detected, the controller will clear the RX_DIS bit and the address byte data will be stored in the RX-FIFO.

**Program Sequence Example:**

1. Program FUN_SEL(UA_FUN_SEL[2:0]) to select RS-485 function.

2. Program the RX_DIS(UA_FCR[8]) bit to determine whether to store the received data before an address byte is detected (bit 9 = “1”).

3. Program the RS485_NMM by setting RS485_NMM(UA_ALT_CSR[8]).

4. When an address byte is detected (bit 9 = “1”), hardware will set RLS_IS(UA_ISR[2]) and RS485_ADD_DETF(UA_FSR[3]) flag.

5. Software can decide whether to accept the following data byte by setting RX_DIS(UA_FCR[8]).

6. Repeat step 4 and step 5.

**RS-485 Auto Address Detection Operation Mode (AAD)**

In RS-485 Auto Address Detection Operation Mode, the receiver will ignore any data until an address byte is detected (bit9 =1) and the address byte data match the ADDR_MATCH(UA_ALT_CSR[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 and address byte data not match the ADDR_MATCH(UA_ALT_CSR[31:24]) value. In RS-485 AAD mode, don't fill any value to RX_DIS(UA_CTL[2]) bit.

**Program Sequence example:**

1. Program FUN_SEL(UART_FUN_SEL[1:0]) to select RS-485 function.

2. Program the RS485_AAD(UA_ALT_CSR[9]).

3. When an address byte is detected (bit9 = "1"), hardware will compare the address byte and the ADDR_MATCH (UA_ALT_CSR[31:24]) value.

4. If the address byte matches the ADDR_MATCH(UA_ALT_CSR[31:24]) value, hardware will set RLS_IS(UART_ISR[2]) and RS485_ADD_DETF(UA_FSR[3]). And the receiver will sorted address byte to FIFO and accept the following data transfer and stored data in FIFO until next address byte be detected.

However if the address byte does not match the ADDR_MATCH(UA_ALT_CSR[31:24]) value, hardware will ignored the address byte data and ignored the following data transfer.

5. Respect step 3 and step 4.

**RS-485 Auto Direction Mode (AUD)**

Another option function of RS-485 controllers is RS-485 auto direction control function. The RS-485 driver control is implemented using the RTS control signal from an asynchronous serial port to enable the RS-485 driver. The RTS line is connected to the RS-485 driver enable such that setting the RTS line to high (logic 1) enables the RS-485 driver. Setting the RTS line to low (logic 0) puts the driver into the tri-state condition. User can setting LEV_RTS(UA_MCR[9]) to change the RTS driving level.

**Program Sequence example:**

Program FUN_SEL(UA_FUN_SEL[2:0]) to select RS-485 function.

Program the RX_DIS(UA_FCR[8]) to determine enable or disable RS-485 receiver

Program the RS485_NMM(UA_ALT_CSR[6]) or RS485_AAD(UA_ALT_CSR[9]) mode.

If the RS-485_AAD mode is selected, the ADDR_MATCH is programmed for auto address match value.

Determine auto direction control by programming RS-485_AUD.

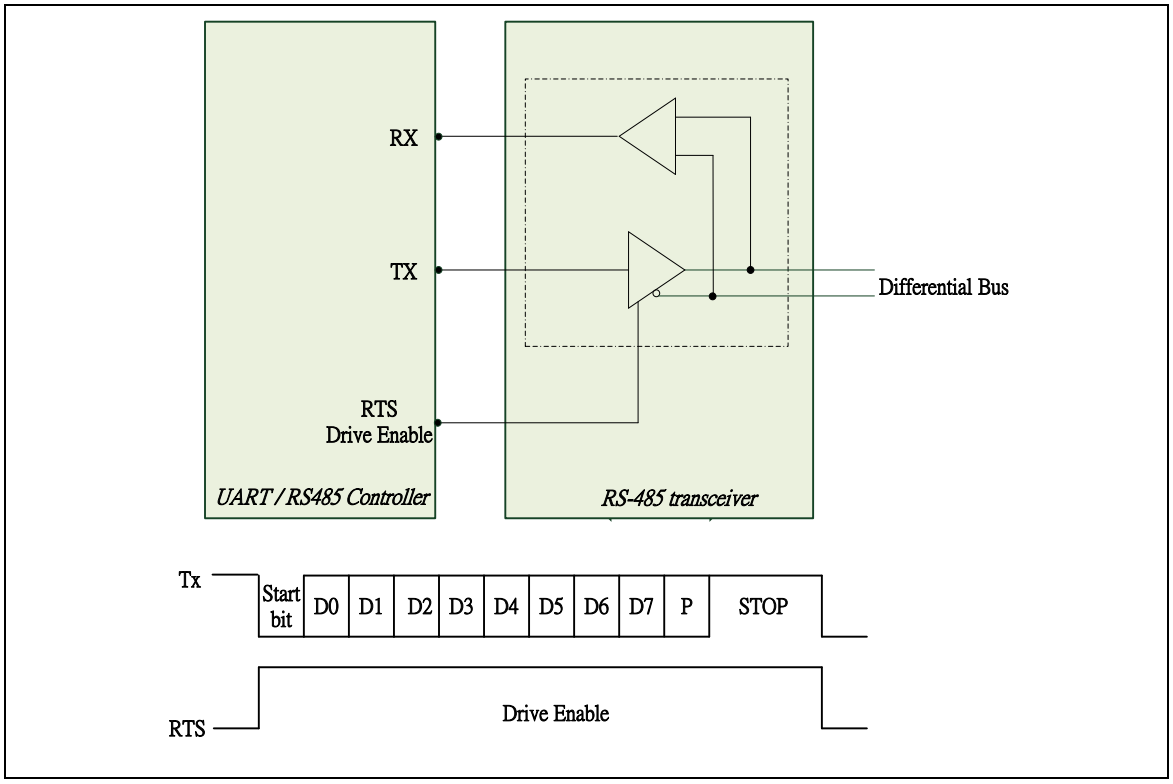



Figure 5.16-10 RS-485 Frame Structure

5.16.5.8 *LIN (Local Interconnection Network) Mode*

The UART supports LIN function. The LIN mode is selected by setting the (FUN_SEL(UA_FUN_SEL[2:0]) = 001). The UART support LIN break/delimiter generation and break/delimiter detection in LIN master mode, support header detection and automatic resynchronization in LIN slave mode.

## Structure of LIN Frame

According to the LIN protocol, all information is transmitted packed as frames; a frame consist (provided by the master task) a header and a response (provided by a slave task). That is any communication on the LIN bus is started by the master sending a header, followed by the response. The header (provided by the master task) consists of a break field and sync field followed by a frame identifier (frame ID). The frame identifier uniquely defines the purpose of the frame. The slave task appointed for providing the response associated with the frame ID and the response consists of a data field and a checksum field. The following diagram is the structure of LIN function mode.

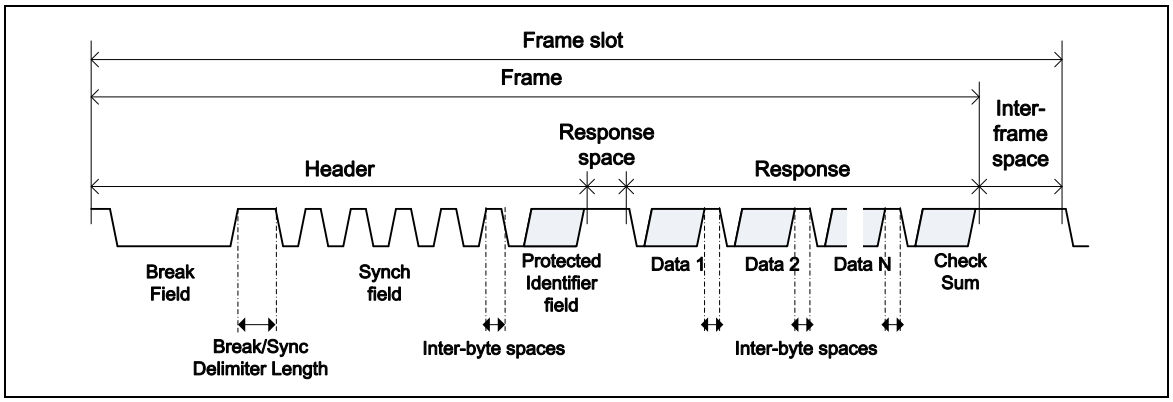



Figure 5.16-11 LIN Frame Structure

**LIN Master Mode**

The UART controller supports LIN master mode. In LIN mode, each byte field is initialed by a start bit with value zero (dominant), followed by 8 data bits (WLS(UA_LCR[1:0]) = 11) and no parity bit, LSB is first and ended by 1 stop bit (NSB(UA_LCR[2]) = 1) with value one (recessive) in accordance with the LIN standard. In LIN master mode, software may need some initial process, and the initialization process flow of LIN master is shown as follows:

1. Select the desired baud-rate by setting the UA_BAUD register.

2. Select LIN function mode by setting UA_FUN_SEL register.

3. Configure the data length to 8 bits by setting (WLS(UA_LCR[1:0]) = 11) and disable parity check by clearing PBE(UA_LCR[3]) bit and configure the stop bit to 1 by clearing NSB(UA_LCR[2]) bit in UA_LCR register.

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 mode. 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 LIN_HEAD_SEL(UA_LIN_CTL[23:22]).

(1). If the header selected is “break field”, software must handle the following sequence to sending a complete header to bus by filled sync data (0x55) and frame ID data to UA_THR register.

(2). If the header selected is “break field and sync field”, software must handle the sequence to sending a complete header to bus by filled frame ID data to UA_THR register.

(3). If the header selected 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 LIN_PID(UA_LIN_CTL[31:25]).

When operating in header selected is “break field, sync field and frame ID field” mode, the frame ID parity bit can be calculated by software or hardware depending on the LIN_IDPEN(UA_LIN_CTL[9]) bit setting.

When operating in LIN data transmission, software can monitor the LIN bus transfer state by hardware or software. User can enable hardware monitoring by setting BIT_ERR_EN (UA_LIN_CTL[12]), and when operating in LIN transmitter state, if the input pin (SIN) state is not equal to the output pin (SOUT) state that the hardware wills generator an interrupt to CPU. User also can monitor the LIN bus transfer state by check the read back data in UA_RBR register. The following sequence is a program sequence example:

Procedure without software error monitoring in master mode:

1. Choose the hardware transmission header field by setting LIN_HEAD_SEL(UA_LIN_CTL[23:22]]).

2. Request header transmission by setting the LIN_SHD(UA_LIN_CTL[8]) bit.

3. Wait for the TE_FLAG(UA_FSR[28]) flag.

**Note1:** The break + delimiter default setting is 13 dominant bits and 1 delimiter bit, software can change it by setting LIN_BKFL(UA_LIN_CTL[19:16]) and LIN_BS_LEN(UA_LIN_CTL[21:20]), to change the dominant bits.

**Note2:** The break/sync delimiter length default setting is 1 bit time and the inter-byte space is 1 bit time, software can change it by setting LIN_BS_LEN(UA_LIN_CTL[21:20]) and DLY(UA_TOR[15:8]).

**Note3:** If the header includes “break field, sync field and frame ID field”, software must fill frame ID in LIN_PID(UA_LIN_CTL[31:24]) before trigger header transmission (setting the LIN_SHD(UA_LIN_CTL[8]) bit. The frame ID parity can be generated by software or hardware depends on LIN_IDPEN(UA_LIN_CTL[9]). If the parity generated by software (LIN_IDPEN(UA_LIN_CTL[9]) = 0), software must fill 8 bit data (include 2 bit parity) in this field, and if the parity generated by hardware (LIN_IDPEN(UA_LIN_CTL[9]) = 1), software fill ID0~ID5, hardware will calculi P0 and P1.

Procedure with software error monitoring in Master mode:

1. Choose if the hardware transmission header field only includes “break field” by setting LIN_HEAD_SEL(UA_LIN_CTL[23:22] = 00).

2. Enable break detection function by setting LIN_BKDET_EN(UA_LIN_CTL[10]) bit.

3. Request break + delimiter transmission by setting the LIN_SHD(UA_LIN_CTL[8]) bit.

4. Wait for the LIN_BKDET_F(UA_LIN_SR[8]) flag.

5. Request sync field transmission by writing 0x55 into UA_THR register.

6. Wait for the RDA_IF(UA_ISR[0]) flag and read back the UART_RBR register.

7. Request header frame ID transmission by writing the protected identifier value in the UA_THR register.

8. Wait for the RDA_IF(UA_ISR[0]) flag and read back the UA_RBR register.

**LIN break and delimiter detection**

When software enable the break detection function by setting LIN_BKDET_EN(UA_LIN_CTL[10]), the break detection circuit is activated.

When the break detection function is enabled, the circuit looks at the input SIN pin for a start signal. If more than 11 bits are detected as 0, and are followed by a delimiter character, the LIN_BKDET_F (UA_LIN_SR[8]) flag is set at the end of break field. If the LIN_RX_BRK_IEN(UA_IER [8]) = 1, an interrupt will be generated. The behavior of the break detection and break flag is shown in the following figure.

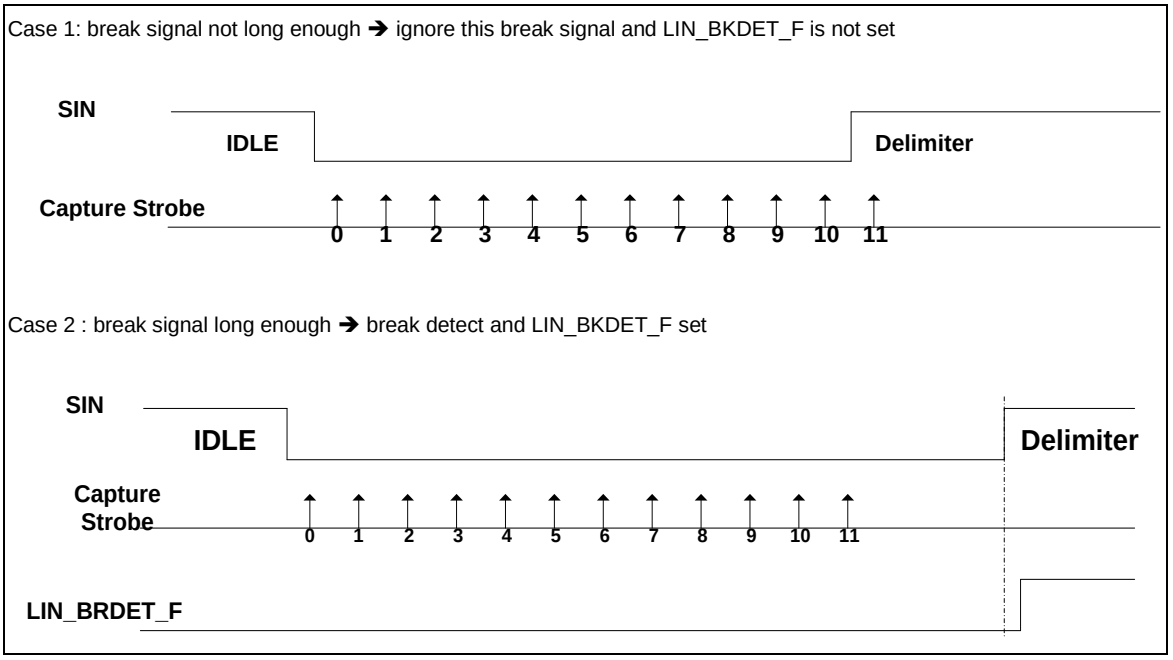



Figure 5.16-12 Break Detection in LIN Mode

If the UART controller detects the “break field”, the received data will not be stored in UA_THR and the frame error will not set when the controller send the “break field” at this time. However if the break detection circuit detect “break field” but the controller doesn’t send the “break field”, the received data will be stored in UA_THR and the frame error will be set. The behavior of the break detection and frame error flag is shown in following figure.

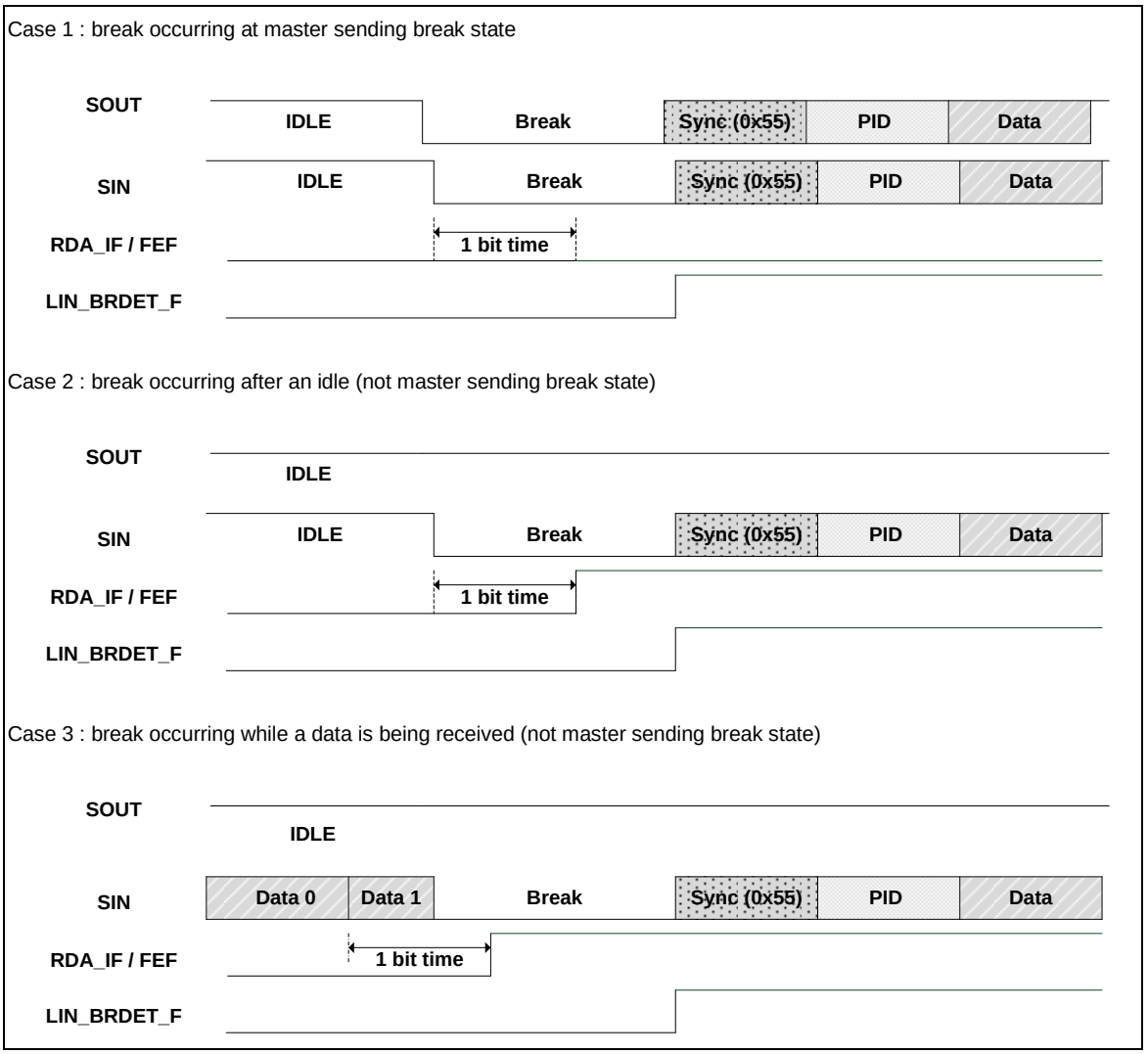



Figure 5.16-13 Relationship between Break Detection and Frame Error Detection

**LIN response transmission**

The LIN master can transmitter response (master is the publisher of the response) and receive response (master is the subscriber of the response). When the master is the publisher of the response, the master send response by writing UA_THR register, and if the master is the subscriber of the response, the master will received n data bytes by other slave node (the maximum of n is 8).

### LIN Slave Mode

The UART controller support LIN slave mode by setting the LINS_EN(UA_LIN_CTL[0]) bit. In LIN mode, each byte field is initialed by a start bit with value zero (dominant), followed by 8 data bits (WLS(UA_LCR[1:0]) = 11) and no parity bit, LSB is first and ended by 1 stop bit (NSB(UA_LCR[2]) = 1) with value one (recessive) in accordance with the LIN standard. In LIN slave mode, software may need some initial process, and the initialization process flow of LIN slave mode is shown as follows:

1. Select the desired baud-rate by setting the UA_BAUD register.

2. Select LIN function mode by setting FUN_SEL(UA_FUN_SEL[2:0]) bits.

3. Configure the data length to 8 bits by setting WLS(UA_LCR[1:0] = 11) and disable parity check by clearing PBE(UA_LCR[3]) bit and configure the stop bit to 1 by clearing NSB(UA_LCR[2]) bit.

4. Enable LIN slave mode by setting the LINS_EN(UA_LIN_CTL[0]) bit.

**LIN header reception**

According to the LIN protocol, a slave node must wait for a valid header which coming from the master node. Then application has to take following action (depending 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 slave header detection function by setting LINS_HDET_EN (UA_LIN_CTL[1]) to detect complete frame header (receive “break field”, “sync field” and “frame ID field”). When a LIN header is received, the LINS_HDET_F(UA_LIN_SR[0]) flag will be set (If the LIN_RX_BRK_IEN(UA_IER [8]) =1), an interrupt will be generated).

User can enable frame ID parity check function by setting LIN_IDPEN (UA_LIN_CTL[9]) bit, and when the frame header be received and received frame ID parity is not correct, the LIN_IDPERR_F (UA_LIN_SR[1]) bit (If the LIN_RX_BRK_IEN(UA_IER[8]) =1, an interrupt will be generated) will be set together with the LINS_HDET_F(UA_LIN_SR[0]) flag. User also can put LIN in mute mode by setting LIN_MUTE_EN (UA_LIN_CTL[4]) bit. 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 support automatic resynchronization function to avoid clock deviation error, user can enable this feature by setting LINS_ARS_EN(UA_LIN_CTL[2]) bit.

**LIN response transmission**

The LIN slave can transmitter response (slave is the publisher of the response) and receive response (slave is the subscriber of the response). When the slave is the publisher of the response, the slave send response by writing UA_THR register, and if the slave is the subscriber of the response, the slave received n data bytes by other slave node (the maximum of n is 8).

**Note:** When LIN data transmission is active, software can monitor the LIN bus transfer state by hardware or software. User can enable hardware monitoring by setting BIT_ERR_EN(UA_LIN_CTL[12]) bit, and when operating in LIN transmitter state, if the input pin (SIN) state is not equal to the output pin (SOUT) state that the hardware wills generator an interrupt to CPU. User also can monitor the LIN bus transfer state by check the read back data in UA_RBR register. The following sequence is a program sequence example:

**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 LIN_HERR_F (UA_LIN_SR [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 assert.

Software write 1 to LINS_SYNC_F(UA_LIN_SR[3]) bit to re-search new frame header.

**Mute mode and LIN wake-up**

In mute mode, it allows detection of headers only and prevents the reception of any other characters. User can enable mute mode by setting LIN_MUTE_EN(UA_LIN_CTL[4]) bit and wake-up condition can be selected by LIN_HEAD_SEL(UA_LIN_CTL[23:22]).

**Note:** It is recommended to put LIN in mute mode by setting LIN_MUTE_EN(UA_LIN_CTL[4]) bit after checksum transmission.

The LIN controller leaves mute mode in the following conditions.

While LIN_HEAD_SEL(UA_LIN_CTL[23:22]) is “break field”, when detect a valid LIN break + delimiter, the controller will enable the receiver (left mute mode) and the following data (sync data and frame ID data) will be stored in RX-FIFO.

While LIN_HEAD_SEL(UA_LIN_CTL[23:22]) is “break field and sync field”, when detect a valid LIN break + delimiter and valid sync field without frame error, the controller will enable the receiver (left mute mode) and the following data (ID data) will be stored in RX-FIFO.

While LIN_HEAD_SEL(UA_LIN_CTL[23:22]) is “break field, sync field and ID field”, when detect a valid LIN break + delimiter and valid sync field without frame error and the following ID data (without frame error) match LIN_PID(UA_LIN_CTL[31:24]) , the controller will enable the receiver (left mute mode) and the following data will be stored in RX-FIFO.

**Slave mode without automatic resynchronization**

User can disable automatic resynchronization function to fix the communication baud rate. When operating in without automatic resynchronization mode, software needs some initial process, and the initialization process flow of without automatic resynchronization mode is shown as follows:

1. Select the desired baud-rate by setting the UA_BAUD register.
2. Select LIN function mode by setting FUN_SEL(UA_FUN_SEL[2:0]) = 001.
3. Enable LIN slave mode by setting the LINS_EN(UA_LIN_CTL[0]) bit.

**Slave mode with automatic resynchronization**

User can enable automatic resynchronization function by setting LINS_ARS_EN(UA_LIN_CTL[2]) bit. In automatic resynchronization mode, the controller will adjust the baud rate generator after each sync field reception. The other program sequence is similar to Slave mode without automatic resynchronization section.

When automatic resynchronization function is enabled, after each LIN break field, the time duration between five falling edges is sampled on engine clock and the result of this measurement is stored in an internal 13-bit register and the UA_BAUD register value will automatically updated at the end of the fifth falling edge. If the measure timer (13bit) overflow before five falling edges, then the header error flag LIN_HERR_F(UA_LIN_SR[1]) will be set.

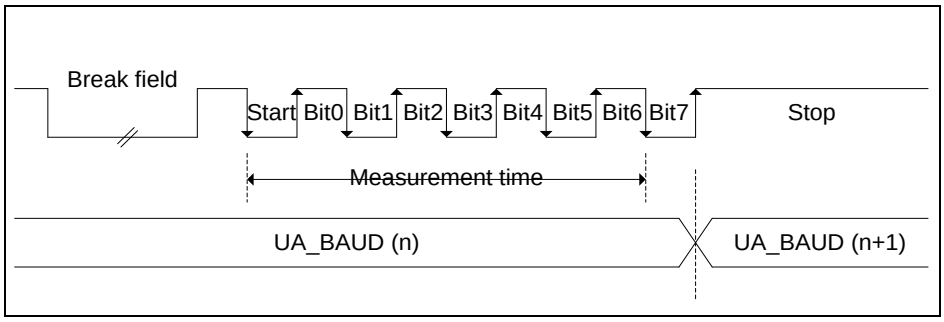



Figure 5.16-14 LIN Sync Field Measurement

When operating in automatic resynchronization mode, software must select the desired baud rate by setting the UA_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 engine clock and the result of this measurement is stored in an internal 13-bit register (BAUD_LIN) and the result will be updated to UA_BAUD register automatically.

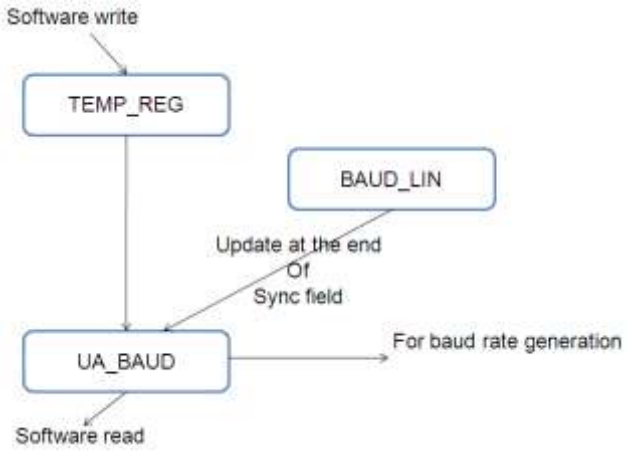
In order to guarantee the transmission baud rate, the baud rate generator must reload the initial value before each new break reception. The initial value is programmed by the application during initialization (TEMP_REG). User can setting LINS_DUM_EN(UA_LIN_CTL[3]) bit to enable auto reload initial baud rate value function. If the LINS_DUM_EN(UA_LIN_CTL[3]) is set, when received the next character, hardware will auto reload the initial value to UA_BAUD, and when the UA_BAUD be updated, the LINS_DUM_EN(UA_LIN_CTL[3]) bit will be cleared automatically. The behavior of LIN updated method as shown in the following figure.

**Note1:** It is recommended to set the LINS_DUM_EN(UA_LIN_CTL[3]) bit before every checksum reception.

**Note2:** When header error been detected, user must writing 1 to LINS_SYNC_F(UA_LIN_SR[3]) bit (UA_LIN_SR[3]) to re-search new frame header. When user 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 mode2 (DIV_X_EN(UA_BAUD[29]) and DIV_X_ONE(UA_BAUD[28]) must be 1).

Case1: UA_BAUD read/write operation when LINS_DUM_EN(UA_LIN_CTL[3]) = 0




Case2: UA_BAUD read/write operation when LINS_DUM_EN(UA_LIN_CTL[3]) = 1

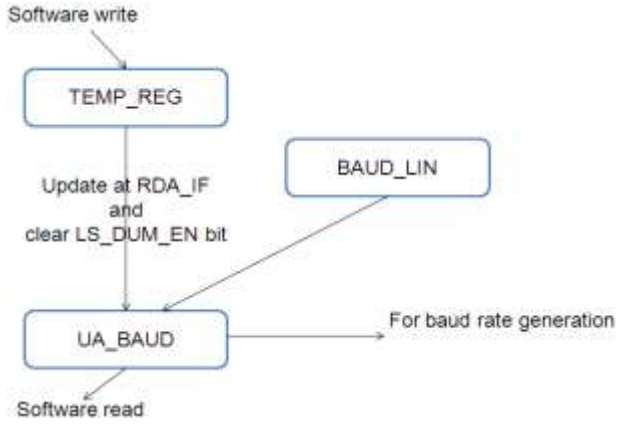



Figure 5.16-15 UA_BAUD Update Method

**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 great than 15%, the header error flag LINS_HERR_F(UA_LIN_SR[1]) will be set.

If the difference between 15% and 14%, the header error flag (LINS_HERR_F(UA_LIN_SR[1])) may either set or not (depending on the data dephasing).

Check2: Based on measurement of time between each falling edge of the sync field.

If the difference great than 19%, the header error flag (LINS_HERR_F(UA_LIN_SR[1])) will be set.

If the difference between 19% and 15%, the header error flag (LINS_HERR_F(UA_LIN_SR[1])) may either set or not (depending on the drift of the incoming data).

**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 LINS_DUM_EN(UA_LIN_CTL[3]) register (It is recommend setting the LINS_DUM_EN(UA_LIN_CTL[3]) bit before every checksum reception)

**LIN header error detection**

In LIN slave function mode, when user enables header detection function by setting LINS_HDET_EN(UA_LIN_CTL[1]) bit, the hardware will handle the header detect flow. If the header has error, the LIN header error flag LINS_HERR_F(UA_LIN_SR[1]) will be set and an interrupt is generated if the LIN_RX_BRK_IEN(UA_IER[8]) bit is set. When header error been detect, user must to reset the detect circuit to re-search new frame header by writing 1 to LINS_SYNC_F (UA_LIN_SR[3]) bit to re-search new frame header.

The LIN header error flag, LIN_HERR_F(UA_LIN_SR[1]), is set if one of the following conditions occurs:

Break Delimiter is too short (less 0.5 bit time).

Frame error in sync field or Identifier field.

The sync field data is not 0x55 (without 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.

5.16.5.9 *Interrupt*

The UART Controller supports seven types of interrupts including

(1). LIN function interrupt (INT_LIN)

(2). Buffer error interrupt (INT_BUF_ERR)

(3). time-out interrupt (INT_TOUT)

(4). MODEM status interrupt (INT_MODEM)

(5). Line status interrupt (break error, parity error, framing error or RS-485 interrupt) (INT_RLS)

(6). Receiver threshold level reaching interrupt (INT_RDA)

(7). Transmitter FIFO empty interrupt (INT_THRE)

| UART Interrupt Source | Interrupt Enable Bit | Interrupt Indicator To Interrupt Controller | Interrupt Flag | Flag Cleared By |
|---|---|---|---|---|
| LIN interrupt | LIN_RX_BRK_IEN | LIN_INT | LIN_IF | Write ‘1’ to LINS_HDET_F/LIN_BKDET_F/BIT_ERR_F/LINS_IDPERR_F/LINS_HERR_F |
| Buffer Error Interrupt INT_BUF_ERR | BUF_ERR_IEN | BUF_ERR_INT | BUF_ERR_IF = (TX_OVER_IF or RX_OVER_IF) | Write ‘1’ to TX_OVER_IF/ RX_OVER_IF |
| RX Time-out Interrupt INT_TOUT | RTO_IEN | TOUT_INT | TOUT_IF | Read UA_RBR |
| Modem Status Interrupt INT_MODEM | MODEM_IEN | MODEM_INT | MODEM_IF = (DCTSF) | Write ‘1’ to DCTSF |
| Receive Line Status Interrupt INT_RLS | RLS_IEN | RLS_INT | RLS_IF = (BIF or FEF or PEF or RS-485_ADD_DETFor SC_OVEREF) | Write ‘1’ to BIF/FEF/PEF/ RS-485_ADD_DETF/ TX_OVER_ERETRY/ |

| | | | | |
|---|---|---|---|---|
| | | | | RX_OVER_ERETRY |
| Transmit Holding Register Empty Interrupt INT_THRE | THRE_IEN | THRE_INT | THRE_IF | Write UA_THR |
| Receive Data Available Interrupt INT_RDA | RDA_IEN | RDA_INT | RDA_IF | Read UA_RBR |

Table 5.16-3 UART Interrupt Sources and Flag List in Software Mode

5.16.5.10 *Time-Out*

The time-out counter resets and starts counting (the counting clock = baud rate clock) whenever the RX FIFO receives a new data word. Once the content of time-out counter (TOUT_CNT) is equal to that of time-out interrupt comparator TOIC (UA_TOR[7:0]), a receiver time-out interrupt (INT_TOUT) is generated if RTO_IEN (UA_IER[4]). A new incoming data word or RX FIFO empty clears INT_TOUT. 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

### 5.16.6 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UART Base Address:**<br>**UART0_BA = 0xB800_0000**<br>**UART1_BA = 0xB800_0100**<br>**UART2_BA = 0xB800_0200**<br>**UART3_BA = 0xB800_0300**<br>**UART4_BA = 0xB800_0400**<br>**UART5_BA = 0xB800_0500**<br>**UART6_BA = 0xB800_0600**<br>**UART7_BA = 0xB800_0700**<br>**UART8_BA = 0xB800_0800**<br>**UART9_BA = 0xB800_0900**<br>**UART10_BA = 0xB800_0A00** | | | | |
| **UAn_RBR**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x000 | R | UART n Receive Buffer Register | Undefined |
| **UAn_THR**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x000 | W | UART n Transmit Holding Register | Undefined |
| **UAn_IER**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x004 | R/W | UART n Interrupt Enable Register | 0x0000_0000 |
| **UAn_FCR**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x008 | R/W | UART n FIFO Control Register | 0x0000_0000 |
| **UAn_LCR**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x00C | R/W | UART n Line Control Register | 0x0000_0000 |
| **UAn_MCR**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x010 | R/W | UART n Modem Control Register | 0x0000_0000 |
| **UAn_MSR**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x014 | R/W | UART n Modem Status Register | 0x0000_0000 |
| **UAn_FSR**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x018 | R/W | UART n FIFO Status Register | 0x1040_4000 |
| **UAn_ISR**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x01C | R/W | UART n Interrupt Status Register | 0x0000_0002 |
| **UAn_TOR**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x020 | R/W | UART n Time-out Register | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **UAn_BAUD**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x024 | R/W | UART n Baud Rate Divisor Register | 0x0F00_0000 |
| **UAn_IRCR**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x028 | R/W | UART n IrDA Control Register | 0x0000_0040 |
| **UAn_ALT_CSR**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x02C | R/W | UART n Alternate Control/Status Register | 0x0000_000C |
| **UAn_FUN_SEL**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x030 | R/W | UART n Function Select Register | 0x0000_0000 |
| **UAn_LIN_CTL**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x034 | R/W | UART n LIN Control Register | 0x000C_0000 |
| **UAn_LIN_SR**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x038 | R/W | UART n LIN Status Register | 0x0000_0000 |

### 5.16.7 Register Description

### UART n Receive Buffer Register (UAn_RBR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UAn_RBR**<br>**n=0,1,2,3,4,5,6, 7,8,9,10** | UARTn_BA+0x000 | R | UART n Receive 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** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **RBR** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **RBR** | **Receive Buffer (Read Only)**<br>By reading this register, the UART will return an 8-bit data received from RX pin (LSB first). |

## UART n Transmit Holding Register (UAn_THR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UAn_THR**<br>**n=0,1,2,3,4,5,6, 7,8,9,10** | UARTn_BA+0x000 | W | UART n Transmit Holding 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** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **THR** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **THR** | **Transmit Holding Register**<br>By writing to this register, the UART will send out an 8-bit data through the TX pin (LSB first). |

## UART n Interrupt Enable Register (UAn_IER)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UAn_IER**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x004 | R/W | UART n 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** | | **AUTO_CTS_EN** | **AUTO_RTS_EN** | **TIME_OUT_EN** | **Reserved** | | **LIN_RX_BRK_IEN** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | **WAKE_EN** | **BUF_ERR_IEN** | **RTO_IEN** | **MODEM_IEN** | **RLS_IEN** | **THRE_IEN** | **RDA_IEN** |

| Bits | Description | |
|---|---|---|
| [31:14] | **Reserved** | Reserved. |
| [13] | **AUTO_CTS_EN** | **CTS Auto Flow Control Enable Control**<br>0 = CTS auto flow control Disabled.<br>1 = CTS auto flow control Enabled.<br>When CTS auto-flow is enabled, the UART will send data to external device when CTS input assert (UART will not send data to device until CTS is asserted). |
| [12] | **AUTO_RTS_EN** | **RTS Auto Flow Control Enable Control**<br>0 = RTS auto flow control Disabled.<br>1 = RTS auto flow control Enabled.<br>When RTS auto-flow is enabled, if the number of bytes in the RX FIFO equals the RTS_TRI_LEV (UA_FCR [19:16]), the UART will de-assert RTS signal. |
| [11] | **TIME_OUT_EN** | **Time-out Counter Enable Control**<br>0 = Time-out counter Disabled.<br>1 = Time-out counter Enabled. |
| [10:9] | **Reserved** | Reserved. |
| [8] | **LIN_RX_BRK_IEN** | **LIN RX Break Field Detected Interrupt Enable Control**<br>0 = Lin bus RX break filed interrupt masked off.<br>1 = Lin bus RX break filed interrupt Enabled.<br>**Note:** This field is used for LIN function mode. |
| [7] | **Reserved** | Reserved. |
| [6] | **WAKE_EN** | **UART Wake-up Function Enable Control**<br>1 = UART wake-up function Enabled when the chip is in Power-down mode, an external 0 = UART wake-up function Disabled. |

| | | |
|---|---|---|
| | | **Note:** CTS change will wake up chip from Power-down mode. |
| [5] | **BUF_ERR_IEN** | **Buffer Error Interrupt Enable Control**<br>0 = INT_BUF_ERR masked off.<br>1 = INT_BUF_ERR Enabled. |
| [4] | **RTO_IEN** | **RX Time-out Interrupt Enable Control**<br>0 = INT_TOUT masked off.<br>1 = INT_TOUT Enabled. |
| [3] | **MODEM_IEN** | **Modem Status Interrupt Enable Control**<br>0 = INT_MODEM masked off.<br>1 = INT_MODEM Enabled. |
| [2] | **RLS_IEN** | **Receive Line Status Interrupt Enable Control**<br>0 = INT_RLS masked off.<br>1 = INT_RLS Enabled. |
| [1] | **THRE_IEN** | **Transmit Holding Register Empty Interrupt Enable Control**<br>0 = INT_THRE masked off.<br>1 = INT_THRE Enabled. |
| [0] | **RDA_IEN** | **Receive Data Available Interrupt Enable Control**<br>0 = INT_RDA masked off.<br>1 = INT_RDA Enabled. |

## UART n FIFO Control Register (UAn_FCR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UAn_FCR**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x008 | R/W | UART n FIFO Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:20] | **Reserved** | Reserved. |
| [19:16] | **RTS_TRI_LEV** | **RTS Trigger Level for Auto-flow Control Use**<br>0000 = 1 Trigger Level (Bytes).<br>0001 = 4 Trigger Level (Bytes).<br>0010 = 8 Trigger Level (Bytes).<br>0011 = 14 Trigger Level (Bytes).<br>0100 = 30/14 (High-speed/Normal Speed) Trigger Level (Bytes).<br>0101 = 46/14 (High-speed/Normal Speed) Trigger Level (Bytes).<br>0110 = 62/14 (High-speed/Normal Speed) Trigger Level (Bytes).<br>others = 62/14 (High-speed/Normal Speed) Trigger Level (Bytes).<br>**Note:** This field is used for auto RTS flow control. |
| [15:9] | **Reserved** | Reserved. |
| [8] | **RX_DIS** | **Receiver Disable**<br>The receiver is disabled or not.<br>0 = Receiver Enabled.<br>1 = Receiver Disabled.<br>**Note:** This field is used for RS-485 Normal Multi-drop mode. It should be programmed before RS485_NMM(UA_ALT_CSR [8]) is programmed. |
| [7:4] | **RFITL** | **RX FIFO Interrupt (INT_RDA) Trigger Level**<br>When the number of bytes in the receive FIFO equals the RFITL, the RDA_IF(UA_ISR[0]) will be set (if RDA_IEN(UA_IER[0] is enabled, an interrupt will generated).<br>0000 = 1 Trigger Level (Bytes).<br>0001 = 4 Trigger Level (Bytes).<br>0011 = 8 Trigger Level (Bytes).<br>011 = 14 Trigger Level (Bytes). |

| | | |
|---|---|---|
| | | 0100 = 30/14 (High-speed/Normal Speed) Trigger Level (Bytes).<br>0101 = 46/14 (High-speed/Normal Speed) Trigger Level (Bytes).<br>0110 = 62/14 (High-speed/Normal Speed) Trigger Level (Bytes).<br>others = 62/14 (High-speed/Normal Speed) Trigger Level (Bytes). |
| [3] | **Reserved** | Reserved. |
| [2] | **TFR** | **TX Field Software Reset**<br>When TFR(UA_FCR[2]) is set, all the byte in the transmit FIFO and TX internal state machine are cleared.<br>0 = No effect.<br>1 = Reset the TX internal state machine and pointers.<br>**Note:** This bit will auto clear needs at least 3 UART engine clock cycles. |
| [1] | **RFR** | **RX Field Software Reset**<br>When RFR(UA_FCR[1]) is set, all the byte in the receiver FIFO and RX internal state machine are cleared.<br>0 = No effect.<br>1 = Reset the RX internal state machine and pointers.<br>**Note:** This bit will be automatically cleared for at least 3 UART engine clock cycles. |
| [0] | **Reserved** | Reserved. |

## UART n Line Control Register (UAn_LCR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UAn_LCR**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x00C | R/W | UART n 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** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | **BCB** | **SPE** | **EPE** | **PBE** | **NSB** | **WLS** | |

| Bits | Description | |
|---|---|---|
| [31:7] | **Reserved** | Reserved. |
| [6] | **BCB** | **Break Control**<br>0 = The serial data output (TX) is not forced to the Spacing State (logic 0).<br>1 = The serial data output (TX) is forced to the Spacing State (logic 0).<br>**Note:** This bit acts only on TX and has no effect on the transmitter logic. |
| [5] | **SPE** | **Stick Parity Enable Control**<br>0 = Stick parity Disabled.<br>1 = If bit 3 and 4 are logic 1, the parity bit is transmitted and checked as logic 0. If bit 3 is 1 and bit 4 is 0 then the parity bit is transmitted and checked as 1. |
| [4] | **EPE** | **Even Parity Enable Control**<br>0 = Odd number of logic 1's is transmitted and checked in each word.<br>1 = Even number of logic 1's is transmitted and checked in each word.<br>**Note:** This bit is effective only when bit 3 (parity bit enable) is set. |
| [3] | **PBE** | **Parity Bit Enable Control**<br>0 = No parity bit.<br>1 = Parity bit is generated on each outgoing character and is checked on each incoming data. |
| [2] | **NSB** | **Number of "STOP Bit"**<br>0= One " STOP bit" is generated in the transmitted data.<br>1= One and a half " STOP bit" is generated in the transmitted data when 5-bit word length is selected;.<br>**Note:** Two "STOP bit" is generated when 6-, 7- and 8-bit word length is selected. |
| [1:0] | **WLS** | **Word Length Selection**<br>00 = 5-bit character length. |

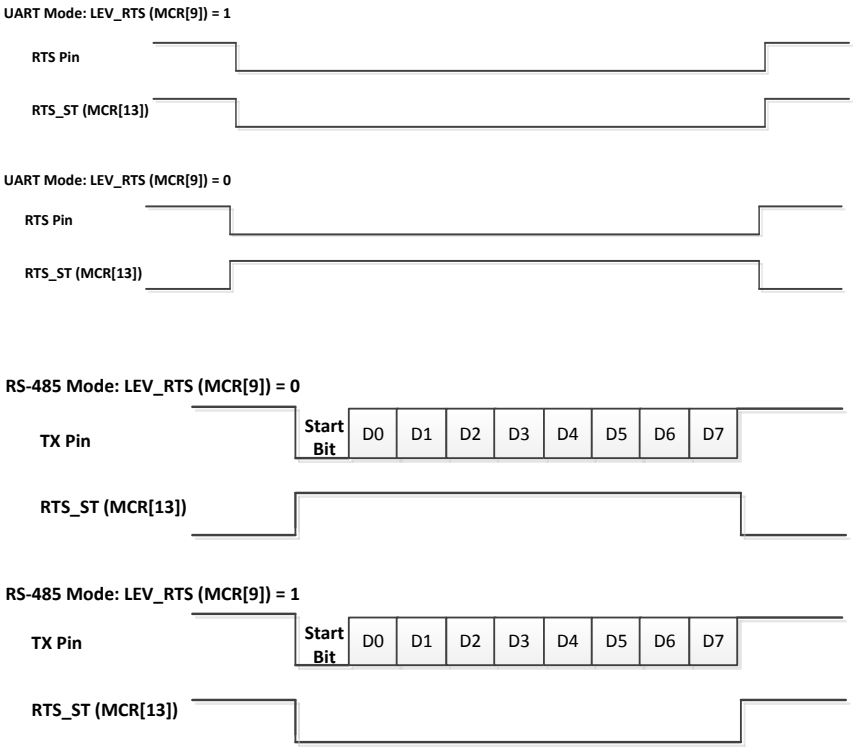
| | | |
|---|---|---|
| | | 01 = 6-bit character length.<br>10 = 7-bit character length.<br>11 = 8-bit character length. |

### UART n MODEM Control Register (UAn_MCR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UAn_MCR**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x010 | R/W | UART n Modem 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** | | **RTS_ST** | **Reserved** | | | **LEV_RTS** | **Reserved** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | | | | **RTS** | **Reserved** |

| Bits | Description | |
|---|---|---|
| [31:14] | **Reserved** | Reserved. |
| [13] | **RTS_ST** | **RTS Pin State (Read Only)**<br>0 = This output pin status of RTS is ‘0’.<br>1 = This output pin status of RTS is ‘1’. |
| [12:10] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [9] | **LEV_RTS** | **RTS Trigger Level**<br>This bit can change the RTS trigger level.<br>0= Low level triggered.<br>1= High level triggered.<br>  |
| [8:2] | **Reserved** | Reserved. |
| [1] | **RTS** | **RTS (Request-to-send) Signal**<br>If the LEV_RTS(UA_MCR[9]) set to low level triggered,<br>0 = Drive RTS pin to logic 1.<br>1 = Drive RTS pin to logic 0.<br>If the LEV_RTS(UA_MCR[9] set to high level triggered,<br>0 = Drive RTS pin to logic 0.<br>1 = Drive RTS pin to logic 1. |
| [0] | **Reserved** | Reserved. |

## UART n Modem Status Register (UAn_MSR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UAn_MSR**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x014 | R/W | UART n Modem 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** | | | | | | | **LEV_CTS** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | **CTS_ST** | **Reserved** | | | **DCTSF** |

| Bits | Description | |
|---|---|---|
| [31:9] | **Reserved** | Reserved. |
| [8] | **LEV_CTS** | **CTS Trigger Level**<br>This bit can change the CTS trigger level.<br>0= Low level triggered.<br>1= High level triggered. |
| [7:5] | **Reserved** | Reserved. |
| [4] | **CTS_ST** | **CTS Pin Status (Read Only)**<br>0 = The pin of CTS is Low.<br>1 = The pin of CTS is High. |
| [3:1] | **Reserved** | Reserved. |
| [0] | **DCTSF** | **Detect CTS State Change Flag (Read Only)**<br>0 = CTS input no change state.<br>1= CTS input has change state, and it will generate Modem interrupt to CPU when MODEM_IEN(UA_IER [3]) is set to 1.<br>**Note:** Software can write 1 to clear this bit to 0 |

## UART n FIFO Status Register (UAn_FSR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UAn_FSR**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x018 | R/W | UART n FIFO Status Register | 0x1040_4000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | TE_FLAG | Reserved | | | TX_OVER_IF |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TX_FULL | TX_EMPTY | TX_POINTER | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RX_FULL | RX_EMPTY | RX_POINTER | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | BIF | FEF | PEF | RS485_ADD_DETF | SC_OVEREF | Reserved | RX_OVER_IF |

| Bits | Description | |
|---|---|---|
| [31:29] | **Reserved** | Reserved. |
| [28] | **TE_FLAG** | **Transmitter Empty Flag (Read Only)**<br>0 = TX FIFO (UA_THR) is not empty.<br>1 = TX FIFO (UA_THR) is empty and the STOP bit of the last byte has been transmitted.<br>**Note:** Bit is cleared automatically when TX FIFO is not empty or the last byte transmission has not completed. |
| [27:25] | **Reserved** | Reserved. |
| [24] | **TX_OVER_IF** | **TX Overflow Error Interrupt Flag (Read Only)**<br>0 = TX FIFO is not over-run.<br>1 = TX FIFO is full, an additional write to UA_THR will cause this bit to logic 1.<br>**Note:** This bit is read only, but it can be cleared by writing '1' to it. |
| [23] | **TX_FULL** | **Transmitter FIFO Full (Read Only)**<br>0 = TX FIFO is not full.<br>0 = TX FIFO is full.<br>**Note:** This bit is set when TX_POINTER is equal to 64/16(High Speed UART/Low Speed UART), otherwise is cleared by hardware. |
| [22] | **TX_EMPTY** | **Transmitter FIFO Empty (Read Only)**<br>0 = TX FIFO is empty.<br>1 = TX FIFO is not empty.<br>**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 THR (TX FIFO not empty). |
| [21:16] | **TX_POINTER** | **TX FIFO Pointer (Read Only)**<br>This field indicates the TX FIFO Buffer Pointer. When CPU writes one byte into |

| | | |
|---|---|---|
| | | UA_THR, TX_POINTER increases one. When one byte of TX FIFO is transferred to Transmitter Shift Register, TX_POINTER decreases one. |
| [15] | **RX_FULL** | **Receiver FIFO Full (Read Only)**<br>0 = RX FIFO is not full.<br>1 = RX FIFO is full.<br>**Note:** This bit is set when RX_POINTER is equal to 64/16(High Speed UART/Low Speed UART), otherwise is cleared by hardware. |
| [14] | **RX_EMPTY** | **Receiver FIFO Empty (Read Only)**<br>0 = RX FIFO is not empty.<br>1 = RX FIFO is empty.<br>**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] | **RX_POINTER** | **RX FIFO Pointer (Read Only)**<br>This field indicates the RX FIFO Buffer Pointer. When UART receives one byte from external device, RX_POINTER increases one. When one byte of RX FIFO is read by CPU, RX_POINTER decreases one. |
| [7] | **Reserved** | Reserved. |
| [6] | **BIF** | **Break Interrupt Flag (Read Only)**<br>0 = No break event active.<br>1 = 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) and is reset whenever the CPU writes 1 to this bit.<br>**Note:** This bit is read only, but it can be cleared by writing ‘1’ to it. |
| [5] | **FEF** | **Framing Error Flag (Read Only)**<br>0 = No any frame data error in “stop bit”.<br>1 = 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 a logic 0), and is reset whenever the CPU writes 1 to this bit.<br>**Note:** This bit is read only, but it can be cleared by writing ‘1’ to it. |
| [4] | **PEF** | **Parity Error Flag (Read Only)**<br>0 = No parity error event .<br>1 = The received character does not have a valid “parity bit”, and is reset whenever the CPU writes 1 to this bit.<br>**Note:** This bit is read only, but it can be cleared by writing ‘1’ to it. |
| [3] | **RS485_ADD_DETF** | **RS-485 Address Byte Detection Flag (Read Only)**<br>0 = No address bit is detected in RS-485 mode.<br>1 = In RS-485 mode, the receiver detect any address byte received address byte character (bit9 = ‘1’) bit”, and it is reset whenever the CPU writes 1 to this bit.<br>**Note:** This field is used for RS-485 function mode.<br>**Note:** This bit is read only, but it can be cleared by writing ‘1’ to it. |
| [2] | **SC_OVEREF** | **Smart Card over Error Retry Flag**<br>0 = No over the Error Retry event.<br>1 = Error Retry event active.<br>**Note:** This field is used for SC function mode.<br>**Note:** This bit is read only, but it can be cleared by writing ‘1’ to it. |
| [1] | **Reserved** | Reserved. |
| [0] | **RX_OVER_IF** | **RX Overflow Error IF (Read Only)** |

| | | |
|---|---|---|
| | | 0 = RX FIFO no overflow.<br>1 = RX FIFO overflow.<br>**Note:** If the number of bytes of received data is greater than RX_FIFO (UA_RBR) size, 64/16 bytes of High Speed UART/Low Speed UART, this bit will be set.<br>**Note:** This bit is read only, but it can be cleared by writing '1' to it. |

## UART n Interrupt Status Control Register (UAn_ISR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UAn_ISR**<br>**n=0,1,2,3,4,5,6,7, 8,9,10** | UARTn_BA+0x01C | R/W | UART n 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** |
| **LIN_INT** | **Reserved** | **BUF_ERR_ INT** | **TOUT_INT** | **MODEM_INT** | **RLS_INT** | **THRE_INT** | **RDA_INT** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **LIN_IF** | **Reserved** | **BUF_ERR_IF** | **TOUT_IF** | **MODEM_IF** | **RLS_IF** | **THRE_IF** | **RDA_IF** |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15] | **LIN_INT** | **LIN Bus Interrupt Indicator (Read Only)**<br>This bit is set if LIN_RX_BRK_IEN(UA_IER[8]) and LIN_IF(UA_ISR[7]) are both set to 1.<br>0 = No LIN RX Break interrupt is generated.<br>1 = LIN RX Break interrupt is generated. |
| [14] | **Reserved** | Reserved. |
| [13] | **BUF_ERR_INT** | **Buffer Error Interrupt Indicator (Read Only)**<br>This bit is set if BUF_ERR_IEN(UA_IER[5]) and BUF_ERR_IF(UA_ISR[5]) are both set to 1.<br>0 = No buffer error interrupt is generated.<br>1 = The buffer error interrupt is generated. |
| [12] | **TOUT_INT** | **Time-out Interrupt Indicator (Read Only)**<br>This bit is set if TIME_OUT_EN(UA_IER[11]) and TOUT_IF(UA_ISR[4]) are both set to 1.<br>0 = No Tout interrupt is generated.<br>1 = Tout interrupt is generated. |
| [11] | **MODEM_INT** | **MODEM Status Interrupt Indicator (Read Only)**<br>This bit is set if MODEM_IEN(UA_IER[3]) and MODEM_IF(UA_ISR[3]) are both set to 1.<br>0 = No Modem interrupt is generated.<br>1 = Modem interrupt is generated. |
| [10] | **RLS_INT** | **Receive Line Status Interrupt Indicator (Read Only)**<br>This bit is set if RLS_IEN(UA_IER[2]) and RLS_IF(UA_ISR[2]) are both set to 1.<br>0 = No RLS interrupt is generated.<br>1 = RLS interrupt is generated. |

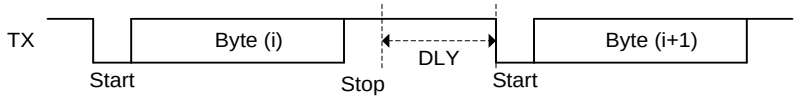
| | | |
|---|---|---|
| [9] | **THRE_INT** | **Transmit Holding Register Empty Interrupt Indicator (Read Only)**<br>This bit is set if THRE_IEN(UA_IER[1]) and THRE_IF(UA_ISR[1]) are both set to 1.<br>0 = No THRE interrupt is generated.<br>1 = THRE interrupt is generated. |
| [8] | **RDA_INT** | **Receive Data Available Interrupt Indicator (Read Only)**<br>This bit is set if RDA_IEN(UA_IER[0]) and RDA_IF(UA_ISR[0]) are both set to 1.<br>0 = No RDA interrupt is generated.<br>1 = RDA interrupt is generated. |
| [7] | **LIN_ IF** | **LIN Bus Flag (Read Only)**<br>This bit is set when LIN slave header detect (LINS_HDET_F(UA_LIN_SR[0]) = 1), LIN break detect (LIN_BKDET_F(UA_LIN_SR[8]) = 1), bit error detect (BIT_ERR_F(UA_LIN_SR[9]) = 1), LIN slave ID parity error (LINS_IDPERR_F(UA_LIN_SR[2])) or LIN slave header error detect (LINS_HERR_F(UA_LIN_SR[1])) If LIN_RX_BRK_IEN(UA_IER[8]) is enabled the LIN interrupt will be generated.<br>**Note:** This bit is cleared when both LINS_HDET_F(UA_LIN_SR[0]) and LIN_BKDET_F(UA_LIN_SR[8]) and BIT_ERR_F(UA_LIN_SR[9]) and LINS_IDPENR_F(UA_LIN_SR[2]) and LINS_HERR_F(UA_LIN_SR[1]) are cleared |
| [6] | **Reserved** | Reserved. |
| [5] | **BUF_ERR_IF** | **Buffer Error Interrupt Flag (Read Only)**<br>This bit is set when the TX or RX FIFO overflows (TX_OVER_IF(UA_FSR[24])) or RX_OVER_IF(UA_FSR[0]) is set). When BUF_ERR_IF is set, the transfer maybe is not correct. If BUF_ERR_IEN(UA_IER[5]) is enabled, the buffer error interrupt will be generated.<br>**Note:** This bit is cleared when both TX_OVER_IF(UA_FSR[24]) and RX_OVER_IF(UA_FSR[0]) are cleared. |
| [4] | **TOUT_IF** | **Time-out Interrupt Flag (Read Only)**<br>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. If RTO_IEN(UA_IER[4]) is enabled, the Time-out interrupt will be generated.<br>**Note:** This bit is read only and user can read UA_RBR (RX is in active) to clear it. |
| [3] | **MODEM_IF** | **MODEM Interrupt Flag (Read Only)**<br>This bit is set when the CTS pin has state change (DCTSF=1). If MODEM_IEN(UA_IER[3]) is enabled, the Modem interrupt will be generated.<br>**Note:** This bit is read only and reset to 0 when bit DCTSF is cleared by a write 1 on DCTSF. |
| [2] | **RLS_IF** | **Receive Line Interrupt Flag (Read Only)**<br>This bit is set when the RX receive data have parity error, framing error or break error (at least one of 3 bits, BIF(UA_FSR[6]), FEF(UA_FSR[5]) and PEF(UA_FSR[4]), is set). If RLS_IEN(UA_IER[2]) is enabled, the RLS interrupt will be generated.<br>**Note:** In RS-485 function mode, this field include “receiver detect any address byte received address byte character (bit9 = ‘1’) bit".<br>**Note:** In SC function mode, this field includes “error retry over flag ".<br>**Note:** This bit is read only and reset to 0 when all bits of BIF(UA_FSR[6]), FEF(UA_FSR[5]) and PEF(UA_FSR[4]) are cleared. |
| [1] | **THRE_IF** | **Transmit Holding Register Empty Interrupt Flag (Read Only)**<br>This bit is set when the last data of TX FIFO is transferred to Transmitter Shift Register. If THRE_IEN(UA_IER[1]) is enabled, the THRE interrupt will be generated.<br>**Note:** This bit is read only and it will be cleared when writing data into THR (TX FIFO not empty). |

| | | |
|---|---|---|
| [0] | **RDA_IF** | **Receive Data Available Interrupt Flag (Read Only)**<br>When the number of bytes in the RX FIFO equals the RFITL then the RDA_IF will be set. If RDA_IEN(UA_IER[0]) is enabled, the RDA interrupt will be generated.<br>**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 n Time-out Register (UAn_TOR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UAn_TOR**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x020 | R/W | UART n 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**<br>This field is use to programming the transfer delay time between the last stop bit and next start bit.<br>  <br>**Note:** The counter clock is baud rate clock |
| [7:0] | **TOIC** | **Time-out Interrupt Comparator**<br>The time-out counter resets and starts counting (the counting clock = baud rate clock) whenever the RX FIFO receives a new data word.<br>**Note1:** 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. |

## UART n Baud Rate Divider Register (UAn_BAUD)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UAn_BAUD**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x024 | R/W | UART n Baud Rate Divisor Register | 0x0F00_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | DIV_X_EN | DIV_X_ONE | DIVIDER_X | | | |
| 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] | **DIV_X_EN** | **Divider X Enable Control**<br>0 = Divider X Disabled (the equation of M = 16).<br>1 = Divider X Enabled (the equation of M = X+1, but DIVIDER_X(UA_BAUD[27:24]) must >= 8).<br>Refer to the table below for more information.<br>**Note:** In IrDA mode, this bit must disable.<br>**Note:** The BRD = Baud Rate Divider, and the baud rate equation is Baud Rate = Clock / [M * (BRD + 2)]; The default value of M is 16. |
| [28] | **DIV_X_ONE** | **Divider X Equal to 1**<br>0 = Divider M = X (the equation of M = X+1, but DIVIDER_X(UA_BAUD[27:24]) must >= 8).<br>1 = Divider M = 1 (the equation of M = 1, but BRD(UA_BAUD[15:0]) must >= 3). |
| [27:24] | **DIVIDER_X** | **Divider X**<br>The baud rate divider M = X+1. |
| [23:16] | **Reserved** | Reserved. |
| [15:0] | **BRD** | **Baud Rate Divider**<br>The field indicated the baud rate divider |

## UART n IrDA Control Register (UAn_IRCR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UAn_IRCR**<br>**n=0,1,2,3,4,5,6,7,8,9,10** | UARTn_BA+0x028 | R/W | UART n 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** |
| **TPS** | **INV_RX** | **INV_TX** | **Reserved** | | | **TX_SELECT** | **Reserved** |

| Bits | Description | |
|---|---|---|
| [31:7] | **Reserved** | Reserved. |
| [7] | **TPS** | **TX Pulse Width**<br>0 = IrDA TX output Pulse width is equal to UART's 3/16 bit frame.<br>1 = IrDA TX output Pulse width is fixed 1.6us. |
| [6] | **INV_RX** | **Inver RX**<br>0 = No inversion.<br>1 = Inverse RX input signal. |
| [5] | **INV_TX** | **Inver TX**<br>0 = No inversion.<br>1 = Inverse TX output signal. |
| [4:2] | **Reserved** | Reserved. |
| [1] | **TX_SELECT** | **TX Select**<br>0 = Select IrDA receiver.<br>1 = Select IrDA transmitter. |
| [0] | **Reserved** | Reserved. |

**Note:** In IrDA mode, the DIV_X_EN(UA_BAUD[29]) register must be disabled (the baud equation must be Clock / 16 * (BRD).

## UART n Alternate Control/Status Register (UAn_ALT_CSR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UAn_ALT_CSR**<br>**n=0,1,2,3,4,5,6, 7,8,9,10** | UARTn_BA+0x02C | R/W | UART n Alternate Control/Status Register | 0x0000_000C |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **ADDR_MATCH** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **RS485_ADD_EN** | **Reserved** | | | | **RS485_AUD** | **RS485_AAD** | **RS485_NMM** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **LIN_TX_EN** | **LIN_RX_EN** | **Reserved** | | **UA_LIN_BKFL** | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **ADDR_MATCH** | **Address Match Value**<br>This field contains the RS-485 address match values.<br>**Note:** This field is used for RS-485 auto address detection mode. |
| [23:16] | **Reserved** | Reserved. |
| [15] | **RS485_ADD_EN** | **RS-485 Address Detection Enable Control**<br>This bit is use to enable RS-485 address detection mode.<br>0 = address detection mode Disabled.<br>1 = Address detection mode Enabled.<br>**Note:** This field is used for RS-485 any operation mode. |
| [14:11] | **Reserved** | Reserved. |
| [10] | **RS485_AUD** | **RS-485 Auto Direction Mode (AUD)**<br>0 = RS-485 Auto Direction Operation (AUO) mode Disabled.<br>1 = RS-485 Auto Direction Operation (AUO) mode Enabled.<br>**Note:** It can be active with RS-485_AAD or RS-485_NMM operation mode. |
| [9] | **RS485_AAD** | **RS-485 Auto Address Detection Operation Mode (AAD)**<br>0 = RS-485 Auto Address Detection (AAD) Operation mode Disabled.<br>1 = RS-485 Auto Address Detection (AAD) Operation mode Enabled.<br>**Note:** It can't be active with RS485_NMM operation mode. |
| [8] | **RS485_NMM** | **RS-485 Normal Multi-drop Operation Mode (NMM)**<br>0 = RS-485 Normal Multi-drop Operation Mode (NMM) Disabled.<br>1 = RS-485 Normal Multi-drop Operation Mode (NMM) Enabled.<br>**Note:** It can't be active with RS-485_AAD operation mode. |
| [7] | **LIN_TX_EN** | **LIN TX Break Mode Enable Control** |

| | | |
|---|---|---|
| | | The LIN TX header can be “break field” or “break and sync field” or “break, sync and frame ID field” depending on the setting LIN_HEAD_SEL(UA_LIN_CTL[23:22]) register.<br>0 = Send LIN TX header Disabled.<br>1 = Send LIN TX header Enabled.<br>**Note:** When transmitter header field (it may be “break” or “break + sync” or “break + sync + frame ID” selected by LIN_HEAD_SEL(UA_LIN_CTL[23:22]) field) transfer operation finished, this bit will be cleared automatically. |
| [6] | **LIN_RX_EN** | **LIN RX Enable Control**<br>0 = LIN RX mode Disabled.<br>1 = LIN RX mode Enabled. |
| [5:4] | **Reserved** | Reserved. |
| [3:0] | **LIN_BKFL** | **LIN Break Field Length**<br>This field indicates a 4-bit LIN TX break field count.<br>**Note1:** This break field length is UA_LIN_BKFL + 1.<br>**Note2:** According to LIN spec, the reset value is 0xC (break field length = 13). |

### UART n Function Select Register (UAn_FUN_SEL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UAn_FUN_SEL**<br>**n=0,1,2,3,4,5,6, 7,8,9,10** | UARTn_BA+0x030 | R/W | UART n Function Select Register | 0x0000_0000 |

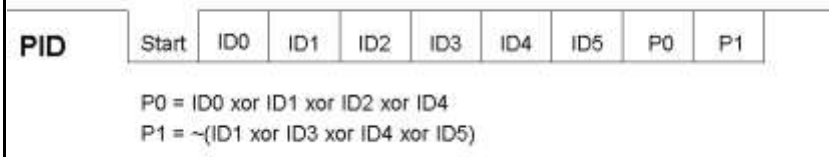
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** |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

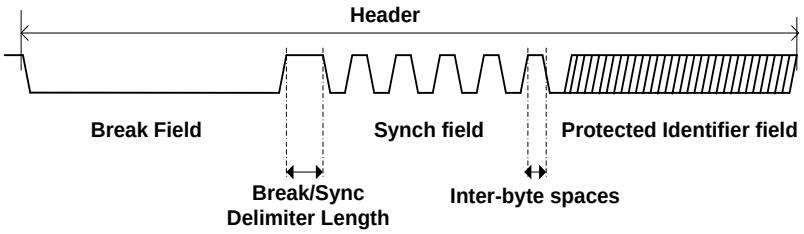
| Bits | Description | |
|---|---|---|
| [31:3] | **Reserved** | Reserved. |
| [2:0] | **FUN_SEL** | **Function Select Enable Control**<br>000 = UART function.<br>001 = LIN function Enabled.<br>010 = IrDA function Enabled.<br>011 = RS-485 function Enabled.<br>100 = Smart-Card function Enabled. |

## UART n LIN Control Register (UAn_LIN_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| UAn_LIN_CTL<br>n=0,1,2,3,4,5,6,7,8,9,10 | UARTn_BA+0x034 | R/W | UART n LIN Control Register | 0x000C_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| LIN_PID | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| LIN_HEAD_SEL | | LIN_BS_LEN | | LIN_BKFL | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LIN_LBDL | Reserved | | BIT_ERR_EN | LIN_RX_DIS | LIN_BKDET_EN | LIN_IDPEN | LIN_SHD |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | LIN_MUTE_EN | LINS_DUM_EN | LINS_ARS_EN | LINS_HDET_EN | LINS_EN |

| Bits | Description | |
|---|---|---|
| [31:24] | **LIN_PID** | **This Field Contains the LIN Frame ID Value in LIN Function Mode, the Frame ID Parity Can Be Generated by Software or Hardware Depending on UA_LIN_CTL [LIN_IDPEN]**<br>If the parity generated by hardware (LIN_IDPEN(UA_LIN_CTL[9]) = 1), user fill ID0~ID5, hardware will calculi P0 and P1, otherwise user must filled frame ID and parity in this field.<br>  <br>P0 = ID0 xor ID1 xor ID2 xor ID4<br>P1 = ~(ID1 xor ID3 xor ID4 xor ID5)<br>**Note1:** User can fill any 8-bit value to this field and the bit 24 indicates ID0 (LSB first)<br>**Note2:** This field can be used for LIN Master mode or Slave mode. |
| [23:22] | **LIN_HEAD_SEL** | **LIN Header Selection**<br>00 = LIN header includes “break field”.<br>01 = LIN header includes “break field” and “sync field”.<br>10 = LIN header includes “break field”, “sync field” and “frame ID field”.<br>11 = LIN header includes “break field”, “sync field” and “frame ID field”, but this mode only supports Receiver mode, not support transmitter mode. This mode difference with mode “10”; in this mode, the receiver will receive ID field (not check the LIN_PID(UA_LIN_CTL[31:24]) register) and when received ID field the LINS_HDET_F will be asserted (if LINS_HDET_EN(UA_LIN_CTL[1]) be set).<br>**Note:** This bit is used to master mode for LIN to sending header field (LIN_SHD = 1) or used to slave to indicates wake-up condition from mute mode (LIN_MUTE_EN(UA_LIN_CTL[4])). |

| | | |
|---|---|---|
| [21:20] | **LIN_BS_LEN** | **LIN Break/Sync Delimiter Length**<br>00 = LIN break/sync delimiter length is 1 bit time.<br>10 = The LIN break/sync delimiter length is 2 bit time.<br>10 = The LIN break/sync delimiter length is 3 bit time.<br>11 = The LIN break/sync delimiter length is 4 bit time.<br>  <br>**Note:** This bit used for LIN master to send header field. |
| [19:16] | **LIN_BKFL** | **LIN Break Field Length**<br>This field indicates a 4-bit LIN TX break field count.<br>**Note1:** These registers are shadow registers of LIN_BKFL(UA_ALT_CSR[3:0]), User can read/write it by setting LIN_BKFL(UA_ALT_CSR[3:0]) or LIN_BKFL(UA_LIN_CTL[19:16]).<br>**Note2:** This break field length is UA_LIN_BKFL + 1.<br>**Note3:** According to LIN spec, the reset value is 0XC (break field length = 13). |
| [15:13] | **Reserved** | Reserved. |
| [12] | **BIT_ERR_EN** | **Bit Error Detect Enable Control**<br>0 = Bit error detection function Disabled.<br>1 = Bit error detection Enabled.<br>**Note:** In LIN function mode, when occur bit error, hardware will generate an interrupt to CPU (INT_LIN) and the BIT_ERR_F(UA_LIN_SR [9]) flag will be asserted. |
| [11] | **LIN_RX_DIS** | If the receiver is be enabled (LIN_RX_DIS = 0), all received byte data will be accepted and stored in the RX-FIFO, and if the receiver is disabled (LIN_RX_DIS = 1), all received byte data will be ignore.<br>0 = Bit error detection function Disabled.<br>1 = Bit error detection Enabled.<br>**Note:** This bit is only valid when operating in LIN function mode (FUN_SEL(UA_FUN_SEL[2:0]) = 001). |
| [10] | **LIN_BKDET_EN** | **LIN Break Detection Enable Control**<br>When detect great than 11/10 bits are detected as 0, and are followed by a delimiter character, the LIN_BKDET_F(UA_LIN_SR[8]) flag is set at the end of break field. If the (LIN_RX_BRK_IEN(UA_IER[8])) =1, an interrupt will be generated.<br>0 = LIN break detection Disabled.<br>1 = LIN break detection Enabled. |
| [9] | **LIN_IDPEN** | **LIN ID Parity Enable Control**<br>0 = LIN frame ID parity Disabled.<br>1 = LIN frame ID parity Enabled.<br>**Note1:** This bit can be used for LIN master to sending header field (LIN_SHD(UA_LIN_CTL[8]) = 1 and( LIN_HEAD_SEL(UA_LIN_CTL[23:22]) = 10) or be used for enable LIN slave received frame ID parity checked.<br>**Note2:** This bit is only used when operation header transmitter is in LIN_HEAD_SEL(UA_LIN_CTL[23:22]) = 10. |

| | | |
|---|---|---|
| [8] | **LIN_SHD** | **LIN TX Send Header Enable Control**<br>The LIN TX header can be “break field” or “break and sync field” or “break, sync and frame ID field” depending on the setting LIN_HEAD_SEL(UA_LIN_CTL[23:22]) register.<br>0 = Send LIN TX header Disabled.<br>1 = Send LIN TX header Enabled.<br>**Note1:** When transmitter header field (it may be “break” or “break + sync” or “break + sync + frame ID” selected by LIN_HEAD_SEL(UA_LIN_CTL[23:22]) field) transfer operation finished, this bit will be cleared automatically. |
| [7:5] | **Reserved** | Reserved. |
| [4] | **LIN_MUTE_EN** | **LIN Mute Mode Enable Control**<br>0 = LIN mute mode. Disabled<br>1 = LIN mute mode Enabled.<br>**Note:** The wake-up condition from mute mode and each control and interactions of this field are explained in character 1.1.4.3.2 (LIN slave mode). |
| [3] | **LINS_DUM_EN** | **LIN Slave Divider Update Method Enable Control**<br>0 = UA_BAUD is updated as soon as UA_BAUD is writing by software (if no automatic resynchronization update occurs at the same time).<br>1 = UA_BAUD is updated at the next received character. User must set the bit before checksum reception.<br>**Note1:** This bit only valid in LIN slave mode (LINS_EN(UA_LIN_CTL[0]) = 1).<br>**Note2:** This bit is used for LIN slave automatic resynchronization mode (for non-automatic resynchronization mode, this bit should be kept cleared).<br>**Note3:** The control and interactions of this field are explained in character 1.1.4.3.2.2 (Slave mode with automatic resynchronization). |
| [2] | **LINS_ARS_EN** | **LIN Slave Automatic Resynchronization Mode Enable Control**<br>0 = LIN automatic resynchronization Disabled.<br>1 = LIN automatic resynchronization Enabled.<br>**Note1:** This bit only valid in LIN slave mode (LINS_EN(UA_LIN_CTL[0]) = 1).<br>**Note2:** When operating in Automatic Resynchronization mode, the baud rate setting must be mode2 (DIV_X_EN(UA_BAUD[29]) and DIV_X_ONE(UA_BAUD[28]) must be 1).<br>**Note3:** The control and interactions of this field are explained in character 1.1.4.3.2.2 (Slave mode with automatic resynchronization). |
| [1] | **LINS_HDET_EN** | **LIN Slave Header Detection Enable Control**<br>0 = LIN slave header detection Disabled.<br>1 = LIN slave header detection Enabled.<br>**Note1:** This bit only valid in LIN slave mode (LINS_EN(UA_LIN_CTL[0]) = 1).<br>**Note2:** In LIN function mode, when header field (break + sync + frame ID) is detected, hardware will generate an interrupt to CPU (INT_LIN) and the LINS_HDET_F(UA_LIN_SR[0]) flag will be asserted. |
| [0] | **LINS_EN** | **LIN Slave Mode Enable Control**<br>0 = LIN slave mode Disabled.<br>1 = LIN slave mode Enabled. |

## UART n LIN Status Register (UAn_LIN_SR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UAn_LIN_SR**<br>**n=0,1,2,3,4,5,6, 7,8,9,10** | UARTn_BA+0x038 | R/W | UART n 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** | | | | | | **BIT_ERR_F** | **LIN_BKDET_F** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | | **LINS_SYNC_F** | **LINS_IDPERR_F** | **LINS_HERR_F** | **LINS_HDET_F** |

| Bits | Description | |
|---|---|---|
| [31:10] | **Reserved** | Reserved. |
| [9] | **BIT_ERR_F** | **Bit Error Detect Status Flag (Read Only)**<br>0 = The input pin (SIN) state is equals to the output pin (SOUT) state.<br>1 = The input pin (SIN) state not equals to the output pin (SOUT) state.<br>**Note1:** This bit is read only, but can be cleared by writing 1 to it.<br>**Note2:** This bit is only valid when enable bit error detection function (BIT_ERR_EN(UA_LIN_CTL[12]) = 1). |
| [8] | **LIN_BKDET_F** | **LIN Break Detection Flag (Read Only)**<br>This bit is set by hardware when a break is detected and be cleared by writing 1 to it.<br>0 = LIN break not detected.<br>1 = LIN break detected.<br>**Note1:** This bit is read only, but can be cleared by writing 1 to it.<br>**Note2:** This bit is only valid when enable LIN break detection function (LIN_BKDET_EN(UA_LIN_CTL[10])). |
| [7:4] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [3] | **LINS_SYNC_F** | **LIN Slave Sync Field**<br>This bit indicates that the LIN sync field is being analyzed. When the receiver header have some error been detect, user must to reset the internal circuit to re-search new frame header by writing 1 to this bit.<br>0 = The current character is not at LIN sync state.<br>1 = The current character is at LIN sync state.<br>**Note1:** This bit only valid in LIN Slave mode (LINS_EN(UA_LIN_CTL[0]) = 1).<br>**Note2:** This bit is read only, but can be cleared by writing 1 to it.<br>**Note3:** When user writing 1 to it, hardware will reload the initial baud-rate and re-search new frame header, the control and interactions of this field are explained in character 1.1.4.3.2.2 (Slave mode with automatic resynchronization). |
| [2] | **LINS_IDPERR_F** | **LIN Slave ID Parity Error Flag (Read Only)**<br>This bit is set by hardware when receipted frame ID parity is not correct.<br>0 = no active.<br>1 = Receipted frame ID parity is not correct.<br>**Note1:** This bit is read only, but can be cleared by writing 1 to it.<br>**Note2:** This bit is only valid in LIN slave mode (LINS_EN(UA_LIN_CTL[0]) = 1) and LIN frame ID parity check function (LIN_IDPEN(UA_LIN_CTL[9])) is enabled. |
| [1] | **LINS_HERR_F** | **LIN Slave Header Error Flag (Read Only)**<br>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 include “break delimiter is too short”, “frame error in sync field or Identifier field”, “sync field data is not 0x55 without 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”.<br>0 = LIN header error not detected.<br>1 = LIN header error detected.<br>**Note1:** This bit is read only, but can be cleared by writing 1 to it.<br>**Note2:** This bit is only valid in LIN slave mode (LINS_EN(UA_LIN_CTL[0]) = 1) and LIN slave header detection function (LINS_HDET_EN(UA_LIN_CTL[1])) is enabled. |
| [0] | **LINS_HDET_F** | **LIN Slave Header Detection Flag (Read Only)**<br>This bit is set by hardware when a LIN header is detected in LIN slave mode and be cleared by writing 1 to it.<br>0 = LIN header not detected.<br>1 = LIN header detected (break + sync + frame ID).<br>**Note1:** This bit is read only, but can be cleared by writing 1 to it.<br>**Note2:** This bit is only valid in LIN slave mode ([LINS_EN(UA_LIN_CTL[0]) = 1) and LIN slave header detection function (LINS_HDET_EN(UA_LIN_CTL[1])) is enabled.<br>**Note3:** When the ID parity check (LIN_IDPEN(UA_LIN_CTL[9]) = 1) is enabled, if hardware detect complete harder (“break + sync + frame ID”), the LINS_HEDT_F(UA_LIN_SR[0]) will be set no matter the frame ID is corrected or not. |

## 5.17 Smart Card Host Interface (SC)

### 5.17.1 Overview

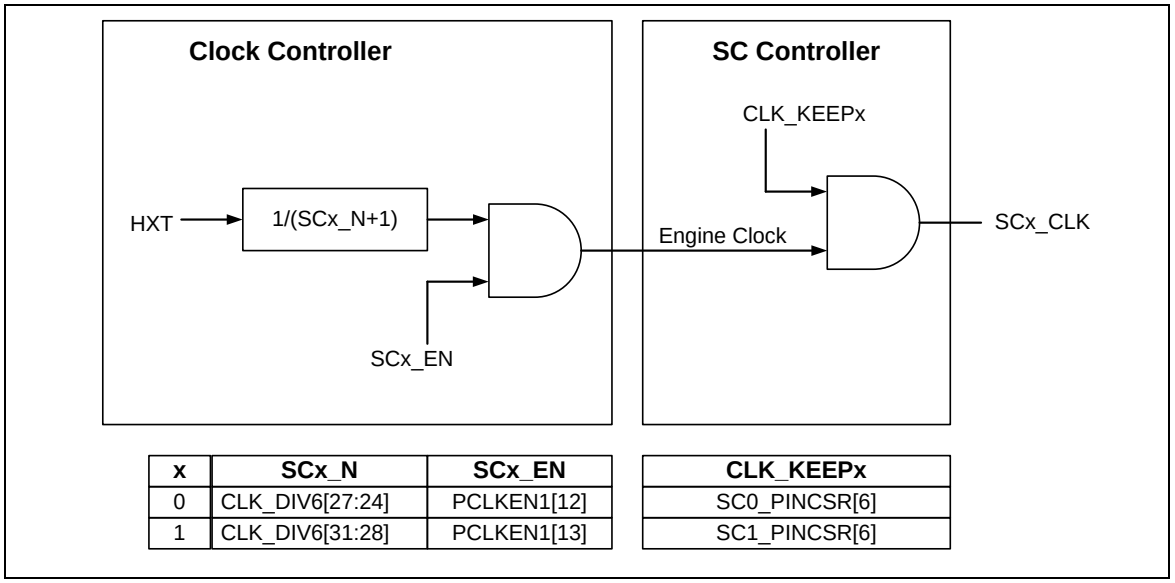
The Smart Card Interface controller (SC controller) is based on ISO/IEC 7816-3 standard and fully compliant with PC/SC Specifications. It also provides status of card insertion/removal.

### 5.17.2 Features

- ISO-7816-3 T = 0, T = 1 compliant
- EMV2000 compliant
- Up to two 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)
- A 24-bit and two 8-bit timers for Answer to Request (ATR) and waiting times processing
- Supports auto inverse convention function
- Supports transmitter and receiver error retry and error number limiting function
- Supports hardware activation sequence, hardware warm reset sequence and 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 for each channel
  - 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 (SC_EGT[7:0])
  - Programmable even, odd or no parity bit generation and detection
  - Programmable stop bit, 1- or 2- stop bit generation

### 5.17.3 Block Diagram

The SC clock control and block diagram are shown as follows. The PCLK should be higher or equal than the frequency of peripheral clock (SCx_CLK).




| x | SCx_N | SCx_EN | CLK_KEEPx |
|---|---|---|---|
| 0 | CLK_DIV6[27:24] | PCLKEN1[12] | SC0_PINCSR[6] |
| 1 | CLK_DIV6[31:28] | PCLKEN1[13] | SC1_PINCSR[6] |

Figure 5.17-1 SC Clock Control Diagram (4-bit Prescale Counter in Clock Controller)

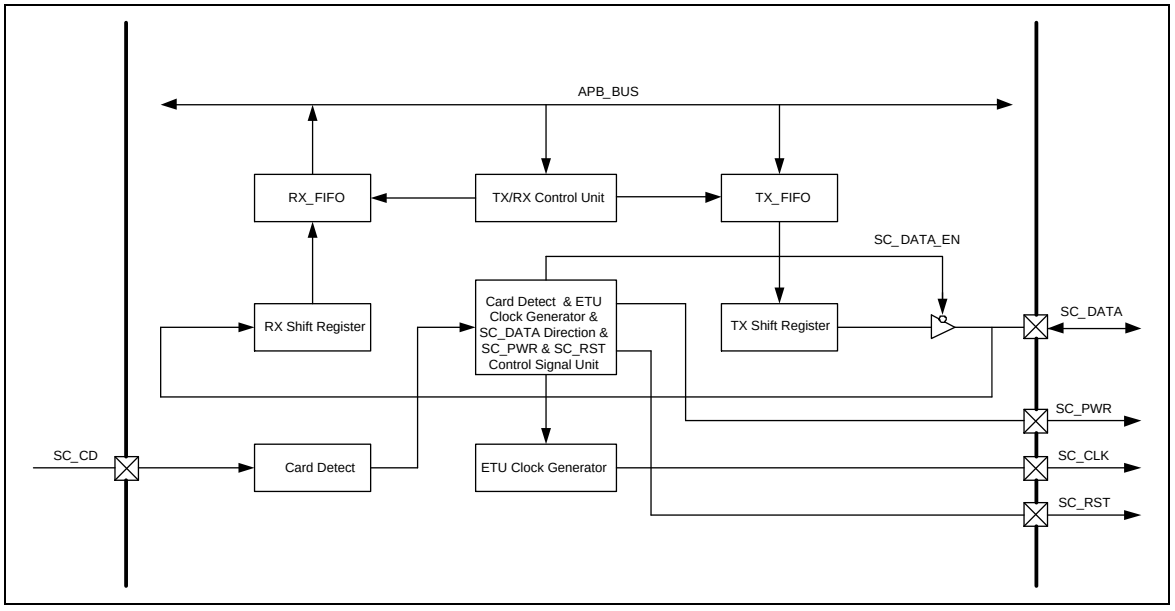



Figure 5.17-2 SC Controller Block Diagram

### 5.17.4 Basic Configuration

The SC function pins are configured in SYS_GPA_MFPL, SYS_GPB_MFPL, SYS_GPE_MFPL and SYS_GPE_MPFH Multiple Function Pin Registers.

SC Host Controller Pin description is shown as follows:

| Pin | Type | Description |
|---|---|---|
| SC_DATA | Bi-direction | SC Host Controller DATA |
| SC_CD | Input | SC Host Controller Card Detect |
| SC_PWR | Output | SC Host Controller Power ON/OFF Switch for RTC_CALMD |
| SC_CLK | Output | SC Host Controller Clock |
| SC_RST | Output | SC Host Controller Reset |

Table 5.17-1 SC Host Controller Pin Description

UART Pin description is shown as follows:

| Pin | Type | Description |
|---|---|---|
| SC_DATA | Input | UART Receive Data |
| SC_CLK | Output | UART Transmit Data |

Table 5.17-2 UART Pin Description

### 5.17.5 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 show as follows.

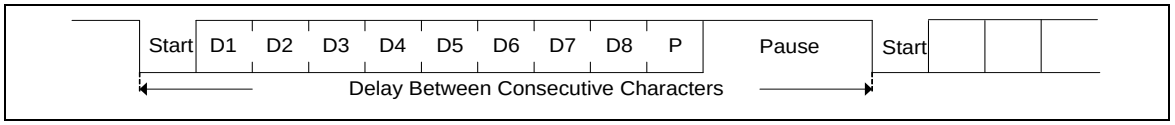



Figure 5.17-3 SC Data Character

The Smart Card Interface controller supports hardware activation, warm reset and deactivation sequence. The activation, Warm Reset and Deactivation and sequence are shown as follows.

#### 5.17.5.1 *Activation, Warm Reset and Deactivation Sequence*

**Activation**

The activation sequence is show as Follows.

1. Set SC_RST to low by programming RSTSTS (SC_PINCTL[18]) to ‘0’
2. Set SC_PWR at high level by programming PWRSTS (SC_PINCTL[18]) to ‘1’ and SC_DAT at high level (reception mode) by programming DATSTS (SC_PINCTL[16]) to ‘1’ .
3. Enable SC_CLK clock by programming CLKKEEP (SC_PINCTL[6]) to ‘1’.

4. De-assert SC_RST to high by programming RSTSTS (SC_PINCTL[18]) to '1'.

The activation sequence can be controlled in two ways. The procedure is shown as follows:

Software Timing Control:

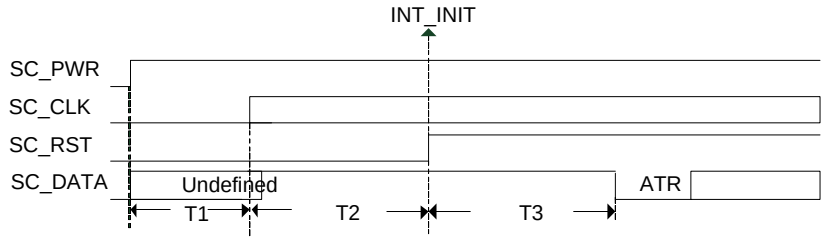
Set SC_PINCTL and SC_TMRCTLx (x = 0, 1, 2) to process the activation sequence. SC_PWR, SC_CLK, SC_RST and SC_DATA pin state can be programmed by SC_PINCTL. The programming method is shown in Activation description. The activation sequence timing can be controlled by setting SC_TMRCTLx (x = 0, 1, 2). This programming procedure provides user has a flexible timing setting for activation sequence.

Hardware Timing Control:

Set ACTEN (SC_ALTCTL[3]) to '1' and the interface will perform the activation sequence by hardware. The SC_PWR to SC_CLK start (T1) and SC_CLK start to SC_RST assert (T2) can be selected by programming INITSEL (SC_ALTCTL[9:8]). This programming procedure provides user has a simple setting for activation sequence.

Following is the activation control sequence generated by hardware:

1. Set activation timing by setting INITSEL (SC_ALTCTL[9:8]).
2. TMR0 can be selected by setting TMRSEL (SC_CTL[14:13]) is '01', '10' or '11'.
3. Set operation mode OPMODE (SC_TMRCTL0[27:24]) to '0011' and give an Answer to Request (ATR) value by setting CNT (SC_TMRCTL0[23:0]) register.
4. When hardware de-asserts SC_RST to high, hardware will generator an interrupt INTIF (SC_INTSTS[8]) to CPU at the same time INITIEN (SC_INTEN[8]) = "1".
5. If the TMR0 decreases the counter to "0" (start from SC_RST de-assert) and the card does not response ATR before that time, hardware will generate interrupt TMR0IF (SC_INTSTS[3]) to CPU.




| Time | Comment |
|---|---|
| T1 | SC_PWR to SC_CLK Start |
| T2 | SC_CLK Start to SC_RST Assert |
| T3 | SC_CLK Start to ATR Appear |

| INIT_SEL (SC_ ALTCT[9:8]) | T1 | T2 |
|---|---|---|
| 00 | 85 | 489 |
| 01 | 133 | 537 |
| 10 | 165 | 569 |
| 11 | 165 | 42060 |

Unit: SC Clock

Note : The values are measured by chip I/O pin and the real value will depend on system design

Figure 5.17-4 SC Activation Sequence

**Warm Reset**

The warm reset sequence is showed as follows.

1. Set SC_RST to low by programming RSTSTS (SC_PINCTL[18]) to '0'.
2. Set SC_DAT to high by programming DATSTS (SC_PINCTL[16]) to '1'.
3. Set SC_RST to high by programming RSTSTS (SC_PINCTL[18]) to '1'.

The warm reset sequence can be controlled in two ways. The procedure is shown as follows.

Software Timing Control:

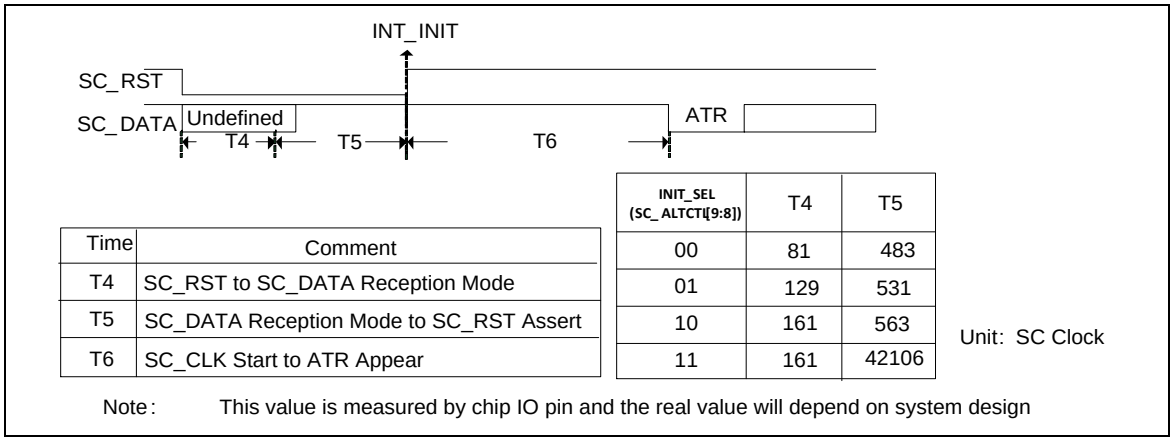
Set SC_PINCTL and SC_TMRCTLx (x = 0, 1, 2) to process the warm reset sequence. SC_RST and SC_DATA pin state can be programmed by SC_PINCTL. The warm reset sequence timing can be controlled by setting SC_TMRCTLx (x = 0, 1, 2). This programming procedure provides user has a flexible timing setting for warm reset sequence.

Hardware Timing Control:

Set WARSTEN (SC_ALTCTL[4]) to '1' and the interface will perform the warm reset sequence by hardware. The SC_RST to SC_DATA reception mode (T4) and SC_DATA reception mode to SC_RST assert (T5) can be selected by programming INITSEL (SC_ALTCTL[9:8]). This programming procedure provides user has a simple setting for warm reset sequence.

Following is THE warm reset control sequence by hardware:

1. Set warm reset timing by setting INITSEL (SC_ALTCTL[9:8]).
2. Select TMR0 by setting TMRSEL (SC_CTL[14:13]) register (TMRSEL can be set to '01', '10', or '11').
3. Set operation mode OPMODE (SC_TMRCTL0[27:24]) to '0011' and give an Answer to Request value by setting CNT (SC_TMRCTL0[23:0]) register.
4. SetCNTEN0 (SC_ALTCTL[5]) and WARSTEN (SC_ALTCTL[4]) to start counting.
5. When hardware de-asserts SC_RST to high, hardware will generate an interrupt INTIF (SC_INTSTS[8]) to CPU at the same time (INITIEN (SC_INTEN[8]) = '1').
6. If the TMR0 decrease the counter to "0" (start from SC_RST) and the card does not response ATR before that time, hardware will generate interrupt TMR0IF (SC_INTSTS[3]) to CPU.




| 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 |

| INIT_SEL (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 5.17-5 SC Warm Reset Sequence

**Deactivation**

The deactivation sequence is showed as follows:

1. Set SC_RST to low by programming RSTSTS (SC_PINCTL[18]) to '0'.
2. Stop SC_CLK by programming CLKKEEP (SC_PINCTL[6]) to '0'.
3. Set SC_DATA to state low by programming DATSTS (SC_PINCTL[16]) to '0'.
4. Deactivate SC_PWR by programming PWRSTS (SC_PINCTL[18]) to '0'.

The deactivation sequence can be controlled in two ways. The procedure is shown as follows.

Software Timing Control:

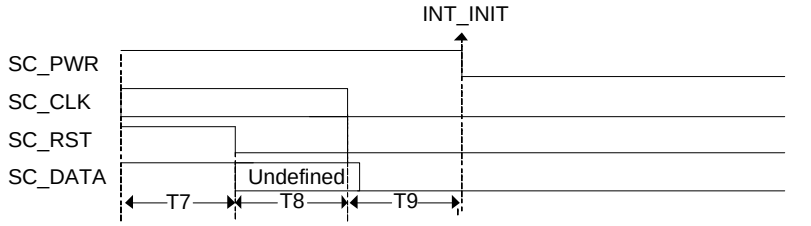
Set SC_PINCTL and SC_TMRCTL0 to process the deactivation sequence. SC_PWR, SC_CLK, SC_RST and SC_DATA pin state can be programmed by SC_PINCTL. The deactivation sequence timing can be controlled by setting SC_TMRCTL0. This programming procedure provides user has a flexible timing setting for deactivation sequence.

Hardware Timing Control:

DACTEN (SC_ALTCTL[2]) to '1' and the interface will perform the deactivation sequence by hardware. The Deactivation Trigger to SC_RST low (T7), SMC_RST low to SC_CLK (T8) and stop SC_CLK to stop SC_PWR (T9) time can be selected by programming INITSEL (SC_ALTCTL[9:8]). This programming procedure provides user has a simple setting for deactivation sequence.

The SC controller also supports auto deactivation sequence when the card removal detection is enabled by setting ADAC_CDEN (SC_ALTCTL[11]).

When hardware de-asserts SC_PWR to low, the controller will generate an interrupt to CPU at the same time (if INITIEN (SC_IER[8]) = 1).




| 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 5.17-6 SC Deactivation Sequence

The Program Sequence Flow is shown as follows:

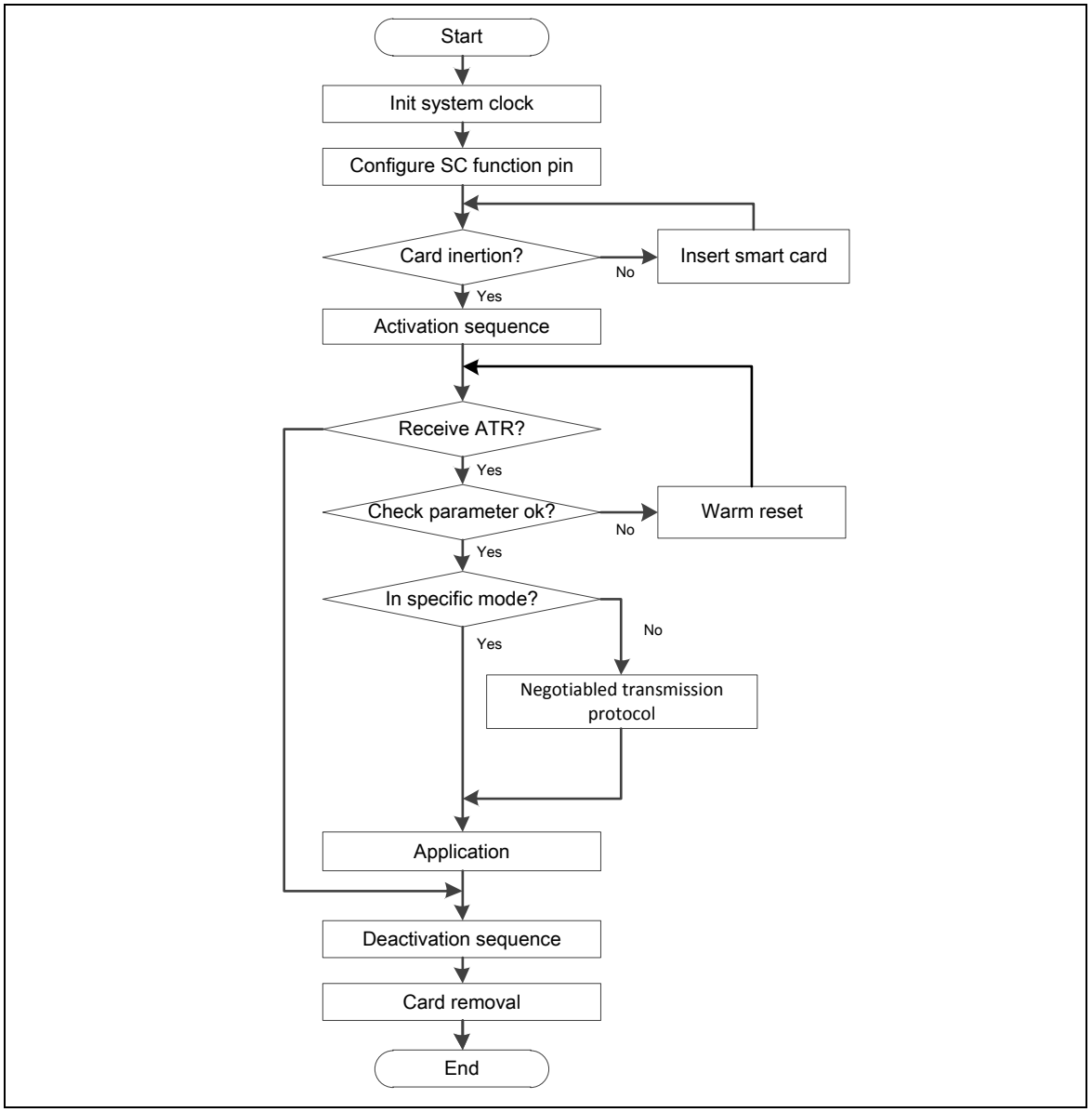



Figure 5.17-7 Basic Operation Flow

5.17.5.2 *Initial Character TS*

According to 7816-3, the initial character TS has two possible patterns shown in the following figure. 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. Software can set AUTOCEN (SC_CTL[3]) and then the operating convention will be decided by hardware. Software can also set the CONSEL (SC_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 (SC_CTL[3]) register, the setting step must be done before Answer to Request state and the first data must be 0x3B or 0x3F. After hardware received first data and stored it at buffer, the hardware will decided the convention and

change the CONSEL (SC_CTL[5:4]) register automatically. If the first data is neither 0x3B nor 0x3F, the hardware will generate an interrupt (if ACERRIEN (SC_INTEN[10]) = '1') to CPU.

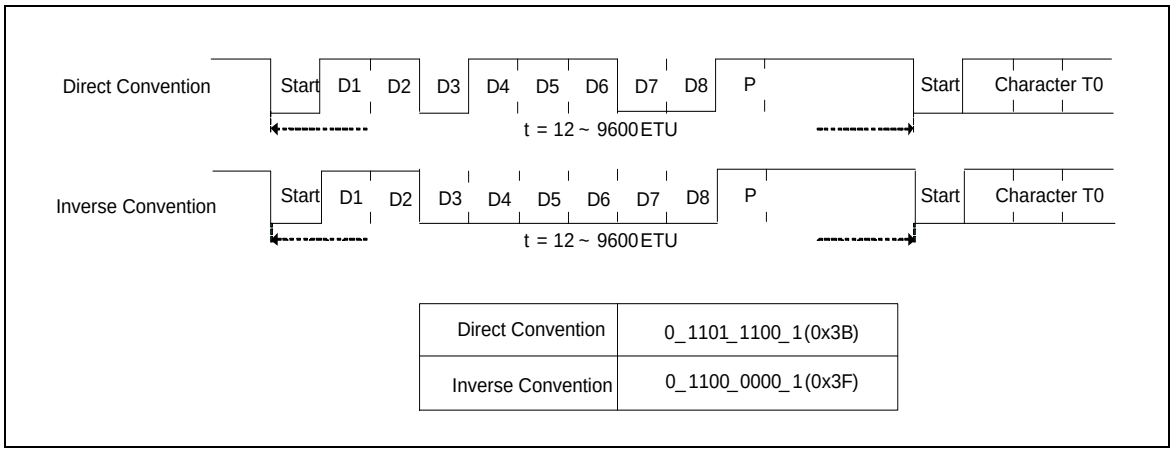



Figure 5.17-8 Initial Character TS

5.17.5.3 *Error Signal and Character Repetition*

According to ISO7816-3 T=0 mode description, as shown in following, if the receiver receives a wrong parity bit, it will pull the SC_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. Software can enable re-transmit function by setting TXRTYEN (SC_CTL[23]). Software can also define the retry (re-transmit) number limitation in TXRTY (SC_CTL[22:20]). The re-transmit number is up to TXRTY +1 and if the re-transmit number is equal to TXRTY +1, TXOVERR flag will be set by hardware and if TERRIEN (SC_INTEN [2]), SC controller will generate a transfer error interrupt to CPU. Software can also define the received retry number limitation in RXRTY (SC_CTL[18:16]) register. 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 flag will be set by hardware and if TERRIEN (SC_INTEN[2]), SC controller will generate a transfer error interrupt to CPU.

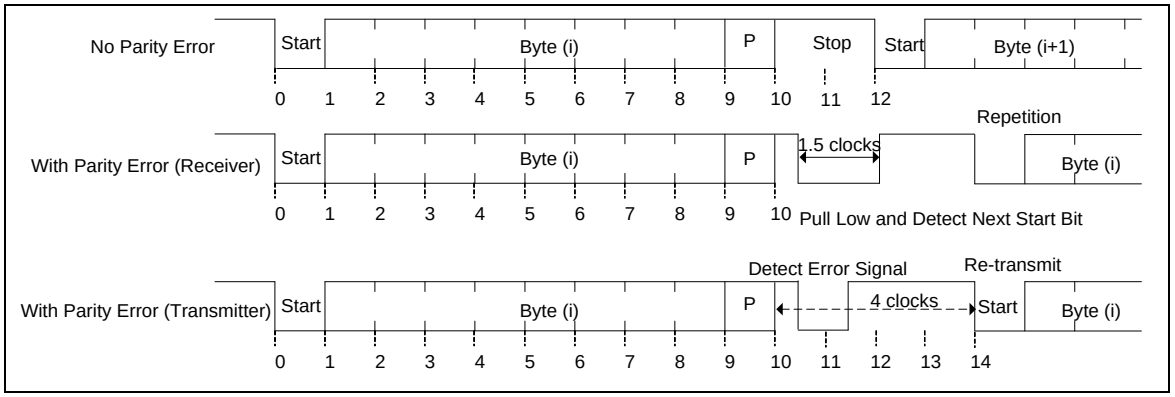



Figure 5.17-9 SC Error Signal

5.17.5.4 *Internal Time-out Counter*

The smart card interface includes a 24-bit time-out counter and two 8 bit time-out counters. These counters help the controller in processing different real-time interval. Each counter can be set to start counting once the trigger enable bit has been written or a START bit has been detected.

The following is the programming flow:

Enable counter by setting TMRSEL (SC_CTL[14:13]). Select operation mode OPMODE (SC_TMRCTLx[27:24]) and give a count value CNT (SC_TMRCTLx[23:0]) by setting SC_TMRCTLx register. Set CNTEN0 (SC_ALTCTL[5]), CNTEN1 (SC_ALTCTL[6]) or CNTEN2 (SC_ALTCTL[7]) is to start counting.

The SC_TMRCTL0, SC_TMRCTL1 and SC_TMRCTL2 timer operation mode are listed in below table.

**Note:** Only SC_TMRCTL0 supports mode 0011.

| **OPMODE (SC_TMRCTLx [27:24]) (X=0 ~2)** | **Operation Description** | |
|---|---|---|
| 0000 | The down counter started when CNTENx (SC_ALTCTL[7:5]) enabled and ended when counter time-out. The time-out value will be CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) + 1. | |
| | Start | Start counting when CNTENx (SC_ALTCTL[7:5]) enabled |
| | End | When the down counter equals to 0, hardware will set TMRxIF (SC_INTSTS[5:3]) and clear CNTENx (SC_ALTCTL[7:5]) automatically. |
| 0001 | The down counter started when the first START bit (reception or transmission) detected and ended when counter time-out. The time-out value will be CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) + 1. | |
| | Start | Start counting when the first START bit (reception or transmission) detected after CNTENx (SC_ALTCTL[7:5]) set to 1. |
| | End | When the down counter equals to 0, hardware will set TMRxIF (SC_INTSTS[5:3]) and clear CNTENx (SC_ALTCTL[7:5]) automatically. |
| 0010 | The down counter started when the first START bit (reception) detected and ended when counter time-out. The time-out value will be CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) + 1. | |
| | Start | Start counting when the first START bit (reception) detected bit after CNTENx (SC_ALTCTL[7:5]) set to 1. |
| | End | When the down counter equals to 0, hardware will set TMRxIF (SC_INTSTS[5:3]) and clear CNTENx (SC_ALTCTL[7:5]) automatically. |
| 0011 | The down counter is only used for hardware activation, warm reset sequence to measure ATR timing.<br>The timing starts when SC_RST de-assertion and ends when ATR response received or time-out.<br>If the counter decreases to 0 before ATR response received, hardware will generate an interrupt to CPU. The time-out value will be CNT (SC_TMRCTL0[23:0]) + 1. | |
| | Start | Start counting when SC_RST de-assertion after CNTEN0 (SC_ALTCTL[5]) set to 1.<br>It is used for hardware activation, warm reset mode. |
| | End | When the down counter equals to 0 before ATR response received, hardware will set TMR0IF (SC_INTSTS[3]) and clear CNTEN0 (SC_ALTCTL[5]) automatically.<br>When ATR received and down counter does not equal to 0, hardware will clear CNTEN0 (SC_ALTCTL[5]) automatically. |
| 0100 | Same as 0000, but when the down counter equals to 0, hardware will set TMRxIF (SC_INTSTS[5:3]) and counter will re-load the CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) value and re-count until software clears CNTENx (SC_ALTCTL[7:5]).<br>When ACTSTSx (SC_ALTCTL[15:13]) = 1, software can change CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) value at any time. When the down counter equals to 0, counter will reload the new value of CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) and re-count.<br>The time-out value will be CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) + 1. | |

| | | |
|---|---|---|
| 0101 | Same as 0001, but when the down counter equals to 0, hardware will set TMRxIF (SC_INTSTS[5:3]) and counter will re-load the CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) value. When the next START bit is detected, counter will re-count until software clears CNTENx (SC_ALTCTL[7:5]).<br>When ACTSTSx (SC_ALTCTL[15:13]) = 1 software can change CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) value at any time. When the down counter equal to 0, it will reload the new value of CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) and re-counting.<br>The time-out value will be CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) + 1. | |
| 0110 | Same as 0010, but when the down counter equals to 0, it will set TMRxIF (SC_INTSTS[5:3]) and counter will re-load the CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) value. When the next START bit is detected, counter will re-count until software clears CNTENx (SC_ALTCTL[7:5]).<br>When ACTSTSx (SC_ALTCTL[15:13]) = 1, software can change CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) value at any time. When the down counter equals to 0, counter will reload the new value of CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) and re-count.<br>The time-out value will be CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) + 1. | |
| 0111 | The down counter started when the first START bit (reception or transmission) detected and ended when software clears CNTENx (SC_ALTCTL[7:5]) bit. If next START bit detected, counter will reload the new value of CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) and re-counting.<br>If the counter decreases to 0 before the next START bit detected, hardware will generate an interrupt to CPU. The time-out value will be CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) + 1. | |
| | Start | Start counting when the first START bit detected after CNTENx (SC_ALTCTL[7:5]) set to 1. |
| | End | Stop counting after CNTENx (SC_ALTCTL[7:5]) set to 0. |
| 1000 | The up counter starts when CNTENx (SC_ALTCTL[7:5]) enabled and ends when CNTENx (SC_ALTCTL[7:5]) disabled. This count value will be stored in CNTx (SC_TMRDAT0[23:0], SC_TMRDAT1_2[7:0], SC_TMRDAT1_2[15:8]). In this mode, hardware cannot generate any interrupt to CPU. The real count value will be CNTx (SC_TMRDAT0[23:0], SC_TMRDAT1_2[7:0], SC_TMRDAT1_2[15:8]) + 1. | |
| | Start | Start counting after CNTENx (SC_ALTCTL[7:5]) set to 1, and the start count value is 0 (hardware will ignore CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) value). |
| | End | Stop counting after CNTENx (SC_ALTCTL[7:5]) set to 0 and the value stored to CNTx (SC_TMRDAT0[23:0], SC_TMRDAT1_2[7:0], SC_TMRDAT1_2[15:8]) register. |
| 1111 | Down counter starts when software set CNTENx (SC_ALTCTL[7:5]) bit or any START bit been detected and ends when software clears CNTENx (SC_ALTCTL[7:5]) bit. If next START bit detected, counter will reload the new value of CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) and re-counting.<br>If the counter decreases to “0” before the next START bit be detected, hardware will generate an interrupt to CPU. The time-out value will be CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0])+1. | |
| | Start | Start count when the CNTENx (SC_ALTCTL[7:5]) set to “1” or any START bit (CNTENx (SC_ALTCTL[7:5]) must be set) be detected. |
| | End | Stop count after CNTENx (SC_ALTCTL[7:5]) set to “0”. |

#### 5.17.5.5 *UART Mode*

When the UARTEN (SC_UARTCTL[0]) bit set, the Smart Card Interface controller can also be used as base UART function. The following is the program example for UART mode.

Program example:

1. Set UARTEN (SC_UARTCTL[0]) bit to enter UART mode.
2. Do software reset by setting RXRST (SC_ALTCTL[1]) and TXRST (SC_ALTCTL[0]) bit to ensure that all state machine return idle state.
3. Fill “0” to CONSEL (SC_CTL[5:4]) and AUTOCEN (SC_CTL[3]) field. (In UART mode, those fields must be “0”)

4. Select the UART baud rate by setting ETURDIV (SC_ETUCR[11:0]) fields. For example, if smartcard module clock is 12 MHZ and target baud rate is 115200bps, ETURDIV should fill with (12000000 / 115200 - 1).
5. Select the data format include data length (by setting WLS (SC_UARTCTL[5:4]), parity format (by setting OPE (SC_UARTCTL[7]) and PBOFF (SC_UARTCTL[6])) and stop bit length (by setting NSB (SC_CTL[15]) or EGT (SC_EGT[7:0])).
6. Select the receiver buffer trigger level by setting RXTRGLV (SC_CTL[7:6]) field and select the receiver buffer time-out value by setting RFTM (SC_RXTOUT[8:0]) field.
7. Write SC_DAT (SC_DAT[7:0]) (TX) register or read the SC_DAT (SC_DAT[7:0]) (RX) register can perform UART function.

### 5.17.6 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SC Base Address:**<br>**SC0_BA = 0xB800_5000**<br>**SC1_BA = 0xB800_5400** | | | | |
| **SC_DAT**<br>**x = 0,1** | SCx_BA+0x000 | R/W | SC Receiving/Transmit Holding Buffer Register | 0xXXXX_XXXX |
| **SC_CTL**<br>**x = 0,1** | SCx_BA+0x004 | R/W | SC Control Register | 0x0000_0000 |
| **SC_ALTCTL**<br>**x = 0,1** | SCx_BA+0x008 | R/W | SC Alternate Control Register | 0x0000_0000 |
| **SC_EGT**<br>**x = 0,1** | SCx_BA+0x00C | R/W | SC Extend Guard Time Register | 0x0000_0000 |
| **SC_RXTOUT**<br>**x = 0,1** | SCx_BA+0x010 | R/W | SC Receive Buffer Time-out Register | 0x0000_0000 |
| **SC_ETUCTL**<br>**x = 0,1** | SCx_BA+0x014 | R/W | SC ETU Control Register | 0x0000_0173 |
| **SC_INTEN**<br>**x = 0,1** | SCx_BA+0x018 | R/W | SC Interrupt Enable Control Register | 0x0000_0000 |
| **SC_INTSTS**<br>**x = 0,1** | SCx_BA+0x01C | R/W | SC Interrupt Status Register | 0x0000_0002 |
| **SC_STATUS**<br>**x = 0,1** | SCx_BA+0x020 | R/W | SC Status Register | 0x0000_0202 |
| **SC_PINCTL**<br>**x = 0,1** | SCx_BA+0x024 | R/W | SC Pin Control State Register | 0x0000_00x0 |
| **SC_TMRCTL0**<br>**x = 0,1** | SCx_BA+0x028 | R/W | SC Internal Timer Control Register 0 | 0x0000_0000 |
| **SC_TMRCTL1**<br>**x = 0,1** | SCx_BA+0x02C | R/W | SC Internal Timer Control Register 1 | 0x0000_0000 |
| **SC_TMRCTL2**<br>**x = 0,1** | SCx_BA+0x030 | R/W | SC Internal Timer Control Register 2 | 0x0000_0000 |
| **SC_UARTCTL**<br>**x = 0,1** | SCx_BA+0x034 | R/W | SC UART Mode Control Register | 0x0000_0000 |
| **SC_TMRDAT0**<br>**x = 0,1** | SCx_BA+0x038 | R | SC Timer Current Data Register 0 | 0x0000_07FF |
| **SC_TMRDAT1**<br>**x = 0,1** | SCx_BA+0x03C | R | SC Timer Current Data Register 1 | 0x0000_7F7F |

### 5.17.7 Register Description

**SC Receiving/Transmit Holding Buffer Register (SC_DAT)**

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SC_DAT**<br>**x=0,1** | SCx_BA+0x000 | R/W | SC Receiving/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** | **Receiving/ Transmit Holding Buffer**<br>Write Operation:<br>By writing data to DAT, the SC will send out an 8-bit data.<br>**Note:** If SCEN (SC_CTL[0]) is not enabled, DAT cannot be programmed.<br><br>Read Operation:<br>By reading DAT, the SC will return an 8-bit received data. |

## SC Control Register (SC_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SC_CTL<br>x=0,1 | SCx_BA+0x004 | 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**<br>Due to synchronization, software should check this bit before writing a new value to RXRTY and TXRTY.<br>0 = Synchronizing is completion, user can write new data to RXRTY and TXRTY.<br>1 = Last value is synchronizing.<br>**Note:** This bit is read only. |
| [29:27] | **Reserved** | Reserved. |
| [26] | **CDLV** | **Card Detect Level**<br>0 = When hardware detects the card detect pin (SC_CD) from high to low, it indicates a card is detected.<br>1 = When hardware detects the card detect pin from low to high, it indicates a card is detected.<br>**Note:** Software must select card detect level before Smart Card engine enabled. |
| [25:24] | **CDDBSEL** | **Card Detect De-bounce Selection**<br>This field indicates the card detect de-bounce selection.<br>00 = De-bounce sample card insert once per 384 (128 * 3) peripheral clocks and de-bounce sample card removal once per 128 peripheral clocks.<br>01 = De-bounce sample card insert once per 192 (64 * 3) peripheral clocks and de-bounce sample card removal once per 64 peripheral clocks.<br>10 = De-bounce sample card insert once per 96 (32 * 3) peripheral clocks and de-bounce sample card removal once per 32 peripheral clocks.<br>11 = De-bounce sample card insert once per 48 (16 * 3) peripheral clocks and de-bounce sample card removal once per 16 peripheral clocks. |
| [23] | **TXRTYEN** | **TX Error Retry Enable Bit**<br>This bit enables transmitter retry function when parity error has occurred. |

| | | |
|---|---|---|
| | | 0 = TX error retry function Disabled.<br>1 = TX error retry function Enabled. |
| [22:20] | **TXRTY** | **TX Error Retry Count Number**<br>This field indicates the maximum number of transmitter retries that are allowed when parity error has occurred.<br>**Note1:** The real retry number is TXRTY + 1, so 8 is the maximum retry number.<br>**Note2:** 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**<br>This bit enables receiver retry function when parity error has occurred.<br>0 = RX error retry function Disabled.<br>1 = RX error retry function Enabled.<br>**Note:** Software must fill in the RXRTY value before enabling this bit. |
| [18:16] | **RXRTY** | **RX Error Retry Count Number**<br>This field indicates the maximum number of receiver retries that are allowed when parity error has occurred.<br>**Note1:** The real retry number is RXRTY + 1, so 8 is the maximum retry number.<br>**Note2:** 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**<br>This field indicates the length of stop bit.<br>0 = The stop bit length is 2 ETU.<br>1= The stop bit length is 1 ETU.<br>**Note:** The default stop bit length is 2. SMC and UART adopt NSB to program the stop bit length. |
| [14:13] | **TMRSEL** | **Timer Selection**<br>00 = All internal timer function Disabled.<br>01 = Internal 24 bit timer Enabled. Software can configure it by setting SC_TMRCTL0[23:0]. SC_TMRCTL1 and SC_TMRCTL2 will be ignored in this mode.<br>10 = Internal 24 bit timer and 8 bit internal timer Enabled. Software can configure the 24 bit timer by setting SC_TMRCTL0[23:0] and configure the 8 bit timer by setting SC_TMRCTL1[7:0]. SC_TMRCTL2 will be ignored in this mode.<br>11 = Internal 24 bit timer and two 8 bit timers Enabled. Software can configure them by setting SC_TMRCTL0[23:0], SC_TMRCTL1[7:0] and SC_TMRCTL2[7:0]. |
| [12:8] | **BGT** | **Block Guard Time (BGT)**<br>Block guard time means the minimum bit length 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, software must fill in 15 (real block guard time = 16.5) to this field; in T = 1 mode, software must fill in 21 (real block guard time = 22.5) to it.<br>**Note:** The real block guard time is BGT + 1. |

| | | |
|---|---|---|
| [7:6] | **RXTRGLV** | **Rx Buffer Trigger Level**<br>When the number of bytes in the receiving buffer equals the RXTRGLV, the RDAIF will be set (if SC_INTEN[RDAIEN] is enabled, an interrupt will be generated).<br>00 = INTR_RDA Trigger Level with 01 Bytes.<br>01 = INTR_RDA Trigger Level with 02 Bytes.<br>10 = INTR_RDA Trigger Level with 03 Bytes.<br>11 = Reserved. |
| [5:4] | **CONSEL** | **Convention Selection**<br>00 = Direct convention.<br>01 = Reserved.<br>10 = Reserved.<br>11 = Inverse convention.<br>**Note:** If AUTOCEN (SC_CTL[3]) enabled, this fields are ignored. |
| [3] | **AUTOCEN** | **Auto Convention Enable Bit**<br>0 = Auto-convention Disabled.<br>1 = Auto-convention Enabled. When hardware receives TS in answer to reset state and the TS is direct convention, CONSEL (SC_CTL[5:4]) will be set to 00 automatically, otherwise if the TS is inverse convention, and CONSEL (SC_CTL[5:4]) will be set to 11.<br>If software enables auto convention function, the setting step must be done before Answer to Reset state and the first data must be 0x3B or 0x3F. After hardware received first data and stored it at buffer, hardware will decide the convention and change the CONSEL (SC_CTL[5:4]) bits automatically. If the first data is not 0x3B or 0x3F, hardware will generate an interrupt INT_ACON_ERR (if ACERRIE (SC_INTEN[10]) = 1) to CPU. |
| [2] | **TXOFF** | **TX Transition Disable Bit**<br>0 = The transceiver Enabled.<br>1 = The transceiver Disabled. |
| [1] | **RXOFF** | **RX Transition Disable Bit**<br>0 = The receiver Enabled.<br>1 = The receiver Disabled.<br>**Note:** If AUTOCEN (SC_CTL[3]) is enabled, this field must be ignored. |
| [0] | **SCEN** | **SC Engine Enable Bit**<br>Set this bit to 1 to enable SC operation. If this bit is cleared, SC will force all transition to IDLE state. |

### SC Alternate Control Register (SC_ALTCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SC_ALTCTL x=0,1 | SCx_BA+0x008 | R/W | SC Alternate Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | OUTSEL |
| 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 | |
|---|---|---|
| [31:17] | **Reserved** | Reserved. |
| [16] | **OUTSEL** | **Smartcard Data Pin Output Mode Selection**<br>Use this bit to select smartcard data pin (SC_DATA) output mode.<br>0 = Quasi mode.<br>1 = Open-drain mode. |
| [15] | **ACTSTS2** | **Internal Timer2 Active State (Read Only)**<br>This bit indicates the timer counter status of timer2.<br>0 = Timer2 is not active.<br>1 = Timer2 is active. |
| [14] | **ACTSTS1** | **Internal Timer1 Active State (Read Only)**<br>This bit indicates the timer counter status of timer1.<br>0 = Timer1 is not active.<br>1 = Timer1 is active. |
| [13] | **ACTSTS0** | **Internal Timer0 Active State (Read Only)**<br>This bit indicates the timer counter status of timer0.<br>0 = Timer0 is not active.<br>1 = Timer0 is active. |
| [12] | **RXBGTEN** | **Receiver Block Guard Time Function Enable Bit**<br>0 = Receiver block guard time function Disabled.<br>1 = Receiver block guard time function Enabled. |

| | | |
|---|---|---|
| [11] | **ADACEN** | **Auto Deactivation When Card Removal**<br>0 = Auto deactivation Disabled when hardware detected the card removal.<br>1 = Auto deactivation Enabled when hardware detected the card removal.<br>**Note:** When the card is removed, hardware will stop any process and then do deactivation sequence (if this bit is set). If this process completes, hardware will generate an interrupt INITIF to CPU. |
| [11:10] | **Reserved** | Reserved. |
| [9:8] | **INITSEL** | **Initial Timing Selection**<br>This fields indicates the timing of hardware initial state (activation or warm-reset or deactivation).<br>Unit: SC clock<br>Activation: refer to SC Activation Sequence in Figure 5.17-4<br>Warm-reset: refer to Warm-Reset Sequence in Figure 5.17-5.<br>Deactivation: refer to Deactivation Sequence in Figure 5.17-6. |
| [7] | **CNTEN2** | **Internal Timer2 Start Enable Bit**<br>This bit enables Timer 2 to start counting. Software can fill in 0 to stop it and set 1 to reload and count.<br>0 = Stops counting.<br>1 = Starts counting.<br>**Note1:** This field is used for internal 8 bit timer when TMRSEL (SC_CTL[14:13]) = 11. Don't fill CNTEN2 when TMRSEL (SC_CTL[14:13]) = 00 or TMRSEL (SC_CTL[14:13]) = 01 or TMRSEL (SC_CTL[14:13]) = 10.<br>**Note2:** If the operation mode is not in auto-reload mode (SC_TMRCTL2[26] = 0), this bit will be auto-cleared by hardware.<br>**Note3:** This field will be cleared by TXRST (SC_ALTCTL[0]) and RXRST (SC_ALTCTL[1]). So don't fill this bit, TXRST (SC_ALTCTL[0]), and RXRST (SC_ALTCTL[1]) at the same time.<br>**Note4:** If SCEN (SC_CTL[0]) is not enabled, this field cannot be programmed. |
| [6] | **CNTEN1** | **Internal Timer1 Start Enable Bit**<br>This bit enables Timer 1 to start counting. Software can fill in 0 to stop it and set 1 to reload and count.<br>0 = Stops counting.<br>1 = Starts counting.<br>**Note1:** This field is used for internal 8 bit timer when TMRSEL (SC_CTL[14:13]) = 10 or TMRSEL (SC_CTL[14:13]) = 11. Don't fill CNTEN1 when TMRSEL (SC_CTL[14:13]) = 00 or TMRSEL (SC_CTL[14:13]) = 01.<br>**Note2:** If the operation mode is not in auto-reload mode (SC_TMRCTL1[26] = 0), this bit will be auto-cleared by hardware.<br>**Note3:** This field will be cleared by TXRST (SC_ALTCTL[0]) and RXRST (SC_ALTCTL[1]), so don't fill this bit, TXRST (SC_ALTCTL[0]), and RXRST (SC_ALTCTL[1]) at the same time.<br>**Note4:** If SCEN (SC_CTL[0]) is not enabled, this field cannot be programmed. |

| | | |
|---|---|---|
| [5] | **CNTEN0** | **Internal Timer0 Start Enable Bit**<br>This bit enables Timer 0 to start counting. Software can fill in 0 to stop it and set 1 to reload and count.<br>0 = Stops counting.<br>1 = Starts counting.<br>**Note1:** This field is used for internal 24 bit timer when TMRSEL (SC_CTL[14:13]) = 01.<br>**Note2:** If the operation mode is not in auto-reload mode (SC_TMRCTL0[26] = 0), this bit will be auto-cleared by hardware.<br>**Note3:** This field will be cleared by TXRST (SC_ALTCTL[0]) and RXRST (SC_ALTCTL[1]). So don't fill this bit, TXRST and RXRST at the same time.<br>**Note4:** If SCEN (SC_CTL[0]) is not enabled, this field cannot be programmed. |
| [4] | **WARSTEN** | **Warm Reset Sequence Generator Enable Bit**<br>This bit enables SC controller to initiate the card by warm reset sequence.<br>0 = No effect.<br>1 = Warm reset sequence generator Enabled.<br>**Note1:** When the warm reset sequence completed, this bit will be cleared automatically and the INITIF (SC_INTSTS[8]) will be set to 1.<br>**Note2:** This field will be cleared by TXRST (SC_ALTCTL[0]) and RXRST (SC_ALTCTL[1]), so don't fill this bit, TXRST, and RXRST at the same time.<br>**Note3:** If SCEN (SC_CTL[0]) is not enabled, this field cannot be programmed. |
| [3] | **ACTEN** | **Activation Sequence Generator Enable Bit**<br>This bit enables SC controller to initiate the card by activation sequence.<br>0 = No effect.<br>1 = Activation sequence generator Enabled.<br>**Note1:** When the activation sequence completed, this bit will be cleared automatically and the INITIF (SC_INTSTS[8]) will be set to 1.<br>**Note2:** This field will be cleared by TXRST (SC_ALTCTL[0]) and RXRST (SC_ALTCTL[1]), so don't fill this bit, TXRST (SC_ALTCTL[0]), and RXRST (SC_ALTCTL[1]) at the same time.<br>**Note3:** If SCEN (SC_CTL[0]) is not enabled, this field cannot be programmed. |
| [2] | **DACTEN** | **Deactivation Sequence Generator Enable Bit**<br>This bit enables SC controller to initiate the card by deactivation sequence.<br>0 = No effect.<br>1 = Deactivation sequence generator Enabled.<br>**Note1:** When the deactivation sequence completed, this bit will be cleared automatically and the INITIF (SC_INTSTS[8]) will be set to 1.<br>**Note2:** This field will be cleared by TXRST (SC_ALTCTL[0]) and RXRST (SC_ALTCTL[1]). So don't fill this bit, TXRST, and RXRST at the same time.<br>**Note3:** If SCEN (SC_CTL[0]) is not enabled, this field cannot be programmed. |
| [1] | **RXRST** | **Rx Software Reset**<br>When RXRST is set, all the bytes in the receiver buffer and Rx internal state machine will be cleared.<br>0 = No effect.<br>1 = Reset the Rx internal state machine and pointers.<br>**Note:** This bit will be auto cleared after reset is complete. |

| | | |
|---|---|---|
| [0] | **TXRST** | **TX Software Reset**<br>When TXRST is set, all the bytes in the transmit buffer and TX internal state machine will be cleared.<br>0 = No effect.<br>1 = Reset the TX internal state machine and pointers.<br>**Note:** This bit will be auto cleared after reset is complete. |

## SC Extend Guard Time Register (SC_EGT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SC_EGT x=0,1** | SCx_BA+0x00C | R/W | SC Extend 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** | **Extended Guard Time**<br>This field indicates the extended guard timer value.<br>**Note:** The counter is ETU base and the real extended guard time is EGT. |

## SC Receiver Buffer Time-out Register (SC_RXTOUT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SC_RXTOUT x=0,1** | SCx_BA+0x010 | R/W | SC Receive Buffer 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** |
| **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 (ETU Base)**<br>The time-out counter resets and starts counting whenever the RX buffer received a new data word. Once the counter decrease to 1 and no new data is received or CPU does not read data by reading SC_DAT buffer, a receiver time-out interrupt INT_RTMR will be generated (if RXTOIF (SC_INTEN[9]) = 1).<br>**Note1:** The counter unit is ETU based and the interval of time-out is RFTM + 0.5.<br>**Note2:** Filling all 0 to this field indicates to disable this function. |

## SC Clock Divider Control Register (SC_ETUCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SC_ETUCTL x=0,1** | SCx_BA+0x014 | R/W | SC ETU 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** |
| **CMPEN** | **Reserved** | | | **ETURDIV** | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ETURDIV** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15] | **CMPEN** | **Compensation Mode Enable Bit**<br>This bit enables clock compensation function. When this bit enabled, hardware will alternate between n clock cycles and n-1 clock cycles, where n is the value to be written into the ETURDIV.<br>0 = Compensation function Disabled.<br>1 = Compensation function Enabled. |
| [14:12] | **Reserved** | Reserved. |
| [11:0] | **ETURDIV** | **ETU Rate Divider**<br>The field indicates the clock rate divider.<br>The real ETU is ETURDIV + 1.<br>**Note:** Software can configure this field, but this field must be greater than 0x004. |

## SC Interrupt Control Register (SC_INTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SC_INTEN x=0,1** | SCx_BA+0x018 | 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**<br>This field is used for auto-convention error interrupt enable.<br>0 = Auto-convention error interrupt Disabled.<br>1 = Auto-convention error interrupt Enabled. |
| [9] | **RXTOIEN** | **Receiver Buffer Time-out Interrupt Enable Bit**<br>This field is used for receiver buffer time-out interrupt enable.<br>0 = Receiver buffer time-out interrupt Disabled.<br>1 = Receiver buffer time-out interrupt Enabled. |
| [8] | **INITIEN** | **Initial End Interrupt Enable Bit**<br>This field is used for activation (ACTEN (SC_ALTCTL[3]) = 1), deactivation (DACTEN (SC_ALTCTL[2]) = 1) and warm reset (WARSTEN (SC_ALTCTL[4]) sequence interrupt enable.<br>0 = Initial end interrupt Disabled.<br>1 = Initial end interrupt Enabled. |
| [7] | **CDIEN** | **Card Detect Interrupt Enable Bit**<br>This field is used for card detect interrupt enable. The card detect status is CINSERT (SC_STATUS[12]).<br>0 = Card detect interrupt Disabled.<br>1 = Card detect interrupt Enabled. |
| [6] | **BGTIEN** | **Block Guard Time Interrupt Enable Bit**<br>This field is used for block guard time interrupt enable.<br>0 = Block guard time Disabled.<br>1 = Block guard time Enabled. |
| [5] | **TMR2IEN** | **Timer2 Interrupt Enable Bit** |

| | | |
|---|---|---|
| | | This field is used for TMR2 interrupt enable.<br>0 = Timer2 interrupt Disabled.<br>1 = Timer2 interrupt Enabled. |
| [4] | **TMR1IEN** | **Timer1 Interrupt Enable Bit**<br>This field is used to enable the TMR1 interrupt.<br>0 = Timer1 interrupt Disabled.<br>1 = Timer1 interrupt Enabled. |
| [3] | **TMR0IEN** | **Timer0 Interrupt Enable Bit**<br>This field is used to enable TMR0 interrupt enable.<br>0 = Timer0 interrupt Disabled.<br>1 = Timer0 interrupt Enabled. |
| [2] | **TERRIEN** | **Transfer Error Interrupt Enable Bit**<br>This field is used for transfer error interrupt enable. The transfer error states is at SC_STATUS register which includes receiver break error BEF (SC_STATUS[6]), frame error FEF (SC_STATUS[5]), parity error PEF (SC_STATUS[4]), receiver buffer overflow error RXOV (SC_STATUS[0]), transmit buffer overflow error TXOV (SC_STATUS[8]), receiver retry over limit error RXOVERR (SC_STATUS[22]) and transmitter retry over limit error TXOVERR (SC_STATUS[30]).<br>0 = Transfer error interrupt Disabled.<br>1 = Transfer error interrupt Enabled. |
| [1] | **TBEIEN** | **Transmit Buffer Empty Interrupt Enable Bit**<br>This field is used for transmit buffer empty interrupt enable.<br>0 = Transmit buffer empty interrupt Disabled.<br>1 = Transmit buffer empty interrupt Enabled. |
| [0] | **RDAIEN** | **Receive Data Reach Interrupt Enable Bit**<br>This field is used for received data reaching trigger level RXTRGLV (SC_CTL[7:6]) interrupt enable.<br>0 = Receive data reach trigger level interrupt Disabled.<br>1 = Receive data reach trigger level interrupt Enabled. |

## SC Interrupt Status Register (SC_INTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SC_INTSTS x=0,1** | SCx_BA+0x01C | 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** | **RBTOIF** | **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 (Read Only)**<br>This field indicates auto convention sequence error. If the received TS at ATR state is neither 0x3B nor 0x3F, this bit will be set.<br>**Note:** This bit is read only, but it can be cleared by writing 1 to it. |
| [9] | **RBTOIF** | **Receiver Buffer Time-out Interrupt Status Flag (Read Only)**<br>This field is used for receiver buffer time-out interrupt status flag.<br>**Note:** This field is the status flag of receiver buffer time-out state. If software wants to clear this bit, software must read all receivers buffer remaining data by reading SC_DAT buffer. |
| [8] | **INITIF** | **Initial End Interrupt Status Flag (Read Only)**<br>This field is used for activation (ACTEN (SC_ALTCTL[3])), deactivation (DACTEN (SC_ALTCTL[2])) and warm reset (WARSTEN (SC_ALTCTL[4])) sequence interrupt status flag.<br>**Note:** This bit is read only, but it can be cleared by writing 1 to it. |
| [7] | **CDIF** | **Card Detect Interrupt Status Flag (Read Only)**<br>This field is used for card detect interrupt status flag. The card detect status is CINSERT (SC_STATUS[12])] and CREMOVE (SC_STATUS[11]).<br>**Note:** This field is the status flag of CINSERT (SC_STATUS[12]) or CREMOVE (SC_STATUS[11]). So if software wants to clear this bit, software must write 1 to this field. |
| [6] | **BGTIF** | **Block Guard Time Interrupt Status Flag (Read Only)**<br>This field is used for block guard time interrupt status flag.<br>**Note1:** This bit is valid when RXBGTEN (SC_ALTCTL[12]) is enabled.<br>**Note2:** This bit is read only, but it can be cleared by writing 1 to it. |
| [5] | **TMR2IF** | **Timer2 Interrupt Status Flag (Read Only)** |

| | | |
|---|---|---|
| | | This field is used for TMR2 interrupt status flag.<br>**Note:** This bit is read only, but it can be cleared by writing 1 to it. |
| [4] | **TMR1IF** | **Timer1 Interrupt Status Flag (Read Only)**<br>This field is used for TMR1 interrupt status flag.<br>**Note:** This bit is read only, but it can be cleared by writing 1 to it. |
| [3] | **TMR0IF** | **Timer0 Interrupt Status Flag (Read Only)**<br>This field is used for TMR0 interrupt status flag.<br>**Note:** This bit is read only, but it can be cleared by writing 1 to it. |
| [2] | **TERRIF** | **Transfer Error Interrupt Status Flag (Read Only)**<br>This field is used for transfer error interrupt status flag. The transfer error states is at SC_STATUS register which includes receiver break error BEF (SC_STATUS[6]), frame error FEF (SC_STATUS[5]), parity error PEF (SC_STATUS[4]) and receiver buffer overflow error RXOV (SC_STATUS[0]), transmit buffer overflow error TXOV (SC_STATUS[8]), receiver retry over limit error RXOVERR (SC_STATUS[22]) and transmitter retry over limit error TXOVERR (SC_STATUS[30]).<br>**Note:** This field is the status flag of BEF (SC_STATUS[6]), FEF (SC_STATUS[5]), PEF (SC_STATUS[4]), RXOV (SC_STATUS[0]), TXOV (SC_STATUS[8]), RXOVERR (SC_STATUS[22]) or TXOVERR (SC_STATUS[30]). So, if software wants to clear this bit, software must write 1 to each field. |
| [1] | **TBEIF** | **Transmit Buffer Empty Interrupt Status Flag (Read Only)**<br>This field is used for transmit buffer empty interrupt status flag.<br>**Note:** This field is the status flag of transmit buffer empty state. If software wants to clear this bit, software must write data to DAT (SC_DAT[7:0]) buffer and then this bit will be cleared automatically. |
| [0] | **RDAIF** | **Receive Data Reach Interrupt Status Flag (Read Only)**<br>This field is used for received data reaching trigger level RXTRGLV (SC_CTL[7:6]) interrupt status flag.<br>**Note:** This field is the status flag of received data reaching RXTRGLV (SC_CTL[7:6]). If software reads data from SC_DAT and receiver buffer data byte number is less than RXTRGLV (SC_CTL[7:6]), this bit will be cleared automatically. |

## SC Transfer Status Register (SC_STATUS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SC_STATUS x=0,1** | SCx_BA+0x020 | R/W | SC Status Register | 0x0000_0202 |

| 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)**<br>0 = This bit is cleared automatically when TX transfer is finished or the last byte transmission has completed.<br>1 = This bit is set by hardware when TX transfer is in active and the STOP bit of the last byte has been transmitted. |
| [30] | **TXOVERR** | **Transmitter over Retry Error (Read Only)**<br>This bit is set by hardware when transmitter re-transmits over retry number limitation.<br>**Note:** This bit is read only, but it can be cleared by writing 1 to it. |
| [29] | **TXRERR** | **Transmitter Retry Error (Read Only)**<br>This bit is set by hardware when transmitter re-transmits.<br>**Note1:** This bit is read only, but it can be cleared by writing 1 to it.<br>**Note2:** This bit is a flag and cannot generate any interrupt to CPU. |
| [28:26] | **Reserved** | Reserved. |
| [25:24] | **TXPOINT** | **Transmit Buffer Pointer Status Flag (Read Only)**<br>This field indicates the TX buffer pointer status flag. When CPU writes data into SC_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)**<br>This bit is set by hardware when RX transfer is in active.<br>This bit is cleared automatically when RX transfer is finished. |
| [22] | **RXOVERR** | **Receiver over Retry Error (Read Only)**<br>This bit is set by hardware when RX transfer error retry over retry number limit.<br>**Note1:** This bit is read only, but it can be cleared by writing 1 to it.<br>**Note2:** If CPU enables receiver retries function by setting RXRTY (SC_CTL[19]), the PEF (SC_STATUS[4]) flag will be ignored (hardware will not set PEF (SC_STATUS[4])). |

| | | |
|---|---|---|
| [21] | **RXRERR** | **Receiver Retry Error (Read Only)**<br>This bit is set by hardware when RX has any error and retries transfer.<br>**Note1:** This bit is read only, but it can be cleared by writing 1 to it.<br>**Note2** This bit is a flag and cannot generate any interrupt to CPU.<br>**Note3:** If CPU enables receiver retry function by setting RXRTYEN (SC_CTL[19]) , the PEF (SC_STATUS[4]) flag will be ignored (hardware will not set PEF (SC_STATUS[4])). |
| [20:18] | **Reserved** | Reserved. |
| [17:16] | **RXPOINT** | **Receiver Buffer Pointer Status Flag (Read Only)**<br>This field indicates the RX buffer pointer status flag. When SC receives one byte from external device, RXPOINT (SC_STATUS[17:16]) increases one. When one byte of RX buffer is read by CPU, RX_POINT (SC_STATUS[17:16]) decreases one. |
| [15:14] | **Reserved** | Reserved. |
| [13] | **CDPINSTS** | **Card Detect Status of SC_CD Pin Status (Read Only)**<br>This bit is the pin status flag of SC_CD.<br>0 = The SC_CD pin state at low.<br>1 = The SC_CD pin state at high. |
| [12] | **CINSERT** | **Card Detect Insert Status of SC_CD Pin (Read Only)**<br>This bit is set whenever card has been inserted.<br>0 = No effect.<br>1 = Card insert.<br>**Note1:** This bit is read only, but it can be cleared by writing “1” to it.<br>**Note2:** The card detect engine will start after SCEN (SC_CTL[0]) set. |
| [11] | **CREMOVE** | **Card Detect Removal Status of SC_CD Pin (Read Only)**<br>This bit is set whenever card has been removal.<br>0 = No effect.<br>1 = Card removed.<br>**Note1:** This bit is read only, but it can be cleared by writing “1” to it.<br>**Note2:** Card detect engine will start after SC_CEN (SC_CTL[0]) set. |
| [10] | **TXFULL** | **Transmit Buffer Full Status Flag (Read Only)**<br>This bit indicates TX buffer full or not. This bit is set when TX pointer is equal to 4, otherwise is cleared by hardware. |
| [9] | **TXEMPTY** | **Transmit Buffer Empty Status Flag (Read Only)**<br>This bit indicates TX buffer empty or not.<br>When the last byte of TX buffer has been transferred to Transmitter Shift Register, hardware sets this bit high. It will be cleared when writing data into DAT (SC_DAT[7:0]) (TX buffer not empty). |
| [8] | **TXOV** | **TX Overflow Error Interrupt Status Flag (Read Only)**<br>If TX buffer is full, an additional write to DAT (SC_DAT[7:0]) will cause this bit be set to “1” by hardware.<br>**Note:** This bit is read only, but it can be cleared by writing 1 to it. |
| [7] | **Reserved** | Reserved. |
| [6] | **BEF** | **Receiver Break Error Status Flag (Read Only)**<br>This bit is set to a 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 + stop bits).<br>**Note1:** This bit is read only, but it can be cleared by writing 1 to it. |

| | | |
|---|---|---|
| | | **Note2:** If CPU sets receiver retries function by setting RXRTYEN (SC_CTL[19]) , hardware will not set this flag. |
| [5] | **FEF** | **Receiver Frame Error Status Flag (Read Only)**<br>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 a logic 0).<br>**Note1:** This bit is read only, but it can be cleared by writing 1 to it.<br>**Note2:** If CPU sets receiver retries function by setting RXRTY_EN (SC_CTL[19]), hardware will not set this flag. |
| [4] | **PEF** | **Receiver Parity Error Status Flag (Read Only)**<br>This bit is set to logic 1 whenever the received character does not have a valid “parity bit”.<br>**Note1:** This bit is read only, but it can be cleared by writing 1 to it.<br>**Note2:** If CPU sets receiver retries function by setting RXRTY_EN (SC_CTL[19]), hardware will not set this flag. |
| [3] | **Reserved** | Reserved. |
| [2] | **RXFULL** | **Receiver Buffer Full Status Flag (Read Only)**<br>This bit indicates RX buffer full or not.<br>This bit is set when RX pointer is equal to 4, otherwise it is cleared by hardware. |
| [1] | **RXEMPTY** | **Receiver Buffer Empty Status Flag (Read Only)**<br>This bit indicates RX buffer empty or not.<br>When the last byte of Rx buffer has been read by CPU, hardware sets this bit high. It will be cleared when SC receives any new data. |
| [0] | **RXOV** | **RX Overflow Error Status Flag (Read Only)**<br>This bit is set when RX buffer overflow.<br>If the number of received bytes is greater than Rx Buffer size (4 bytes), this bit will be set.<br>**Note:** This bit is read only, but it can be cleared by writing 1 to it. |

## SC PIN Control State Register (SC_PINCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SC_PINCTL x=0,1** | SCx_BA+0x024 | R/W | SC Pin Control State Register | 0x0000_00x0 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | **SYNC** | **Reserved** | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | **RSTSTS** | **PWRSTS** | **DATSTS** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | **SCDOSTS** | **PWRINV** | **Reserved** | **SCDOUT** | **Reserved** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | **CLKKEEP** | **Reserved** | | | | **SCRST** | **PWREN** |

| Bits | Description | |
|---|---|---|
| [31] | **Reserved** | Reserved. |
| [30] | **SYNC** | **SYNC Flag Indicator**<br>Due to synchronization, software should check this bit when writing a new value to SC_PINCTL register.<br>0 = Synchronizing is completion, user can write new data to SC_PINCTL register.<br>1 = Last value is synchronizing.<br>**Note:** This bit is read only. |
| [29:19] | **Reserved** | Reserved. |
| [18] | **RSTSTS** | **SCRST Pin Signals**<br>This bit is the pin status of SC_RST.<br>0 = SC_RST pin is low.<br>1 = SC_RST pin is high.<br>**Note:** When SC is operated at activation, warm reset or deactivation mode, this bit will be changed automatically. This bit is not allowed to program when SC is operated at these modes. |
| [17] | **PWRSTS** | **SC_PWR Pin Signal**<br>This bit is the pin status of SC_PWR.<br>0 = SC_PWR pin to low.<br>1 = SC_PWR pin to high.<br>**Note:** When SC is operated at activation, warm reset or deactivation mode, this bit will be changed automatically. This bit is not allowed to program when SC is operated at these modes. |
| [16] | **DATSTS** | This bit is the pin status of SC_DAT.<br>0 = The SC_DAT pin is low. |

| | | |
|---|---|---|
| | | 1 = The SC_DAT pin is high. |
| [15:13] | **Reserved** | Reserved. |
| [12] | **SCDOSTS** | **SC Data Pin Output Status**<br>This bit is the pin status of SCDATAOUT.<br>0 = SCDATAOUT pin to low.<br>1 = SCDATAOUT pin to high. |
| [11] | **PWRINV** | **SC_PWR Pin Inverse**<br>This bit is used for inverse the SC_PWR pin.<br>There are four kinds of combination for SC_PWR pin setting by PWRINV (SC_PINCTL[11]) and PWREN (SC_PINCTL[0]). PWRINV (SC_PINCTL[11]) is bit 1 and PWREN (SC_PINCTL[0]) is bit 0 for SC_PWR Pin as high or low voltage selection.<br>00 = SC_PWR Pin is 0.<br>01 = SC_PWR Pin is 1.<br>10 = SC_PWR Pin is 1.<br>11 = SC_PWR Pin is 0.<br>**Note:** Software must select PWRINV (SC_PINCTL[11]) before Smart Card is enabled by SCEN (SC_CTL[0]). |
| [10] | **Reserved** | Reserved. |
| [9] | **SCDOUT** | **SC Data Output Pin**<br>This bit is the pin status of SCDATOUT but user can drive SCDATOUT pin to high or low by setting this bit.<br>0 = Drive SCDATOUT pin to low.<br>1 = Drive SCDATOUT pin to high.<br>**Note:** When SC is at activation, warm reset or deactivation mode, this bit will be changed automatically. So don't fill this field when SC is in these modes. |
| [8:7] | **Reserved** | Reserved. |
| [6] | **CLKKEEP** | **SC Clock Enable Bit**<br>0 = SC clock generation Disabled.<br>1 = SC clock always keeps free running.<br>**Note:** When operating in activation, warm reset or deactivation mode, this bit will be changed automatically. So don't fill this field when operating in these modes. |
| [5:2] | **Reserved** | Reserved. |
| [1] | **SCRST** | **SC_RST Pin Signal**<br>This bit is the pin status of SC_RST but user can drive SC_RST pin to high or low by setting this bit.<br>Write this field to drive SC_RST pin.<br>0 = Drive SC_RST pin to low.<br>1 = Drive SC_RST pin to high.<br>Read this field to get SC_RST pin status.<br>0 = SC_RST pin status is low.<br>1 = SC_RST pin status is high.<br>**Note:** When operating at activation, warm reset or deactivation mode, this bit will be changed automatically. So don't fill this field when operating in these modes. |
| [0] | **PWREN** | **SC_PWREN Pin Signal**<br>Software can set PWREN (SC_PINCTL[0]) and PWRINV (SC_PINCTL[11]) to decide SC_PWR pin is in high or low level. |

| | | |
|---|---|---|
| | | Write this field to drive SC_PWR pin.<br>Refer PWRINV (SC_PINCTL[11]) description for programming SC_PWR pin voltage level.<br>Read this field to get SC_PWR pin status.<br>0 = SC_PWR pin status is low.<br>1 = SC_PWR pin status is high.<br>**Note:** When operating at activation, warm reset or deactivation mode, this bit will be changed automatically. So don't fill this field when operating in these modes. |

### SC Timer Control Register 0 (SC_TMRCTL0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SC_TMRCTL0 x=0,1** | SCx_BA+0x028 | R/W | SC Internal Timer Control Register 0 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **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 | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27:24] | **OPMODE** | **Timer 0 Operation Mode Selection**<br>This field indicates the internal 24-bit timer operation selection.<br>Refer to **Error! Reference source not found.** for programming Timer0. |
| [23:0] | **CNT** | **Timer 0 Counter Value (ETU Base)**<br>This field indicates the internal timer operation values. |

### SC Timer Control Register 1 (SC_TMRCTL1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SC_TMRCTL1 x=0,1** | SCx_BA+0x02C | R/W | SC Internal Timer Control Register 1 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **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 | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27:24] | **OPMODE** | **Timer 1 Operation Mode Selection**<br>This field indicates the internal 8-bit timer operation selection.<br>Refer to **Error! Reference source not found.** for programming Timer1. |
| [23:8] | **Reserved** | Reserved. |
| [7:0] | **CNT** | **Timer 1 Counter Value (ETU Base)**<br>This field indicates the internal timer operation values. |

### SC Timer Control Register 2 (SC_TMRCTL2)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SC_TMRCTL2 x=0,1** | SCx_BA+0x030 | R/W | SC Internal Timer Control Register 2 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **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 | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27:24] | **OPMODE** | **Timer 2 Operation Mode Selection**<br>This field indicates the internal 8-bit timer operation selection.<br>Refer to **Error! Reference source not found.** for programming Timer2. |
| [23:8] | **Reserved** | Reserved. |
| [7:0] | **CNT** | **Timer 2 Counter Value (ETU Base)**<br>This field indicates the internal timer operation values. |

## SC UART Mode Control Register (SC_UARTCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SC_UARTCTL x=0,1 | SCx_BA+0x034 | 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**<br>0 = Even number of logic 1's are transmitted or check the data word and parity bits in receiving mode.<br>1 = Odd number of logic 1's are transmitted or check the data word and parity bits in receiving mode.<br>**Note:** This bit has effect only when PBOFF bit is '0'. |
| [6] | **PBOFF** | **Parity Bit Disable Control**<br>0 = Parity bit is generated or checked between the "last data word bit" and "stop bit" of the serial data.<br>1 = Parity bit is not generated (transmitting data) or checked (receiving data) during transfer.<br>**Note:** In smart card mode, this field must be '0' (default setting is with parity bit). |
| [5:4] | **WLS [1:0]** | **Word Length Selection**<br>00 = Word length is 8 bits.<br>01 = Word length is 7 bits.<br>10 = Word length is 6 bits.<br>11 = Word length is 5 bits.<br>**Note:** In smart card mode, this WLS must be '00'. |
| [3:1] | **Reserved** | Reserved. |
| [0] | **UARTEN** | **UART Mode Enable Bit**<br>0 = Smart Card mode.<br>1 = UART mode.<br>**Note1:** When operating in UART mode, user must set CONSEL (SC_CTL[5:4]) = 00 and AUTOCEN (SC_CTL[3]) = 0.<br>**Note2:** When operating in Smart Card mode, user must set UARTEN (SC_UARTCTL[0]) |

| Bits | Description | |
|---|---|---|
| | | = 0.<br>**Note3:** When UART is enabled, hardware will generate a reset to reset FIFO and internal state machine. |

### SC Timer Current Data Register 0 (SC_TMRDAT0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SC_TMRDAT0 x=0,1** | SCx_BA+0x038 | R | SC Timer Current Data Register 0 | 0x0000_07FF |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **CNT0** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **CNT0** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **CNT0** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **CNT0** | **Timer0 Current Data Value (Read Only)**<br>This field indicates the current count values of timer0. |

### SC Timer Current Data Register 1 (SC_TMRDAT1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SC_TMRDAT1 x=0,1** | SCx_BA+0x03C | R | SC Timer Current Data Register 1 | 0x0000_7F7F |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **CNT2** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **CNT1** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:8] | **CNT2** | **Timer2 Current Data Value (Read Only)**<br>This field indicates the current count values of timer2. |
| [7:0] | **CNT1** | **Timer1 Current Data Value (Read Only)**<br>This field indicates the current count values of timer1. |

## 5.18 I²C Synchronous Serial Interface Controller (I²C)

### 5.18.1 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.

Serial, 8-bit oriented bi-directional data transfers can be up to 100 KBit/s in Standard-mode, 400 KBit/s in the Fast-mode, or 3.4 Mbit/s in the High-speed mode. Only 100kbps and 400kbps modes are supported directly in this chip.

Data transfer is synchronized to SCL signal between a Master and a Slave with byte-by-byte basis. Each data byte is 8 bits long. There is one SCL clock pulse for each data bit with the MSB being transmitted first. 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).

### 5.18.2 Features

- Compatible with Philips I²C standard, support master mode
- Multi Master Operation.
- Clock stretching and wait state generation.
- Provide multi-byte transmit operation, up to 4 bytes can be transmitted in a single transfer
- Software programmable acknowledge bit.
- Arbitration lost interrupt, with automatic transfer cancellation.
- Start/Stop/Repeated Start/Acknowledge generation.
- Start/Stop/Repeated Start detection.
- Bus busy detection.
- Supports 7 bit addressing mode.
- Fully static synchronous design with one clock domain.
- Software mode I²C.

### 5.18.3 Block Diagram

The block diagram of I²C Serial Interface controller is shown as following.

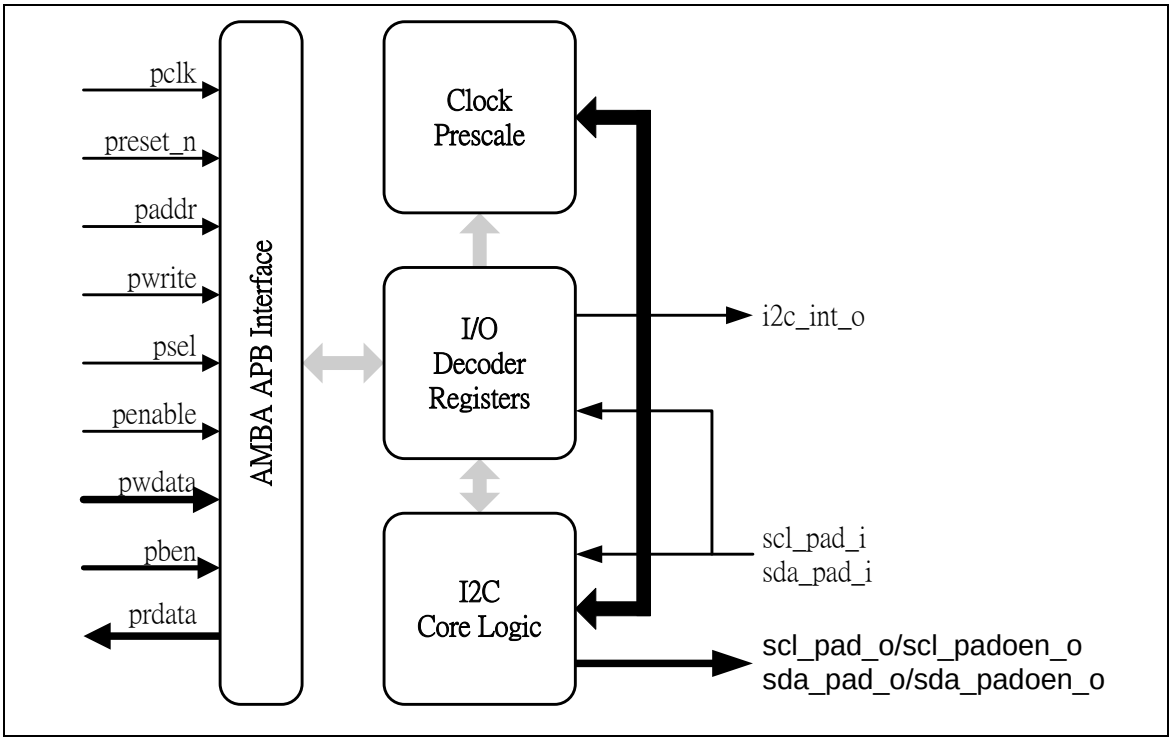



Figure 5.18-1 I²C Block Diagram

**NOTE1:** scl_pad_o and sda_pad_o are always tied to 1'b0.

**NOTE2:** scl_padoen_o and sda_padoen_o are active low signals.

### 5.18.4 Basic Configuration

Before using I²C functionality, it's necessary to configure I/O pins as the I2C function and enable I²C's clock.

Write 0x8 to MFP_GPG0 (SYS_GPG_MFPL[3:0]) and MFP_GPG1 (SYS_GPG_MFPL[7:4]) configures pin PG.0 and PG.1 to be I2C0_SCL and I2C0_SDA respectively.

Write 0x8 to MFP_GPG2 (SYS_GPG_MFPL[11:8]) and MFP_GPG3 (SYS_GPG_MFPL[15:12]) configures pin PG.2 and PG.3 to be I2C1_SCL and I2C1_SDA respectively.

Write 0x8 to MFP_GPH2 (SYS_GPH_MFPL[11:8]) and MFP_GPH3 (SYS_GPH_MFPL[15:12]) configures pin PH.2 and PH.3 to be I2C1_SCL and I2C1_SDA respectively.

Write 0x8 to MFP_GPI3 (SYS_GPI_MFPL[15:12]) and MFP_GPI4 (SYS_GPI_MFPL[19:16]) configures pin PI.3 and PI.4 to be I2C1_SCL and I2C1_SDA respectively.

Please note that configure PG.2, PH.2 and PI.3 to be I2C1_SCL functionality in the same time or configure PG.3, PH.3 and PI.4 to be I2C1_SDA functionality in the same time is prohibited.

To enable I²C's clock, please refer to register CLK_PCLKEN1. Set I2C0 (CLK_PCLKEN1[0]) high to enable I²C 0 clock while set I2C1 (CLK_PCLKEN1[1]) high to enable I²C 1 clock.

### 5.18.5 Functional Description

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 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 the following figure for more detailed I²C BUS Timing.

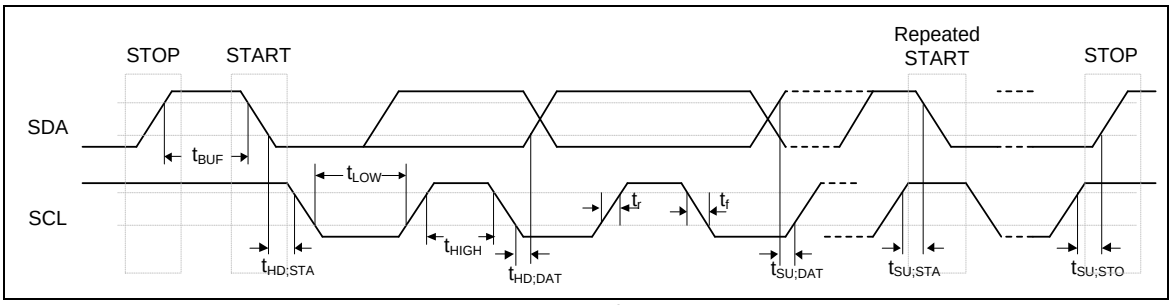



Figure 5.18-2 I²C Bus Timing

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. To enable this port, the bit ENS1 in I2CON should be set to '1'. The I²C hardware 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 as the SDA and SCL are open-drain pins

#### 5.18.5.1 *I²C Protocol*

The following figure shows the typical I²C protocol. Normally, a standard communication consists of four parts:

1). START or Repeated START signal generation

2). Slave address transfer

3). Data transfer

4). STOP signal generation

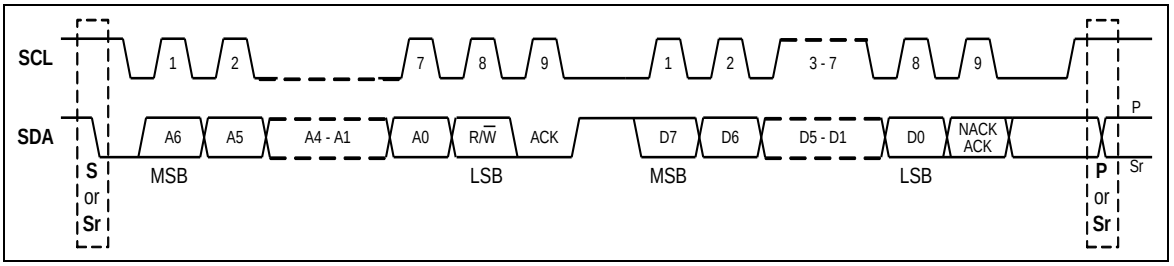
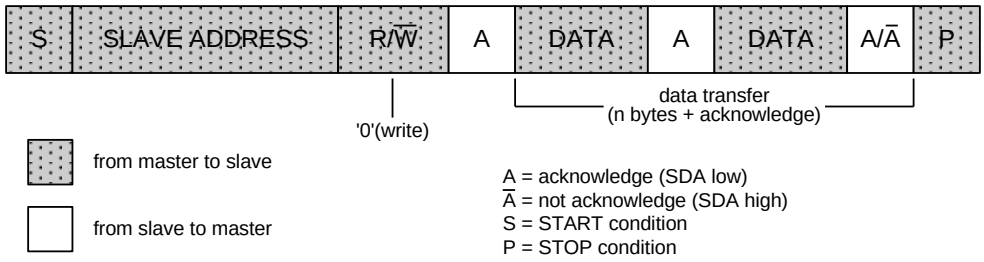
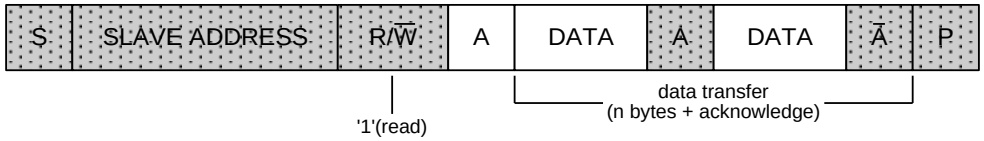



Figure 5.18-3 Data transfer on the I²C-bus




A master-transmitter addressing a slave receiver with a 7-bit address. The transfer direction is not changed




A master reads a slave immediately after the first byte (address)

5.18.5.2 *START or Repeated START signal*

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.

5.18.5.3 *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.

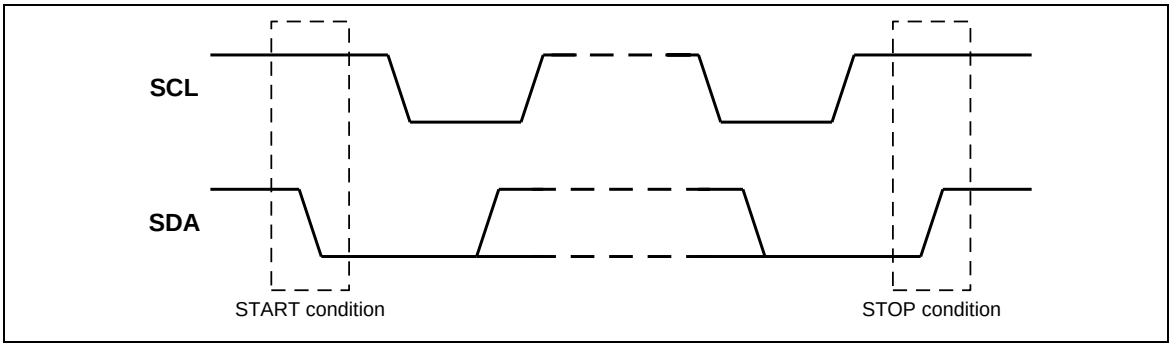
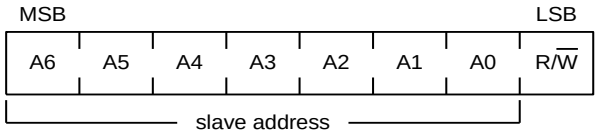



Figure 5.18-4 START and STOP conditions

5.18.5.4 *Slave Address Transfer*

The first byte of data transferred by the master immediately after the START signal is the slave address (SLA). This is a 7-bits calling address followed by a Read/Write (R/W) bit. The R/W bit signals of the slave indicate the data transfer direction. No two slaves in the system can have the same address. Only the slave with an address that matches the one transmitted by the master will respond by returning an acknowledge bit by pulling the SDA low at the 9th SCL clock cycle.

The core treats a Slave Address Transfer as any other write action. Store the slave device's address in the Transmit Register (TxR) and set the WRITE bit. The core will then transfer the slave address on the bus.




The first byte after the START procedure

5.18.5.5 *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 the 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.

To write data to a slave, store the data to be transmitted in the Transmit Register (TxR) and set the WRITE bit. To read data from a slave, set the READ bit. During a transfer the core set the I2C_TIP flag, indicating that a Transfer is In Progress. When the transfer is done the I2C_TIP flag is cleared, the IF flag set if enabled, then an interrupt generated. The Receive Register (RxR) contains valid data after the IF flag has been set. The software may issue a new write or read command when the I2C_TIP flag is cleared.

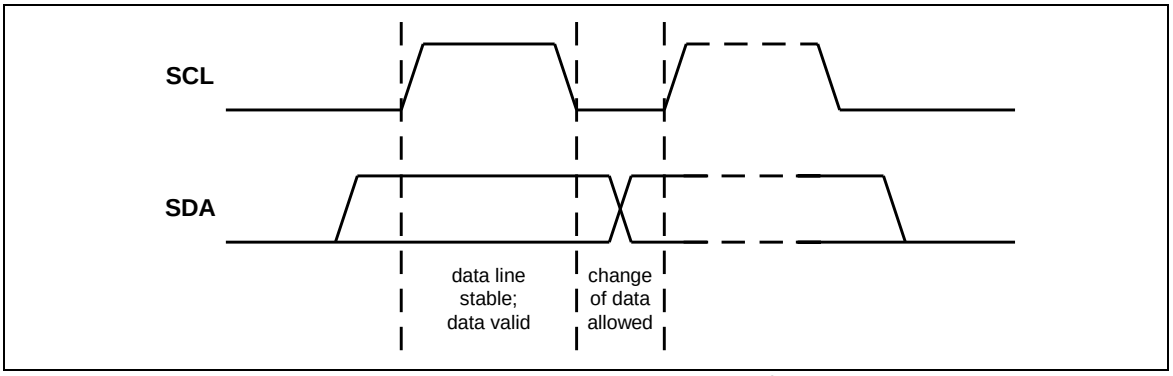



Figure 5.18-5 Bit transfer on the I$^2$C-bus

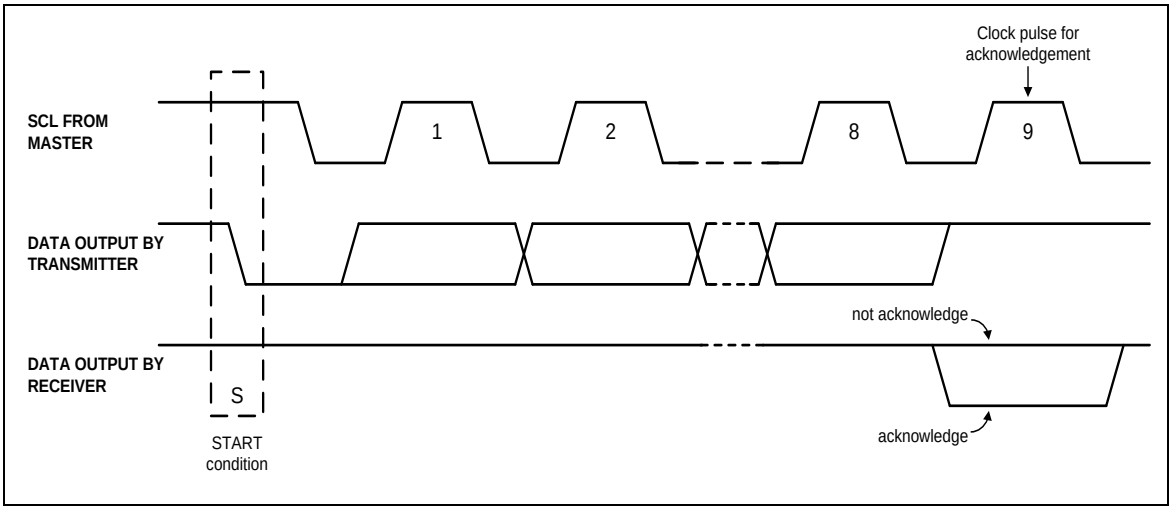



Figure 5.18-6 Acknowledge on the I$^2$C-bus

### 5.18.5.6 *Data transfer on the I$^2$C bus*

The following figure 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.

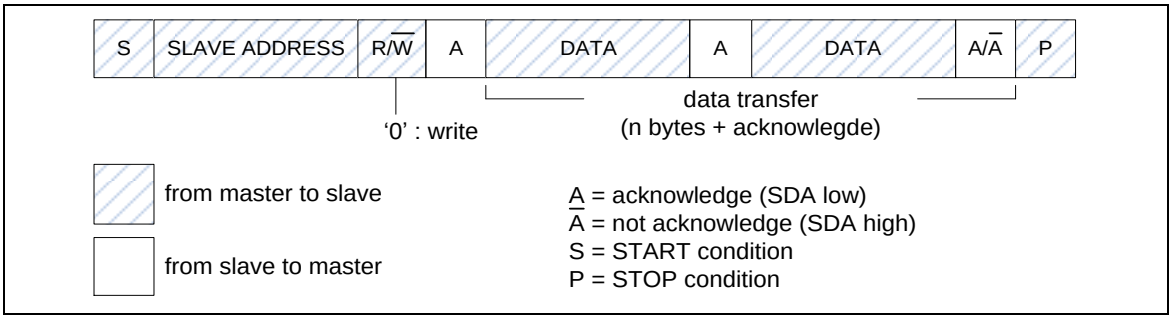



Figure 5.18-7 Master Transmits Data to Slave

The following figure 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.

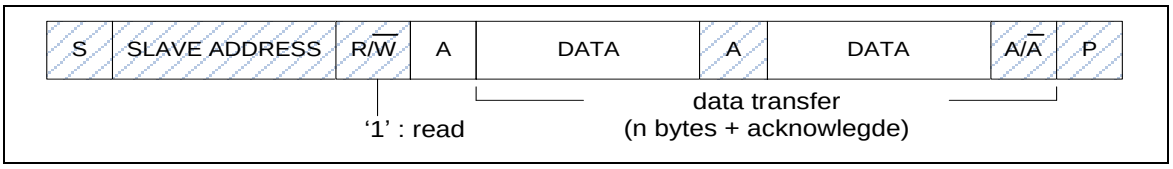



Figure 5.18-8 Master Reads Data from Slave

For the data transmission, the $I^2C$ core used 32-bit transmit buffer and provide multi-byte transmit function. Set CSR[Tx_NUM] to a value that you want to transmit. $I^2C$ core will always issue a transfer from the highest byte first. For example, if CSR[Tx_NUM] = 0x3, Tx[31:24] will be transmitted first, then Tx[23:16], and so on.

In case of a data transfer, all bits will be treated as data.

In case of a slave address transfer, the first 7 bits will be treated as 7-bit address and the LSB represent the R/W bit. In this case, LSB = 1, reading from slave and LSB = 0, writing to slave.

### 5.18.5.7 *I2C Programming Examples*

**Example 1**

Write 1 byte of data to a slave (using multi-byte transmit mode).

Slave address = 0x51 (7b'1010001)

Data to write = 0xAC

**$I^2C$ Sequence:**

generate start command

write slave address + write bit

receive acknowledge from slave

write data

receive acknowledge from slave

generate stop command

**Commands:**

Write a value into DIVIDER to determine the frequency of serial clock.

Set Tx_NUM = 0x1 and set I2C_EN = 1 to enable $I^2C$ core.

Write 0xA2 (address + write bit) to Transmit Register (TxR[15:8]) and 0xAC to TxR[7:0].

Set START bit and WRITE bit.

— Wait for interrupt or I2C_TIP flag to negate —

Read I2C_RxACK bit from CSR Register, it should be '0'.

Set Tx_NUM = 0x0.

Set STOP bit.

— Wait for interrupt or I2C_TIP flag to negate —

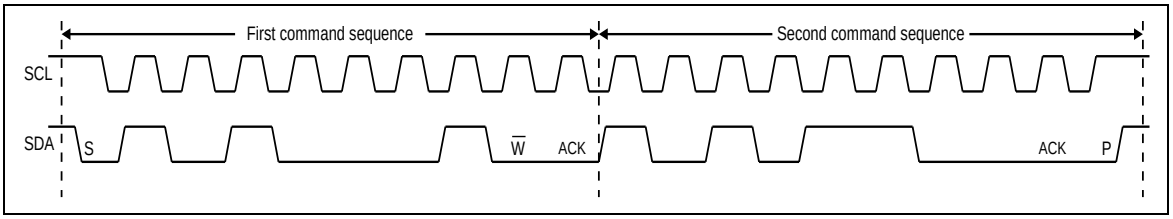



Figure 5.18-9 Write 1 byte of data to a slave

**NOTE:** Please note that the time for the Interrupt Service Routine is not shown here. It is assumed that the ISR is much faster then the $I^2C$ cycle time, and therefore not visible.

**Example 2**

Read a byte of data from an $I^2C$ memory device (using single byte transfer mode).

Slave address = 0x4E (7'b1001110)

Memory location to read from = 0x20

**$I^2C$ sequence:**

generate start signal

write slave address + write bit, then receive acknowledge from slave

write memory location, then receive acknowledge from slave

generate repeated start signal

write slave address + read bit, then receive acknowledge from slave

read byte from slave

write not acknowledge (NACK) to slave, indicating end of transfer

generate stop signal

**Commands:**

Write a value into DIVIDER to determine the frequency of serial clock.

Set Tx_NUM = 0x0 and set I2C_EN = 1 to enable $I^2C$ core.

Write 0x9C (address + write bit) to TxR[7:0], set START bit and WRITE bit.

— Wait for interrupt or I2C_TIP flag to negate —

Read I2C_RxACK bit from CSR Register, it should be '0'.

Write 0x20 to TxR[7:0], set WRITE bit.

— Wait for interrupt or I2C_TIP flag to negate —

Read I2C_RxACK bit from CSR Register, it should be '0'.

Write 0x9D (address + read bit) to TxR[7:0], set START bit, set WRITE bit.

— Wait for interrupt or I2C_TIP flag to negate —

Set READ bit, set ACK to '1' (NACK), set STOP bit.

Read out received data from RxR, it will put on RxR[7:0].

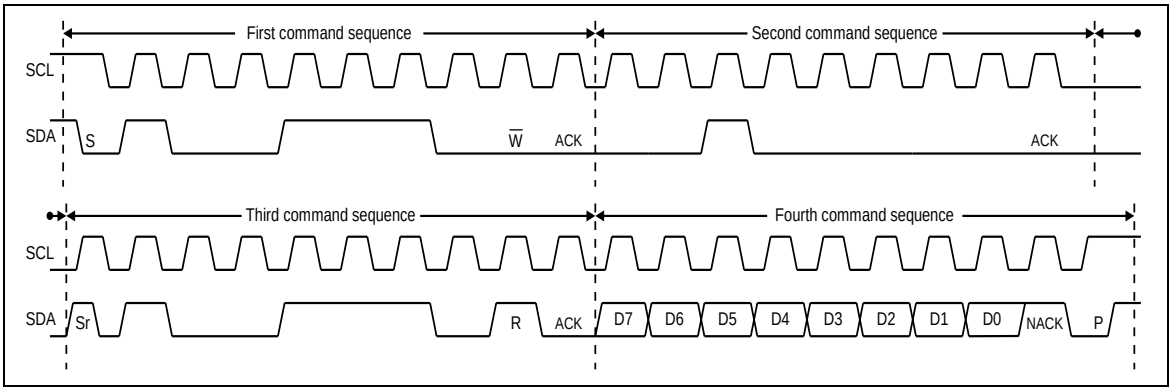



Figure 5.18-10 Read a byte of data

**NOTE:** Please note that the time for the Interrupt Service Routine is not shown here. It is assumed that the ISR is much faster then the $I^2C$ cycle time, and therefore not visible.

### 5.18.5.8 *Software $I^2C$ Operation*

The software $I^2C$ function contains 3 registers for software to control the output enable of pad actually. The implementation of software $I^2C$ is shown bellow.

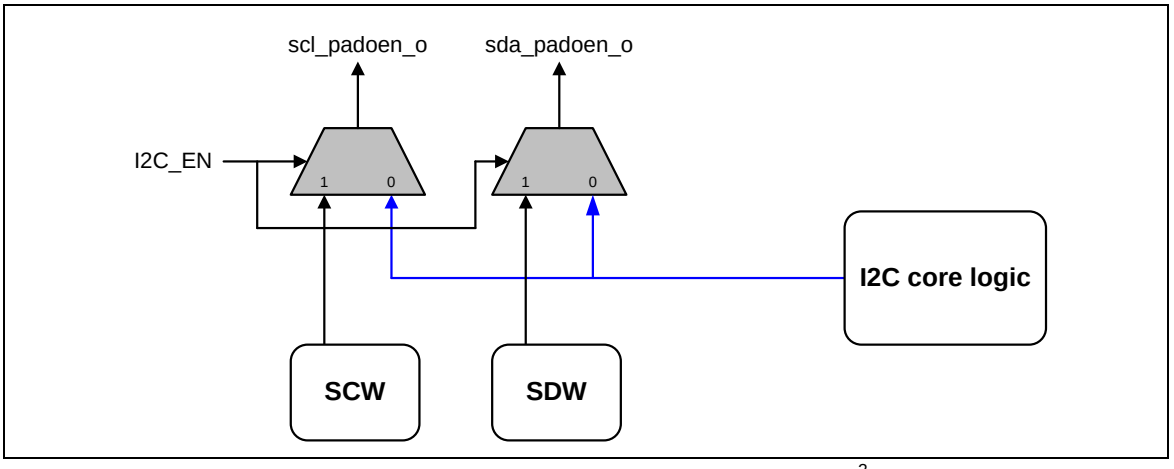



Figure 5.18-11 Implementation of Software $I^2C$

The other three registers – SCR and SDR just represent the status of input port - scl_pad_i and

sda_pad_i.

Software can read/write this register at any time, but the output enable – scl_padoen_o and sda_padoen_o are controlled by software only when I2C_EN = 0.

### 5.18.6 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **$I^2C$ Base Address:**<br>**I2C0_BA = 0xB800_6000**<br>**I2C1_BA = 0xB800_6100** | | | | |
| **I2Cn_CSR**<br>**n=0,1** | I2Cn_BA+0x000 | R/W | I2C n Control and Status Register | 0x0000_0000 |
| **I2Cn_DIVIDER**<br>**n=0,1** | I2Cn_BA+0x004 | R/W | I2C n Clock Prescale Register | 0x0000_0000 |
| **I2Cn_CMDR**<br>**n=0,1** | I2Cn_BA+0x008 | R/W | I2C n Command Register | 0x0000_0000 |
| **I2Cn_SWR**<br>**n=0,1** | I2Cn_BA+0x00C | R/W | I2C n Software Mode Control Register | 0x0000_003F |
| **I2Cn_RXR**<br>**n=0,1** | I2Cn_BA+0x010 | R | I2C n Data Receive Register | 0x0000_0000 |
| **I2Cn_TXR**<br>**n=0,1** | I2Cn_BA+0x014 | R/W | I2C n Data Transmit Register | 0x0000_0000 |

**NOTE:** The reset value of SWR is 0x3F only when SCR, SDR and SER are connected to pull high resistor.

### 5.18.7 Register Description

## I2C n Control and Status Register (I2Cn_CSR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2Cn_CSR n=0,1** | I2Cn_BA+0x000 | R/W | I2C n 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** | | | | **I2C_RxACK** | **I2C_BUSY** | **I2C_AL** | **I2C_TIP** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | **Tx_NUM** | | **Reserved** | **IF** | **IE** | **I2C_EN** |

| Bits | Description | |
|---|---|---|
| [31:12] | **Reserved** | Reserved. |
| [11] | **I2C_RxACK** | **Received Acknowledge From Slave (Read Only)**<br>This flag represents acknowledge from the addressed slave.<br>0 = Acknowledge received (ACK).<br>1 = Not acknowledge received (NACK). |
| [10] | **I2C_BUSY** | **$I^2C$ Bus Busy (Read Only)**<br>0 = After STOP signal detected.<br>1 = After START signal detected. |
| [9] | **I2C_AL** | **Arbitration Lost (Read Only)**<br>This bit is set when the $I^2C$ core lost arbitration. Arbitration is lost when:<br>A STOP signal is detected, but no requested.<br>The master drives SDA high, but SDA is low. |
| [8] | **I2C_TIP** | **Transfer in Progress (Read Only)**<br>0 = Transfer complete.<br>1 = Transferring data.<br>**NOTE:** When a transfer is in progress, you will not allow writing to any register of the $I^2C$ master core except SWR. |
| [5:4] | **Tx_NUM** | **Transmit Byte Counts**<br>These two bits represent how many bytes are remained to transmit. When a byte has been transmitted, the Tx_NUM will decrease 1 until all bytes are transmitted (Tx_NUM = 0x0) or NACK received from slave. Then the interrupt signal will assert if IE was set.<br>0x0 = Only one byte is left for transmission.<br>0x1 = Two bytes are left to for transmission.<br>0x2 = Three bytes are left for transmission.<br>0x3 = Four bytes are left for transmission. |

| | | |
|---|---|---|
| [3] | **Reserved** | Reserved. |
| [2] | **IF** | **Interrupt Flag**<br>The Interrupt Flag is set when:<br>Transfer has been completed.<br>Transfer has not been completed, but slave responded NACK (in multi-byte transmit mode).<br>Arbitration is lost.<br>**NOTE:** This bit is read only, but can be cleared by writing 1 to this bit. |
| [1] | **IE** | **Interrupt Enable**<br>0 = Disable $I^2C$ Interrupt.<br>1 = Enable $I^2C$ Interrupt. |
| [0] | **I2C_EN** | **$I^2C$ Core Enable**<br>0 = Disable $I^2C$ core, serial bus outputs are controlled by SDW/SCW.<br>1 = Enable $I^2C$ core, serial bus outputs are controlled by $I^2C$ core. |

## I2C n Prescale Register (I2Cn_DIVIDER)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2Cn_DIVIDER n=0,1** | I2Cn_BA+0x004 | R/W | I2C n 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** |
| **DIVIDER[15:8]** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **DIVIDER[7:0]** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [15:0] | **DIVIDER** | **Clock Prescale Register**<br>It is used to prescale the SCL clock line. Due to the structure of the $I^2C$ interface, the core uses a 5*SCL clock internally. The prescale register must be programmed to this 5*SCL frequency (minus 1). Change the value of the prescale register only when the "I2C_EN" bit is cleared.<br>Example: pclk = 32MHz, desired SCL = 100KHz.<br>$$prescale = \frac{32MHz}{5*100KHz} - 1 = 63(dec) = 3F(hex)$$ |

## I2C n Command Register (I2Cn_CMDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2Cn_CMDR n=0,1** | I2Cn_BA+0x008 | R/W | I2C n 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** | | | **START** | **STOP** | **READ** | **WRITE** | **ACK** |

**NOTE:** Software can write this register only when I2C_EN = 1.

| Bits | Description | |
|---|---|---|
| [31:5] | **Reserved** | Reserved. |
| [4] | **START** | **Generate Start Condition**<br>Generate (repeated) start condition on $I^2C$ bus. |
| [3] | **STOP** | **Generate Stop Condition**<br>Generate stop condition on $I^2C$ bus. |
| [2] | **READ** | **Read Data From Slave**<br>Retrieve data from slave. |
| [1] | **WRITE** | **Write Data to Slave**<br>Transmit data to slave. |
| [0] | **ACK** | **Send Acknowledge to Slave**<br>When $I^2C$ behaves as a receiver, sent ACK (ACK = '0') or NACK (ACK = '1') to slave. |

**NOTE:** The START, STOP, READ and WRITE bits are cleared automatically while transfer finished. READ and WRITE cannot be set concurrently.

## I2C n Software Mode Register (I2Cn_SWR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2Cn_SWR n=0,1** | I2Cn_BA+0x00C | R/W | I2C n Software Mode Control 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** | | | **SDR** | **SCR** | **Reserved** | **SDW** | **SCW** |

| Bits | Description | |
|---|---|---|
| [31:5] | **Reserved** | Reserved. |
| [4] | **SDR** | **Serial Interface SDA Status (Read Only)**<br>0 = SDA is Low.<br>1 = SDA is High. |
| [3] | **SCR** | **Serial Interface SCK Status (Read Only)**<br>0 = SCL is Low.<br>1 = SCL is High. |
| [2] | **Reserved** | Reserved. |
| [1] | **SDW** | **Serial Interface SDA Output Control**<br>0 = SDA pin is driven Low.<br>1 = SDA pin is tri-state. |
| [0] | **SCW** | **Serial Interface SCK Output Control**<br>0 = SCL pin is driven Low.<br>1 = SCL pin is tri-state. |

**NOTE:** This register is used as software mode of $I^2C$. Software can read/write this register no matter I2C_EN is 0 or 1. But SCL and SDA are controlled by software only when I2C_EN = 0.

## I2C n Data Receive Register (I2Cn_RXR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2Cn_RXR n=0,1** | I2Cn_BA+0x010 | R | I2C n Data Receive 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** |
| **Rx** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **Rx** | **Data Receive Register**<br>The last byte received via $I^2C$ bus will put on this register. The $I^2C$ core only used 8-bit receive buffer. |

## I2C n Data Transmit Register (I2Cn_TXR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2Cn_TXR n=0,1** | I2Cn_BA+0x014 | R/W | I2C n 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**<br>32-bit transmit buffer. Refer to Section of “Data transfer on the $I^2C$ bus” for more detail information. |

## 5.19 SPI Interface Controller (SPI)

### 5.19.1 Overview

The SPI is a synchronous serial interface performs a serial-to-parallel conversion on data characters received from the peripheral, and a parallel-to-serial conversion on data characters received from CPU. This interface can drive up to 2 external peripherals and is seen as the master.

### 5.19.2 Features

- Support master mode
- Full duplex synchronous serial data transfer
- Variable length of transfer word up to 32 bits
- Provide burst mode operation, transmit/receive can be executed up to four times in one transfer
- MSB or LSB first data transfer
- Rx and Tx on both rising or falling edge of serial clock independently
- Support 2 slave/device select lines
- Support dual IO and quad mode for SPI flash access

### 5.19.3 Block Diagram

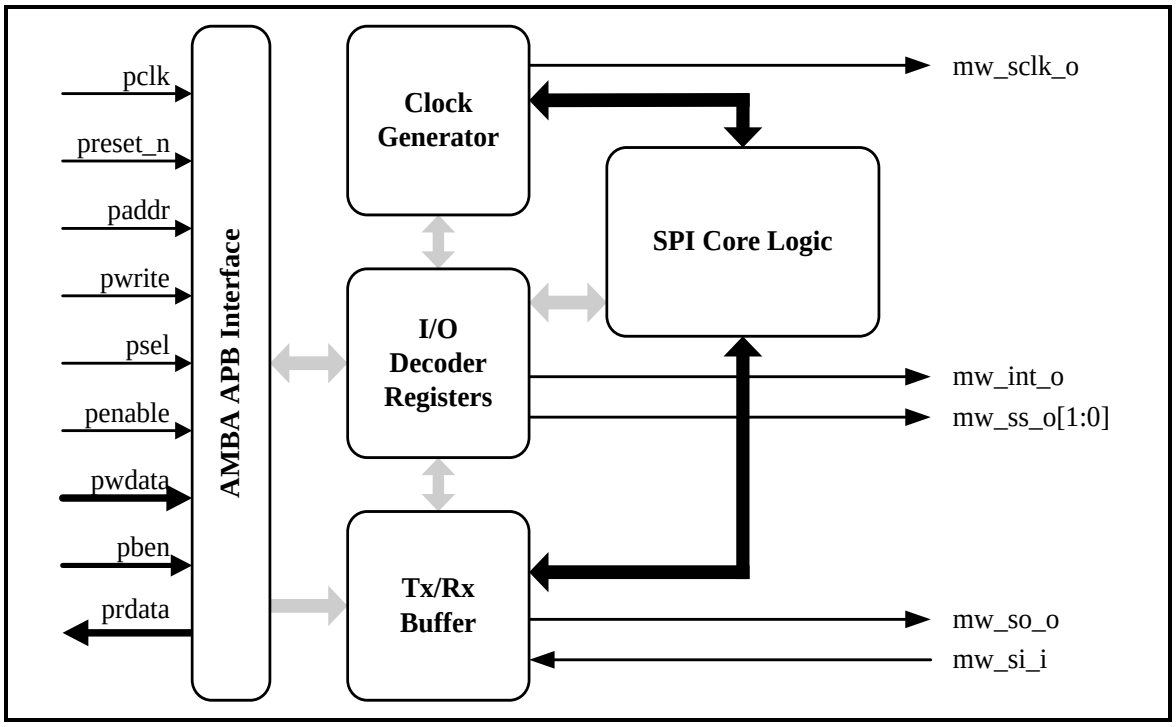



Figure 5.19-1 SPI Block Diagram

Pin descriptions:

mw_sclk_o: SPI serial clock output pin.

mw_int_o: SPI interrupt signal output.

mw_ss_o: SPI slave/device select signal output.

mw_so_o: SPI serial data output pin (to slave device).

mw_si_i: SPI serial data input pin (from slave device).

### 5.19.4 Basic Configuration

Before using SPI functionality, it's necessary to configure I/O pins as the SPI function and enable SPI's clock.

Write 0xB to MFP_GPB6 (SYS_GPB_MFPL[27:24]), MFP_GPB7 (SYS_GPB_MFPL[31:28]), MFP_GPB8 (SYS_GPB_MFPH[3:0]), MFP_GPB9 (SYS_GPB_MFPH[7:4]), MFP_GPB10 (SYS_GPB_MFPH[11:8]) and MFP_GPB11 (SYS_GPB_MFPH[15:12]) configures pin PB.6, PB.7, PB.8, PB.9, PB.10 and PB.11 to be SPI0_SS0, SPI0_CLK, SPI0_DATA0, SPI0_DATA1, SPI0_DATA2 and SPI0_DATA3 resepctively.

Write 0xB to MFP_GPB12 (SYS_GPB_MFPH[19:16]), MFP_GPB13 (SYS_GPB_MFPH[23:20]), MFP_GPB14 (SYS_GPB_MFPH[27:24]), MFP_GPB15 (SYS_GPB_MFPH[31:28]), MFP_GPG4 (SYS_GPG_MFPL[19:16]) and MFP_GPG5 (SYS_GPG_MFPL[23:20]) configures pin PB.12, PB.13, PB.14, PB.15, PG.4 and PG.5 to be SPI1_SS0, SPI1_CLK, SPI1_DATA0, SPI1_DATA1, SPI1_DATA2 and SPI1_DATA3 resepctively.

Write 0xB to MFP_GPI5 (SYS_GPI_MFPL[23:20]), MFP_GPI6 (SYS_GPI_MFPL[27:24]), MFP_GPI7 (SYS_GPI_MFPL[31:28]) and MFP_GPI8 (SYS_GPI_MFPH[3:0]) configures pin PI.5, PI.6, PI.7 and PI.8 to be SPI1_SS0, SPI1_CLK, SPI1_DATA0 and SPI1_DATA1 resepctively.

Please note that configure different pins to be same functionality is prohibited. For example, please don't configure PB.14 and PI.6 to be SPI1_DATA0 functionality in the same time.

To enable SPI's clock, please refer to register CLK_PCLKEN1. Set SPI0 (CLK_PCLKEN1[4]) high to enable SPI0 clock while set SPI1 (CLK_PCLKEN1[5]) high to enable SPI1 clock.

### 5.19.5 Function Description

This SPI controller can be set as Master to communicate with the off-chip SPI slave device.

It can generate an interrupt signal when data transfer is finished and can be cleared by writing 1 to the interrupt flag.

The active level of device/slave select signal can be chosen to low active or high active, which depends on the peripheral it's connected.

Writing a divisor into DIVIDER register can program the frequency of serial clock output. This master core contains four 32-bit transmit/receive buffers, and can provide burst mode operation. The maximum bits can be transmitted/received is 32 bits, and can transmit/receive data up to four times successive.

It also supports Dual IO and Quad IO transfer mode.

5.19.5.1 *Operation Setting Description*

**Clock Polarity**

The CLK_POL bit (CNTRL[31]) defines the SPI clock idle state. If CLK_POL = 1, the output SPI clock is idle at high state; if CLK_POL = 0, it is idle at low state.

**Transmit/Receive Number**

This Tx_NUM (CNTRL[9:8] specifies how many transmit/receive numbers should be executed in one transfer.

**Transmit/Receive Bit Length**

The bit length of a transaction word is defined in Tx_BIT_LEN bit field (CNTRL[7:3]) and can be configured up to 32-bit length in a transaction word for transmitting and receiving.

**LSB/MSB First**

The LSB bit (CNTRL[10]) defines the bit transfer sequence in a transaction. If the LSB bit is set to 1, the transfer sequence is LSB first. The bit 0 will be transferred firstly. If the LSB bit is cleared to 0, the transfer sequence is MSB first.

**Edge**

The TX_NEG bit (CNTRL[2]) defines the data transmitted out either on negative edge or on positive edge of SPI clock.

The Rx_NEG bit (CNTRL[1]) defines the data received either on negative edge or on positive edge of SPI clock.

**Note:** The settings of TX_NEG and RX_NEG are mutual exclusive. In other words, do not transmit and receive data at the same clock edge.

**Suspend Interval**

These four bits provide the configuration of suspend interval between two successive transmit/receive in a transfer. The default value is 0x0. When CNTRL[Tx_NUM] = 00, setting this field has no effect on transfer. The desired interval is obtained according to the following equation (from the last falling edge of current sclk to the first rising edge of next sclk).

(CNTRL[SLEEP] + 2)*period of SCLK

**Clock Divider**

The value of DIVIDER (DIVIDER[15:0]) is the frequency divider of the system clock pclk to generate the serial clock on the output mw_sclk_o. The desired frequency is obtained according to the following equation:

$$f_{sclk} = \frac{f_{pclk}}{(DIVIDER+1)*2}$$

**Auto Select**

The auto select function is controlled in ASS bit (SSR[3]).

If this bit is cleared, slave select signals are asserted and de-asserted by setting and clearing related bits in SSR register.

If this bit is set, mw_ss_o signals are generated automatically. It means that device/slave select signal, which is set in SSR register is asserted by the SPI controller when transmit/receive is started by setting CNTRL[GO_BUSY], and is de-asserted after every transmit/receive is finished.

**Slave Select**

The SPI controller can drive up to two off-chip slave devices on SS[1:0] (SSR[1:0]) through the slave select output pins mw_ss_o[1:0]. The active state of slave select signal can be programmed to low or high active in SS_LVL bit (SSR[2]). The selection of slave select conditions depends on what type of peripheral slave/master device is connected.

5.19.5.2 *SPI Timing Diagram*

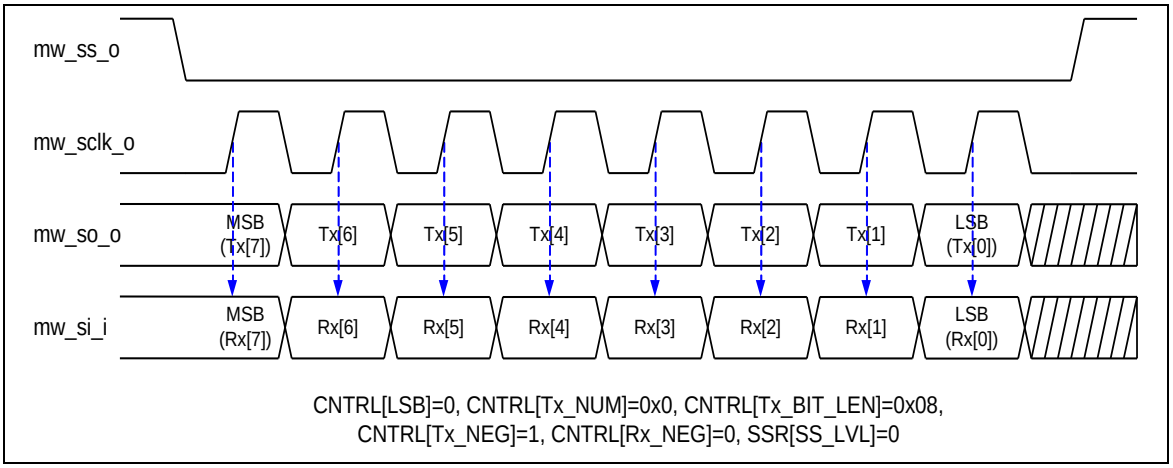



Figure 5.19-2 Normal SPI Timing

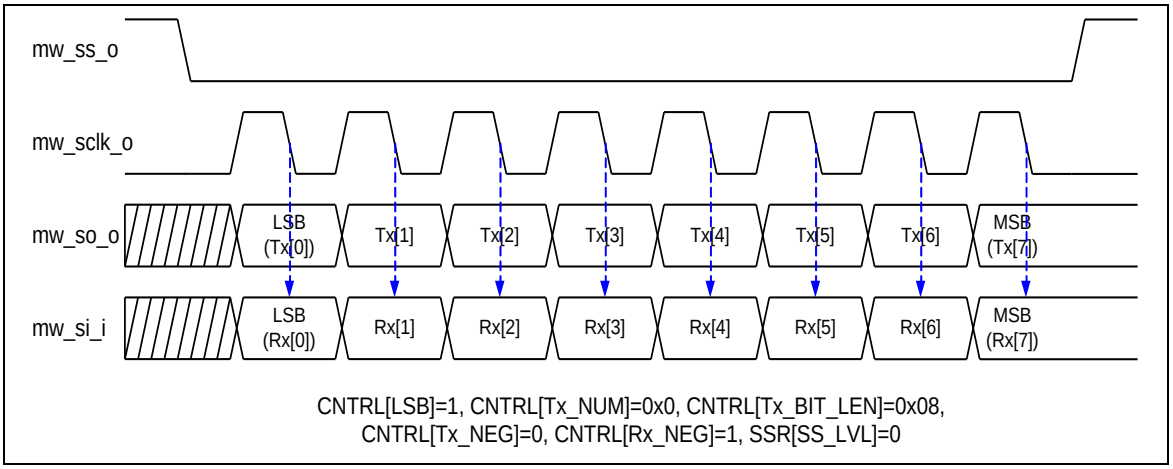



Figure 5.19-3 Alternate Phase SCLK Clock Timing

5.19.5.3 *Dual and Quad IO Mode*

This SPI controller also supports dual and quad IO transfer for SPI Flash when set the DUALM or QUAD bit (CNTRL[21:20]) to 1. The DIR_2QM bit (CNTRL[20]) is used to define the direction of the transfer data. When set the DIR_2QM bit to 1, the controller will send the data to external device. When the DIR_2QM bit set 0, the controller will read the data from the external device. This function supports 8, 16, 24, and 32-bits of bit length.

If both the DUALM (or QUADM) and DIR_2QM bits are set as 1, the mw_so_o[0] is the lowest bit data output and the mw_so_o[1] (or mw_so_o[3]) will be set as the uppermost bit data output. If the DUALM (or QUADM) is set as 1 and DIR_2QM is set as 0, both the mw_si_i will be set as data input ports. The following Figure shows the dual IO function timing diagram. The Quad mode timing is similar to the dual IO mode.

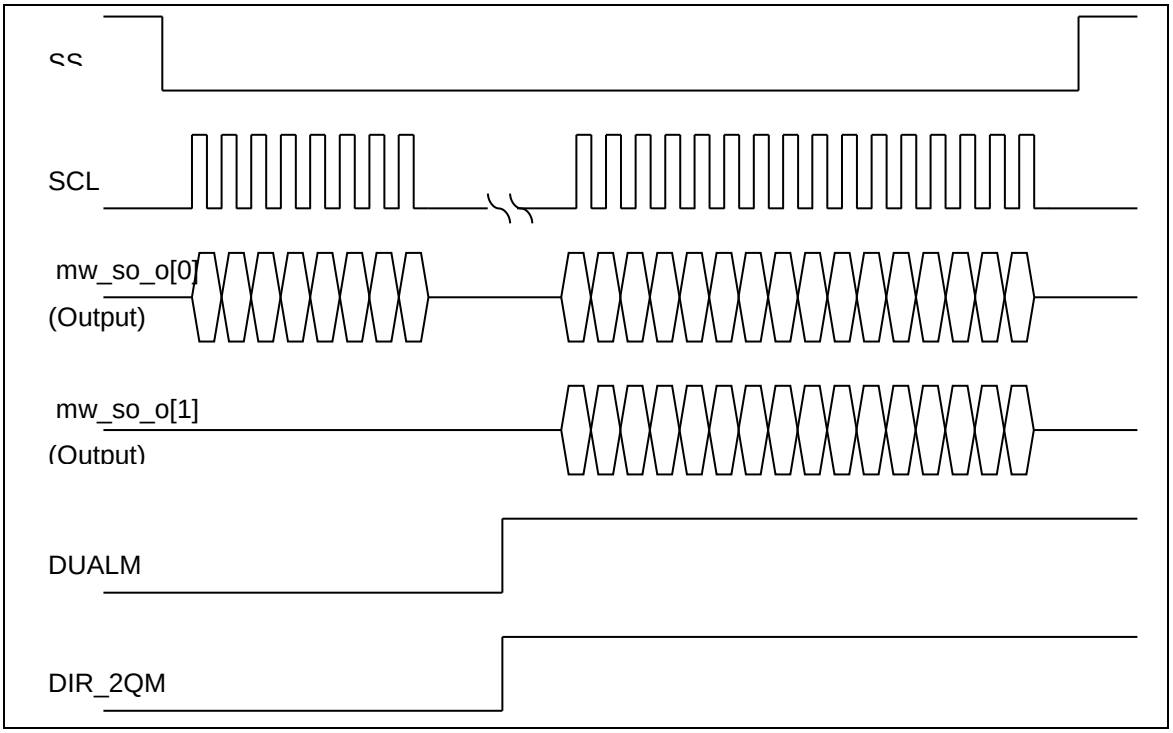



Figure 5.19-4 Dual-IO output Sequence

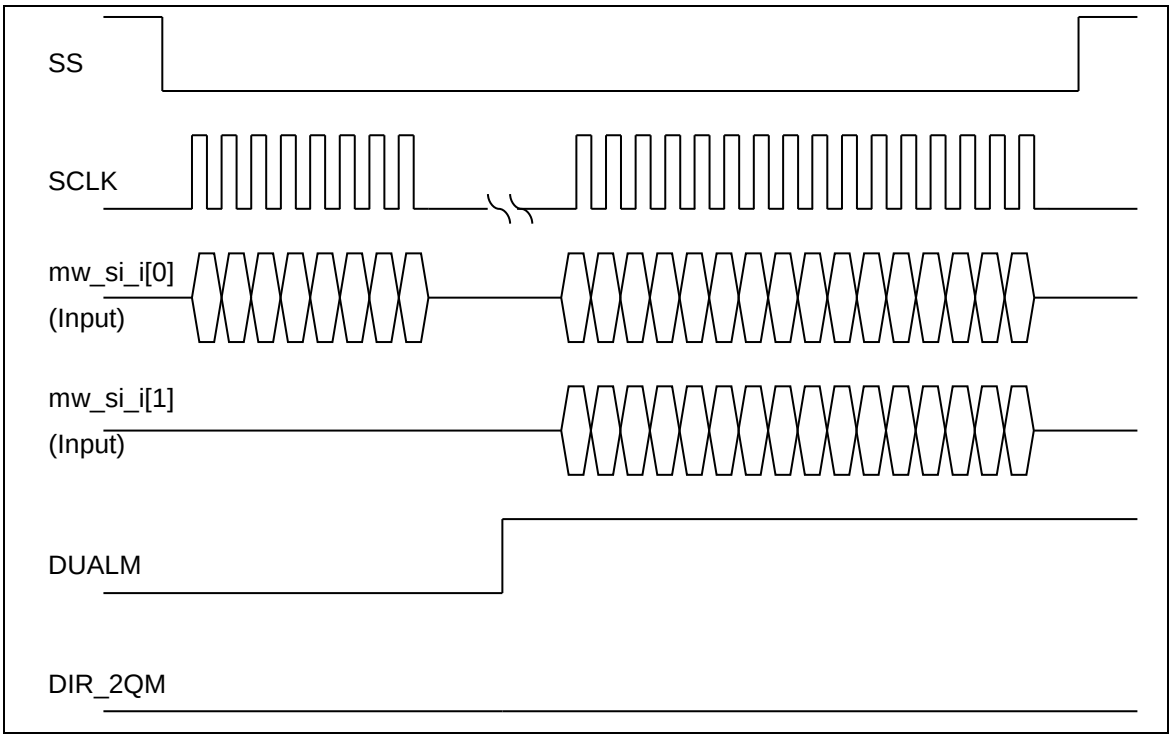



Figure 5.19-5 Dual-IO Input Sequence

5.19.5.4 *SPI Programming Example*

If a device with following specifications:

Data bit latches on positive edge of serial clock

Data bit drives on negative edge of serial clock

Data is transferred with the MSB first

Only one byte transmits/receives in a transfer

Chip select signal is active low

Do following actions basically (Should refer to the specification of device for the detailed steps):

Write a divisor into DIVIDER to determine the frequency of serial clock.

Write in SSR, set ASS = 0, SS_LVL = 0 and SSR[0] or SSR[1] to 1 to activate the device you want to access.

When transmit (write) data to device:

Write the data you want to transmit into Tx0[7:0].

When receive (read) data from device:

Write 0xFFFFFFFF into Tx0.

Write in CNTRL, set Rx_NEG = 0, Tx_NEG = 1, Tx_BIT_LEN = 0x08, Tx_NUM = 0x0, LSB = 0, SLEEP = 0x0 and GO_BUSY = 1 to start the transfer.

— Wait for interrupt (if IE = 1) or polling the GO_BUSY bit until it turns to 0 —

Read out the received data from Rx0.

Go to 3) to continue data transfer or set SSR[0] or SSR[1] to 0 to inactivate the device.

### 5.19.6 Registers Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SPI Base Address:**<br>**SPI0_BA = 0xB800_6200**<br>**SPI1_BA = 0xB800_6300** | | | | |
| **SPIn_CNTRL**<br>**n=0,1** | SPIn_BA+0x000 | R/W | SPI n Control and Status Register | 0x0000_0004 |
| **SPIn_DIVIDER**<br>**n=0,1** | SPIn_BA+0x004 | R/W | SPI n Clock Divider Register | 0x0000_0000 |
| **SPIn_SSR**<br>**n=0,1** | SPIn_BA+0x008 | R/W | SPI n Slave Select Register | 0x0000_0000 |
| **SPIn_RX0**<br>**n=0,1** | SPIn_BA+0x010 | R | SPI n Data Receive Register 0 | 0x0000_0000 |
| **SPIn_RX1**<br>**n=0,1** | SPIn_BA+0x014 | R | SPI n Data Receive Register 1 | 0x0000_0000 |
| **SPIn_RX2**<br>**n=0,1** | SPIn_BA+0x018 | R | SPI n Data Receive Register 2 | 0x0000_0000 |
| **SPIn_RX3**<br>**n=0,1** | SPIn_BA+0x01C | R | SPI n Data Receive Register 3 | 0x0000_0000 |
| **SPIn_TX0**<br>**n=0,1** | SPIn_BA+0x010 | W | SPI n Data Transmit Register 0 | 0x0000_0000 |
| **SPIn_TX1**<br>**n=0,1** | SPIn_BA+0x014 | W | SPI n Data Transmit Register 1 | 0x0000_0000 |
| **SPIn_TX2**<br>**n=0,1** | SPIn_BA+0x018 | W | SPI n Data Transmit Register 2 | 0x0000_0000 |
| **SPIn_TX3**<br>**n=0,1** | SPIn_BA+0x01C | W | SPI n Data Transmit Register 3 | 0x0000_0000 |

NOTE 1: When software programs CNTRL, the GO_BUSY bit should be written last.

### 5.19.7 Register Description

### SPI n Control and Status Register (SPIn_CNTRL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SPIn_CNTRL n=0,1 | SPIn_BA+0x000 | R/W | SPI n Control and Status Register | 0x0000_0004 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CLK_POL | Reserved | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | DUALM | QUADM | DIR_2QM | Reserved | | IE | IF |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SLEEP | | | | Reserved | LSB | Tx_NUM | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Tx_BIT_LEN | | | | | Tx_NEG | Rx_NEG | GO_BUSY |

| Bits | Description | |
|---|---|---|
| [31] | **CLK_POL** | **Clock Polarity**<br>0 = Normal polarity.<br>1 = Reverse polarity. |
| [22] | **DUALM** | **Dual I/O Mode Enable Control**<br>0 = Dual I/O mode Disabled.<br>1 = Dual I/O mode Enabled. |
| [21] | **QUADM** | **Quad I/O Mode Enable Control**<br>0 = Quad I/O mode Disabled.<br>1 = Quad I/O mode Enabled. |
| [20] | **DIR_2QM** | **Quad or Dual I/O Mode Direction Control**<br>0 = Quad or Dual Input mode.<br>1 = Quad or Dual Output mode. |
| [17] | **IE** | **Interrupt Enable**<br>0 = Disable SPI Interrupt.<br>1 = Enable SPI Interrupt. |
| [16] | **IF** | **Interrupt Flag**<br>0 = The transfer dose not finish yet.<br>1 = The transfer is done. The interrupt flag is set if it was enable.<br>**NOTE:** This bit is read only, but can be cleared by writing 1 to this bit. |
| [15:12] | **SLEEP** | **Suspend Interval**<br>0000 = SPI transfer suspended 2 SCLK clock cycle.<br>0001 = SPI transfer suspended 3 SCLK clock cycle.<br>0010 = SPI transfer suspended 4 SCLK clock cycle.<br>0011 = SPI transfer suspended 5 SCLK clock cycle. |

| | | |
|---|---|---|
| | | 0100 = SPI transfer suspended 6 SCLK clock cycle.<br>0101 = SPI transfer suspended 7 SCLK clock cycle.<br>0110 = SPI transfer suspended 8 SCLK clock cycle.<br>0111 = SPI transfer suspended 9 SCLK clock cycle.<br>1000 = SPI transfer suspended 10 SCLK clock cycle.<br>1001 = SPI transfer suspended 11 SCLK clock cycle.<br>1010 = SPI transfer suspended 12 SCLK clock cycle.<br>1011 = SPI transfer suspended 13 SCLK clock cycle.<br>1100 = SPI transfer suspended 14 SCLK clock cycle.<br>1101 = SPI transfer suspended 15 SCLK clock cycle.<br>1110 = SPI transfer suspended 16 SCLK clock cycle.<br>1111 = SPI transfer suspended 17 SCLK clock cycle. |
| [10] | **LSB** | **Send LSB First**<br>0 = The MSB Is Transmitted/Received First.<br>1 = The LSB Is Transmitted/Received First. |
| [9:8] | **Tx_NUM** | **Transmit/Receive Numbers**<br>00 = Only one transmit/receive will be executed in one transfer.<br>01 = Two successive transmit/receive will be executed in one transfer.<br>10 = Three successive transmit/receive will be executed in one transfer.<br>11 = Four successive transmit/receive will be executed in one transfer. |
| [7:3] | **Tx_BIT_LEN** | **Transmit Bit Length**<br>This field specifies how many bits are transmitted in one transmit/receive. Up to 32 bits can be transmitted. The transmit bit length is Tx_BIT_LEN+1 |
| [2] | **Tx_NEG** | **Transmit on Negative Edge**<br>0 = Transmitted data output signal is changed on the rising edge of mw_sclk_o.<br>1 = Transmitted data output signal is changed on the falling edge of mw_sclk_o. |
| [1] | **Rx_NEG** | **Receive on Negative Edge**<br>0 = Received data input signal is latched on the rising edge of mw_sclk_o.<br>1 = Received data input signal is latched on the falling edge of mw_sclk_o. |
| [0] | **GO_BUSY** | **Go and Busy Status**<br>0 = Writing 0 to this bit has no effect.<br>1 = Writing 1 to this bit starts the transfer. This bit remains set during the transfer and is automatically cleared after transfer finished.<br>**NOTE:** All registers should be set before writing 1 to the GO_BUSY bit in the CNTRL register. When a transfer is in progress, writing to any register of the SPI master core has no effect. |

### SPI n Divider Register (SPIn_DIVIDER)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SPIn_DIVIDER n=0,1** | SPIn_BA+0x004 | R/W | SPI n 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** |
| **DIVIDER** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **DIVIDER** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [15:0] | **DIVIDER** | **Clock Divider Register**<br>The divider value of serial clock.<br>**NOTE:** Suggest DIVIDER should be at least 1. |

## SPI n Slave Select Register (SPIn_SSR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SPIn_SSR n=0,1** | SPIn_BA+0x008 | R/W | SPI n Slave 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** | | | | **AUTOSS** | **SS_LVL** | **SSR** | |

| Bits | Description | |
|---|---|---|
| [3] | **AUTOSS** | **Automatic Slave Select Function Enable Control**<br>0 = Automatic slave select function Disabled.<br>1 = Automatic slave select function Enabled. |
| [2] | **SS_LVL** | **Slave Select Active Level**<br>0 = The slave select signal mw_ss_o is active on low-level.<br>1 = The slave select signal mw_ss_o is active on high-level. |
| [1:0] | **SSR** | **Slave Select Register**<br>If SSR[ASS] bit =0,.<br>0 = sets the line back to inactive state.<br>1 = sets the proper mw_ss_o line to an active state.<br>If SSR[ASS] bit =1,.<br>0 = The controller will be driven to inactive state for the rest of the time.<br>1 = select appropriate mw_ss_o line to be automatically driven to active state for the duration of transmit/receive.<br>**NOTE:** This interface can only drive one device/slave at a given time. Therefore, the slave select of the selected device must be set to its active level before starting any read or write transfer.<br>**NOTE:** (The active level of mw_ss_o is specified in SSR[SS_LVL]). |

## SPI n Data Receive Register (SPIn_RX)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SPIn_RX0 n=0,1** | SPIn_BA+0x010 | R | SPI n Data Receive Register 0 | 0x0000_0000 |
| **SPIn_RX1 n=0,1** | SPIn_BA+0x014 | R | SPI n Data Receive Register 1 | 0x0000_0000 |
| **SPIn_RX2 n=0,1** | SPIn_BA+0x018 | R | SPI n Data Receive Register 2 | 0x0000_0000 |
| **SPIn_RX3 n=0,1** | SPIn_BA+0x01C | R | SPI n Data Receive Register 3 | 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**<br>The Data Receive Registers hold the value of received data of the last executed transfer. Valid bits depend on the transmit bit length field in the CNTRL register. For example, if CNTRL[Tx_BIT_LEN] is set to 0x08 and CNTRL[Tx_NUM] is set to 0x0, bit Rx0[7:0] holds the received data.<br>**NOTE:** The Data Receive Registers are read only registers. A Write to these registers will actually modify the Data Transmit Registers because those registers share the same FFs. |

### SPI n Data Transmit Register (SPIn_TX)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SPIn_TX0 n=0,1** | SPIn_BA+0x010 | W | SPI n Data Transmit Register 0 | 0x0000_0000 |
| **SPIn_TX1 n=0,1** | SPIn_BA+0x014 | W | SPI n Data Transmit Register 1 | 0x0000_0000 |
| **SPIn_TX2 n=0,1** | SPIn_BA+0x018 | W | SPI n Data Transmit Register 2 | 0x0000_0000 |
| **SPIn_TX3 n=0,1** | SPIn_BA+0x01C | W | SPI n Data Transmit Register 3 | 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**<br>The Data Transmit Registers hold the data to be transmitted in the next transfer. Valid bits depend on the transmit bit length field in the CNTRL register. For example, if CNTRL[Tx_BIT_LEN] is set to 0x08 and the CNTRL[Tx_NUM] is set to 0x0, the bit Tx0[7:0] will be transmitted in next transfer. If CNTRL[Tx_BIT_LEN] is set to 0x00 and CNTRL[Tx_NUM] is set to 0x3, the core will perform four 32-bit transmit/receive successive using the same setting (the order is Tx0[31:0], Tx1[31:0], Tx2[31:0], Tx3[31:0]).<br>**NOTE:** The RxX and TxX registers share the same flip-flops, which means that what is received from the input data line in one transfer will be transmitted on the output data line in the next transfer if no write access to the TxX register is executed between the transfers. |

## 5.20 I²S Controller (I²S)

### 5.20.1 Overview

The I²S controller consists of I²S and PCM protocols to interface with external audio CODEC. The I²S and PCM interface supports 8, 16, 18, 20 and 24-bit left/right precision in record and playback. When operating in 18/20/24-bit precision, each left/right-channel sample is stored in a 32-bit word. Each left/right-channel sample has 24/20/18 MSB bits of valid data and other LSB bits are the padding zeros. When operating in 16-bit precision, right-channel sample is stored in MSB of a 32-bit word and left-channel sample is stored in LSB of a 32-bit word.

The following are the property of the DMA.

- When 16-bit precision, the DMA always 8-beat incrementing burst (FIFO_TH = 0) or 4-beat incrementing burst (FIFO_TH = 1).
- When 24/20/18-bit precision, the DMA always 16-beat incrementing burst (FIFO_TH = 0) or 8-beat incrementing burst (FIFO_TH = 1).
- Always bus lock when 4-beat or 8-beat or 16-beat incrementing burst.
- When reach eighth, quarter, middle and end address of destination address, a DMA_IRQ is triggered to CPU automatically.

An AHB master port and an AHB slave port are offered in I²S controller.

### 5.20.2 Features

- Support I²S interface record and playback
  - Left/right channel
  - 8, 16, 20, 24-bit data precision
  - Mater and slave mode
- Support PCM interface record and playback
  - Two slots
  - 8, 16, 20, 24-bit data precision
  - Master mode
- Use DMA to playback and record data, with interrupt
- Support two addresses for left/right channel data and different slots

### 5.20.3 Block Diagram

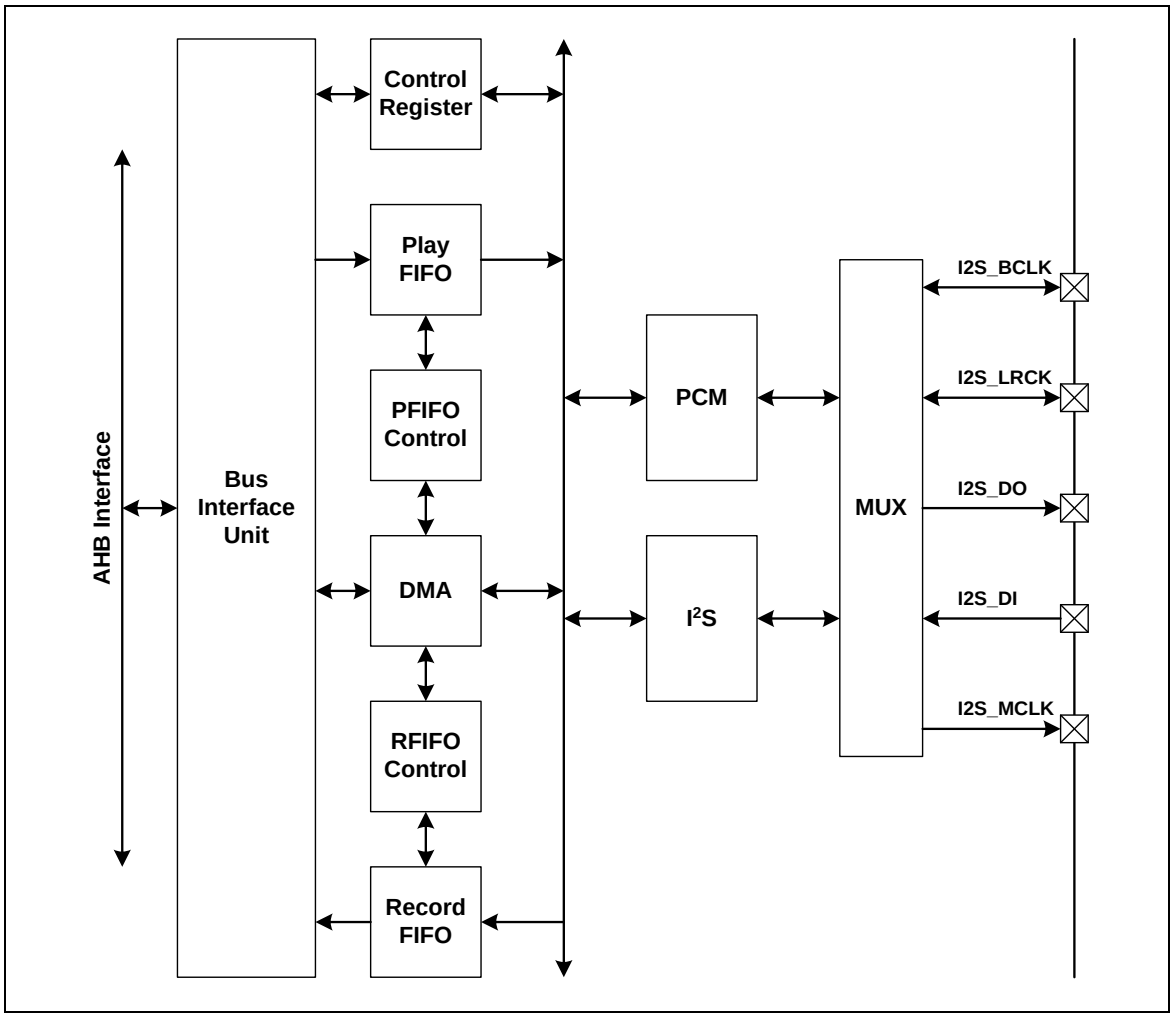



Figure 5.20-1 I²S Controller Block Diagram

### 5.20.4 Basic Configuration

Before using I²S functionality, it's necessary to configure I/O pins as the I²S function and enable I²S's clock.

Write 0x8 to MFP_GPG10 (SYS_GPG_MFPH[11:8]), MFP_GPG11 (SYS_GPG_MFPH[15:12]), MFP_GPG12 (SYS_GPG_MFPH[19:16]), MFP_GPG13 (SYS_GPG_MFPH[23:20]) and MFP_GPG14 (SYS_GPG_MFPH[27:24]) configures pin PG.10, PG.11, PG.12, PG.13 and PG.14 to be I2S_SYSCLK, I2S_DATAO, I2S_DATAI, I2S_BITCLK and I2S_WS resepctively.

To enable I²S's clock, please refer to register CLK_HCLKEN. Set I2S (CLK_HCLKEN[24]) high to enable I²S clock.

### 5.20.5 Functional Description

#### 5.20.5.1 *I²S interface*

The I²S interface signals are shown as the following figure.

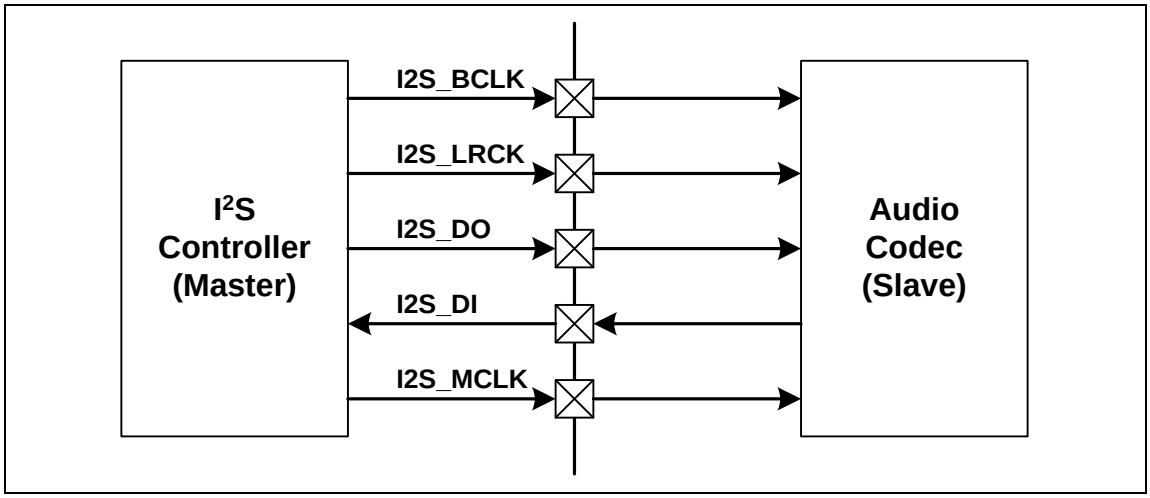



Figure 5.20-2 I²S Interface Signal of Master Mode

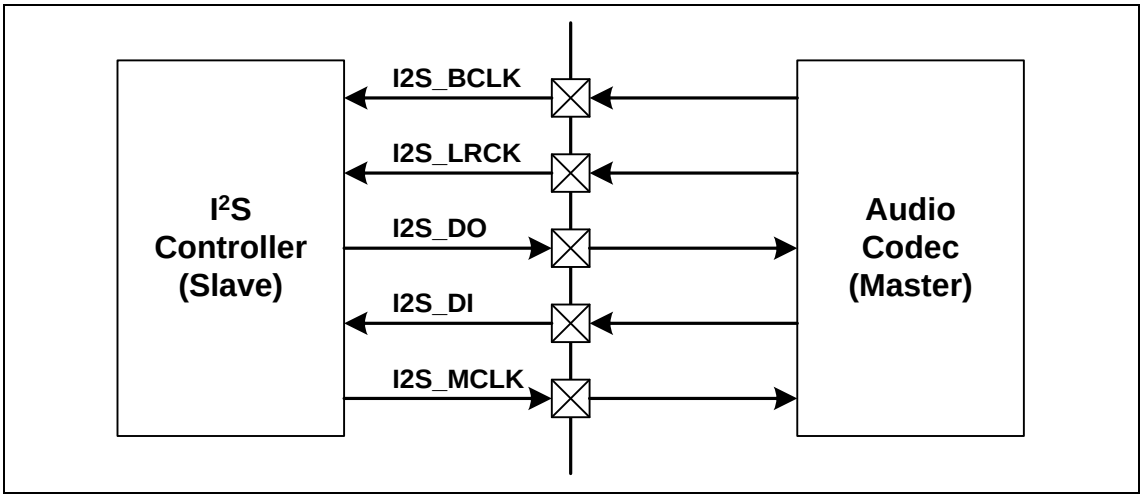



Figure 5.20-3 I²S Interface Signal of Slave Mode

The I²S and MSB-justified format are supported; the timing diagram is shown as following figure.

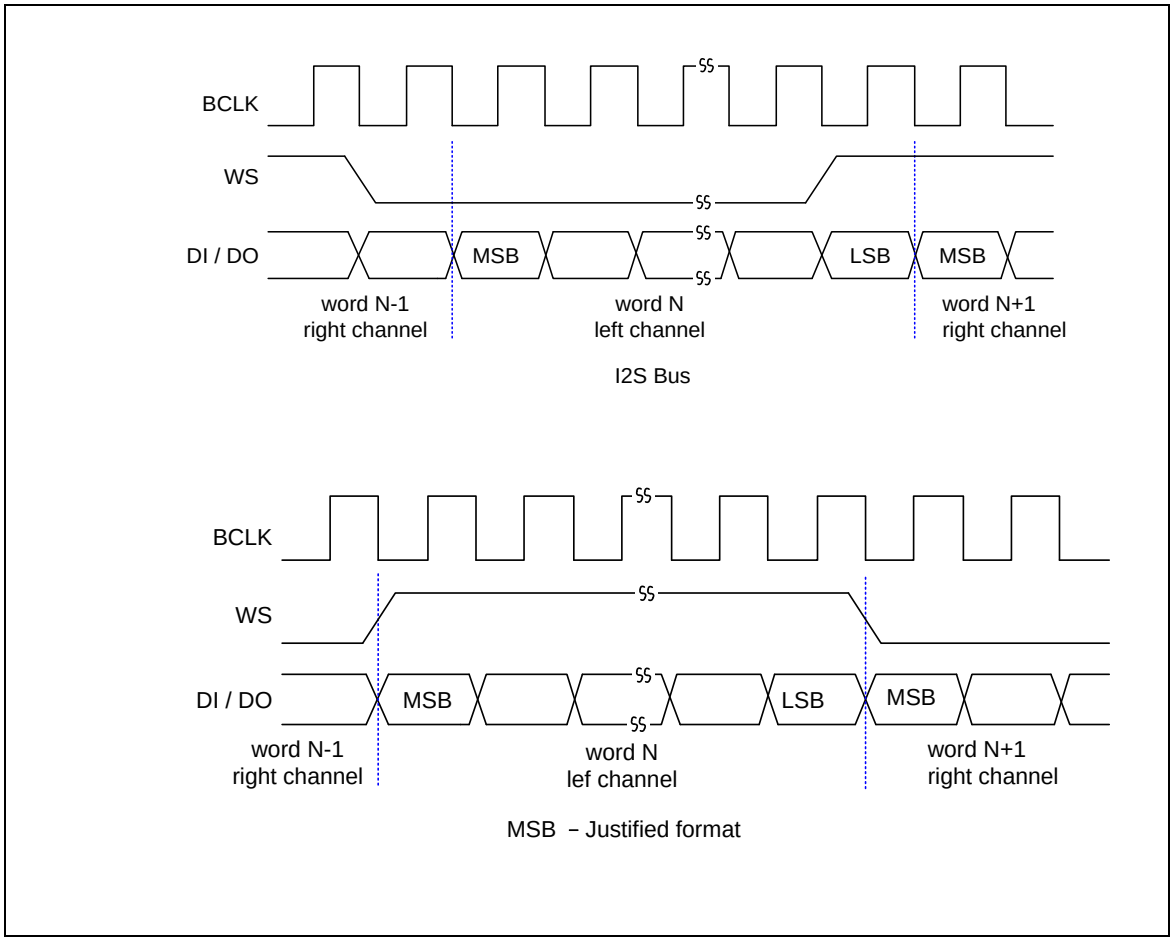



Figure 5.20-4 I²S MSB-Justified Format

The sampling rate, bit shift clock frequency could be set by the control register I2S_CON.

5.20.5.2 *PCM Interface*

The PCM interface signals are shown as the following figure.

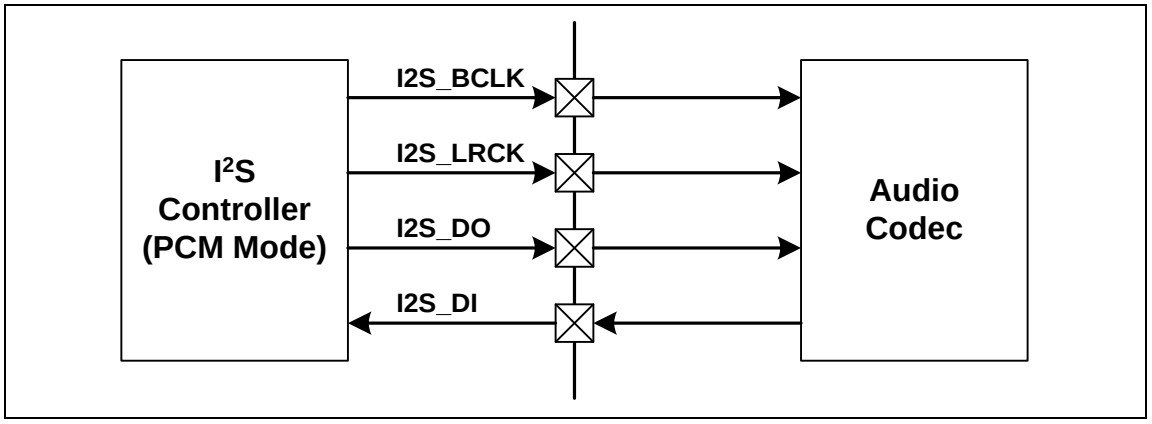



Figure 5.20-5 PCM Mode Signal Interface

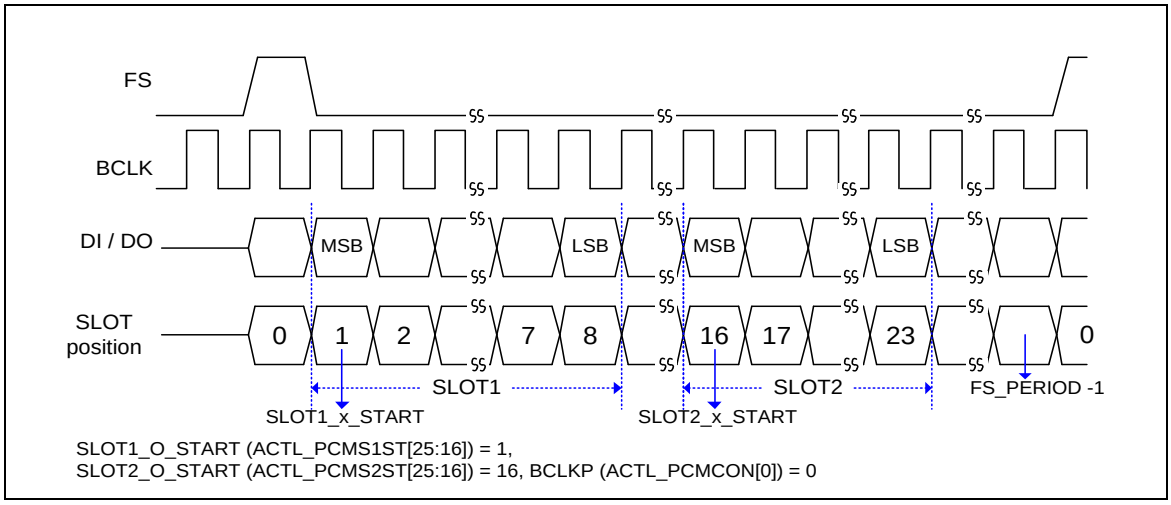



Figure 5.20-6 PCM Mode Interface Waveform

### 5.20.6 Register Map

**R**: read only, **W**: write only, **R/W**: both read and write, **C**: Only value 1 can be written

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **$I^2S$ Base Address:**<br>**I2S_BA = 0xB000_9000** | | | | |
| **I2S_GLBCON** | I2S_BA+0x000 | R/W | I2S Global Control Register | 0x0000_0000 |
| **I2S_RESET** | I2S_BA+0x004 | R/W | I2S Sub Block Reset Control Register | 0x0000_0000 |
| **I2S_RDESB** | I2S_BA+0x008 | R/W | I2S Record DMA Destination Base Address Register | 0x0000_0000 |
| **I2S_RDES_LENGTH** | I2S_BA+0x00C | R/W | I2S Record DMA Destination Length Register | 0x0000_0000 |
| **I2S_RDESC** | I2S_BA+0x010 | R | I2S Record DMA Destination Current Address Register | 0x0000_0000 |
| **I2S_PDESB** | I2S_BA+0x014 | R/W | I2S Play DMA Destination Base Address Register | 0x0000_0000 |
| **I2S_PDES_LENGTH** | I2S_BA+0x018 | R/W | I2S Play DMA Destination Length Register | 0x0000_0000 |
| **I2S_PDESC** | I2S_BA+0x01C | R | I2S Play DMA Destination Current Address Register | 0x0000_0000 |
| **I2S_RSR** | I2S_BA+0x020 | R/W | I2S Record Status Register | 0x0000_0000 |
| **I2S_PSR** | I2S_BA+0x024 | R/W | I2S Play Status Register | 0x0000_0000 |
| **I2S_CON** | I2S_BA+0x028 | R/W | I2S Control Register | 0x0000_0000 |
| **I2S_COUNTER** | I2S_BA+0x02C | R/W | I2S Play DMA Down Counter Register | 0xFFFF_FFFF |
| **I2S_PCMCON** | I2S_BA+0x030 | R/W | I2S PCM Mode Control Register | 0x0000_0000 |
| **I2S_PCMS1ST** | I2S_BA+0x034 | R/W | I2S PCM Mode Slot 1 Start Register | 0x0000_0000 |
| **I2S_PCMS2ST** | I2S_BA+0x038 | R/W | I2S PCM Mode Slot 2 Start Register | 0x0000_0000 |
| **I2S_RDESB2** | I2S_BA+0x040 | R/W | I2S Record DMA Destination Base Address 2 Register | 0x0000_0000 |
| **I2S_PDESB2** | I2S_BA+0x044 | R/W | I2S Play DMA Destination Base Address 2 Register | 0x0000_0000 |

### 5.20.7 Register Description

### I2S Global Control Register (I2S_GLBCON)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| I2S_GLBCON | I2S_BA+0x000 | R/W | I2S Global Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | R_DMA_IRQ_EN | P_DMA_IRQ_EN | R_FIFO_FULL_IRQ_EN | R_FIFO_EMPTY_IRQ_EN | P_FIFO_FULL_IRQ_EN | P_FIFO_EMPTY_IRQ_EN |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| R_DMA_IRQ_SEL | | P_DMA_IRQ_SEL | | R_DMA_IRQ | P_DMA_IRQ | BITS_SELECT | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FIFO_TH | Reserved | | IRQ_DMA_CNTER_EN | IRQ_DMA_DATA_ZERO_EN | Reserved | BLOCK_EN | |

| Bits | Description | |
|---|---|---|
| [31:20] | **Reserved** | Reserved. |
| [21] | **R_DMA_IRQ_EN** | **Record DMA Interrupt Request Enable Bit**<br>0: not allowed to generation R_DMA_IRQ<br>1: allowed to generation R_DMA_IRQ<br>The R_DMA_IRQ_EN bit is read/write |
| [20] | **P_DMA_IRQ_EN** | **Playback DMA Interrupt Request Enable Bit**<br>0: not allowed to generation P_DMA_IRQ<br>1: allowed to generation P_DMA_IRQ<br>The P_DMA_IRQ_EN bit is read/write |
| [19] | **R_FIFO_FULL_IRQ_EN** | **Record FIFO Full Interrupt Request Enable Bit**<br>0: not allowed to generation R_FIFO_FULL_IRQ<br>1: allowed to generation R_FIFO_FULL_IRQ<br>The R_FIFO_FULL_IRQ_EN bit is read/write |
| [18] | **R_FIFO_EMPTY_IRQ_EN** | **Record FIFO Empty Interrupt Request Enable Bit**<br>0: not allowed to generation R_FIFO_EMPTY_IRQ<br>1: allowed to generation R_FIFO_EMPTY_IRQ<br>The R_FIFO_EMPTY_IRQ_EN bit is read/write |
| [17] | **P_FIFO_FULL_IRQ_EN** | **Playback FIFO Full Interrupt Request Enable Bit**<br>0: not allowed to generation P_FIFO_FULL_IRQ<br>1: allowed to generation P_FIFO_FULL_IRQ<br>The P_FIFO_FULL_IRQ_EN bit is read/write |

| | | |
|---|---|---|
| [16] | **P_FIFO_EMPTY_IRQ_EN** | **Playback FIFO Empty Interrupt Request Enable Bit**<br>0: not allowed to generation P_FIFO_EMPTY_IRQ<br>1: allowed to generation P_FIFO_EMPTY_IRQ<br>The P_FIFO_EMPTY_IRQ_EN bit is read/write |
| [15:14] | **R_DMA_IRQ_SEL[1:0]** | **Record DMA Interrupt Request Selection Bits**<br>00: When record DMA address reach DMA record destination **end** address, the R_DMA_RIA_IRQ will be issued.<br>01: When record DMA address reach each **half** of DMA record destination end address, the R_DMA_RIA_IRQ will be issued.<br>10: When record DMA address reach each **quarter** of DMA record destination end address, the R_DMA_RIA_IRQ will be issued.<br>11: When record DMA address reach each **eighth** of DMA record destination end address, the R_DMA_RIA_IRQ will be issued.<br>The R_DMA_IRQ_SEL bits are read/write |
| [13:12] | **P_DMA_IRQ_SEL[1:0]** | **Play DMA Interrupt Request Selection Bits**<br>00: When play DMA address reach DMA play destination **end** address, the P_DMA_RIA_IRQ will be issued.<br>01: When play DMA address reach each **half** of DMA play destination **end** address, the P_DMA_RIA_IRQ will be issued.<br>10: When play DMA address reach each **quarter** of DMA play destination end address, the P_DMA_RIA_IRQ will be issued.<br>11: When play DMA address reach each **eighth** of DMA play destination end address, the P_DMA_RIA_IRQ will be issued.<br>The P_DMA_IRQ_SEL bits are read/write |
| [11] | **R_DMA_IRQ** | **Record DMA Interrupt Request Bit**<br>When R_DMA_RIA_IRQ or R_FIFO_FULL or R_FIFO_EMPTY is set to “1” in record and these corresponding interrupt enable bits are set to “1”, the R_DMA_IRQ bit will be set to 1 automatically, and this bit could be cleared to 0 by CPU writing “1”. The bit is hardwired to ARM926 as interrupt request signal with an inverter.<br>The R_DMA_IRQ bit is read/write |
| [10] | **P_DMA_IRQ** | **Playback DMA Interrupt Request Bit**<br>When P_DMA_RIA_IRQ (I2S_PSR[0]) or DMA_DATA_ZERO_IRQ (I2S_PSR[3]) or DMA_CNTER_IRQ (I2S_PSR[4]) or P_FIFO_FULL (I2S_PSR[2]) or P_FIFO_EMPTY (I2S_PSR[1]) is set to 1 in playback and these corresponding interrupt enable bits are set to “1”, the bit P_DMA_IRQ will be set to 1, and this bit could be clear to 0 by CPU writing “1”. And the bit is hardwired to ARM926 as interrupt request signal with an inverter.<br>The P_DMA_IRQ bit is read/write |
| [9:8] | **BITS_SELECT** | **BITS Selection**<br>00: data format is 8-bits of a channel.<br>01: data format is 16-bits of a channel.<br>10: data format is 24-bits of a channel.<br>11: reserve<br>The BITS_SELECT bit is read/write |
| [7] | **FIFO_TH** | **FIFO Threshold Control Bit**<br>0: The FIFO threshold is 8 levels.<br>1: The FIFO threshold is 4 levels.<br>The FIFO_TH bit is read/write |
| [6:5] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [4] | **IRQ_DMA_CNTER_EN** | **IRQ_DMA Counter Function Enable Bit**<br>0: not allowed to set P_DMA_IRQ (I2S_CON[10]) if I2S_PSR[4] is set to 1.<br>1: allowed to set P_DMA_IRQ (I2S_CON[10]) if (I2S_PSR[4]) is set to 1.<br>The IRQ_DMA_CNTER_EN bit is read/write |
| [3] | **IRQ_DMA_DATA_ZERO_EN** | **IRQ_DMA_DATA Zero and Sign Detect Enable Bit**<br>0: not allowed to set P_DMA_IRQ (I2S_CON[10]) if I2S_PSR[3] is set to 1.<br>1: allowed to set P_DMA_IRQ (I2S_CON[10]) if I2S_PSR[3] is set to 1.<br>The IRQ_DMA_DATA_ZERO_EN bit is read/write |
| [2] | **Reserved** | Reserved. |
| [1:0] | **BLOCK_EN** | **Block Function Enable Register**<br>00: The $I^2S$ & PCM interface is disabled.<br>01: The $I^2S$ interface is enabled.<br>10: The PCM interface is enabled.<br>11: reserved<br>The register is read/write |

## I2S Sub Block Reset Control Register (I2S_RESET)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| I2S_RESET | I2S_BA+0x004 | R/W | I2S Sub Block Reset Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | SPLIT_DATA | Reserved | | | RESET |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RECORD_SINGLE | | PLAY_SINGLE | | Reserved | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | RECORD | PLAY | DMA_CNTER_EN | DMA_DATA_ZERO_EN | Reserved | | BLOCK_RESET |

| Bits | Description | |
|---|---|---|
| [31:17] | **Reserved** | Reserved. |
| [20] | **SPLIT_DATA** | **SPLIT Left/Right and Slot1/Slot2 Data**<br>**Note: this bit work at stereo/dual slot mode only**<br>1: Left channel data at I2S_RDESB / I2S_PDESB address, Right channel data at I2S_RDESB2 / I2S_PDESB2<br>0: Left/Right channel data place at I2S_RDESB / I2S_PDESB address<br>The DATA_TYPE bit is read/write<br>**If SPLIT=0, 8bit-data, L=8bit left/slot0 data, R=8bit right/slot1 data, address at** I2S_RDESB / I2S_PDESB. |

| 0xC | 0x8 | 0x4 | 0x0 |
|---|---|---|---|
| R,L,R,L | R,L,R,L | R,L,R,L | R,L,R,L |

**If SPLIT=0, 16bit-data, L=16bit left/slot0 data, R=16bit right/slot1 data, address at** I2S_RDESB / I2S_PDESB.

| 0xC | 0x8 | 0x4 | 0x0 |
|---|---|---|---|
| R,L | R,L | R,L | R,L |

**If SPLIT=0, 24bit-data, L=24bit left/slot0 data, R=24bit right/slot1 data, address at** I2S_RDESB / I2S_PDESB.

| 0xC | 0x8 | 0x4 | 0x0 |
|---|---|---|---|
| R | L | R | L |

**If SPLIT=1, 8bit-data, L=8bit left/slot0 data, R=8bit right/slot1 data, address at** I2S_RDESB / I2S_PDESB.

| 0xC | 0x8 | 0x4 | 0x0 |
|---|---|---|---|
| L,L,L,L | L,L,L,L | L,L,L,L | L,L,L,L |

<table>
<tr><td></td><td></td><td>
address at I2S_RDESB2 / I2S_PDESB2
<table>
<tr><td>0xC</td><td>0x8</td><td>0x4</td><td>0x0</td></tr>
<tr><td>R,R,R,R</td><td>R,R,R,R</td><td>R,R,R,R</td><td>R,R,R,R</td></tr>
</table>
If SPLIT=1, 16bit-data, L=16bit left/slot0 data, R=16bit right/slot1 data, address at I2S_RDESB / I2S_PDESB.
<table>
<tr><td>0xC</td><td>0x8</td><td>0x4</td><td>0x0</td></tr>
<tr><td>L,L</td><td>L,L</td><td>L,L</td><td>L,L</td></tr>
</table>
address at I2S_RDESB2 / I2S_PDESB2
<table>
<tr><td>0xC</td><td>0x8</td><td>0x4</td><td>0x0</td></tr>
<tr><td>R,R</td><td>R,R</td><td>R,R</td><td>R,R</td></tr>
</table>
If SPLIT=1, 24bit-data, L=24bit left/slot0 data, R=24bit right/slot1 data, address at I2S_RDESB / I2S_PDESB.
<table>
<tr><td>0xC</td><td>0x8</td><td>0x4</td><td>0x0</td></tr>
<tr><td>L</td><td>L</td><td>L</td><td>L</td></tr>
</table>
address at I2S_RDESB2 / I2S_PDESB2
<table>
<tr><td>0xC</td><td>0x8</td><td>0x4</td><td>0x0</td></tr>
<tr><td>R</td><td>R</td><td>R</td><td>R</td></tr>
</table>
</td></tr>
<tr><td>[19:17]</td><td>Reserved</td><td>Reserved.</td></tr>
<tr><td>[16]</td><td>RESET</td><td>Audio Controller Reset Control Bit<br>0: The audio controller is normal operation.<br>1: The whole audio controller is reset.<br>The RESET bit is read/write</td></tr>
<tr><td>[15:14]</td><td>RECORD_SINGLE [1:0]</td><td>Record Single/Dual Channel Select Bits<br>00: RESERVED.<br>01: The record only selects i2s left channel/pcm slot0.<br>10: The record only selects i2s right channel/pcm slot1.<br>11: The record is dual channel.<br>The RECORD_SINGLE[1:0] bits are read/write</td></tr>
<tr><td>[13:12]</td><td>PLAY_SINGLE [1:0]</td><td>Playback Single/Dual Channel Select Bits<br>In I²S mode,<br>00: reserved.<br>01: reserved.<br>10: The playback is mono mode.<br>11: The playback is stereo mode.<br>In PCM mode,<br>00: The playback is dual slot with slot1 data.<br>01: The playback is dual slot with slot0 data.<br>10: The playback is mono data. slot0 only.<br>11: The playback is dual slot.</td></tr>
</table>

| | | |
|---|---|---|
| | | The PLAY_SINGLE[1:0] bits are read/write |
| [11:7] | **Reserved** | Reserved. |
| [6] | **RECORD** | **I²S/PCM Record Control Bit**<br>0: The record path of I²S/PCM is disabled.<br>1: The record path of I²S/PCM is enabled.<br>The RECORD bit is read/write |
| [5] | **PLAY** | **I²S/PCM Playback Control Bit**<br>0: The playback path of I²S/PCM is disabled.<br>1: The playback path of I²S/PCM is enabled.<br>The PLAY bit is read/write |
| [4] | **DMA_CNTER_EN** | **DMA Counter Function Enable Bit**<br>This function is supported to count playback data for software monitoring. When one playback data is transferred to codec, the DMA counter subtracts 1. When the I2S_COUNTER [31:0] register is Zero that set DMA_CNTER_IRQ bit =1.<br>0: The DMA counter function is disabled.<br>1: The DMA counter function is enabled.<br>The DMA_CNTER_EN bit is read/write |
| [3] | **DMA_DATA_ZERO_EN** | **DMA_DATA Zero and Sign Detect Enable Bit**<br>0: The DMA_DATA zero and sign detect function is disabled.<br>1: The DMA_DATA zero and sign detect function is enabled.<br>The DMA_DATA_ZERO_EN bit is read/write |
| [2:1] | **Reserved** | Reserved. |
| [0] | **BLOCK_RESET** | **I²S/PCM RESET Control Bit**<br>0: Release the I²S/PCM function block from reset mode.<br>1: Force the I²S/PCM function block to reset mode.<br>This bit is read/write |

### I2S Record DMA Destination Base Address Register (I2S_RDESB)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_RDESB** | I2S_BA+0x008 | R/W | I2S Record DMA Destination Base Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **AUDIO_RDESB** | **32-bit Record Destination Base Address**<br>This bit field indicates the record destination base address of DMA.<br>The AUDIO_RDESB [31:0] bits are read/write. |

### I2S Record DMA Destination Length Register (I2S_RDES_LENGTH)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_RDES_LENGTH** | I2S_BA+0x00C | R/W | I2S Record DMA Destination Length Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **AUDIO_RDES_L** | **32-bit Record Destination Address Length**<br>The AUDIO_RDES_L [31:0] bits are read/write. The minimum value for 16-bits mode is 0x20 and for 24-bits mode is 0x40. |

### I2S Record DMA Destination Current Address Register (I2S_RDESC)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_RDESC** | I2S_BA+0x010 | R | I2S Record DMA Destination Current Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **AUDIO_RDESC** | **32-bit Record Destination Current Address**<br>This bit field indicates the current address of DMA record destination.<br>The AUDIO_RDESC [31:0] bits are read only. |

## I2S Play DMA Destination Base Address Register (I2S_PDESB)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_PDESB** | I2S_BA+0x014 | R/W | I2S Play DMA Destination Base Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **AUDIO_PDESB** | **32-bit Play Destination Base Address**<br>This bit field indicates the play destination base address of DMA.<br>The AUDIO_PDESB [31:0] bits are read/write. |

## I2S Play DMA Destination Length Register (I2S_PDES_LENGTH)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| I2S_PDES_LENGTH | I2S_BA+0x018 | R/W | I2S Play DMA Destination Length Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | AUDIO_PDES_L | **32-bit Play Destination Address Length**<br>The AUDIO_PDES_L [31:0] bits are read/write. The minimum value for 16-bits mode is 0x20 and for 24-bits mode is 0x40. |

### I2S Play DMA Destination Current Address Register (I2S_PDESC)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_PDESC** | I2S_BA+0x01C | R | I2S Play DMA Destination Current Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **AUDIO_PDESC** | **32-bit Play Destination Current Address**<br>This bit field indicates the current address of DMA play destination.<br>The AUDIO_PDESC [31:0] bits are read only. |

## I2S Record Status Register (I2S_RSR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_RSR** | I2S_BA+0x020 | R/W | I2S Record 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** |
| **R_DMA_RIA_SN** | | | **Reserved** | | **R_FIFO_FULL** | **R_FIFO_EMPTY** | **R_DMA_RIA_IRQ** |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:5] | **R_DMA_RIA_SN** | **Record DMA Reach Indicative Address Section Number Bit**<br>R_DMA_IRQ_SEL (I2S_CON[15:14]) = 01, **R_DMA_RIA_SN[2:0]= 1, 0.**<br>R_DMA_IRQ_SEL (I2S_CON[15:14]) = 10, **R_DMA_RIA_SN[2:0]= 1, 2, 3, 0.**<br>R_DMA_IRQ_SEL (I2S_CON[15:14]) = 11, **R_DMA_RIA_SN[2:0]= 1, 2, 3, 4, 5, 6, 7, 0.**<br>The R_DMA_RIA_SN[2:0] bits are read only |
| [4:3] | **Reserved** | Reserved. |
| [2] | **R_FIFO_FULL** | **Record FIFO Full Indicator Bit**<br>When record FIFO is full and the record data is written into record FIFO, the R_FIFO_FULL bit is set to 1. This bit indicates the full error of record FIFO is happened.<br>0: the full error of record FIFO is not happened.<br>1: the full error of record FIFO is happened.<br>The R_FIFO_FULL bit is readable, and only can be clear by write “1” to this bit. |
| [1] | **R_FIFO_EMPTY** | **Record FIFO EMPTY Indicator Bit**<br>When record FIFO is empty and the record data is read from record FIFO, the R_FIFO_EMPTY bit is set to 1. This bit indicates the empty error of record FIFO is happened.<br>0: the empty error of record FIFO is not happened.<br>1: the empty error of record FIFO is happened.<br>The R_FIFO_EMPTY bit is readable, and only can be clear by write “1” to this bit. |
| [0] | **R_DMA_RIA_IRQ** | **Record DMA Reach Indicative Address Interrupt Request Bit**<br>0: Record DMA address does not reach the indicative address by R_DMA_IRQ_SEL (I2S_CON[15:14]).<br>1: Record the DMA address reach the indicative address by R_DMA_IRQ_SEL (I2S_CON[15:14]).<br>The R_DMA_RIA_IRQ bit is readable, and only can be clear by write “1” to this bit |

### I2S Play Status Register (I2S_PSR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| I2S_PSR | I2S_BA+0x024 | R/W | I2S Play 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 |
| P_DMA_RIA_SN | | | DMA_CNTER_IRQ | DMA_DATA_ZERO_IRQ | P_FIFO_FULL | P_FIFO_EMPTY | P_DMA_RIA_IRQ |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:5] | **P_DMA_RIA_SN** | **Play DMA Reach Indicative Address Section Number Bit**<br>P_DMA_IRQ_SEL (I2S_CON[13:12]) = 01, **P_DMA_RIA_SN[2:0]= 1, 0.**<br>P_DMA_IRQ_SEL (I2S_CON[13:12]) = 10, **P_DMA_RIA_SN[2:0]= 1, 2, 3, 0.**<br>P_DMA_IRQ_SEL (I2S_CON[13:12]) = 11, **P_DMA_RIA_SN[2:0]= 1, 2, 3, 4, 5, 6, 7, 0.**<br>The P_DMA_RIA_SN[2:0] bits are read only |
| [4] | **DMA_CNTER_IRQ** | **DMA Counter IRQ**<br>When one playback data is transferred to codec, the DMA counter subtracts 1. The counting of playback data number is used for software monitoring. If the DMA counter I2S_COUNTER [31:0] is Zero, this bit DMA_CNTER_IRQ will be set to 1.<br>0: DMA counter I2S_COUNTER [31:0] has not counted to zero.<br>1: DMA counter I2S_COUNTER [31:0] has counted down to zero.<br>The DMA_CNTER_IRQ bit is readable , and only can be clear by **write "1" to clear this bit** |
| [3] | **DMA_DATA_ZERO_IRQ** | **DMA_DATA Zero IRQ**<br>0: not found the all data bit of playback DMA is zero or its sign bit does not change (two channels).<br>1: found all the data bit of playback DMA is zero or its sign bit change (two channels).<br>The DMA_DATA_ZERO_IRQ bit is readable , and only can be clear by **write "1" to clear this bit** |

| | | |
|---|---|---|
| [2] | **P_FIFO_FULL** | **Playback FIFO Full Indicator Bit**<br>When playback FIFO is full and the playback data is written into playback FIFO, the P_FIFO_FULL bit is set to 1. This bit indicates the full error of playback FIFO is happened.<br>0: the full error of playback FIFO is not happened.<br>1: the full error of playback FIFO is happened.<br>The TP_FIFO_FULL bit is readable, and only can be clear by write “1” to this bit. |
| [1] | **P_FIFO_EMPTY** | **Playback FIFO EMPTY Indicator Bit**<br>When playback FIFO is empty and the playback data is read from playback FIFO, the P_FIFO_EMPTY bit is set to 1. This bit indicates the empty error of playback FIFO is happened.<br>0: the empty error of playback FIFO is not happened.<br>1: the empty error of playback FIFO is happened.<br>The P_FIFO_EMPTY bit is readable, and only can be clear by write “1” to this bit. |
| [0] | **P_DMA_RIA_IRQ** | **Playback DMA Reach Indicative Address Interrupt Request Bit**<br>0: Playback DMA address does not reach the specific address by P_DMA_IRQ_SEL (I2S_CON[13:12]) bits.<br>1: Playback DMA address reach the indicative address by P_DMA_IRQ_SEL (I2S_CON[13:12]) bits.<br>The P_DMA_RIA_IRQ bit is readable, and only can be clear by write “1” to this bit |

## I2S Control Register (I2S_CON)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_CON** | I2S_BA+0x028 | R/W | I2S Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:21] | **Reserved** | Reserved. |
| [20] | **SLAVE** | **I²S Slave Mode Selection Bit**<br>0 = I²S Master mode.<br>1 = I²S Slave mode.<br>The SLAVE bit is read/write |
| [19:16] | **PRS** | **I²S Frequency PRE_SCALER Selection Bits (FPLL Is the Input PLL Frequency, MCLK Is the Output Main Clock)**<br>0000 = MCLK=FPLL/1.<br>0001 = MCLK=FPLL/2.<br>0010 = MCLK=FPLL/3.<br>0011 = MCLK=FPLL/4.<br>0100 = MCLK=FPLL/5.<br>0101 = MCLK=FPLL/6.<br>0110 = MCLK=FPLL/7.<br>0111 = MCLK=FPLL/8.<br>1000 = RESERVED.<br>1001 = MCLK=FPLL/10.<br>1010 = RESERVED.<br>1011 = MCLK=FPLL/12.<br>1100 = RESERVED.<br>1101 = MCLK=FPLL/14.<br>1110 = RESERVED.<br>1111 = MCLK=FPLL/16.<br>(when the division factor is 3/5/7, the duty cycle of MCLK is not 50%, the high duration is 0.5*FPLL)<br>The PSR[3:0] bits are read/write |

| | | |
|---|---|---|
| [15:8] | **Reserved** | Reserved. |
| [7:5] | **BCLK_DIV** | **I²S Serial Data Clock Frequency Selection Bit**<br>This bit field is used to decide the relationship of frequency between PLL and I²S serial data clock. The frequency of I²S serial data clock follows the formula below:<br>$\square\square\square\square\square\square\square\square\square\square(\text{BCLK}) = \frac{\square\square\square\square\square\square\square\square\square\square(\text{PLL})}{(1 + \text{PRS}[3:0])} \times \frac{1}{(\text{BCLK\_DIV} + 1) \times 2}$ |
| [4] | **MCLK_SEL** | **MCLK Clock Selection Bit**<br>0 = I²S MCLK output will follow the PRS [3:0] setting.<br>1 = I²S MCLK output will be the same with the input frequency of PLL.<br>The MCLK_SEL bit is read/write |
| [3] | **FORMAT** | **I²S Format Selection Bit**<br>0 = I²S compatible format is selected.<br>1 = MSB-justified format is selected.<br>The FORMAT bit are read/write |
| [2:0] | **Reserved** | Reserved. |

### I2S Play DMA Down Counter Register (I2S_COUNTER)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_COUNTER** | I2S_BA+0x02C | R/W | I2S Play DMA Down Counter Register | 0xFFFF_FFFF |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **COUNTER** | **Play DMA Down Counter**<br>This bit field is used to count playback data number for software monitoring. When one playback data is transferred to codec, the DMA counter subtracts 1. If the I2S_COUNTER [31:0] register is Zero, the DMA_CNTER_IRQ (I2S_PSR[4]) will be set to 1.<br>The I2S_COUNTER [31:0] bits are read and write. |

## I2S PCM Mode Control Register (I2S_PCMCON)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_PCMCON** | I2S_BA+0x030 | R/W | I2S PCM Mode Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:28] | **PCM_MCLK_PRS** | **PCM MCLK Frequency PRE_SCALER Selection Bits (FPLL Is the Input PLL Frequency, MCLK Is the Output Main Clock)**<br>0000: PCM_MCLK=FPLL/1.<br>0001: PCM_MCLK=FPLL/2.<br>0010: PCM_MCLK=FPLL/3.<br>0011: PCM_MCLK=FPLL/4.<br>0100: PCM_MCLK=FPLL/5.<br>0101: PCM_MCLK=FPLL/6.<br>0110: PCM_MCLK=FPLL/7.<br>0111: PCM_MCLK=FPLL/8.<br>1000: RESERVED<br>1001: PCM_MCLK=FPLL/10.<br>1010: RESERVED<br>1011: PCM_MCLK=FPLL/12.<br>1100: RESERVED<br>1101: PCM_MCLK=FPLL/14.<br>1110: RESERVED<br>1111: PCM_MCLK=FPLL/16.<br>(when the division factor is 3/5/7, the duty cycle of MCLK is not 50%, the high duration is 0.5*FPLL)<br>The **PCM_MCLK_PRS** [3:0] bits are read/write |
| [27:26] | **Reserved** | Reserved. |
| [25:16] | **FS_PERIOD** | **FS Pulse Period**<br>BCLK counts between two FS pulse.<br>use this register to set sample rate<br>The register are read/write |

| | | |
|---|---|---|
| [15:8] | **PCM_PRS** | **PCM_BCLK Frequency PRE_SCALER Selection Bits**<br>**BCLK = PCM_MCLK / (2*(PCM_PRS+1)).** |
| [7:1] | **Reserved** | Reserved. |
| [0] | **BCLKP** | **BCLK Polarity**<br>0: send data at rising edge, latch data at falling edge<br>1: send data at falling edge, latch data at rising edge<br>The BCLKP bit is read/write |

## I2S PCM Mode Slot 1 Start Register (I2S_PCMS1ST)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_PCMS1ST** | I2S_BA+0x034 | R/W | I2S PCM Mode Slot 1 Start Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | **SLOT1_O_START** | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **SLOT1_O_START** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | | **SLOT1_I_START** | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **SLOT1_I_START** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:26] | **Reserved** | Reserved. |
| [25:16] | **SLOT1_O_START** | **Slot 1 Data Out Start Position**<br>This bit field is used to set the start position of slot1 output data.<br>Example.<br>For Short Frame Sync, set SLOT1_O_START to 1<br>For Long Frame Sync, set SLOT1_O_START to 0 |
| [15:10] | **Reserved** | Reserved. |
| [9:0] | **SLOT1_I_START** | **Slot 1 Data in Start Position**<br>This bit field is used to set the start position of slot1 input data.<br>Example.<br>For Short Frame Sync, set SLOT1_I_START to 1<br>For Long Frame Sync, set SLOT1_I_START to 0 |

## I2S PCM Mode Slot 2 Start Register (I2S_PCMS2ST)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_PCMS2ST** | I2S_BA+0x038 | R/W | I2S PCM Mode Slot 2 Start Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | **SLOT2_O_START** | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **SLOT2_O_START** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | | **SLOT2_I_START** | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **SLOT2_I_START** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:26] | **Reserved** | Reserved. |
| [25:16] | **SLOT2_O_START** | **Slot 2 Data Out Start Position**<br>This bit field is used to set the start position of slot2 output data. |
| [15:10] | **Reserved** | Reserved. |
| [9:0] | **SLOT2_I_START** | **Slot 2 Data in Start Position**<br>This bit field is used to set the start position of slot2 input data. |

### I2S Record DMA Destination Base Address 2 Register (I2S_RDESB2)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_RDESB2** | I2S_BA+0x040 | R/W | I2S Record DMA Destination Base Address 2 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **AUDIO_RDESB2** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **AUDIO_RDESB2** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **AUDIO_RDESB2** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **AUDIO_RDESB2** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:0] | **AUDIO_RDESB2** | **32-bit Record Destination Base Address for Right Channel**<br>This bit field indicates the record destination base address of DMA.<br>The AUDIO_RDESB2 [31:0] bits are read/write. |

### I2S Play DMA Destination Base Address 2 Register (I2S_PDESB2)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_PDESB2** | I2S_BA+0x044 | R/W | I2S Play DMA Destination Base Address 2 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **AUDIO_PDESB2** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **AUDIO_PDESB2** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **AUDIO_PDESB2** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **AUDIO_PDESB2** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:0] | **AUDIO_PDESB2** | **32-bit Play Destination Base Address for Right Channel**<br>This bit field indicates the play destination base address of DMA.<br>The AUDIO_PDESB2 [31:0] bits are read/write. |

## 5.21 Ethernet MAC Controller (EMAC)

### 5.21.1 Overview

This chip provides 2 Ethernet MAC Controller (EMAC) for Network application.

The Ethernet MAC controller consists of IEEE 802.3/Ethernet protocol engine with internal CAM function for recognizing Ethernet MAC addresses; Transmit-FIFO, Receive-FIFO, TX/RX state machine controller, time stamping engine for IEEE 1588, Magic Packet parsing engine and status controller.

The EMAC supports RMII (Reduced MII) interface to connect with external Ethernet PHY.

### 5.21.2 Features

- Supports IEEE Std. 802.3 CSMA/CD protocol
- Supports Ethernet frame time stamping for IEEE Std. 1588 – 2002 protocol
- Supports both half and full duplex for 10 Mbps or 100 Mbps operation
- Supports RMII interface
- Supports MII Management function to control external Ethernet PHY
- Supports pause and remote pause function for flow control
- Supports long frame (more than 1518 bytes) and short frame (less than 64 bytes) reception
- Supports 16 entries CAM function for Ethernet MAC address recognition
- Supports Magic Packet recognition to wake system up from power-down mode
- Supports 256 bytes transmit FIFO and 256 bytes receive FIFO
- Supports DMA function

### 5.21.3 Block Diagram

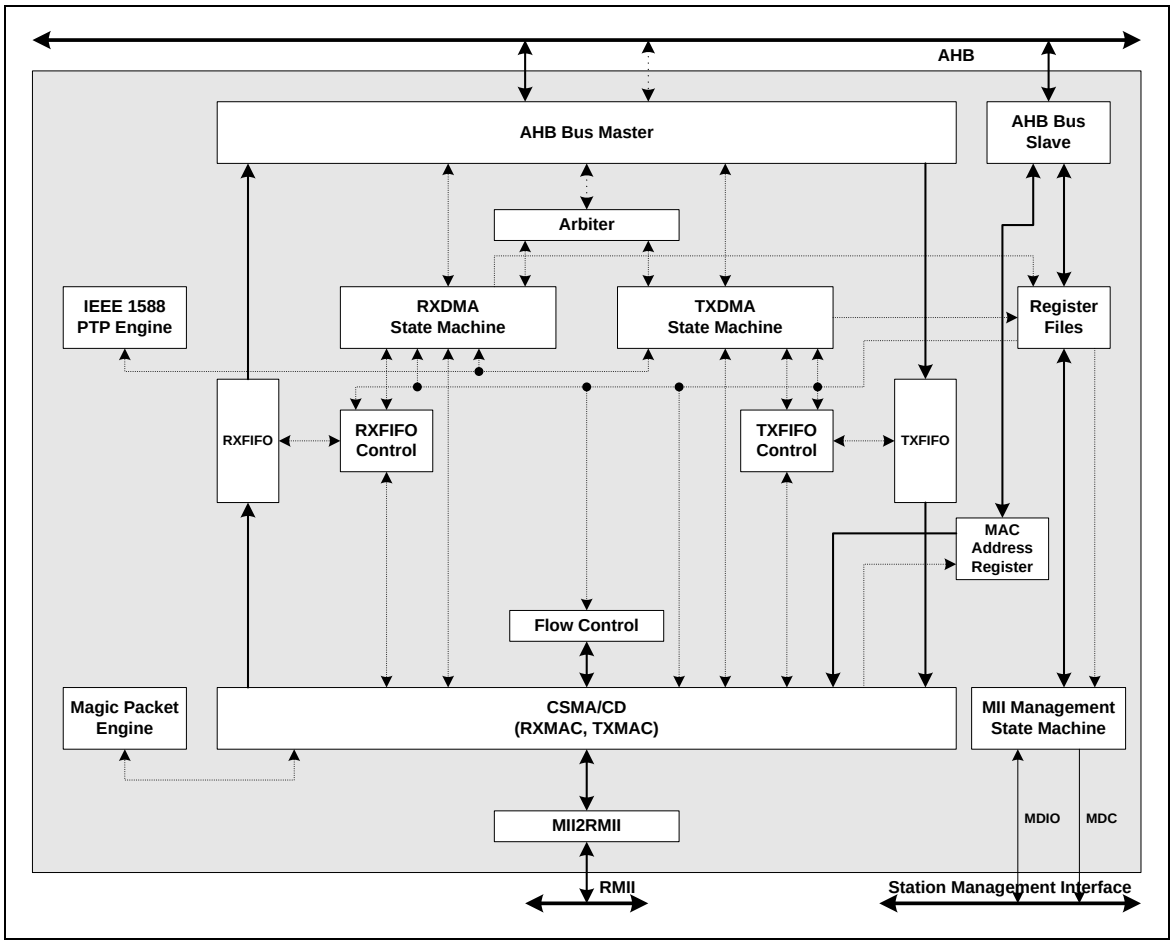



Figure 5.21-1 Ethernet MAC Controller Block Diagram

### 5.21.4 Basic Configuration

Before using EMAC functionality, it's necessary to configure I/O pins as the EMAC function and enable EMAC's clock.

Write 0x111111 to SYS_GPE_MFPL[31:8] and write 0x1111 to SYS_GPE_MFPH[15:0] configures pin PE[11:2] to be EMAC0 functionality.

Write 0x11111111 to SYS_GPF_MFPL[31:0] and write 0x11 to SYS_GPF_MFPH[7:0] configures pin PE[9:0] to be EMAC1 functionality.

To enable EMAC's clock, please refer to register CLK_HCLKEN. Set EMAC0 (CLK_HCLKEN[16]) high to enable EMAC0's clock while set EMAC1 (CLK_HCLKEN[17]) high to enable EMAC1's clock.

### 5.21.5 Functional Description

#### 5.21.5.1 *Arbiter*

In the EMAC, there are two different bus requests, RXREQ and TXREQ respectively. Arbiter does the arbitration between the RXREQ and TXREQ, and then decides which one can request the AHB bus. The arbitration results are shown below:

| RXREQ | TXREQ | Granted |
|---|---|---|
| 0 | 0 | Neither TXDMA nor RXDMA granted. |
| 0 | 1 | TXDMA granted. |
| 1 | 0 | RXDMA granted. |
| 1 | 1 | If TXFIFO valid data byte count is less than RXFIFO free space byte count, TXDMA granted. |
| 1 | 1 | If RXFIFO free space byte count is less than or equal to TXFIFO valid space byte count, RXDAM granted. |

Table 5.21-1 Arbiter Arbitration Results

#### 5.21.5.2 *TXDMA State Machine*

The TXDMA state machine transfers data from the system memory to the internal 256 bytes transmit FIFO through the AHB master. Then, the TXDMA state machine will request the transmit MAC to send the data out. During the transmission process, the TXDMA will fetch the transmit descriptor first. Through the buffer address field of the transmit descriptor, the TXDMA fetch the frame data from the system and store it into the internal 256 bytes transmit FIFO. Then, the transmit MAC will read frame data from the transmit FIFO and send the frame out. After the finish of the frame transmission, the TXDMA updates the transmit status of current frame and write the transmit descriptor back to the system memory to indicate the frame transmission has finished.

#### 5.21.5.3 *RXDMA State Machine*

The RXDMA state machine transfers data from the internal 256 bytes receiving FIFO to the system memory through AHB master. During the receiving process, the RXDMA will fetch the received descriptor first. Through the buffer address field of the received descriptor, the RXDMA will know memory space which is allocated to store the incoming frame. After the received MAC indicates there is a new incoming frame, the RXDMA starts to transfer the frame data from the internal received FIFO to the system memory. After the receiving process has finished, the RXDMA will update the receiving status of current frame and write the received descriptor back to system memory to indicate a new incoming frame is in the system memory.

#### 5.21.5.4 *Flow Control*

This block implements the flow control function while EMAC operates in the full duplex mode. The flow control function is defined in the IEEE 802.3 Std. chapter 31. The type of flow control frame defined in the IEEE 802.3 Std. is only the PAUSE frame at the moment. The control frame transmission and reception is programmable through the control registers.

To receive a control frame, software must set the bit ACP (Accept Control Packet) of register EMACn_MCMDR (MAC Command Register). While a PAUSE frame is received, the flow control

function will pause the transmission process after the current transmitting frame has been transmitted out.

To transmit a control frame out, software must program the destination MAC address of control frame into the register pair {EMACn_CAM13M, EMACn_CAM13L}, source MAC address into the register pair {EMACn_CAM14M, EMACn_CAM14L}, and configure length/type, op-code and operand of control frame into the register pair {EMACn_CAM15M, EMACn_CAM15L}, and then set the bit SDPZ (EMACn_MCMDR[16]). The bit SDPZ will be cleared while the control frame has been transmitted out.

#### 5.21.5.5 *MII Management State Machine*

The MII management function of EMAC is compliant to IEEE 802.3 Std. Through the MII management interface, software can access the control and status registers of the external PHY chip. Two programmable registers EMACn_MIID (MAC MII Management Data Register) and EMACn_MIIDA (MAC MII Management Data Control and Address Register) are for MII management function. Set the bit BUSY (EMACn_MIIDA[17]) will trigger the MII management state machine. After the MII management cycle is finished, the BUSY bit will be cleared automatically.

#### 5.21.5.6 *Media Access Control (MAC)*

The function of Ethernet MAC fully meets the requirements defined by the IEEE802.3u specification. The following paragraphs describe the frame structure and the operation of the transmission and receiving.

The transmission data frame sent from the transmit DMA will be encapsulated by the MAC before transmitting onto the MII bus. The sent data will be assembled with the preamble, the start frame delimiter (SFD), the frame check sequence and the padding for enforcing those less than 64 bytes to meet the minimum size frame and CRC sequence. The outgoing frame format will be as follows.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 110101010 --- 10101010 | 10101011 | d0 | d1 | d2 | -- | dn | Padding | CRC31 | CRC30 | --- | CRC0 |

Figure 5.21-2 Ethernet Frame Format

As mentioned by the above format, the preamble is a consecutive 7-byte long with the pattern “10101010” and the SFD is a one byte 10101011 data. The padding data will be all 0 value if the sent data frame is less than 64 bytes. The padding disable function specified in the bit P of the transmit descriptor is used to control if the MAC needs to pad data at the end of frame data or not when the transmitted data frame is less than 64 bytes. The padding data will not be appended if the padding disable bit is set to be high. The bits CRC0 ... CRC31 are the 32 bits cyclic redundancy check (CRC) sequence. The CRC encoding is defined by the following polynomial specified by the IEEE802.3. This 32 bits CRC appending function will be disabled if the Inhibit CRC of the transmission descriptor is set to high.

The MAC also performs many other transmission functions specified by the IEEE802.3, including the inter-frame spacing function, collision detection, collision enforcement, collision back off and retransmission. The collision back-off timer is a function of the integer slot time, 512 bit times. The number of slot times to delay between the current transmission attempt to the next attempt is determined by a uniformly distributed random integer algorithm specified by the IEEE802.3. The MAC performs the receive functions specified by the IEEE 802.3 including the address recognition function, the frame check sequence validation, the frame disassembly, framing and

collision filtering.

### 5.21.5.7 *Time Stamping Engine for IEEE 1588*

The EMAC supports a time stamping engine for IEEE Std. 1588. In this time stamping engine, a 64-bit counter implemented to generate the reference timing, the registers EMACn_TSSEC and ETSLSR.

In frame transmission, if TSEN (EMACn_TSCTL[0]) and TTSEN of TXDES 0 (TXDMA Descriptor Word 0) are both high, EMAC would store the 64-bit reference timing value to TXDES 1 (TXDMA Descriptor Word 1) and TXDES 2 (TXDMA Descriptor Word 2) when frame transmission completed.

In frame reception, if TSEN (EMACn_TSCTL[0]) is high, EMAC would store the 64-bit reference timing value to RXDES 1 (RXDMA Descriptor Word 1) and RXDES 3 (RXDMA Descriptor Word 3) when the frame reception finished.

The figure shown below describes how the 64-bit counter works to generate the reference timing.

The 64-bit counter formed by two 32-bit counters, the EMACn_TSSEC and EMACn_TSSUBSEC, a updated using the EMAC's input reference clock, the HCLK. Two difference methods, controlled by TSMODE (EMACn_TSCTL[3]), implemented to increase 32-bit EMACn_TSSUBSEC counter by value configured in register EMACn_TSINC. When TSMODE (EMACn_TSCTL[3]) is low, TSLSR counter increased in every clock. When TSMODE (EMACn_TSCTL[3]) is high, TSLSR counter increased only when accumulator is overflow.

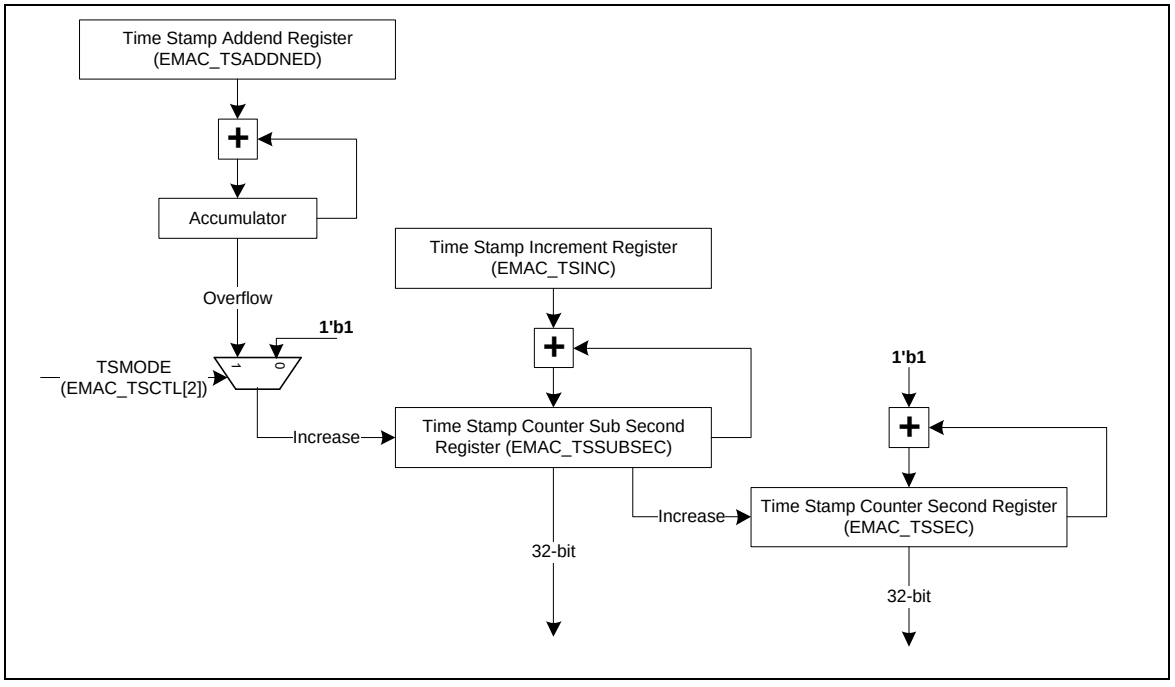



Figure 5.21-3 64-bit Reference Timing Counter

### 5.21.5.8 *Magic Packet Parsing Engine*

The EMAC supports a Magic packet parsing engine for recognizing Magic packet. The Magic packet is a broadcast frame which payload includes 6 bytes of 0xFF, followed by 16 repetitions of

48-bit MAC address defined by registers EMACn_CAM0M and EMACn_CAM0L.

The MGP_WAKE (EMACn_MCMDR[6]) controls if the Magic packet parsing engine enabled. If MGP_WAKE (EMACn_MCMDR[6]) is high, EMAC will set bit MGPR (EMACn_MISTA[15]) high to indicate Magic packet received. At the same time, EMAC generates an event to wake system up from power-down mode. If WOLIEN (EMACn_MIEN[15]) is high, EMAC generates an RX interrupt to CPU simultaneously.

### 5.21.6 DMA Descriptors Data Structure

A link-list data structure named as descriptor is used to keep the control, status and data information of each frame. Through the descriptor, CPU and EMAC exchange the information for frame reception and transmission.

Two different descriptors defined in EMAC. One named as RXDMA descriptor for frame reception and the other named as TXDMA descriptor for frame transmission. Each RXDMA or TXDMA descriptor consists of four words. The descriptor keeps the much control, status information and the details of descriptor are described below.

#### 5.21.6.1 *RXDMA Descriptor Data Structure*

The RXDMA descriptor consists of four 32-bit words. The data structure of RXDMA descriptor shown in figure below.

| | 31 | 15 | 0 |
|---|---|---|---|
| RXDES 0 | OWN | Receive Frame Status | Receive Frame Byte Count |
| RXDES 1 | Receive Frame Buffer Starting Address / Time Stamp Least Significant 32-Bit | | |
| RXDES 2 | Reserved | | |
| RXDES 3 | Next RxDMA Descriptor Starting Address / Time Stamp Most Significant 32-Bit | | |

Figure 5.21-4 RXDMA Descriptor Data Structure

### RXDES 0: RXDMA Descriptor Word 0

The RXDMA descriptor word 0 contains a descriptor ownership indicator, receive frame status, and receive frame byte count. The detail description of RXDES 0 is shown below.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Owner | Reserved | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RTSAS | RP | ALIE | RXGD | PTLE | Reserved | CRCE | RXINTR |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RBC | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RBC | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31] | **Owner** | **Ownership**<br>The ownership field defines which one, the CPU or EMAC, is the owner of each RX descriptor. Only the owner has right to modify the RX descriptor and the others can read the RX descriptor only.<br>If the O=1'b1 indicates the EMAC RXDMA is the owner of RX descriptor and the RX descriptor is available for frame reception. After the frame reception completed, EMAC RXDMA modified ownership field to 1'b0.<br>If the O=1'b0 indicates the CPU is the owner of RX descriptor. After the CPU completed the frame processing, it modified the ownership field to 1'b1 and released the RX descriptor to EMAC RXDMA.<br>0 = The owner is CPU.<br>1 = The owner is EMAC. |
| [30:24] | **Reserved** | Reserved. |
| [23] | **RTSAS** | **RX Time Stamp Active Status**<br>This bit is to indicate the time stamping circuit stamped this incoming frame successfully. When this bit set high, RX Descriptor Word 1 and RX Descriptor Word 3 keep the time stamp value recorded when this incoming frame is received completely.<br>0 = RX Descriptor Word 1 and RX Descriptor Word 3 does not keep the time stamp value.<br>1 = RX Descriptor Word 1 and RX Descriptor Word 3 keep the time stamp value. |
| [22] | **RP** | **Runt Packet**<br>The RP indicates the frame stored in the data buffer pointed by RX descriptor is a short frame (frame length is less than 64 bytes).<br>0 = The frame is not a short frame.<br>1 = The frame is a short frame. |
| [21] | **ALIE** | **Alignment Error**<br>The ALIE indicates the frame stored in the data buffer pointed by RX descriptor is not a multiple of byte.<br>0 = The frame is a multiple of byte.<br>1 = The frame is not a multiple of byte. |

| | | |
|---|---|---|
| [20] | **RXGD** | **Frame Reception Complete**<br>The RXGD indicates the frame reception has completed and stored in the data buffer pointed by RX descriptor.<br>0 = The frame reception does not complete yet.<br>1 = The frame reception completed. |
| [19] | **PTLE** | **Packet Too Long**<br>The PTLE indicates the frame stored in the data buffer pointed by RX descriptor is a long frame (frame length is greater than 1518 bytes).<br>0 = The frame is not a long frame.<br>1 = The frame is a long frame. |
| [18] | **Reserved** | Reserved. |
| [17] | **CRCE** | **CRC Error**<br>The CRCE indicates the frame stored in the data buffer pointed by RX descriptor incurred CRC error.<br>0 = The frame does not incur CRC error.<br>1 = The frame incurred CRC error. |
| [16] | **RXINTR** | **Receive Interrupt**<br>The RXINTR indicates the frame stored in the data buffer pointed by RX descriptor caused an interrupt condition.<br>0 = The frame does not cause an interrupt.<br>1 = The frame caused an interrupt. |
| [15:0] | **RBC** | **Receive Byte Count**<br>The RBC indicates the byte count of the frame stored in the data buffer pointed by RX descriptor. The four bytes CRC field is also included in the receive byte count. But if the SPCRC (EMACn_MCMDR[5]) is enabled, the four bytes CRC field will be excluded from the receive byte count. |

### RXDES 1: RXDMA Descriptor Word 1

The RXDMA descriptor word 1 contains the received frame buffer starting address or time stamp least significant 32-bit value. The detail description of RXDES 1 is shown below.

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

| Bits | Description | |
|---|---|---|
| [31:0] | **RXBSA** | **Receive Buffer Starting Address**<br>The RXBSA is the buffer starting address to store the received packet. |

| Bits | Description | |
|---|---|---|
| [31:0] | **TSLSB** | **Time Stamp Least Significant 32-bit**<br>If TSEN (EMACn_TSCR[0]) enabled, Ethernet MAC controller would store time stamp least signification 32-bit value, register EMACn_TSLSR, into this field when it writes back RX Descriptor to system memory. |

### RXDES 2: RXDMA Descriptor Word 2

The RXDMA descriptor word 2 is reserved.

| 31 | 30 | 29 | 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. |

### RXDES 3: RXDMA Descriptor Word 3

The RXDMA descriptor word 3 contains the next RXDMA descriptor starting address or time stamp most significant 32-bit value. The detail description of RXDES 3 is shown below.

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

| Bits | Description | |
|---|---|---|
| [31:0] | **NRXDSA** | **Next RX Descriptor Starting Address**<br>NRXDSA is the starting address of the next RX descriptor. When Ethernet MAC controller fetches the next RX descriptor, it ignored the bits [1:0] of NRXDSA. |

| Bits | Description | |
|---|---|---|
| [31:0] | **TSMSB** | **Time Stamp Most Significant 32-bit**<br>If TSEN (EMACn_TSCR[0]) enabled, Ethernet MAC controller would store time stamp most signification 32-bit value, register EMACn_TSMSR, into this field when it writes back RX Descriptor to system memory. |

5.21.6.2 *TXDMA Descriptor Data Structure*

The TXDMA descriptor consists of four 32-bit words. The data structure of TXDMA descriptor shown in figure below.

| | 31 | 15 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|
| TXDES 0 | OWN | Reserved | T | I | C | P |
| TXDES 1 | Transmit Frame Buffer Starting Address / Time Stamp Least Significant 32-Bit | | | | | |
| TXDES 2 | Transmit Frame Status | Transmit Frame Byte Count | | | | |
| TXDES 3 | Next TxDMA Descriptor Starting Address / Time Stamp Most Significant 32-Bit | | | | | |

Figure 5.21-5 TXDMA Descriptor Data Structure

### TXDES 0: TXDMA Descriptor Word 0

The TXDMA descriptor word 0 contains a descriptor ownership indicator. In addition, it also contains control bits for transmit frame padding, CRC append, interrupt enable and time stamping control. The detail description of TXDES 0 is shown below.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Owner | 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 | | | | TTSEN | INTEN | CRCAPP | PADEN |

| Bits | Description | |
|---|---|---|
| [31] | **Owner** | **Ownership**<br>The ownership field defines which one, the CPU or EMAC, is the owner of each TX descriptor. Only the owner has right to modify the TX descriptor and the other can read the TX descriptor only.<br>If the O=1'b1 indicates the EMAC TXDMA is the owner of TX descriptor and the TX descriptor is available for frame transmission. After the frame transmission completed, EMAC TXDMA modify ownership field to 1'b0 and return the ownership of TX descriptor to CPU.<br>If the O=1'b0 indicates the CPU is the owner of TX descriptor. After the CPU prepares new frame to wait transmission, it modifies the ownership field to 1'b1 and releases the TX descriptor to EMAC TXDMA.<br>0 = The owner is CPU.<br>1 = The owner is EMAC. |
| [30:4] | **Reserved** | Reserved. |
| [3] | **TTSEN** | **TX Time Stamp Enable Control**<br>When this bit set high and IEEE 1588 PTP function is also enabled, the embedded time stamping circuit would stamp this frame when SFD of frame is transmitted out on MII/RMII.<br>0 = IEEE 1588 time stamp function Disabled for this frame.<br>1 = IEEE 1588 time stamp function Enabled for this frame. |
| [2] | **INTEN** | **Transmit Interrupt Enable Control**<br>The INTEN controls the interrupt trigger circuit after the frame transmission completed. If the INTEN enabled, the EMAC will trigger interrupt after frame transmission completed. Otherwise, the interrupt doesn't be triggered.<br>0 = Frame transmission interrupt masked.<br>1 = Frame transmission interrupt Enabled. |

| | | |
|---|---|---|
| [1] | **CRCAPP** | **CRC Append**<br>The CRCAPP control the CRC append during frame transmission. If CRCAPP is enabled, the 4-bytes CRC checksum will be appended to frame at the end of frame transmission.<br>0 = 4-bytes CRC appending Disabled.<br>1 = 4-bytes CRC appending Enabled. |
| [0] | **PADEN** | **Padding Enable Control**<br>The PADEN control the PAD bits appending while the length of transmission frame is less than 60 bytes. If PADEN is enabled, EMAC does the padding automatically.<br>0 = PAD bits appending Disabled.<br>1 = PAD bits appending Enabled. |

### TXDES 1: TXDMA Descriptor Word 1

The TXDMA descriptor word 1 contains the transmit frame buffer starting address or time stamp least significant 32-bit value. The detail description of TXDES 1 is shown below.

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

| Bits | Description | |
|---|---|---|
| [31:2] | **TXBSA** | **Transmit Buffer Starting Address**<br>The TXBSA is the starting address of buffer where transmit packet data stored. |

| Bits | Description | |
|---|---|---|
| [31:0] | **TSLSB** | **Time Stamp Least Significant 32-bit**<br>If TSEN (EMACn_TSCR[0]) and TTSEN of TX Descriptor word 0 both enabled, Ethernet MAC controller would store time stamp least signification 32-bit value, register EMACn_TSLSR, into this field when it writes back TX Descriptor to system memory. |

### TXDES 2: TXDMA Descriptor Word 2

The TXDMA descriptor word 2 contains transmit frame status, and transmit frame byte count. The detail description of TXDES 2 is shown below.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CCNT | | | | TTSAS | SQE | PAU | TXHA |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| LC | TXABT | NCS | EXDEF | TXCP | Reserved | DEF | TXINTR |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TBC | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TBC | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **CCNT** | **Collision Count**<br>The CCNT indicates the how many collisions found consecutively during a packet transmission. If the packet incurred 16 consecutive collisions during transmission, the CCNT is 0x0 and bit TXABT is set high. |
| [27] | **TTSAS** | **TX Time Stamp Active Status**<br>This bit is to indicate the time stamping circuit stamped this frame successfully. When this bit set high, TX Descriptor Word 1 and TX Descriptor Word 3 keep the time stamp value recorded when SFD of frame is transmitted out on MII/RMII.<br>0 = TX Descriptor Word 1 and TX Descriptor Word 3 does not keep the time stamp value.<br>1 = TX Descriptor Word 1 and TX Descriptor Word 3 keep the time stamp value. |
| [26] | **SQE** | **SQE Error**<br>The SQE indicates the SQE error found at end of packet transmission on 10Mbps half-duplex mode. The SQE error check will only be done while both bit EnSQE of MCMDR is enabled and EMAC is operating on 10Mbps half-duplex mode.<br>0 = No SQE error found at end of packet transmission.<br>1 = SQE error found at end of packet transmission. |
| [25] | **PAU** | **Transmission Paused**<br>The PAU indicates the next normal packet transmission process will be paused temporally because EMAC received a PAUSE control frame, or software sets the bit SDPZ of MCMDR and enables EMAC to transmit a PAUSE control frame out.<br>0 = Next normal packet transmission process continue normally.<br>1 = Next normal packet transmission process paused. |
| [24] | **TXHA** | **Transmission Halted**<br>The TXHA indicates the next normal packet transmission process will be halted because the bit TXON of MCMDR is disabled by software.<br>0 = Next normal packet transmission process continue normally.<br>1 = Next normal packet transmission process halted. |

| | | |
|---|---|---|
| [23] | **LC** | **Late Collision**<br>The LC indicates the collision found in the outside of 64 bytes collision window. This means after the 64 bytes of a frame has been transmitted out to the network, the collision still found. The late collision check will only be done while EMAC is operating on half-duplex mode.<br>0 = No collision found in the outside of 64 bytes collision window.<br>1 = Collision found in the outside of 64 bytes collision window. |
| [22] | **TXABT** | **Transmission Abort**<br>The TXABT indicates the packet incurred 16 consecutive collisions during transmission, and then the transmission process for this packet is aborted. The transmission abort is only available while EMAC is operating on half-duplex mode.<br>0 = Packet does not incur 16 consecutive collisions during transmission.<br>1 = Packet incurred 16 consecutive collisions during transmission. |
| [21] | **NCS** | **No Carrier Sense**<br>The NCS indicates the MII I/F signal CRS does not active at the start of or during the packet transmission. The NCS is only available while EMAC is operating on half-duplex mode.<br>0 = CRS signal does not active at the start of or during the packet transmission.<br>1 = CRS signal actives correctly. |
| [20] | **EXDEF** | **Defer Exceed**<br>The EXDEF indicates the frame waiting for transmission has deferred over 0.32768ms on 100Mbps mode, or 3.2768ms on 10Mbps mode. The deferral exceed check will only be done while bit NDEF of MCMDR is disabled, and EMAC is operating on half-duplex mode.<br>0 = Frame waiting for transmission did not defer over 0.32768ms (100Mbps) or 3.2768ms (10Mbps).<br>1 = Frame waiting for transmission deferred over 0.32768ms (100Mbps) or 3.2768ms (10Mbps). |
| [19] | **TXCP** | **Transmission Complete**<br>The TXCP indicates the packet transmission has completed correctly.<br>0 = The packet transmission does not complete.<br>1 = The packet transmission completed. |
| [18] | **Reserved** | Reserved. |
| [17] | **DEF** | **Transmission Deferred**<br>The DEF indicates the packet transmission has deferred once. The DEF is only available while EMAC is operating on half-duplex mode.<br>0 = Packet transmission does not defer.<br>1 = Packet transmission deferred once. |
| [16] | **TXINTR** | **Transmit Interrupt**<br>The TXINTR indicates the packet transmission would trigger an interrupt condition.<br>0 = The packet transmission would not trigger an interrupt.<br>1 = The packet transmission would trigger an interrupt. |
| [15:0] | **TBC** | **Transmit Byte Count**<br>The TBC indicates the byte count of the frame stored in the data buffer pointed by TX descriptor for transmission. |

### TXDES 3: TXDMA Descriptor Word 3

The TXDMA descriptor word 3 contains the next TXDMA descriptor starting address or time stamp most significant 32-bit value. The detail description of TXDES 3 is shown below.

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

| Bits | Description | |
|---|---|---|
| [31:0] | **NTXDSA** | **Next TX Descriptor Starting Address**<br>NTXDSA is the starting address of the next TX descriptor. When Ethernet MAC controller fetch the next TX descriptor, it ignored the bits [1:0] of NTXDSA. |

| Bits | Description | |
|---|---|---|
| [31:0] | **TSMSB** | **Time Stamp Most Significant 32-bit**<br>If TSEN (EMACn_TSCR[0]) and TTSEN of TX Descriptor word 0 are both enabled, Ethernet MAC controller would store time stamp most signification 32-bit value, register EMACn_TSMSR, into this field when it writes back TX Descriptor to system memory. |

### 5.21.7 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMAC Base Address:**<br>**EMAC0_BA = 0xB000_2000**<br>**EMAC1_BA = 0xB000_3000** | | | | |
| **EMACn_CAMCMR**<br>**n=0,1** | EMACn_BA+0x000 | R/W | EMAC n CAM Command Register | 0x0000_0000 |
| **EMACn_CAMEN**<br>**n=0,1** | EMACn_BA+0x004 | R/W | EMAC n CAM Enable Register | 0x0000_0000 |
| **EMACn_CAM0M**<br>**n=0,1** | EMACn_BA+0x008 | R/W | EMAC n CAM 0 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM0L**<br>**n=0,1** | EMACn_BA+0x00C | R/W | EMAC n CAM 0 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM1M**<br>**n=0,1** | EMACn_BA+0x010 | R/W | EMAC n CAM 1 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM1L**<br>**n=0,1** | EMACn_BA+0x014 | R/W | EMAC n CAM 1 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM2M**<br>**n=0,1** | EMACn_BA+0x018 | R/W | EMAC n CAM 2 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM2L**<br>**n=0,1** | EMACn_BA+0x01C | R/W | EMAC n CAM 2 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM3M**<br>**n=0,1** | EMACn_BA+0x020 | R/W | EMAC n CAM 3 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM3L**<br>**n=0,1** | EMACn_BA+0x024 | R/W | EMAC n CAM 3 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM4M**<br>**n=0,1** | EMACn_BA+0x028 | R/W | EMAC n CAM 4 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM4L**<br>**n=0,1** | EMACn_BA+0x02C | R/W | EMAC n CAM 4 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM5M**<br>**n=0,1** | EMACn_BA+0x030 | R/W | EMAC n CAM 5 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM5L**<br>**n=0,1** | EMACn_BA+0x034 | R/W | EMAC n CAM 5 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM6M**<br>**n=0,1** | EMACn_BA+0x038 | R/W | EMAC n CAM 6 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM6L**<br>**n=0,1** | EMACn_BA+0x03C | R/W | EMAC n CAM 6 Least Significant Word Register | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **EMACn_CAM7M n=0,1** | EMACn_BA+0x040 | R/W | EMAC n CAM 7 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM7L n=0,1** | EMACn_BA+0x044 | R/W | EMAC n CAM 7 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM8M n=0,1** | EMACn_BA+0x048 | R/W | EMAC n CAM 8 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM8L n=0,1** | EMACn_BA+0x04C | R/W | EMAC n CAM 8 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM9M n=0,1** | EMACn_BA+0x050 | R/W | EMAC n CAM 9 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM9L n=0,1** | EMACn_BA+0x054 | R/W | EMAC n CAM 9 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM10M n=0,1** | EMACn_BA+0x058 | R/W | EMAC n CAM 10 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM10L n=0,1** | EMACn_BA+0x05C | R/W | EMAC n CAM 10 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM11M n=0,1** | EMACn_BA+0x060 | R/W | EMAC n CAM 11 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM11L n=0,1** | EMACn_BA+0x064 | R/W | EMAC n CAM 11 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM12M n=0,1** | EMACn_BA+0x068 | R/W | EMAC n CAM 12 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM12L n=0,1** | EMACn_BA+0x06C | R/W | EMAC n CAM 12 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM13M n=0,1** | EMACn_BA+0x070 | R/W | EMAC n CAM 13 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM13L n=0,1** | EMACn_BA+0x074 | R/W | EMAC n CAM 13 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM14M n=0,1** | EMACn_BA+0x078 | R/W | EMAC n CAM 14 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM14L n=0,1** | EMACn_BA+0x07C | R/W | EMAC n CAM 14 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM15M n=0,1** | EMACn_BA+0x080 | R/W | EMAC n CAM 15 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM15L n=0,1** | EMACn_BA+0x084 | R/W | EMAC n CAM 15 Least Significant Word Register | 0x0000_0000 |
| **EMACn_TXDLSA n=0,1** | EMACn_BA+0x088 | R/W | EMAC n Transmit Descriptor Link List Start Address Register | 0xFFFF_FFFC |
| **EMACn_RXDLSA n=0,1** | EMACn_BA+0x08C | R/W | EMAC n Receive Descriptor Link List Start Address Register | 0xFFFF_FFFC |

| | | | | |
|---|---|---|---|---|
| **EMACn_MCMDR n=0,1** | EMACn_BA+0x090 | R/W | EMAC n MAC Command Register | 0x0040_0000 |
| **EMACn_MIID n=0,1** | EMACn_BA+0x094 | R/W | EMAC n MII Management Data Register | 0x0000_0000 |
| **EMACn_MIIDA n=0,1** | EMACn_BA+0x098 | R/W | EMAC n MII Management Control and Address Register | 0x0090_0000 |
| **EMACn_FFTCR n=0,1** | EMACn_BA+0x09C | R/W | EMAC n FIFO Threshold Control Register | 0x0000_0101 |
| **EMACn_TSDR n=0,1** | EMACn_BA+0x0A0 | W | EMAC n Transmit Start Demand Register | Undefined |
| **EMACn_RSDR n=0,1** | EMACn_BA+0x0A4 | W | EMAC n Receive Start Demand Register | Undefined |
| **EMACn_DMARFC n=0,1** | EMACn_BA+0x0A8 | R/W | EMAC n Maximum Receive Frame Control Register | 0x0000_0800 |
| **EMACn_MIEN n=0,1** | EMACn_BA+0x0AC | R/W | EMAC n MAC Interrupt Enable Register | 0x0000_0000 |
| **EMACn_MISTA n=0,1** | EMACn_BA+0x0B0 | R/W | EMAC n MAC Interrupt Status Register | 0x0000_0000 |
| **EMACn_MGSTA n=0,1** | EMACn_BA+0x0B4 | R/W | EMAC n MAC General Status Register | 0x0000_0000 |
| **EMACn_MPCNT n=0,1** | EMACn_BA+0x0B8 | R/W | EMAC n Missed Packet Count Register | 0x0000_7FFF |
| **EMACn_MRPC n=0,1** | EMACn_BA+0x0BC | R | EMAC n MAC Receive Pause Count Register | 0x0000_0000 |
| **EMACn_DMARFS n=0,1** | EMACn_BA+0x0C8 | R/W | EMAC n DMA Receive Frame Status Register | 0x0000_0000 |
| **EMACn_CTXDSA n=0,1** | EMACn_BA+0x0CC | R | EMAC n Current Transmit Descriptor Start Address Register | 0x0000_0000 |
| **EMACn_CTXBSA n=0,1** | EMACn_BA+0x0D0 | R | EMAC n Current Transmit Buffer Start Address Register | 0x0000_0000 |
| **EMACn_CRXDSA n=0,1** | EMACn_BA+0x0D4 | R | EMAC n Current Receive Descriptor Start Address Register | 0x0000_0000 |
| **EMACn_CRXBSA n=0,1** | EMACn_BA+0x0D8 | R | EMAC n Current Receive Buffer Start Address Register | 0x0000_0000 |
| **EMACn_TSCTL n=0,1** | EMACn_BA+0x100 | R/W | EMAC n Time Stamp Control Register | 0x0000_0000 |
| **EMACn_TSSEC n=0,1** | EMACn_BA+0x110 | R/W | EMAC n Time Stamp Counter Second Register | 0x0000_0000 |
| **EMACn_TSSUBSEC n=0,1** | EMACn_BA+0x114 | R/W | EMAC n Time Stamp Counter Sub Second Register | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **EMACn_TSINC n=0,1** | EMACn_BA+0x118 | R/W | EMAC n Time Stamp Increment Register | 0x0000_0000 |
| **EMACn_TSADDEND n=0,1** | EMACn_BA+0x11C | R/W | EMAC n Time Stamp Addend Register | 0x0000_0000 |
| **EMACn_UPDSEC n=0,1** | EMACn_BA+0x120 | R/W | EMAC n Time Stamp Update Second Register | 0x0000_0000 |
| **EMACn_UPDSUBSEC n=0,1** | EMACn_BA+0x124 | R/W | EMAC n Time Stamp Update Sub Second Register | 0x0000_0000 |
| **EMACn_ALMSEC n=0,1** | EMACn_BA+0x128 | R/W | EMAC n Time Stamp Alarm Second Register | 0x0000_0000 |
| **EMACn_ALMSUBSEC n=0,1** | EMACn_BA+0x12C | R/W | EMAC n Time Stamp Alarm Sub Second Register | 0x0000_0000 |

### 5.21.8 Register Description

## EMAC n CAM Command Register (EMACn_CAMCMR)

The EMAC supports CAM function for destination MAC address recognition. The EMACn_CAMCMR control the CAM comparison function, and unicast, multicast, and broadcast packet reception.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_CAMCMR**<br>**n=0,1** | EMACn_BA+0x000 | R/W | EMAC n CAM 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** | | | **ECMP** | **CCCAM** | **ABP** | **AMP** | **AUP** |

| Bits | Description | |
|---|---|---|
| [31:5] | **Reserved** | Reserved. |
| [4] | **ECMP** | **CAM Compare Enable Control**<br>The ECMP controls the enable of CAM comparison function for destination MAC address recognition. If software wants to receive a packet with specific destination MAC address, configures the MAC address into CAM 12~0, then enables that CAM entry and set ECMP to 1.<br>0 = CAM comparison function for destination MAC address recognition disabled.<br>1 = CAM comparison function for destination MAC address recognition enabled. |
| [3] | **CCAM** | **Complement CAM Compare**<br>The CCAM controls the complement of the CAM comparison result. If the ECMP and CCAM are both enabled, the incoming packet with specific destination MAC address configured in CAM entry will be dropped. And the incoming packet with destination MAC address does not configured in any CAM entry will be received.<br>0 = The CAM comparison result does not complement.<br>1 = The CAM comparison result complemented. |
| [2] | **ABP** | **Accept Broadcast Packet**<br>The ABP controls the broadcast packet reception. If ABP is enabled, EMAC receives all incoming packet its destination MAC address is a broadcast address.<br>0 = EMAC receives packet depends on the CAM comparison result.<br>1 = EMAC receives all broadcast packets. |

| | | |
|---|---|---|
| [1] | **AMP** | **Accept Multicast Packet**<br>The AMP controls the multicast packet reception. If AMP is enabled, EMAC receives all incoming packet its destination MAC address is a multicast address.<br>0 = EMAC receives packet depends on the CAM comparison result.<br>1 = EMAC receives all multicast packets. |
| [0] | **AUP** | **Accept Unicast Packet**<br>The AUP controls the unicast packet reception. If AUP is enabled, EMAC receives all incoming packet its destination MAC address is a unicast address.<br>0 = EMAC receives packet depends on the CAM comparison result.<br>1 = EMAC receives all unicast packets. |

**CAMCMR Setting and Comparison Results**

The following table is the address recognition result in different CAMCMR configuration. The column Result shows the incoming packet type that can pass the address recognition in specific CAM configuration. The C, U, M and B represents the:

C: Indicates the destination MAC address of incoming packet has been configured in CAM entry.

U: Indicates the incoming packet is a unicast packet.

M: Indicates the incoming packet is a multicast packet.

B: Indicates the incoming packet is a broadcast packet.

| ECMP | CCAM | AUP | AMP | ABP | Result | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | No Packet | | | |
| 0 | 0 | 0 | 0 | 1 | B | | | |
| 0 | 0 | 0 | 1 | 0 | M | | | |
| 0 | 0 | 0 | 1 | 1 | M | B | | |
| 0 | 0 | 1 | 0 | 0 | C | U | | |
| 0 | 0 | 1 | 0 | 1 | C | U | B | |
| 0 | 0 | 1 | 1 | 0 | C | U | M | |
| 0 | 0 | 1 | 1 | 1 | C | U | M | B |
| 0 | 1 | 0 | 0 | 0 | C | U | M | B |
| 0 | 1 | 0 | 0 | 1 | C | U | M | B |
| 0 | 1 | 0 | 1 | 0 | C | U | M | B |
| 0 | 1 | 0 | 1 | 1 | C | U | M | B |
| 0 | 1 | 1 | 0 | 0 | C | U | M | B |
| 0 | 1 | 1 | 0 | 1 | C | U | M | B |
| 0 | 1 | 1 | 1 | 0 | C | U | M | B |
| 0 | 1 | 1 | 1 | 1 | C | U | M | B |
| 1 | 0 | 0 | 0 | 0 | C | | | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | C | B | | |
| 1 | 0 | 0 | 1 | 0 | C | M | | |
| 1 | 0 | 0 | 1 | 1 | C | M | B | |
| 1 | 0 | 1 | 0 | 0 | C | U | | |
| 1 | 0 | 1 | 0 | 1 | C | U | B | |
| 1 | 0 | 1 | 1 | 0 | C | U | M | |
| 1 | 0 | 1 | 1 | 1 | C | U | M | B |
| 1 | 1 | 0 | 0 | 0 | U | M | B | |
| 1 | 1 | 0 | 0 | 1 | U | M | B | |
| 1 | 1 | 0 | 1 | 0 | U | M | B | |
| 1 | 1 | 0 | 1 | 1 | U | M | B | |
| 1 | 1 | 1 | 0 | 0 | C | U | M | B |
| 1 | 1 | 1 | 0 | 1 | C | U | M | B |
| 1 | 1 | 1 | 1 | 0 | C | U | M | B |
| 1 | 1 | 1 | 1 | 1 | C | U | M | B |

Table 5.21-2 Different CAMCMR Setting and Type of Received Packet

### EMAC n CAM Enable Register (EMACn_CAMEN)

The EMACn_CAMEN controls the validation of each CAM entry. Each CAM entry must be enabled first before it participates in the destination MAC address recognition.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_CAMEN n=0,1** | EMACn_BA+0x004 | R/W | EMAC n CAM 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** |
| **CAM15EN** | **CAM14EN** | **CAM13EN** | **CAM12EN** | **CAM11EN** | **CAM10EN** | **CAM9EN** | **CAM8EN** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **CAM7EN** | **CAM6EN** | **CAM5EN** | **CAM4EN** | **CAM3EN** | **CAM2EN** | **CAM1EN** | **CAM0EN** |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **CAMxEN** | **CAM Entry X Enable Control**<br>The CAMxEN controls the validation of CAM entry x. The x can be 0 to 15.<br>The CAM entry 13, 14 and 15 are for PAUSE control frame transmission. If software wants to transmit a PAUSE control frame out to network, the enable bits of these three CAM entries all must be enabled first.<br>0 = CAM entry x Disabled.<br>1 = CAM entry x Enabled. |

## EMAC n CAM x Most Significant Word Register (EMACn_CAMxM)

The EMAC is equipped with 16 CAM entries. In these 16 CAM entries, 13 entries (entry 0~12) are to keep destination MAC address for packet recognition, and the other 3 entries (entry 13~15) are for PAUSE control frame transmission. Each CAM entry consists of 6 bytes. Consequently, 2 register are used for each CAM entry.

For packet recognition, a register pair {EMACn_CAMxM, EMACn_CAMxL} represents a CAM entry and can keep a destination MAC address. The corresponding CAM enable bit CAMxEN (EMACn_CAMEN[x]) is also needed be enabled. The x can be the 0 to 12.

The register pairs {EMACn_CAM13M, EMACn_CAM13L}, {EMACn_CAM14M, EMACn_CAM14L} and {EMACn_CAM15M, EMACn_CAM15L} are used for flow control function.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_CAM0M n=0,1** | EMACn_BA+0x008 | R/W | EMAC n CAM 0 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM1M n=0,1** | EMACn_BA+0x010 | R/W | EMAC n CAM 1 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM2M n=0,1** | EMACn_BA+0x018 | R/W | EMAC n CAM 2 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM3M n=0,1** | EMACn_BA+0x020 | R/W | EMAC n CAM 3 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM4M n=0,1** | EMACn_BA+0x028 | R/W | EMAC n CAM 4 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM5M n=0,1** | EMACn_BA+0x030 | R/W | EMAC n CAM 5 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM6M n=0,1** | EMACn_BA+0x038 | R/W | EMAC n CAM 6 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM7M n=0,1** | EMACn_BA+0x040 | R/W | EMAC n CAM 7 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM8M n=0,1** | EMACn_BA+0x048 | R/W | EMAC n CAM 8 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM9M n=0,1** | EMACn_BA+0x050 | R/W | EMAC n CAM 9 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM10M n=0,1** | EMACn_BA+0x058 | R/W | EMAC n CAM 10 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM11M n=0,1** | EMACn_BA+0x060 | R/W | EMAC n CAM 11 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM12M n=0,1** | EMACn_BA+0x068 | R/W | EMAC n CAM 12 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM13M n=0,1** | EMACn_BA+0x070 | R/W | EMAC n CAM 13 Most Significant Word Register | 0x0000_0000 |
| **EMACn_CAM14M n=0,1** | EMACn_BA+0x078 | R/W | EMAC n CAM 14 Most Significant Word Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **CAMxM** | **CAMx Most Significant Word**<br>The CAMxM keeps the bit 47~16 of MAC address. The x can be the 0~12. The register pair {EMACn_CAMxM, EMACn_CAMxL} represents a CAM entry and keeps a MAC address.<br>For example, if the MAC address 00-50-BA-33-BA-44 kept in CAM entry 1, the register EMACn_CAM1M is 32'h0050_BA33 and EMACn_CAM1L is 32'hBA44_0000. |

### EMAC n CAM x Least Significant Word Register (EMACn_CAMxL)

The EMAC is equipped with 16 CAM entries. In these 16 CAM entries, 13 entries (entry 0~12) are to keep destination MAC address for packet recognition, and the other 3 entries (entry 13~15) are for PAUSE control frame transmission. Each CAM entry consists of 6 bytes. Consequently, 2 register are used for each CAM entry.

For packet recognition, a register pair {EMACn_CAMxM, EMACn_CAMxL} represents a CAM entry and can keep a destination MAC address. The corresponding CAM enable bit CAMxEN (EMACn_CAMEN[x]) is also needed be enabled. The x can be the 0 to 12.

The register pairs {EMACn_CAM13M, EMACn_CAM13L}, {EMACn_CAM14M, EMACn_CAM14L} and {EMACn_CAM15M, EMACn_CAM15L} are used for flow control function.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_CAM0L n=0,1** | EMACn_BA+0x00C | R/W | EMAC n CAM 0 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM1L n=0,1** | EMACn_BA+0x014 | R/W | EMAC n CAM 1 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM2L n=0,1** | EMACn_BA+0x01C | R/W | EMAC n CAM 2 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM3L n=0,1** | EMACn_BA+0x024 | R/W | EMAC n CAM 3 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM4L n=0,1** | EMACn_BA+0x02C | R/W | EMAC n CAM 4 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM5L n=0,1** | EMACn_BA+0x034 | R/W | EMAC n CAM 5 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM6L n=0,1** | EMACn_BA+0x03C | R/W | EMAC n CAM 6 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM7L n=0,1** | EMACn_BA+0x044 | R/W | EMAC n CAM 7 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM8L n=0,1** | EMACn_BA+0x04C | R/W | EMAC n CAM 8 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM9L n=0,1** | EMACn_BA+0x054 | R/W | EMAC n CAM 9 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM10L n=0,1** | EMACn_BA+0x05C | R/W | EMAC n CAM 10 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM11L n=0,1** | EMACn_BA+0x064 | R/W | EMAC n CAM 11 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM12L n=0,1** | EMACn_BA+0x06C | R/W | EMAC n CAM 12 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM13L n=0,1** | EMACn_BA+0x074 | R/W | EMAC n CAM 13 Least Significant Word Register | 0x0000_0000 |
| **EMACn_CAM14L n=0,1** | EMACn_BA+0x07C | R/W | EMAC n CAM 14 Least Significant Word Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **CAMxL** | **CAMx Least Significant Word**<br>The CAMxL keeps the bit 15~0 of MAC address. The x can be the 0~14. The register pair {EMACn_CAMxM, EMACn_CAMxL} represents a CAM entry and keeps a MAC address.<br>For example, if the MAC address 00-50-BA-33-BA-44 kept in CAM entry 1, the register EMACn_CAM1M is 32'h0050_BA33 and EMACn_CAM1L is 32'hBA44_0000. |
| [15:0] | **Reserved** | Reserved. |

## EMAC n CAM 15 Most Significant Word Register (EMACn_CAM15M)

The EMAC is equipped with 16 CAM entries. In these 16 CAM entries, 13 entries (entry 0~12) are to keep destination MAC address for packet recognition, and the other 3 entries (entry 13~15) are for PAUSE control frame transmission. Each CAM entry consists of 6 bytes. Consequently, 2 register are used for each CAM entry.

For packet recognition, a register pair {EMACn_CAMxM, EMACn_CAMxL} represents a CAM entry and can keep a destination MAC address. The corresponding CAM enable bit CAMxEN (EMACn_CAMEN[x]) is also needed be enabled. The x can be the 0 to 12.

The register pairs {EMACn_CAM13M, EMACn_CAM13L}, {EMACn_CAM14M, EMACn_CAM14L} and {EMACn_CAM15M, EMACn_CAM15L} are used for flow control function.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_CAM15M n=0,1** | EMACn_BA+0x080 | R/W | EMAC n CAM 15 Most Significant Word Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **Length_Type** | **Length/Type Field of PAUSE Control Frame**<br>In the PAUSE control frame, a length/type field defined and is 16"h8808. |
| [15:0] | **OP_Code** | **OP Code Field of PAUSE Control Frame**<br>In the PAUSE control frame, an op code field defined and is 16"h0001. |

## EMAC n CAM 15 Least Significant Word Register (EMACn_CAM15L)

The EMAC is equipped with 16 CAM entries. In these 16 CAM entries, 13 entries (entry 0~12) are to keep destination MAC address for packet recognition, and the other 3 entries (entry 13~15) are for PAUSE control frame transmission. Each CAM entry consists of 6 bytes. Consequently, 2 register are used for each CAM entry.

For packet recognition, a register pair {EMACn_CAMxM, EMACn_CAMxL} represents a CAM entry and can keep a destination MAC address. The corresponding CAM enable bit CAMxEN (EMACn_CAMEN[x]) is also needed be enabled. The x can be the 0 to 12.

The register pairs {EMACn_CAM13M, EMACn_CAM13L}, {EMACn_CAM14M, EMACn_CAM14L} and {EMACn_CAM15M, EMACn_CAM15L} are used for flow control function.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_CAM15L**<br>**n=0,1** | EMACn_BA+0x084 | R/W | EMAC n CAM 15 Least Significant Word Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Operand** | | | | | | | |
| **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] | **Operand** | **Pause Parameter**<br>In the PAUSE control frame, an operand field defined and controls how much time the destination Ethernet MAC Controller paused. The unit of the operand is a slot time, the 512 bits time. |
| [23:0] | **Reserved** | Reserved. |

## EMAC n Transmit Descriptor Link List Start Address Register (EMACn_TXDLSA)

The TX descriptor defined in EMAC is a link-list data structure. The EMACn_TXDLSA keeps the starting address of this link-list. In other words, the EMACn_TXDLSA keeps the starting address of the 1st TX descriptor. EMACn_TXDLSA must be configured by software before the bit TXON (EMACn_MCMDR[8]) is enabled.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_TXDLSA**<br>**n=0,1** | EMACn_BA+0x088 | R/W | EMAC n Transmit Descriptor Link List Start Address Register | 0xFFFF_FFFC |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **TXDLSA** | **Transmit Descriptor Link-list Start Address**<br>The TXDLSA keeps the start address of transmit descriptor link-list. If the software enables the bit TXON (EMACn_MCMDR[8]), the content of TXDLSA will be loaded into the current transmit descriptor start address register (EMACn_CTXDSA). The TXDLSA does not be updated by EMAC. During the operation, EMAC will ignore the bits [1:0] of TXDLSA. This means that each TX descriptor always must locate at word boundary memory address. |

## EMAC n Receive Descriptor Link List Start Address Register (EMACn_RXDLSA)

The RX descriptor defined in EMAC is a link-list data structure. The EMACn_RXDLSA keeps the starting address of this link-list. In other words, the EMACn_RXDLSA keeps the starting address of the 1st RX descriptor. EMACn_RXDLSA must be configured by software before the bit RXON (EMACn_MCMDR[0]) is enabled.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_RXDLSA n=0,1** | EMACn_BA+0x08C | R/W | EMAC n Receive Descriptor Link List Start Address Register | 0xFFFF_FFFC |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **RXDLSA** | **Receive Descriptor Link-list Start Address**<br>The RXDLSA keeps the start address of receive descriptor link-list. If the S/W enables the bit RXON (EMACn_MCMDR[0]), the content of RXDLSA will be loaded into the current receive descriptor start address register (EMACn_CRXDSA). The RXDLSA does not be updated by EMAC. During the operation, EMAC will ignore the bits [1:0] of RXDLSA. This means that each RX descriptor always must locate at word boundary memory address. |

## EMAC n MAC Command Register (EMACn_MCMDR)

The EMACn_MCMDR provides the control information for EMAC. Some command settings affect both frame transmission and reception, such as bit FDUP (EMACn_MCMDR[18]), the full/half duplex mode selection, or bit OPMOD (EMACn_MCMDR[20]), the 100/10M bps mode selection. Some command settings control frame transmission and reception separately, likes bit TXON (EMACn_MCMDR[8]) and RXON (EMACn_MCMDR[0]).

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_MCMDR**<br>**n=0,1** | EMACn_BA+0x090 | R/W | EMAC n MAC Command Register | 0x0040_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | SWR |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| REFCLKINV | Reserved | | OPMOD | Reserved | FDUP | SQECHKEN | SDPZ |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | NDEF | TXON |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PTP_SRC | MGP_WAKE | SPCRC | AEP | ACP | ARP | ALP | RXON |

| Bits | Description | |
|---|---|---|
| [31:25] | **Reserved** | Reserved. |
| [24] | **SWR** | **Software Reset**<br>The SWR implements a reset function to make the EMAC return default state. The SWR is a self-clear bit. This means after the software reset finished, the SWR will be cleared automatically. Enable SWR can also reset all control and status registers, exclusive of the control bits EnRMII (EMACn_MCMDR[22]), LBK (EMACn_MCMDR[21]) and OPMOD (EMACn_MCMDR[20]).<br>The EMAC re-initial is necessary after the software reset completed.<br>0 = Software reset completed.<br>1 = Software reset Enabled. |
| [23] | **REFCLKINV** | **REFCLK Inverted Control**<br>This bit controls if RMIIx_REFCLK from external PHY is inverted before using by EMAC.<br>0 = RMIIx_REFCLK not inverted.<br>1 = RMIIx_REFCLK inverted. |
| [22:21] | **Reserved** | Reserved. |
| [20] | **OPMOD** | **Operation Mode Selection**<br>The OPMOD defines that if the EMAC is operating on 10M or 100M bps mode. The SWR would not affect OPMOD value.<br>0 = EMAC operates in 10Mbps mode.<br>1 = EMAC operates in 100Mbps mode. |

| | | |
|---|---|---|
| [19] | **Reserved** | Reserved. |
| [18] | **FDUP** | **Full Duplex Mode Selection**<br>The FDUP controls that if EMAC is operating on full or half duplex mode.<br>0 = EMAC operates in half duplex mode.<br>1 = EMAC operates in full duplex mode. |
| [17] | **SQECHKEN** | **SQE Checking Enable Control**<br>The SQECHKEN controls the enable of SQE checking. The SQE checking is only available while EMAC is operating on 10M bps and half duplex mode. In other words, the SQECHKEN cannot affect EMAC operation, if the EMAC is operating on 100M bps or full duplex mode.<br>0 = SQE checking Disabled while EMAC is operating in 10Mbps and Half Duplex mode.<br>1 = SQE checking Enabled while EMAC is operating in 10Mbps and Half Duplex mode. |
| [16] | **SDPZ** | **Send PAUSE Frame**<br>The SDPZ controls the PAUSE control frame transmission.<br>If S/W wants to send a PAUSE control frame out, the CAM entry 13, 14 and 15 must be configured first and the corresponding CAM enable bit of CAMEN register also must be set. Then, set SDPZ to 1 enables the PAUSE control frame transmission.<br>The SDPZ is a self-clear bit. This means after the PAUSE control frame transmission has completed, the SDPZ will be cleared automatically.<br>It is recommended that only enabling SPDZ while EMAC is operating in Full Duplex mode.<br>0 = PAUSE control frame transmission completed.<br>1 = PAUSE control frame transmission Enabled. |
| [15:10] | **Reserved** | Reserved. |
| [9] | **NDEF** | **No Deferral**<br>The NDEF controls the enable of deferral exceed counter. If NDEF is set to high, the deferral exceed counter is disabled. The NDEF is only useful while EMAC is operating on half duplex mode.<br>0 = The deferral exceed counter Enabled.<br>1 = The deferral exceed counter Disabled. |
| [8] | **TXON** | **Frame Transmission ON**<br>The TXON controls the normal packet transmission of EMAC. If the TXON is set to high, the EMAC starts the packet transmission process, including the TX descriptor fetching, packet transmission and TX descriptor modification.<br>It is must to finish EMAC initial sequence before enable TXON. Otherwise, the EMAC operation is undefined.<br>If the TXON is disabled during EMAC is transmitting a packet out, the EMAC stops the packet transmission process after the current packet transmission finished.<br>0 = Packet transmission process stopped.<br>1 = Packet transmission process started. |
| [7] | **PTP_SRC** | **PTP Counter Source Selection**<br>This bit control the PTP counter source is from EMC0 or from EMC1 internally.<br>1'b0: The PTP counter source is from EMC1 internally<br>1'b1: The PTP counter source is from EMC0.<br>**Note:** This bit is only available in EMAC1. In EMAC0, this bit is reserved. |

| | | |
|---|---|---|
| [6] | **MGP_WAKE** | **Magic Packet Wake-up Enable Control**<br>The MGP_WAKE high enables the functionality that Ethernet MAC controller checked if the incoming packet is Magic Packet and wakeup system from Power-down mode.<br>If incoming packet was a Magic Packet and the system was in Power-down, the Ethernet MAC controller would generate a wakeup event to wake system up from Power-down mode.<br>0 = Wake-up by Magic Packet function Disabled.<br>1 = Wake-up by Magic Packet function Enabled. |
| [5] | **SPCRC** | **Strip CRC Checksum**<br>The SPCRC controls if the length of incoming packet is calculated with 4 bytes CRC checksum. If the SPCRC is set to high, 4 bytes CRC checksum is excluded from length calculation of incoming packet.<br>0 = The 4 bytes CRC checksum is included in packet length calculation.<br>1 = The 4 bytes CRC checksum is excluded in packet length calculation. |
| [4] | **AEP** | **Accept CRC Error Packet**<br>The AEP controls the EMAC accepts or drops the CRC error packet. If the AEP is set to high, the incoming packet with CRC error will be received by EMAC as a good packet.<br>0 = Ethernet MAC controller dropped the CRC error packet.<br>1 = Ethernet MAC controller received the CRC error packet. |
| [3] | **ACP** | **Accept Control Packet**<br>The ACP controls the control frame reception. If the ACP is set to high, the EMAC will accept the control frame. Otherwise, the control frame will be dropped. It is recommended that S/W only enable ACP while EMAC is operating on full duplex mode.<br>0 = Ethernet MAC controller dropped the control frame.<br>1 = Ethernet MAC controller received the control frame. |
| [2] | **ARP** | **Accept Runt Packet**<br>The ARP controls the runt packet, which length is less than 64 bytes, reception. If the ARP is set to high, the EMAC will accept the runt packet.<br>Otherwise, the runt packet will be dropped.<br>0 = Ethernet MAC controller dropped the runt packet.<br>1 = Ethernet MAC controller received the runt packet. |
| [1] | **ALP** | **Accept Long Packet**<br>The ALP controls the long packet, which packet length is greater than 1518 bytes, reception. If the ALP is set to high, the EMAC will accept the long packet.<br>Otherwise, the long packet will be dropped.<br>0 = Ethernet MAC controller dropped the long packet.<br>1 = Ethernet MAC controller received the long packet. |
| [0] | **RXON** | **Frame Reception ON**<br>The RXON controls the normal packet reception of EMAC. If the RXON is set to high, the EMAC starts the packet reception process, including the RX descriptor fetching, packet reception and RX descriptor modification.<br>It is necessary to finish EMAC initial sequence before enable RXON. Otherwise, the EMAC operation is undefined.<br>If the RXON is disabled during EMAC is receiving an incoming packet, the EMAC stops the packet reception process after the current packet reception finished.<br>0 = Packet reception process stopped.<br>1 = Packet reception process started. |

## EMAC n MII Management Data Register (EMACn_MIID)

The EMAC provides MII management function to access the control and status registers of the external PHY. The EMACn_MIID register is used to store the data that will be written into the registers of external PHY for write command or the data that is read from the registers of external PHY for read command.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_MIID n=0,1** | EMACn_BA+0x094 | R/W | EMAC n MII Management 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** |
| **DATA** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **DATA** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **DATA** | **MII Management Data**<br>The MIIData is the 16 bits data that will be written into the registers of external PHY for MII Management write command or the data from the registers of external PHY for MII Management read command. |

## EMAC n MII Management Control and Address Register (EMACn_MIIDA)

The EMAC provides MII management function to access the control and status registers of the external PHY. The EMACn_MIIDA register is used to keep the MII management command information, like the register address, external PHY address, MDC clocking rate, read/write etc.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_MIIDA n=0,1** | EMACn_BA+0x098 | R/W | EMAC n MII Management Control and Address Register | 0x0090_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | **MDCON** | **PREAMSP** | **BUSY** | **WRITE** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | **PHYAD** | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | **PHYRAD** | | | | |

| Bits | Description | |
|---|---|---|
| [31:20] | **Reserved** | Reserved. |
| [19] | **MDCON** | **MDC Clock ON Always**<br>The MDC controls the MDC clock generation. If the MDCON is set to high, the MDC clock actives always. Otherwise, the MDC will only active while S/W issues a MII management command.<br>0 = MDC clock only actives while S/W issues a MII management command.<br>1 = MDC clock actives always. |
| [18] | **PREAMSP** | **Preamble Suppress**<br>The PREAMSP controls the preamble field generation of MII management frame. If the PREAMSP is set to high, the preamble field generation of MII management frame is skipped.<br>0 = Preamble field generation of MII management frame not skipped.<br>1 = Preamble field generation of MII management frame skipped. |
| [17] | **BUSY** | **Busy Bit**<br>The BUSY controls the enable of the MII management frame generation. If S/W wants to access registers of external PHY, it set BUSY to high and EMAC generates the MII management frame to external PHY through MII Management I/F. The BUSY is a self-clear bit. This means the BUSY will be cleared automatically after the MII management command finished.<br>0 = MII management command generation finished.<br>1 = MII management command generation Enabled. |

| | | |
|---|---|---|
| [16] | **WRITE** | **Write Command**<br>The WRITE defines the MII management command is a read or write.<br>0 = MII management command is a read command.<br>1 = MII management command is a write command. |
| [15:13] | **Reserved** | Reserved. |
| [12:8] | **PHYAD** | **PHY Address**<br>The PHYAD keeps the address to differentiate which external PHY is the target of the MII management command. |
| [7:5] | **Reserved** | Reserved. |
| [4:0] | **PHYRAD** | **PHY Register Address**<br>The PHYRAD keeps the address to indicate which register of external PHY is the target of the MII management command. |

### MII Management Function Frame Format

In IEEE Std. 802.3 clause 22.2.4, the MII management function is defined. The MII management function is used for the purpose of controlling the PHY and gathering status from the PHY. The MII management frame format is shown as follow.

| | **Management Frame Fields** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **PRE** | **ST** | **OP** | **PHYAD** | **REGAD** | **TA** | **DATA** | **IDLE** |
| **READ** | **1…1** | **01** | **10** | **AAAAA** | **RRRRR** | **Z0** | **DDDDDDDDDDDDDDDD** | **Z** |
| **WRITE** | **1…1** | **01** | **01** | **AAAAA** | **RRRRR** | **10** | **DDDDDDDDDDDDDDDD** | **Z** |

Figure 5.21-6 MII Management Frame Format

### MII Management Function Configure Sequence

| **Read** | **Write** |
|---|---|
| 1. Set appropriate EMACn_MDCCR. | 1. Write data to EMACn_MIID register |
| 2. Set PHYAD and PHYRAD. | 2. Set appropriate EMACn_MDCCR. |
| 3. Set Write to 1"b0 | 3. Set PHYAD and PHYRAD. |
| 4. Set bit BUSY (EMACn_MIIDA[17]) to 1"b1 to send a MII management frame out. | 4. Set Write to 1"b1 |
| 5. Wait BUSY (EMACn_MIIDA[17]) to become 1"b0. | 5. Set bit BUSY (EMACn_MIIDA[17]) to 1"b1 to send a MII management frame out. |
| 6. Read data from EMACn_MIID register. | 6. Wait BUSY (EMACn_MIIDA[17]) to become 1"b0. |
| 7. Finish the read command. | 7. Finish the write command. |

Table 5.21-3 MII Management Function Configure Sequence

## EMAC n FIFO Threshold Control Register (EMACn_FFTCR)

The EMACn_FFTCR defines the high and low threshold of internal FIFOs, including TXFIFO and RXFIFO. The threshold of internal FIFOs is related to EMAC request generation and when the frame transmission starts. The EMACn_FFTCR also defines the burst length of AHB bus cycle for system memory access.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_FFTCR n=0,1** | EMACn_BA+0x09C | R/W | EMAC n FIFO Threshold Control Register | 0x0000_0101 |

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

| Bits | Description | |
|---|---|---|
| [31:22] | **Reserved** | Reserved. |
| [21:20] | **BURSTLEN** | **DMA Burst Length**<br>This defines the burst length of AHB bus cycle while EMAC accesses system memory.<br>00 = 16 words.<br>01 = 16 words.<br>10 = 8 words.<br>11 = 4 words. |
| [19:10] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [9:8] | **TXTHD** | **TXFIFO Low Threshold**<br>The TXTHD controls when TXDMA requests internal arbiter for data transfer between system memory and TXFIFO. The TXTHD defines not only the low threshold of TXFIFO, but also the high threshold. The high threshold is the twice of low threshold always. During the packet transmission, if the TXFIFO reaches the high threshold, the TXDMA stops generate request to transfer frame data from system memory to TXFIFO. If the frame data in TXFIFO is less than low threshold, TXDMA starts to transfer frame data from system memory to TXFIFO.<br>The TXTHD also defines when the TXMAC starts to transmit frame out to network. The TXMAC starts to transmit the frame out while the TXFIFO first time reaches the high threshold during the transmission of the frame. If the frame data length is less than TXFIFO high threshold, the TXMAC starts to transmit the frame out after the frame data are all inside the TXFIFO.<br>00 = Undefined.<br>10 = TXFIFO low threshold is 80B and high threshold is 160B.<br>01 = TXFIFO low threshold is 64B and high threshold is 128B.<br>11 = TXFIFO low threshold is 96B and high threshold is 192B. |
| [7:2] | **Reserved** | Reserved. |
| [1:0] | **RXTHD** | **RXFIFO Low Threshold**<br>The RXTHD controls when RXDMA requests internal arbiter for data transfer between RXFIFO and system memory. The RXTHD defines not only the high threshold of RXFIFO, but also the low threshold. The low threshold is the half of high threshold always. During the packet reception, if the RXFIFO reaches the high threshold, the RXDMA starts to transfer frame data from RXFIFO to system memory. If the frame data in RXFIFO is less than low threshold, RXDMA stops to transfer the frame data to system memory.<br>00 = Depend on the burst length setting. If the burst length is 8 words, high threshold is 8 words, too.<br>01 = RXFIFO high threshold is 64B and low threshold is 32B.<br>10 = RXFIFO high threshold is 128B and low threshold is 64B.<br>11 = RXFIFO high threshold is 192B and low threshold is 96B. |

## EMAC n Transmit Start Demand Register (EMACn_TSDR)

S/W issues a write command to EMACn_TSDR register to make TXDMA to leave Halt state and continue the frame transmission.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_TSDR n=0,1** | EMACn_BA+0x0 A0 | W | EMAC n Transmit Start Demand Register | Undefined |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **TSD** | **Transmit Start Demand**<br>If the TX descriptor is not available for use of TXDMA after the TXON (EMACn_MCMDR[8]) is enabled, the FSM (Finite State Machine) of TXDMA enters the Halt state and the frame transmission is halted. After the S/W has prepared the new TX descriptor for frame transmission, it must issue a write command to EMACn_TSDR register to make TXDMA to leave Halt state and continue the frame transmission.<br>The EMACn_TSDR is a write only register and the value read from this register is undefined.<br>The write to EMACn_TSDR register takes effect only when TXDMA stayed at Halt state. |

### EMAC n Receive Start Demand Register (EMACn_RSDR)

S/W issues a write command to EMACn_RSDR register to make RXDMA to leave Halt state and continue the frame reception.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_RSDR n=0,1** | EMACn_BA+0x0A4 | W | EMAC n Receive Start Demand Register | Undefined |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **RSD** | **Receive Start Demand**<br>If the RX descriptor is not available for use of RXDMA after the RXON (EMACn_MCMDR[0]) is enabled, the FSM (Finite State Machine) of RXDMA enters the Halt state and the frame reception is halted. After the S/W has prepared the new RX descriptor for frame reception, it must issue a write command to EMACn_RSDR register to make RXDMA to leave Halt state and continue the frame reception.<br>The EMACn_RSDR is a write only register and the value read from this register is undefined.<br>The write to EMACn_RSDR register takes effect only when RXDMA stayed at Halt state. |

## EMAC n Maximum Receive Frame Control Register (EMACn_DMARFC)

The EMACn_DMARFC defines the maximum frame length for a received frame that can be stored in the system memory. It is recommend that only use this register while S/W wants to receive a frame which length is greater than 1518 bytes.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_DMARFC n=0,1** | EMACn_BA+0x0A8 | R/W | EMAC n Maximum Receive Frame Control Register | 0x0000_0800 |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **RXMS** | **Maximum Receive Frame Length**<br>The RXMS defines the maximum frame length for received frame. If the frame length of received frame is greater than RXMS, and bit MFLEIEN (EMACn_MIEN[8]) is also enabled, the bit DFOI (EMACn_MISTA[8]) is set and the RX interrupt is triggered.<br>It is recommended that only use RXMS to qualify the length of received frame while S/W wants to receive a frame which length is greater than 1518 bytes. |

## EMAC n MAC Interrupt Enable Register (EMACn_MIEN)

The EMACn_MIEN controls the enable of EMAC interrupt status to generate interrupt. Two interrupts, RXINTR for frame reception and TXINTR for frame transmission, are generated from EMAC to CPU.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_MIEN n=0,1** | EMACn_BA+0x0AC | R/W | EMAC n MAC Interrupt Enable Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | TSALMIEN | Reserved | | | TXBEIEN |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TDUIEN | LCIEN | TXABTIEN | NCSIEN | EXDEFIEN | TXCPIEN | TXUDIEN | TXIEN |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| WOLIEN | CFRIEN | Reserved | | RXBEIEN | RDUIEN | DENIEN | MFLEIEN |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MMPIEN | RPIEN | ALIEIEN | RXGDIEN | LPIEN | RXOVIEN | CRCEIEN | RXIEN |

| Bits | Description | |
|---|---|---|
| [31:29] | **Reserved** | Reserved. |
| [28] | **TSALMIEN** | **Time Stamp Alarm Interrupt Enable Control**<br>The TSALMIEN controls the TSALS (EMACn_MISTA[28]) interrupt generation. If TSALS (EMACn_MISTA[28]) is set, and both TSALMIEN and TXIEN (EMACn_MIEN[16]) enabled, the EMAC generates the TX interrupt to CPU. If TSALMIEN or TXIEN (EMACn_MIEN[16]) disabled, no TX interrupt generated to CPU even the TSALS (EMACn_MISTA[28]) is set.<br>0 = TSALS (EMACn_MISTA[28]) trigger TX interrupt Disabled.<br>1 = TSALS (EMACn_MISTA[28]) trigger TX interrupt Enabled. |
| [27:25] | **Reserved** | Reserved. |
| [24] | **TXBEIEN** | **Transmit Bus Error Interrupt Enable Control**<br>The TXBEIEN controls the TXBERR (EMACn_MISTA[24]) interrupt generation. If TXBERR (EMACn_MISTA[24]) is set, and both TXBEIEN and TXIEN (EMACn_MIEN[16]) are enabled, the EMAC generates the TX interrupt to CPU. If TXBEIEN or TXIEN (EMACn_MIEN[16]) is disabled, no TX interrupt is generated to CPU even the TXBERR (EMACn_MISTA[24]) is set.<br>0 = TXBERR (EMACn_MISTA[24]) trigger TX interrupt Disabled.<br>1 = TXBERR (EMACn_MISTA[24]) trigger TX interrupt Enabled. |

| | | |
|---|---|---|
| [23] | **TDUIEN** | **Transmit Descriptor Unavailable Interrupt Enable Control**<br>The TDUIEN controls the TDU (EMACn_MISTA[23]) interrupt generation. If TDU (EMACn_MISTA[23]) is set, and both TDUIEN and TXIEN (EMACn_MIEN[16]) are enabled, the EMAC generates the TX interrupt to CPU. If TDUIEN or TXIEN (EMACn_MIEN[16]) is disabled, no TX interrupt is generated to CPU even the TDU (EMACn_MISTA[23]) is set.<br>0 = TDU (EMACn_MISTA[23]) trigger TX interrupt Disabled.<br>1 = TDU (EMACn_MISTA[23]) trigger TX interrupt Enabled. |
| [22] | **LCIEN** | **Late Collision Interrupt Enable Control**<br>The LCIEN controls the LC (EMACn_MISTA[22]) interrupt generation. If LC (EMACn_MISTA[22]) is set, and both LCIEN and TXIEN (EMACn_MIEN[16]) are enabled, the EMAC generates the TX interrupt to CPU. If LCIEN or TXIEN (EMACn_MIEN[16]) is disabled, no TX interrupt is generated to CPU even the LC (EMACn_MISTA[22]) is set.<br>0 = LC (EMACn_MISTA[22]) trigger TX interrupt Disabled.<br>1 = LC (EMACn_MISTA[22]) trigger TX interrupt Enabled. |
| [21] | **ExTXABT** | **Transmit Abort Interrupt Enable Control**<br>The TXABTIEN controls the TXABT (EMACn_MISTA[21]) interrupt generation. If TXABT (EMACn_MISTA[21]) is set, and both TXABTIEN and TXIEN (EMACn_MIEN[16]) are enabled, the EMAC generates the TX interrupt to CPU. If TXABTIEN or TXIEN (EMACn_MIEN[16]) is disabled, no TX interrupt is generated to CPU even the TXABT (EMACn_MISTA[21]) is set.<br>0 = TXABT (EMACn_MISTA[21]) trigger TX interrupt Disabled.<br>1 = TXABT (EMACn_MISTA[21]) trigger TX interrupt Enabled. |
| [20] | **NCSIEN** | **No Carrier Sense Interrupt Enable Control**<br>The NCSIEN controls the NCS (EMACn_MISTA[20]) interrupt generation. If NCS (EMACn_MISTA[20]) is set, and both NCSIEN and TXIEN (EMACn_MIEN[16]) are enabled, the EMAC generates the TX interrupt to CPU. If NCSIEN or TXIEN (EMACn_MIEN[16]) is disabled, no TX interrupt is generated to CPU even the NCS (EMACn_MISTA[20]) is set.<br>0 = NCS (EMACn_MISTA[20]) trigger TX interrupt Disabled.<br>1 = NCS (EMACn_MISTA[20]) trigger TX interrupt Enabled. |
| [19] | **EXDEFIEN** | **Defer Exceed Interrupt Enable Control**<br>The EXDEFIEN controls the EXDEF (EMACn_MISTA[19]) interrupt generation. If EXDEF (EMACn_MISTA[19]) is set, and both EXDEFIEN and TXIEN (EMACn_MIEN[16]) are enabled, the EMAC generates the TX interrupt to CPU. If EXDEFIEN or TXIEN (EMACn_MIEN[16]) is disabled, no TX interrupt is generated to CPU even the EXDEF (EMACn_MISTA[19]) is set.<br>0 = EXDEF (EMACn_MISTA[19]) trigger TX interrupt Disabled.<br>1 = EXDEF (EMACn_MISTA[19]) trigger TX interrupt Enabled. |
| [18] | **TXCPIEN** | **Transmit Completion Interrupt Enable Control**<br>The TXCPIEN controls the TXCP (EMACn_MISTA[18]) interrupt generation. If TXCP (EMACn_MISTA[18]) is set, and both TXCPIEN and TXIEN (EMACn_MIEN[16]) are enabled, the EMAC generates the TX interrupt to CPU. If TXCPIEN or TXIEN (EMACn_MIEN[16]) is disabled, no TX interrupt is generated to CPU even the TXCP (EMACn_MISTA[18]) is set.<br>0 = TXCP (EMACn_MISTA[18]) trigger TX interrupt Disabled.<br>1 = TXCP (EMACn_MISTA[18]) trigger TX interrupt Enabled. |

| | | |
|---|---|---|
| [17] | **TXUDIEN** | **Transmit FIFO Underflow Interrupt Enable Control**<br>The TXUDIEN controls the TXEMP (EMACn_MISTA[17]) interrupt generation. If TXEMP (EMACn_MISTA[17]) is set, and both TXUDIEN and TXIEN (EMACn_MIEN[16]) are enabled, the EMAC generates the TX interrupt to CPU. If TXUDIEN or TXIEN (EMACn_MIEN[16]) is disabled, no TX interrupt is generated to CPU even the TXEMP (EMACn_MISTA[17]) is set.<br>0 = TXEMP (EMACn_MISTA[17]) trigger TX interrupt Disabled.<br>1 = TXEMP (EMACn_MISTA[17]) trigger TX interrupt Enabled. |
| [16] | **TXIEN** | **Transmit Interrupt Enable Control**<br>The TXIEN controls the TX interrupt generation.<br>If TXIEN is enabled and TXINTR (EMACn_MISTA[16]) is high, EMAC generates the TX interrupt to CPU. If TXIEN is disabled, no TX interrupt is generated to CPU even any status bit of EMACn_MISTA[24:17] is set and the corresponding bit of EMACn_MIEN is enabled. In other words, if S/W wants to receive TX interrupt from EMAC, this bit must be enabled. And, if S/W doesn‟t want to receive any TX interrupt from EMAC, disables this bit.<br>0 = TXINTR (EMACn_MISTA[16]) is masked and TX interrupt generation Dsiabled.<br>1 = TXINTR (EMACn_MISTA[16]) is not masked and TX interrupt generation Enabled. |
| [15] | **WOLIEN** | **Magic Packet Receive Interrupt Enable Control**<br>The WOLIEN controls the MPR (EMACn_MISTA[15]) interrupt generation. If MPR (EMACn_MISTA[15]) is set, and both WOLIEN and RXIEN (EMACn_MIEN[0]) are enabled, the EMAC generates the RX interrupt to CPU. If WOLIEN or RXIEN (EMACn_MIEN[0]) is disabled, no RX interrupt is generated to CPU even the MPR (EMACn_MISTA[15]) is set.<br>0 = MPR (EMACn_MISTA[15]) trigger RX interrupt Disabled.<br>1 = MPR (EMACn_MISTA[15]) trigger RX interrupt Enabled. |
| [14] | **CFRIEN** | **Control Frame Receive Interrupt Enable Control**<br>The CFRIEN controls the CFR (EMACn_MISTA[14]) interrupt generation. If CFR (EMACn_MISTA[14]) is set, and both CFRIEN and RXIEN (EMACn_MIEN[0]) are enabled, the EMAC generates the RX interrupt to CPU. If CFRIEN or RXIEN (EMACn_MIEN[0]) is disabled, no RX interrupt is generated to CPU even the CFR (EMACn_MISTA[14]) is set.<br>0 = CFR (EMACn_MISTA[14]) trigger RX interrupt Disabled.<br>1 = CFR (EMACn_MISTA[14]) trigger RX interrupt Enabled. |
| [13:12] | **Reserved** | Reserved. |
| [11] | **RXBEIEN** | **Receive Bus Error Interrupt Enable Control**<br>The RXBEIEN controls the RXBERR (EMACn_MISTA[11]) interrupt generation. If RXBERR (EMACn_MISTA[11]) is set, and both RXBEIEN and RXIEN (EMACn_MIEN[0]) are enabled, the EMAC generates the RX interrupt to CPU. If RXBEIEN or RXIEN (EMACn_MIEN[0]) is disabled, no RX interrupt is generated to CPU even the RXBERR (EMACn_MISTA[11]) is set.<br>0 = RXBERR (EMACn_MISTA[11]) trigger RX interrupt Disabled.<br>1 = RXBERR (EMACn_MISTA[11]) trigger RX interrupt Enabled. |
| [10] | **RDUIEN** | **Receive Descriptor Unavailable Interrupt Enable Control**<br>The RDUIEN controls the RDU (EMACn_MISTA[10]) interrupt generation. If RDU (EMACn_MISTA[10]) is set, and both RDUIEN and RXIEN (EMACn_MIEN[0]) are enabled, the EMAC generates the RX interrupt to CPU. If RDUIEN or RXIEN (EMACn_MIEN[0]) is disabled, no RX interrupt is generated to CPU even the RDU (EMACn_MISTA[10]) is set.<br>0 = RDU (EMACn_MISTA[10]) trigger RX interrupt Disabled.<br>1 = RDU (EMACn_MISTA[10]) trigger RX interrupt Enabled. |

| | | |
|---|---|---|
| [9] | **DENIEN** | **DMA Early Notification Interrupt Enable Control**<br>The DENIEN controls the DENI (EMACn_MISTA[9]) interrupt generation. If DENI (EMACn_MISTA[9]) is set, and both DENIEN and RXIEN (EMACn_MIEN[0]) are enabled, the EMAC generates the RX interrupt to CPU. If DENIEN or RXIEN (EMACn_MIEN[0]) is disabled, no RX interrupt is generated to CPU even the DENI (EMACn_MISTA[9]) is set.<br>0 = TDENI (EMACn_MISTA[9]) trigger RX interrupt Disabled.<br>1 = DENI (EMACn_MISTA[9]) trigger RX interrupt Enabled. |
| [8] | **MFLEIEN** | **Maximum Frame Length Exceed Interrupt Enable Control**<br>The MFLEIEN controls the DFOI (EMACn_MISTA[8]) interrupt generation. If DFOI (EMACn_MISTA[8]) is set, and both MFLEIEN and RXIEN (EMACn_MIEN[0]) are enabled, the EMAC generates the RX interrupt to CPU. If MFLEIEN or RXIEN (EMACn_MIEN[0]) is disabled, no RX interrupt is generated to CPU even the DFOI (EMACn_MISTA[8]) is set.<br>0 = DFOI (EMACn_MISTA[8]) trigger RX interrupt Disabled.<br>1 = DFOI (EMACn_MISTA[8]) trigger RX interrupt Enabled. |
| [7] | **MMPIEN** | **Miss Packet Counter Overrun Interrupt Enable Control**<br>The MMPIEN controls the MMP (EMACn_MISTA[7]) interrupt generation. If MMP (EMACn_MISTA[7]) is set, and both MMPIEN and RXIEN (EMACn_MIEN[0]) are enabled, the EMAC generates the RX interrupt to CPU. If MMPIEN or RXIEN (EMACn_MIEN[0]) is disabled, no RX interrupt is generated to CPU even the MMP (EMACn_MISTA[7]) is set.<br>0 = MMP (EMACn_MISTA[7]) trigger RX interrupt Disabled.<br>1 = MMP (EMACn_MISTA[7]) trigger RX interrupt Enabled. |
| [6] | **RPIEN** | **Runt Packet Interrupt Enable Control**<br>The RPIEN controls the RP (EMACn_MISTA[6]) interrupt generation. If RP (EMACn_MISTA[6]) is set, and both RPIEN and RXIEN (EMACn_MIEN[0]) are enabled, the EMAC generates the RX interrupt to CPU. If RPIEN or RXIEN (EMACn_MIEN[0]) is disabled, no RX interrupt is generated to CPU even the RP (EMACn_MISTA[6]) is set.<br>0 = RP (EMACn_MISTA[6]) trigger RX interrupt Disabled.<br>1 = RP (EMACn_MISTA[6]) trigger RX interrupt Enabled. |
| [5] | **ALIEIEN** | **Alignment Error Interrupt Enable Control**<br>The ALIEIEN controls the ALIE (EMACn_MISTA[5]) interrupt generation. If ALIE (EMACn_MISTA[5]) is set, and both ALIEIEN and RXIEN (EMACn_MIEN[0]) are enabled, the EMAC generates the RX interrupt to CPU. If ALIEIEN or RXIEN (EMACn_MIEN[0]) is disabled, no RX interrupt is generated to CPU even the ALIE (EMACn_MISTA[5]) is set.<br>0 = ALIE (EMACn_MISTA[5]) trigger RX interrupt Disabled.<br>1 = ALIE (EMACn_MISTA[5]) trigger RX interrupt Enabled. |
| [4] | **RXGDIEN** | **Receive Good Interrupt Enable Control**<br>The RXGDIEN controls the RXGD (EMACn_MISTA[4]) interrupt generation. If RXGD (EMACn_MISTA[4]) is set, and both RXGDIEN and RXIEN (EMACn_MIEN[0]) are enabled, the EMAC generates the RX interrupt to CPU. If RXGDIEN or RXIEN (EMACn_MIEN[0]) is disabled, no RX interrupt is generated to CPU even the RXGD (EMACn_MISTA[4]) is set.<br>0 = RXGD (EMACn_MISTA[4]) trigger RX interrupt Disabled.<br>1 = RXGD (EMACn_MISTA[4]) trigger RX interrupt Enabled. |

| | | |
|---|---|---|
| [3] | **LPIEN** | **Long Packet Interrupt Enable Control**<br>The LPIEN controls the PTLE (EMACn_MISTA[3]) interrupt generation. If PTLE (EMACn_MISTA[3]) is set, and both LPIEN and RXIEN (EMACn_MIEN[0]) are enabled, the EMAC generates the RX interrupt to CPU. If LPIEN or RXIEN (EMACn_MIEN[0]) is disabled, no RX interrupt is generated to CPU even the PTLE (EMACn_MISTA[3]) is set.<br>0 = PTLE (EMACn_MISTA[3]) trigger RX interrupt Disabled.<br>1 = PTLE (EMACn_MISTA[3]) trigger RX interrupt Enabled. |
| [2] | **RXOVIEN** | **Receive FIFO Overflow Interrupt Enable Control**<br>The RXOVIEN controls the RXOV (EMACn_MISTA[2]) interrupt generation. If RXOV (EMACn_MISTA[2]) is set, and both RXOVIEN and RXIEN (EMACn_MIEN[0]) are enabled, the EMAC generates the RX interrupt to CPU. If RXOVIEN or RXIEN (EMACn_MIEN[0]) is disabled, no RX interrupt is generated to CPU even the RXOV (EMACn_MISTA[2]) is set.<br>0 = RXOV (EMACn_MISTA[2]) trigger RX interrupt Disabled.<br>1 = RXOV (EMACn_MISTA[2]) trigger RX interrupt Enabled. |
| [1] | **CRCEIEN** | **CRC Error Interrupt Enable Control**<br>The CRCEIEN controls the CRCE (EMACn_MISTA[1]) interrupt generation. If CRCE (EMACn_MISTA[1]) is set, and both CRCEIEN and RXIEN (EMACn_MIEN[0]) are enabled, the EMAC generates the RX interrupt to CPU. If CRCEIEN or RXIEN (EMACn_MIEN[0]) is disabled, no RX interrupt is generated to CPU even the CRCE (EMACn_MISTA[1]) is set.<br>0 = CRCE (EMACn_MISTA[1]) trigger RX interrupt Disabled.<br>1 = CRCE (EMACn_MISTA[1]) trigger RX interrupt Enabled. |
| [0] | **RXIEN** | **Receive Interrupt Enable Control**<br>The RXIEN controls the RX interrupt generation.<br>If RXIEN is enabled and RXINTR (EMACn_MISTA[0]) is high, EMAC generates the RX interrupt to CPU. If RXIEN is disabled, no RX interrupt is generated to CPU even any status bit EMACn_MISTA[15:1] is set and the corresponding bit of EMACn_MIEN is enabled. In other words, if S/W wants to receive RX interrupt from EMAC, this bit must be enabled. And, if S/W doesn"t want to receive any RX interrupt from EMAC, disables this bit.<br>0 = RXINTR (EMACn_MISTA[0]) is masked and RX interrupt generation Disabled.<br>1 = RXINTR (EMACn_MISTA[0]) is not masked and RX interrupt generation Enabled. |

## EMAC n MAC Interrupt Status Register (EMACn_MISTA)

The EMACn_MISTA keeps much EMAC statuses, such as frame transmission, reception status and internal FIFO status. The statuses kept in EMACn_MISTA will trigger the reception or transmission interrupt. The EMACn_MISTA is a write clear register and write 1 to corresponding bit clears the status and also clears the interrupt.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_MISTA n=0,1** | EMACn_BA+0x0B0 | R/W | EMAC n MAC Interrupt Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | TSALS | Reserved | | | TXBERR |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TDU | LC | TXABT | NCS | EXDEF | TXCP | TXEMP | TXINTR |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MGPR | CFR | Reserved | | RXBERR | RDU | DENI | DFOI |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MMP | RP | ALIE | RXGD | PTLE | RXOV | CRCE | RXINTR |

| Bits | Description | |
|---|---|---|
| [31:29] | **Reserved** | Reserved. |
| [28] | **TSALS** | **Time Stamp Alarm Interrupt**<br>The TSALS high indicates the EMACn_TSMSR register value equals to EMACn_TSMSAR register and EMACn_TSLSR register value equals to register EMACn_TSMLSR.<br>If TSALS is high and TXALMIEN (EMACn_MIEN[28]) enabled, the TXINTR will be high. Write 1 to this bit clears the TSALS status.<br>0 = EMACn_TSMSR did not equal EMACn_TSMSAR or EMACn_TSLSR did not equal EMACn_TSLSAR.<br>1 = EMACn_TSMSR equals EMACn_TSMSAR and EMACn_TSLSR equals EMACn_TSLSAR. |
| [27:25] | **Reserved** | Reserved. |
| [24] | **TXBERR** | **Transmit Bus Error Interrupt**<br>The TXBERR high indicates the memory controller replies ERROR response while EMAC access system memory through TXDMA during packet transmission process. Reset EMAC is recommended while TXBERR status is high.<br>If the TXBERR is high and TXBEIEN (EMACn_MIEN[24]) is enabled, the TXINTR will be high. Write 1 to this bit clears the TXBERR status.<br>0 = No ERROR response is received.<br>1 = ERROR response is received. |

| | | |
|---|---|---|
| [23] | **TDU** | **Transmit Descriptor Unavailable Interrupt**<br>The TDU high indicates that there is no available TX descriptor for packet transmission and TXDMA will stay at Halt state. Once, the TXDMA enters the Halt state, S/W must issues a write command to TSDR register to make TXDMA leave Halt state while new TX descriptor is available.<br>If the TDU is high and TDUIEN (EMACn_MIEN[23]) is enabled, the TXINTR will be high. Write 1 to this bit clears the TDU status.<br>0 = TX descriptor is available.<br>1 = TX descriptor is unavailable. |
| [22] | **LC** | **Late Collision Interrupt**<br>The LC high indicates the collision occurred in the outside of 64 bytes collision window. This means after the 64 bytes of a frame has been transmitted out to the network, the collision still occurred. The late collision check will only be done while EMAC is operating on half-duplex mode. If the LC is high and LCIEN (EMACn_MIEN[22]) is enabled, the TXINTR will be high. Write 1 to this bit clears the LC status.<br>0 = No collision occurred in the outside of 64 bytes collision window.<br>1 = Collision occurred in the outside of 64 bytes collision window. |
| [21] | **TXABT** | **Transmit Abort Interrupt**<br>The TXABT high indicates the packet incurred 16 consecutive collisions during transmission, and then the transmission process for this packet is aborted. The transmission abort is only available while EMAC is operating on half-duplex mode.<br>If the TXABT is high and TXABTIEN (EMACn_MIEN[21]) is enabled, the TXINTR will be high. Write 1 to this bit clears the TXABT status.<br>0 = Packet does not incur 16 consecutive collisions during transmission.<br>1 = Packet incurred 16 consecutive collisions during transmission. |
| [20] | **NCS** | **No Carrier Sense Interrupt**<br>The NCS high indicates the MII I/F signal CRS does not active at the start of or during the packet transmission. The NCS is only available while EMAC is operating on half-duplex mode. If the NCS is high and NCSIEN (EMACn_MIEN[20]) is enabled, the TXINTR will be high. Write 1 to this bit clears the NCS status.<br>0 = CRS signal actives correctly.<br>1 = CRS signal does not active at the start of or during the packet transmission. |
| [19] | **EXDEF** | **Defer Exceed Interrupt**<br>The EXDEF high indicates the frame waiting for transmission has deferred over 0.32768ms on 100Mbps mode, or 3.2768ms on 10Mbps mode. The deferral exceed check will only be done while bit NDEF of MCMDR is disabled, and EMAC is operating on half-duplex mode.<br>If the EXDEF is high and EXDEFIEN (EMACn_MIEN[19]) is enabled, the TXINTR will be high. Write 1 to this bit clears the EXDEF status.<br>0 = Frame waiting for transmission has not deferred over 0.32768ms (100Mbps) or 3.2768ms (10Mbps).<br>1 = Frame waiting for transmission has deferred over 0.32768ms (100Mbps) or 3.2768ms (10Mbps). |
| [18] | **TXCP** | **Transmit Completion Interrupt**<br>The TXCP indicates the packet transmission has completed correctly.<br>If the TXCP is high and TXCPIEN (EMACn_MIEN[18]) is enabled, the TXINTR will be high. Write 1 to this bit clears the TXCP status.<br>0 = The packet transmission not completed.<br>1 = The packet transmission has completed. |

| | | |
|---|---|---|
| [17] | **TXEMP** | **Transmit FIFO Underflow Interrupt**<br>The TXEMP high indicates the TXFIFO underflow occurred during packet transmission. While the TXFIFO underflow occurred, the EMAC will retransmit the packet automatically without S/W intervention. If the TXFIFO underflow occurred often, it is recommended that modify TXFIFO threshold control, the TXTHD of FFTCR register, to higher level.<br>If the TXEMP is high and TXUDIEN (EMACn_MIEN[17]) is enabled, the TXINTR will be high. Write 1 to this bit clears the TXEMP status.<br>0 = No TXFIFO underflow occurred during packet transmission.<br>1 = TXFIFO underflow occurred during packet transmission. |
| [16] | **TXINTR** | **Transmit Interrupt**<br>The TXINTR indicates the TX interrupt status.<br>If TXINTR high and its corresponding enable bit, TXIEN (EMACn_MIEN[16]), is also high indicates the EMAC generates TX interrupt to CPU. If TXINTR is high but TXIEN (EMACn_MIEN[16]) is disabled, no TX interrupt is generated.<br>The TXINTR is logic OR result of bit logic AND result of EMACn_MISTA[28:17] and EMACn_MIEN[28:17]. In other words, if any bit of EMACn_MISTA[28:17] is high and its corresponding enable bit in EMACn_MIEN[28:17] is also enabled, the TXINTR will be high. Because the TXINTR is a logic OR result, clears EMC_MISTA[28:17] makes TXINTR be cleared, too.<br>0 = No status bit in EMACn_MISTA[28:17] is set or no enable bit in EMACn_MIEN[28:17] is enabled.<br>1 = At least one status in EMACn_MISTA[28:17] is set and its corresponding enable bit in EMACn_MIEN[28:17] is enabled, too. |
| [15] | **MGPR** | **Magic Packet Received Interrupt**<br>The MPR high indicates EMAC receives a Magic Packet. The CFR only available while system is in power down mode and MGP_WAKE is set high.<br>If the MPR is high and WOLIEN (EMACn_MIEN[15]) is enabled, the RXINTR will be high. Write 1 to this bit clears the MPR status.<br>0 = The EMAC does not receive the Magic Packet.<br>1 = The EMAC receives a Magic Packet. |
| [14] | **CFR** | **Control Frame Receive Interrupt**<br>The CFR high indicates EMAC receives a flow control frame. The CFR only available while EMAC is operating on full duplex mode.<br>If the CFR is high and CFRIEN (EMACn_MIEN[14]) is enabled, the RXINTR will be high. Write 1 to this bit clears the CFR status.<br>0 = The EMAC does not receive the flow control frame.<br>1 = The EMAC receives a flow control frame. |
| [13:12] | **Reserved** | Reserved. |
| [11] | **RXBERR** | **Receive Bus Error Interrupt**<br>The RXBERR high indicates the memory controller replies ERROR response while EMAC access system memory through RXDMA during packet reception process. Reset EMAC is recommended while RXBERR status is high.<br>If the RXBERR is high and RXBEIEN (EMACn_MIEN[11]) is enabled, the RXINTR will be high. Write 1 to this bit clears the RXBERR status.<br>0 = No ERROR response is received.<br>1 = ERROR response is received. |

| | | |
|---|---|---|
| [10] | **RDU** | **Receive Descriptor Unavailable Interrupt**<br>The RDU high indicates that there is no available RX descriptor for packet reception and RXDMA will stay at Halt state. Once, the RXDMA enters the Halt state, S/W must issues a write command to RSDR register to make RXDMA leave Halt state while new RX descriptor is available.<br>If the RDU is high and RDUIEN (EMACn_MIEN[10]) is enabled, the RXINTR will be high. Write 1 to this bit clears the RDU status.<br>0 = RX descriptor is available.<br>1 = RX descriptor is unavailable. |
| [9] | **DEN** | **DMA Early Notification Interrupt**<br>The DENI high indicates the EMAC has received the Length/Type field of the incoming packet.<br>If the DENI is high and DENIENI (EMACn_MIEN[9]) is enabled, the RXINTR will be high. Write 1 to this bit clears the DENI status.<br>0 = The Length/Type field of incoming packet has not received yet.<br>1 = The Length/Type field of incoming packet has received. |
| [8] | **DFO** | **Maximum Frame Length Interrupt**<br>The DFOI high indicates the length of the incoming packet has exceeded the length limitation configured in DMARFC register and the incoming packet is dropped. If the DFOI is high and MFLEIEN (EMACn_MIEN[8]) is enabled, the RXINTR will be high. Write 1 to this bit clears the DFOI status.<br>0 = The length of the incoming packet doesn"t exceed the length limitation configured in DMARFC.<br>1 = The length of the incoming packet has exceeded the length limitation configured in DMARFC. |
| [7] | **MMP** | **More Missed Packet Interrupt**<br>The MMP high indicates the MPCNT, Missed Packet Count, has overflow. If the MMP is high and MMPIEN (EMACn_MIEN[7]) is enabled, the RXINTR will be high. Write 1 to this bit clears the MMP status.<br>0 = The MPCNT has not rolled over yet.<br>1 = The MPCNT has rolled over yet. |
| [6] | **RP** | **Runt Packet Interrupt**<br>The RP high indicates the length of the incoming packet is less than 64 bytes and the packet is dropped. If the ARP (EMACn_MCMDR[2]) is set, the short packet is regarded as a good packet and RP will not be set.<br>If the RP is high and RPIEN (EMACn_MIEN[6]) is enabled, the RXINTR will be high. Write 1 to this bit clears the RP status.<br>0 = The incoming frame is not a short frame or S/W wants to receive a short frame.<br>1 = The incoming frame is a short frame and dropped. |
| [5] | **ALIE** | **Alignment Error Interrupt**<br>The ALIE high indicates the length of the incoming frame is not a multiple of byte. If the ALIE is high and ALIEIEN (EMACn_MIEN[5]) is enabled, the RXINTR will be high. Write 1 to this bit clears the ALIE status.<br>0 = The frame length is a multiple of byte.<br>1 = The frame length is not a multiple of byte. |
| [4] | **RXGD** | **Receive Good Interrupt**<br>The RXGD high indicates the frame reception has completed.<br>If the RXGD is high and RXGDIEN (EMACn_MIEN[4]) is enabled, the RXINTR will be high. Write 1 to this bit clears the RXGD status.<br>0 = The frame reception has not complete yet.<br>1 = The frame reception has completed. |

| | | |
|---|---|---|
| [3] | **PTLE** | **Packet Too Long Interrupt**<br>The PTLE high indicates the length of the incoming packet is greater than 1518 bytes and the incoming packet is dropped. If the ALP (EMACn_MCMDR[1]) is set, the long packet will be regarded as a good packet and PTLE will not be set.<br>If the PTLE is high and LPIEN(EMACn_MIEN[3]) is enabled, the RXINTR will be high. Write 1 to this bit clears the PTLE status.<br>0 = The incoming frame is not a long frame or S/W wants to receive a long frame.<br>1 = The incoming frame is a long frame and dropped. |
| [2] | **RXOV** | **Receive FIFO Overflow Interrupt**<br>The RXOV high indicates the RXFIFO overflow occurred during packet reception. While the RXFIFO overflow occurred, the EMAC drops the current receiving packer. If the RXFIFO overflow occurred often, it is recommended that modify RXFIFO threshold control, the RXTHD of FFTCR register, to higher level.<br>If the RXOV is high and RXOVIEN (EMACn_MIEN[2]) is enabled, the RXINTR will be high. Write 1 to this bit clears the RXOV status.<br>0 = No RXFIFO overflow occurred during packet reception.<br>1 = RXFIFO overflow occurred during packet reception. |
| [1] | **CRCE** | **CRC Error Interrupt**<br>The CRCE high indicates the incoming packet incurred the CRC error and the packet is dropped. If the AEP (EMACn_MCMDR[4]) is set, the CRC error packet will be regarded as a good packet and CRCE will not be set.<br>If the CRCE is high and CRCEIEN (EMACn_MIEN[1]) is enabled, the RXINTR will be high. Write 1 to this bit clears the CRCE status.<br>0 = The frame does not incur CRC error.<br>1 = The frame incurred CRC error. |
| [0] | **RXINTR** | **Receive Interrupt**<br>The RXINTR indicates the RX interrupt status.<br>If RXINTR high and its corresponding enable bit, RXIEN (EMACn_MIEN[0]), is also high indicates the EMAC generates RX interrupt to CPU. If RXINTR is high but RXIEN (EMACn_MIEN[0]) is disabled, no RX interrupt is generated.<br>The RXINTR is logic OR result of bit logic AND result of EMACn_MISTA[15:1] and EMACn_MIEN[15:1]. In other words, if any bit of EMACn_MISTA[15:1] is high and its corresponding enable bit in EMACn_MIEN[15:1] is also enabled, the RXINTR will be high.<br>Because the RXINTR is a logic OR result, clears EMACn_MISTA[15:1] makes RXINTR be cleared, too.<br>0 = No status bit in EMACn_MISTA[15:1] is set or no enable bit in EMACn_MIEN[15:1] is enabled.<br>1 = At least one status in EMACn_MISTA[15:1] is set and its corresponding enable bit in EMACn_MIEN[15:1] is enabled, too. |

## EMAC n MAC General Status Register (EMACn_MGSTA)

The EMACn_MGSTA also keeps the statuses of EMAC. But the statuses in the EMACn_MGSTA will not trigger any interrupt. The EMACn_MGSTA is a write clear register and write 1 to corresponding bit clears the status.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_MGSTA n=0,1** | EMACn_BA+0x0B4 | R/W | EMAC n MAC General 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** | | | **RPAU** | **TXHA** | **SQE** | **PAU** | **DEF** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **CCNT** | | | | **Reserved** | **RXFFULL** | **RXHA** | **CFR** |

| Bits | Description | |
|---|---|---|
| [31:13] | **Reserved** | Reserved. |
| [12] | **RPAU** | **Remote Pause Status**<br>The RPAU indicates that remote pause counter down counting actives.<br>After Ethernet MAC controller sent PAUSE frame out successfully, it starts the remote pause counter down counting. When this bit high, it's predictable that remote Ethernet MAC controller wouldn't start the packet transmission until the down counting done.<br>0 = Remote pause counter down counting done.<br>1 = Remote pause counter down counting actives. |
| [11] | **TXHA** | **Transmission Halted**<br>The TXHA high indicates the next normal packet transmission process will be halted because the bit TXON (EMACn_MCMDR[8]) is disabled be S/W.<br>0 = Next normal packet transmission process will go on.<br>1 = Next normal packet transmission process will be halted. |
| [10] | **SQE** | **Signal Quality Error**<br>The SQE high indicates the SQE error found at end of packet transmission on 10Mbps half-duplex mode. The SQE error check will only be done while both bit EnSQE (EMACn_MCMDR[17]) is enabled and EMAC is operating on 10Mbps half-duplex mode.<br>0 = No SQE error found at end of packet transmission.<br>1 = SQE error found at end of packet transmission. |

| | | |
|---|---|---|
| [9] | **PAU** | **Transmission Paused**<br>The PAU high indicates the next normal packet transmission process will be paused temporally because EMAC received a PAUSE control frame.<br>0 = Next normal packet transmission process will go on.<br>1 = Next normal packet transmission process will be paused. |
| [8] | **DEF** | **Deferred Transmission**<br>The DEF high indicates the packet transmission has deferred once. The DEF is only available while EMAC is operating on half-duplex mode.<br>0 = Packet transmission does not defer.<br>1 = Packet transmission has deferred once. |
| [7:4] | **CCNT** | **Collision Count**<br>The CCNT indicates that how many collisions occurred consecutively during a packet transmission. If the packet incurred 16 consecutive collisions during transmission, the CCNT will be 4"h0 and bit TXABT will be set to 1. |
| [3] | **Reserved** | Reserved. |
| [2] | **RXFFULL** | **RXFIFO Full**<br>The RXFFULL indicates the RXFIFO is full due to four 64-byte packets are kept in RXFIFO and the following incoming packet will be dropped.<br>0 = The RXFIFO is not full.<br>1 = The RXFIFO is full and the following incoming packet will be dropped. |
| [1] | **RXHA** | **Receive Halted**<br>The RXHA high indicates the next normal packet reception process will be halted because the bit RXON of MCMDR is disabled be S/W.<br>0 = Next normal packet reception process will go on.<br>1 = Next normal packet reception process will be halted. |
| [0] | **CFR** | **Control Frame Received**<br>The CFR high indicates EMAC receives a flow control frame. The CFR only available while EMAC is operating on full duplex mode.<br>0 = The EMAC does not receive the flow control frame.<br>1 = The EMAC receives a flow control frame. |

## EMAC n Missed Packet Count Register (EMACn_MPCNT)

The EMACn_MPCNT keeps the number of packets that were dropped due to various types of receive errors. The EMACn_MPCNT is a read clear register. In addition, S/W also can write an initial value to EMACn_MPCNT and the missed packet counter will start counting from that initial value. If the missed packet counter is overflow, the MMP (EMACn_MISTA[7]) will be set.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_MPCNT n=0,1** | EMACn_BA+0x0B8 | R/W | EMAC n Missed Packet Count Register | 0x0000_7FFF |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **MPC** | **Miss Packet Count**<br>The MPC indicates the number of packets that were dropped due to various types of receive errors. The following type of receiving error makes missed packet counter increase:<br>1. Incoming packet is incurred RXFIFO overflow.<br>2. Incoming packet is dropped due to RXON is disabled.<br>3. Incoming packet is incurred CRC error. |

### EMAC n MAC Receive Pause Count Register (EMACn_MRPC)

The EMAC supports the PAUSE control frame reception and recognition. If EMAC received a PAUSE control frame, the operand field of the PAUSE control frame will be extracted and stored in the EMACn_MRPC register. The EMACn_MRPC register will keep the same while TX of EMAC is pausing due to the PAUSE control frame is received. The EMACn_MRPC is read only and write to this register has no effect.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_MRPC n=0,1** | EMACn_BA+0x0BC | R | EMAC n MAC Receive Pause Count 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** |
| **MRPC** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **MRPC** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **MRPC** | **MAC Receive Pause Count**<br>The MRPC keeps the operand field of the PAUSE control frame. It indicates how many slot time (512 bit time) the TX of EMAC will be paused. |

## EMAC n DMA Receive Frame Status Register (EMACn_DMARFS)

The EMACn_DMARFS is used to keep the Length/Type field of each incoming Ethernet packet.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_DMARFS**<br>**n=0,1** | EMACn_BA+0x0C8 | R/W | EMAC n DMA Receive Frame 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** |
| **RXFLT** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **RXFLT** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **RXFLT** | **Receive Frame Length/Type**<br>The RXFLT keeps the Length/Type field of each incoming Ethernet packet. If the bit DENIEN (EMACn_MIEN[9]) is enabled and the Length/Type field of incoming packet has received, the bit DENI (EMACn_MISTA[9]) will be set and trigger interrupt. And, the content of Length/Type field will be stored in RXFLT. |

## EMAC n Current Transmit Descriptor Start Address Register (EMACn_CTXDSA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_CTXDSA n=0,1** | EMACn_BA+0x0CC | R | EMAC n Current Transmit Descriptor Start Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **CTXDSA** | **Current Transmit Descriptor Start Address**<br>The CTXDSA keeps the start address of TX descriptor that is used by TXDMA currently. The EMACn_CTXDSA is read only and write to this register has no effect. |

## EMAC n Current Transmit Buffer Start Address Register (EMACn_CTXBSA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_CTXBSA n=0,1** | EMACn_BA+0x0D0 | R | EMAC n Current Transmit Buffer Start Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **CTXBSA** | **Current Transmit Buffer Start Address**<br>The CTXDSA keeps the start address of TX frame buffer that is used by TXDMA currently. The EMACn_CTXBSA is read only and write to this register has no effect. |

## EMAC n Current Receive Descriptor Start Address Register (EMACn_CRXDSA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_CRXDSA n=0,1** | EMACn_BA+0x0D4 | R | EMAC n Current Receive Descriptor Start Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **CRXDSA** | **Current Receive Descriptor Start Address**<br>The CRXDSA keeps the start address of RX descriptor that is used by RXDMA currently. The EMACn_CRXDSA is read only and write to this register has no effect. |

## EMAC n Current Receive Buffer Start Address Register (EMACn_CRXBSA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_CRXBSA n=0,1** | EMACn_BA+0x0D8 | R | EMAC n Current Receive Buffer Start Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **CRXBSA** | **Current Receive Buffer Start Address**<br>The CRXBSA keeps the start address of RX frame buffer that is used by RXDMA currently. The EMACn_CRXBSA is read only and write to this register has no effect. |

## EMAC n Time Stamp Control Register (EMACn_TSCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| EMACn_TSCTL<br>n=0,1 | EMACn_BA+0x100 | R/W | EMAC n Time Stamp 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 | | TSALMEN | Reserved | TSUPDATE | TSMODE | TSIEN | TSEN |

| Bits | Description | |
|---|---|---|
| [31:6] | **Reserved** | Reserved. |
| [5] | **TSALMEN** | **Time Stamp Alarm Enable Control**<br>Set this bit high enable Ethernet MAC controller to set TSALS (EMACn_MISTA[28]) high when EMACn_TSSEC equals to EMACn_ALMSEC and EMACn_TSSUBSEC equals to EMACn_ALMSUBSEC.<br>0 = Alarm disabled when EMACn_TSSEC equals to EMACn_ALMSEC and EMACn_TSSUBSEC equals to EMACn_ALMSUBSEC.<br>1 = Alarm enabled when EMACn_TSSEC equals to EMACn_ALMSEC and EMACn_TSSUBSEC equals to EMACn_ALMSUBSEC. |
| [4] | **Reserved** | Reserved. |
| [3] | **TSUPDATE** | **Time Stamp Counter Time Update Enable Control**<br>Set this bit high enables Ethernet MAC controller to add value of register EMACn_UPDSEC and EMACn_UPDSUBSEC to PTP time stamp counter.<br>After the add operation finished, Ethernet MAC controller clear this bit to low automatically.<br>0 = No action.<br>1 = EMACn_UPDSEC updated to EMACn_TSSEC and EMACn_UPDSUBSEC updated to EMACn_TSSUBSEC. |
| [2] | **TSMODE** | **Time Stamp Fine Update Enable Control**<br>This bit chooses the time stamp counter update mode.<br>0 = Time stamp counter is in coarse update mode.<br>1 = Time stamp counter is in fine update mode. |

| | | |
|---|---|---|
| [1] | **TSIEN** | **Time Stamp Counter Initialization Enable Control**<br>Set this bit high enables Ethernet MAC controller to load value of register EMACn_UPDSEC and EMACn_UPDSUBSEC to PTP time stamp counter.<br>After the load operation finished, Ethernet MAC controller clear this bit to low automatically.<br>0 = Time stamp counter initialization done.<br>1 = Time stamp counter initialization Enabled. |
| [0] | **TSEN** | **Time Stamp Function Enable Control**<br>This bit controls if the IEEE 1588 PTP time stamp function is enabled or not.<br>Set this bit high to enable IEEE 1588 PTP time stamp function while set this bit low to disable IEEE 1588 PTP time stamp function.<br>0 = IEEE 1588 PTP time stamp function Disabled.<br>1 = IEEE 1588 PTP time stamp function Enabled. |

### EMAC n Time Stamp Counter Second Register (EMACn_TSSEC)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_TSSEC n=0,1** | EMACn_BA+0x110 | R/W | EMAC n Time Stamp Counter Second Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **SEC** | **Time Stamp Counter Second**<br>This register reflects the bit [63:32] value of 64-bit reference timing counter. This 32-bit value is used as the second part of time stamp when TSEN (EMACn_TSCTL[0]) is high. |

## EMAC n Time Stamp Counter Sub Second Register (EMACn_TSSUBSEC)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_TSSUBSEC**<br>**n=0,1** | EMACn_BA+0x114 | R/W | EMAC n Time Stamp Counter Sub Second Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **SUBSEC** | **Time Stamp Counter Sub-second**<br>This register reflects the bit [31:0] value of 64-bit reference timing counter. This 32-bit value is used as the sub-second part of time stamp when TSEN (EMACn_TSCTL[0]) is high. |

## EMAC n Time Stamp Increment Register (EMACn_TSINC)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_TSINC n=0,1** | EMACn_BA+0x118 | R/W | EMAC n Time Stamp Increment 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** |
| **CNTINC** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **CNTINC** | **Time Stamp Counter Increment**<br>Time stamp counter increment value.<br>If TSEN (EMACn_TSCTL[0]) is high, EMAC adds EMACn_TSSUBSEC with this 8-bit value every time when it wants to increase the EMACn_TSSUBSEC value. |

## EMAC n Time Stamp Addend Register (EMACn_TSADDEND)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_TSADDEND n=0,1** | EMACn_BA+0x11C | R/W | EMAC n Time Stamp Addend Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **ADDEND** | **Time Stamp Counter Addend**<br>This register keeps a 32-bit value for accumulator to enable increment of EMACn_TSSUBSEC.<br>If TSEN (EMACn_TSCTL[0]) and TSMODE (EMACn_TSCTL[2]) are both high, EMAC increases accumulator with this 32-bit value in each HCLK. Once the accumulator is overflow, it generates an enable to increase EMACn_TSSUBSEC with an 8-bit value kept in register EMACn_TSINC. |

### EMAC n Time Stamp Update Second Register (EMACn_UPDSEC)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_UPDSEC**<br>**n=0,1** | EMACn_BA+0x120 | R/W | EMAC n Time Stamp Update Second Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **SEC** | **Time Stamp Counter Second Update**<br>When TSIEN (EMACn_TSCTL[1]) is high. EMAC loads this 32-bit value to EMACn_TSSEC directly. When TSUPDATE (EMACn_TSCTL[3]) is high, EMAC increases EMACn_TSSEC with this 32-bit value. |

## EMAC n Time Stamp Update Sub Second Register (EMACn_UPDSUBSEC)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_UPDSUBSEC**<br>**n=0,1** | EMACn_BA+0x124 | R/W | EMAC n Time Stamp Update Sub Second Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **SUBSEC** | **Time Stamp Counter Sub-second Update**<br>When TSIEN (EMACn_TSCTL[1]) is high. EMAC loads this 32-bit value to EMACn_TSSUBSEC directly. When TSUPDATE (EMACn_TSCTL[3]) is high, EMAC increases EMACn_TSSUBSEC with this 32-bit value. |

## EMAC n Time Stamp Alarm Second Register (EMACn_ALMSEC)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_ALMSEC**<br>**n=0,1** | EMACn_BA+0x128 | R/W | EMAC n Time Stamp Alarm Second Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **SEC** | **Time Stamp Counter Second Alarm**<br>Time stamp counter second part alarm value.<br>This value is only useful when TSALMEN (EMACn_TSCTL[5]) high. If TSALMEN (EMACn_TSCTL[5]) is high, EMACn_TSSEC equals to EMACn_ALMSEC and EMACn_TSSUBSEC equals to EMACn_ALMSUBSEC, Ethernet MAC controller set TSALS (EMACn_MISTA[28]) high. |

## EMAC n Time Stamp Alarm Sub Second Register (EMACn_ALMSUBSEC)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EMACn_ALMSUBSEC**<br>**n=0,1** | EMACn_BA+0x12C | R/W | EMAC n Time Stamp Alarm Sub Second Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **SUBSEC** | **Time Stamp Counter Sub-second Alarm**<br>Time stamp counter sub-second part alarm value.<br>This value is only useful when TSALMEN (EMACn_TSCTL[5]) high. If TSALMEN (EMACn_TSCTL[5]) is high, EMACn_TSSEC equals to EMACn_ALMSEC and EMACn_TSSUBSEC equals to EMACn_ALMSUBSEC, Ethernet MAC controller set TSALS (EMACn_MISTA[28]) high. |

## 5.22 USB 2.0 Device Controller (USBD)

### 5.22.1 Overview

The USB device controller interfaces the AHB bus and the UTMI bus. The USB controller contains both the AHB master interface and AHB slave interface. CPU programs the USB controller registers through the AHB slave interface. For IN or OUT transfer, the USB device controller needs to write data to memory or read data from memory through the AHB master interface. The USB device controller is complaint with USB 2.0 specification and it contains 12 configurable endpoints in addition to control endpoint. These endpoints could be configured to BULK, INTERRUPT or ISO. The USB device controller has a built-in DMA to relieve the load of CPU.

### 5.22.2 Features

- USB Specification reversion 2.0 compliant
- Supports 12 configurable endpoints in addition to Control Endpoint
- Each of the endpoints can be Isochronous, Bulk or Interrupt and either IN or OUT direction
- Three different operation modes of an in-endpoint – Auto Validation mode, Manual Validation mode, Fly mode
- Supports DMA operation
- 4096 Bytes Configurable RAM used as endpoint buffer
- Supports Endpoint Maximum Packet Size up to 1024 bytes

### 5.22.3 Block Diagram

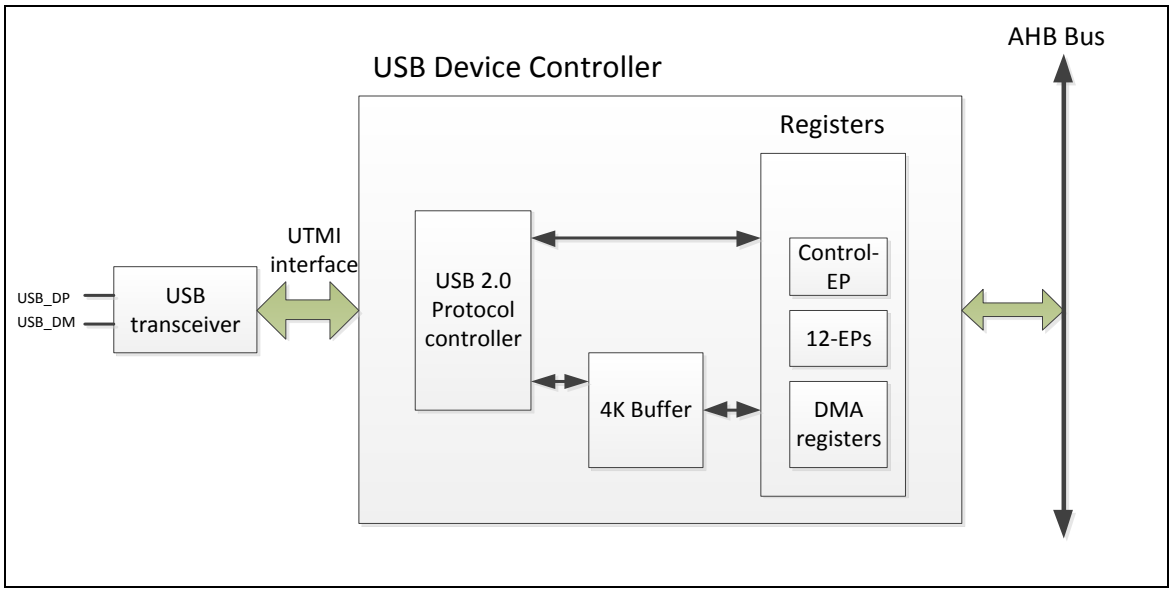



Figure 5.22-1 USB Device Controller Block Diagram

### 5.22.4 Basic Configuration

USB device clock source is derived from PLL and USB PHY. User has to set the PLL related configurations before USB device enabled. Set the USBD (CLK_HCLKEN[19]) bit to enable USB device clock.

In addition, USB device needs a clock from USB PHY for USB 2.0 high speed operation. User has to set PHYEN (USBD_PHYCTL[9]) high to enable USB PHY.

### 5.22.5 Functional Description

#### 5.22.5.1 *Operation of different In-transfer modes*

The data for any in-transfer is written into the internal buffer when in turn is sent to the host on receipt of an in-token. There are three different modes by which the data sent to the host is validated by CPU.

- Auto-Validation Mode
- Manual-Validation Mode
- Fly Mode

#### 5.22.5.2 *Auto-Validation Mode*

If an endpoint is selected to be operating in auto-validation mode, the endpoint responds only with data payload to be equal to EPMPS register. The endpoint controller wait until the amount of data is equal to EPMPS value and then validates the data. If CPU needs to send a short-packet at the end of a transfer, the SHORTTXEN bit of USBD_EPxRSPCTL[6] should be set. When this bit set, any remaining data in the buffer is validated and is sent to the host, for the forthcoming in-token.

This mode requires least intervention of CPU, as most of the work is done by the USB device

controller. The mode can be selected, when the data payload sent to host is always equal to MPS size.

| SHORTTXEN | Data Availability In Buffer | Data Sent/NAK Sent |
|---|---|---|
| 0 | < Max. Packet Size | NAK sent |
| 0 | >= Max. Packet Size | Data payload of max. packet size |
| 1 | < Max. Packet Size | Available data of < max. packet size |
| 1 | >= Max. Packet Size | Data payload of max. packet size sent |

#### 5.22.5.3 *Manual-Validation Mode*

If the endpoint is selected to be operating in manual-validation mode, the endpoint responds only when the data in the buffer is validated by CPU every time. The CPU has to write data into the buffer and then write the count of the data into EPxTXCNT register. Once the data is validating by writing a count into the EPxTXCNT register, the data is sent to the host on receipt of an in-token.

This mode requires intervention of CPU for each transfer. But this would be useful, if the data-count to be sent each time is not fixed, and it is being decided by CPU.

| EPxTXCNT Written | Data Availability In Buffer | Data Sent/NAK Sent |
|---|---|---|
| NO | - | NAK |
| YES | EPxTXCNT | Data payload of EPxTXCNT sent |

#### 5.22.5.4 *Fly Mode*

The fly mode of operation is simplest mode of operation, where there is no validation procedure. The buffer is being filled by CPU. If an in-token is send from the host, the data in the buffer is automatically validated and sent to the host. If the data in the buffer spans more than one packet of maximum packet size, the controller automatically packs to equal to MPS and send it to the host.

This mode requires the least intervention by CPU. This mode is best suited for isochronous data transfer, where the speed of data transfer is more important than the packet size.

| Data Availability In Buffer | Data Sent |
|---|---|
| < Max. Packet Size | Data available sent |
| >= Max. Packet Size | |

#### 5.22.5.5 *Scatter-Gather function*

When enabling the scatter gather DMA function, setting SGEN high and USBD_DMACNT set 8 bytes,

DMA will be enabled to fetch the descriptor which describes the real memory address and length. The descriptor will be an 8-byte format, like the following:

| | Format | | | |
|---|---|---|---|---|
| | [31] | [30] | [29:0] | |
| Word0 | MEM_ADDR[31:0] | | | |
| Word1 | EOT | RD | Reserved | Count[19:0] |

**MEM_ADDR**: It specifies the memory address (AHB address).

**EOT**: end of transfer. When this bit is set to high, it means this is the last descriptor.

**RD**: “1” means read from memory into buffer. “0” means read from buffer into memory.

### 5.22.6 Registers Map

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD Base Address:**<br>**USBD_BA = 0x4001_9000** | | | | |
| **USBD_GINTSTS** | USBD_BA+0x000 | R | Global Interrupt Status Register | 0x0000_0000 |
| **USBD_GINTEN** | USBD_BA+0x008 | R/W | Global Interrupt Enable Register | 0x0000_0001 |
| **USBD_BUSINTSTS** | USBD_BA+0x010 | R/W | USB Bus Interrupt Status Register | 0x0000_0000 |
| **USBD_BUSINTEN** | USBD_BA+0x014 | R/W | USB Bus Interrupt Enable Register | 0x0000_0040 |
| **USBD_OPER** | USBD_BA+0x018 | R/W | USB Operational Register | 0x0000_0002 |
| **USBD_FRAMECNT** | USBD_BA+0x01C | R | USB Frame Count Register | 0x0000_0000 |
| **USBD_FADDR** | USBD_BA+0x020 | R/W | USB Function Address Register | 0x0000_0000 |
| **USBD_TEST** | USBD_BA+0x024 | R/W | USB Test Mode Register | 0x0000_0000 |
| **USBD_CEPDAT** | USBD_BA+0x028 | R/W | Control Endpoint Data Buffer | 0x0000_0000 |
| **USBD_CEPCTL** | USBD_BA+0x02C | R/W | Control Endpoint Control Register | 0x0000_0000 |
| **USBD_CEPINTEN** | USBD_BA+0x030 | R/W | Control Endpoint Interrupt Enable | 0x0000_0000 |
| **USBD_CEPINTSTS** | USBD_BA+0x034 | R/W | Control Endpoint Interrupt Status | 0x0000_1800 |
| **USBD_CEPTXCNT** | USBD_BA+0x038 | R/W | Control Endpoint In Transfer Data Count | 0x0000_0000 |
| **USBD_CEPRXCNT** | USBD_BA+0x03C | R | Control Endpoint Out Transfer Data Count | 0x0000_0000 |
| **USBD_CEPDATCNT** | USBD_BA+0x040 | R | Control Endpoint Data Count | 0x0000_0000 |
| **USBD_SETUP1_0** | USBD_BA+0x044 | R | Setup1 & Setup0 bytes | 0x0000_0000 |
| **USBD_SETUP3_2** | USBD_BA+0x048 | R | Setup3 & Setup2 Bytes | 0x0000_0000 |
| **USBD_SETUP5_4** | USBD_BA+0x04C | R | Setup5 & Setup4 Bytes | 0x0000_0000 |
| **USBD_SETUP7_6** | USBD_BA+0x050 | R | Setup7 & Setup6 Bytes | 0x0000_0000 |
| **USBD_CEPBUFSTART** | USBD_BA+0x054 | R/W | Control Endpoint RAM Start Address Register | 0x0000_0000 |
| **USBD_CEPBUFEND** | USBD_BA+0x058 | R/W | Control Endpoint RAM End Address Register | 0x0000_0000 |
| **USBD_DMACTL** | USBD_BA+0x05C | R/W | DMA Control Status Register | 0x0000_0000 |
| **USBD_DMACNT** | USBD_BA+0x060 | R/W | DMA Count Register | 0x0000_0000 |
| **USBD_EPADAT** | USBD_BA+0x064 | R/W | Endpoint A Data Register | 0x0000_0000 |
| **USBD_EPAINTSTS** | USBD_BA+0x068 | R/W | Endpoint A Interrupt Status Register | 0x0000_0003 |
| **USBD_EPAINTEN** | USBD_BA+0x06C | R/W | Endpoint A Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPADATCNT** | USBD_BA+0x070 | R | Endpoint A Data Available Count Register | 0x0000_0000 |
| **USBD_EPARSPCTL** | USBD_BA+0x074 | R/W | Endpoint A Response Control Register | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **USBD_EPAMPS** | USBD_BA+0x078 | R/W | Endpoint A Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPATXCNT** | USBD_BA+0x07C | R/W | Endpoint A Transfer Count Register | 0x0000_0000 |
| **USBD_EPACFG** | USBD_BA+0x080 | R/W | Endpoint A Configuration Register | 0x0000_0012 |
| **USBD_EPABUFSTART** | USBD_BA+0x084 | R/W | Endpoint A RAM Start Address Register | 0x0000_0000 |
| **USBD_EPABUFEND** | USBD_BA+0x088 | R/W | Endpoint A RAM End Address Register | 0x0000_0000 |
| **USBD_EPBDAT** | USBD_BA+0x08C | R/W | Endpoint B Data Register | 0x0000_0000 |
| **USBD_EPBINTSTS** | USBD_BA+0x090 | R/W | Endpoint B Interrupt Status Register | 0x0000_0003 |
| **USBD_EPBINTEN** | USBD_BA+0x094 | R/W | Endpoint B Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPBDATCNT** | USBD_BA+0x098 | R | Endpoint B Data Available Count Register | 0x0000_0000 |
| **USBD_EPBRSPCTL** | USBD_BA+0x09C | R/W | Endpoint B Response Control Register | 0x0000_0000 |
| **USBD_EPBMPS** | USBD_BA+0x0A0 | R/W | Endpoint B Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPBTXCNT** | USBD_BA+0x0A4 | R/W | Endpoint B Transfer Count Register | 0x0000_0000 |
| **USBD_EPBCFG** | USBD_BA+0x0A8 | R/W | Endpoint B Configuration Register | 0x0000_0022 |
| **USBD_EPBBUFSTART** | USBD_BA+0x0AC | R/W | Endpoint B RAM Start Address Register | 0x0000_0000 |
| **USBD_EPBBUFEND** | USBD_BA+0x0B0 | R/W | Endpoint B RAM End Address Register | 0x0000_0000 |
| **USBD_EPCDAT** | USBD_BA+0x0B4 | R/W | Endpoint C Data Register | 0x0000_0000 |
| **USBD_EPCINTSTS** | USBD_BA+0x0B8 | R/W | Endpoint C Interrupt Status Register | 0x0000_0003 |
| **USBD_EPCINTEN** | USBD_BA+0x0BC | R/W | Endpoint C Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPCDATCNT** | USBD_BA+0x0C0 | R | Endpoint C Data Available Count Register | 0x0000_0000 |
| **USBD_EPCRSPCTL** | USBD_BA+0x0C4 | R/W | Endpoint C Response Control Register | 0x0000_0000 |
| **USBD_EPCMPS** | USBD_BA+0x0C8 | R/W | Endpoint C Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPCTXCNT** | USBD_BA+0x0CC | R/W | Endpoint C Transfer Count Register | 0x0000_0000 |
| **USBD_EPCCFG** | USBD_BA+0x0D0 | R/W | Endpoint C Configuration Register | 0x0000_0032 |
| **USBD_EPCBUFSTART** | USBD_BA+0x0D4 | R/W | Endpoint C RAM Start Address Register | 0x0000_0000 |
| **USBD_EPCBUFEND** | USBD_BA+0x0D8 | R/W | Endpoint C RAM End Address Register | 0x0000_0000 |
| **USBD_EPDDAT** | USBD_BA+0x0DC | R/W | Endpoint D Data Register | 0x0000_0000 |
| **USBD_EPDINTSTS** | USBD_BA+0x0E0 | R/W | Endpoint D Interrupt Status Register | 0x0000_0003 |
| **USBD_EPDINTEN** | USBD_BA+0x0E4 | R/W | Endpoint D Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPDDATCNT** | USBD_BA+0x0E8 | R | Endpoint D Data Available Count Register | 0x0000_0000 |
| **USBD_EPDRSPCTL** | USBD_BA+0x0EC | R/W | Endpoint D Response Control Register | 0x0000_0000 |
| **USBD_EPDMPS** | USBD_BA+0x0F0 | R/W | Endpoint D Maximum Packet Size Register | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **USBD_EPDTXCNT** | USBD_BA+0x0F4 | R/W | Endpoint D Transfer Count Register | 0x0000_0000 |
| **USBD_EPDCFG** | USBD_BA+0x0F8 | R/W | Endpoint D Configuration Register | 0x0000_0042 |
| **USBD_EPDBUFSTART** | USBD_BA+0x0FC | R/W | Endpoint D RAM Start Address Register | 0x0000_0000 |
| **USBD_EPDBUFEND** | USBD_BA+0x100 | R/W | Endpoint D RAM End Address Register | 0x0000_0000 |
| **USBD_EPEDAT** | USBD_BA+0x104 | R/W | Endpoint E Data Register | 0x0000_0000 |
| **USBD_EPEINTSTS** | USBD_BA+0x108 | R/W | Endpoint E Interrupt Status Register | 0x0000_0003 |
| **USBD_EPEINTEN** | USBD_BA+0x10C | R/W | Endpoint E Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPEDATCNT** | USBD_BA+0x110 | R | Endpoint E Data Available Count Register | 0x0000_0000 |
| **USBD_EPERSPCTL** | USBD_BA+0x114 | R/W | Endpoint E Response Control Register | 0x0000_0000 |
| **USBD_EPEMPS** | USBD_BA+0x118 | R/W | Endpoint E Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPETXCNT** | USBD_BA+0x11C | R/W | Endpoint E Transfer Count Register | 0x0000_0000 |
| **USBD_EPECFG** | USBD_BA+0x120 | R/W | Endpoint E Configuration Register | 0x0000_0052 |
| **USBD_EPEBUFSTART** | USBD_BA+0x124 | R/W | Endpoint E RAM Start Address Register | 0x0000_0000 |
| **USBD_EPEBUFEND** | USBD_BA+0x128 | R/W | Endpoint E RAM End Address Register | 0x0000_0000 |
| **USBD_EPFDAT** | USBD_BA+0x12C | R/W | Endpoint F Data Register | 0x0000_0000 |
| **USBD_EPFINTSTS** | USBD_BA+0x130 | R/W | Endpoint F Interrupt Status Register | 0x0000_0003 |
| **USBD_EPFINTEN** | USBD_BA+0x134 | R/W | Endpoint F Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPFDATCNT** | USBD_BA+0x138 | R | Endpoint F Data Available Count Register | 0x0000_0000 |
| **USBD_EPFRSPCTL** | USBD_BA+0x13C | R/W | Endpoint F Response Control Register | 0x0000_0000 |
| **USBD_EPFMPS** | USBD_BA+0x140 | R/W | Endpoint F Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPFTXCNT** | USBD_BA+0x144 | R/W | Endpoint F Transfer Count Register | 0x0000_0000 |
| **USBD_EPFCFG** | USBD_BA+0x148 | R/W | Endpoint F Configuration Register | 0x0000_0062 |
| **USBD_EPFBUFSTART** | USBD_BA+0x14C | R/W | Endpoint F RAM Start Address Register | 0x0000_0000 |
| **USBD_EPFBUFEND** | USBD_BA+0x150 | R/W | Endpoint F RAM End Address Register | 0x0000_0000 |
| **USBD_EPGDAT** | USBD_BA+0x154 | R/W | Endpoint G Data Register | 0x0000_0000 |
| **USBD_EPGINTSTS** | USBD_BA+0x158 | R/W | Endpoint G Interrupt Status Register | 0x0000_0003 |
| **USBD_EPGINTEN** | USBD_BA+0x15C | R/W | Endpoint G Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPGDATCNT** | USBD_BA+0x160 | R | Endpoint G Data Available Count Register | 0x0000_0000 |
| **USBD_EPGRSPCTL** | USBD_BA+0x164 | R/W | Endpoint G Response Control Register | 0x0000_0000 |
| **USBD_EPGMPS** | USBD_BA+0x168 | R/W | Endpoint G Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPGTXCNT** | USBD_BA+0x16C | R/W | Endpoint G Transfer Count Register | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **USBD_EPGCFG** | USBD_BA+0x170 | R/W | Endpoint G Configuration Register | 0x0000_0072 |
| **USBD_EPGBUFSTART** | USBD_BA+0x174 | R/W | Endpoint G RAM Start Address Register | 0x0000_0000 |
| **USBD_EPGBUFEND** | USBD_BA+0x178 | R/W | Endpoint G RAM End Address Register | 0x0000_0000 |
| **USBD_EPHDAT** | USBD_BA+0x17C | R/W | Endpoint H Data Register | 0x0000_0000 |
| **USBD_EPHINTSTS** | USBD_BA+0x180 | R/W | Endpoint H Interrupt Status Register | 0x0000_0003 |
| **USBD_EPHINTEN** | USBD_BA+0x184 | R/W | Endpoint H Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPHDATCNT** | USBD_BA+0x188 | R | Endpoint H Data Available Count Register | 0x0000_0000 |
| **USBD_EPHRSPCTL** | USBD_BA+0x18C | R/W | Endpoint H Response Control Register | 0x0000_0000 |
| **USBD_EPHMPS** | USBD_BA+0x190 | R/W | Endpoint H Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPHTXCNT** | USBD_BA+0x194 | R/W | Endpoint H Transfer Count Register | 0x0000_0000 |
| **USBD_EPHCFG** | USBD_BA+0x198 | R/W | Endpoint H Configuration Register | 0x0000_0082 |
| **USBD_EPHBUFSTART** | USBD_BA+0x19C | R/W | Endpoint H RAM Start Address Register | 0x0000_0000 |
| **USBD_EPHBUFEND** | USBD_BA+0x1A0 | R/W | Endpoint H RAM End Address Register | 0x0000_0000 |
| **USBD_EPIDAT** | USBD_BA+0x1A4 | R/W | Endpoint I Data Register | 0x0000_0000 |
| **USBD_EPIINTSTS** | USBD_BA+0x1A8 | R/W | Endpoint I Interrupt Status Register | 0x0000_0003 |
| **USBD_EPIINTEN** | USBD_BA+0x1AC | R/W | Endpoint I Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPIDATCNT** | USBD_BA+0x1B0 | R | Endpoint I Data Available Count Register | 0x0000_0000 |
| **USBD_EPIRSPCTL** | USBD_BA+0x1B4 | R/W | Endpoint I Response Control Register | 0x0000_0000 |
| **USBD_EPIMPS** | USBD_BA+0x1B8 | R/W | Endpoint I Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPITXCNT** | USBD_BA+0x1BC | R/W | Endpoint I Transfer Count Register | 0x0000_0000 |
| **USBD_EPICFG** | USBD_BA+0x1C0 | R/W | Endpoint I Configuration Register | 0x0000_0092 |
| **USBD_EPIBUFSTART** | USBD_BA+0x1C4 | R/W | Endpoint I RAM Start Address Register | 0x0000_0000 |
| **USBD_EPIBUFEND** | USBD_BA+0x1C8 | R/W | Endpoint I RAM End Address Register | 0x0000_0000 |
| **USBD_EPJDAT** | USBD_BA+0x1CC | R/W | Endpoint J Data Register | 0x0000_0000 |
| **USBD_EPJINTSTS** | USBD_BA+0x1D0 | R/W | Endpoint J Interrupt Status Register | 0x0000_0003 |
| **USBD_EPJINTEN** | USBD_BA+0x1D4 | R/W | Endpoint J Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPJDATCNT** | USBD_BA+0x1D8 | R | Endpoint J Data Available Count Register | 0x0000_0000 |
| **USBD_EPJRSPCTL** | USBD_BA+0x1DC | R/W | Endpoint J Response Control Register | 0x0000_0000 |
| **USBD_EPJMPS** | USBD_BA+0x1E0 | R/W | Endpoint J Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPJTXCNT** | USBD_BA+0x1E4 | R/W | Endpoint J Transfer Count Register | 0x0000_0000 |
| **USBD_EPJCFG** | USBD_BA+0x1E8 | R/W | Endpoint J Configuration Register | 0x0000_00A2 |

| | | | | |
|---|---|---|---|---|
| **USBD_EPJBUFSTART** | USBD_BA+0x1EC | R/W | Endpoint J RAM Start Address Register | 0x0000_0000 |
| **USBD_EPJBUFEND** | USBD_BA+0x1F0 | R/W | Endpoint J RAM End Address Register | 0x0000_0000 |
| **USBD_EPKDAT** | USBD_BA+0x1F4 | R/W | Endpoint K Data Register | 0x0000_0000 |
| **USBD_EPKINTSTS** | USBD_BA+0x1F8 | R/W | Endpoint K Interrupt Status Register | 0x0000_0003 |
| **USBD_EPKINTEN** | USBD_BA+0x1FC | R/W | Endpoint K Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPKDATCNT** | USBD_BA+0x200 | R | Endpoint K Data Available Count Register | 0x0000_0000 |
| **USBD_EPKRSPCTL** | USBD_BA+0x204 | R/W | Endpoint K Response Control Register | 0x0000_0000 |
| **USBD_EPKMPS** | USBD_BA+0x208 | R/W | Endpoint K Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPKTXCNT** | USBD_BA+0x20C | R/W | Endpoint K Transfer Count Register | 0x0000_0000 |
| **USBD_EPKCFG** | USBD_BA+0x210 | R/W | Endpoint K Configuration Register | 0x0000_00B2 |
| **USBD_EPKBUFSTART** | USBD_BA+0x214 | R/W | Endpoint K RAM Start Address Register | 0x0000_0000 |
| **USBD_EPKBUFEND** | USBD_BA+0x218 | R/W | Endpoint K RAM End Address Register | 0x0000_0000 |
| **USBD_EPLDAT** | USBD_BA+0x21C | R/W | Endpoint L Data Register | 0x0000_0000 |
| **USBD_EPLINTSTS** | USBD_BA+0x220 | R/W | Endpoint L Interrupt Status Register | 0x0000_0003 |
| **USBD_EPLINTEN** | USBD_BA+0x224 | R/W | Endpoint L Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPLDATCNT** | USBD_BA+0x228 | R | Endpoint L Data Available Count Register | 0x0000_0000 |
| **USBD_EPLRSPCTL** | USBD_BA+0x22C | R/W | Endpoint L Response Control Register | 0x0000_0000 |
| **USBD_EPLMPS** | USBD_BA+0x230 | R/W | Endpoint L Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPLTXCNT** | USBD_BA+0x234 | R/W | Endpoint L Transfer Count Register | 0x0000_0000 |
| **USBD_EPLCFG** | USBD_BA+0x238 | R/W | Endpoint L Configuration Register | 0x0000_00C2 |
| **USBD_EPLBUFSTART** | USBD_BA+0x23C | R/W | Endpoint L RAM Start Address Register | 0x0000_0000 |
| **USBD_EPLBUFEND** | USBD_BA+0x240 | R/W | Endpoint L RAM End Address Register | 0x0000_0000 |
| **USBD_DMAADDR** | USBD_BA+0x700 | R/W | AHB DMA Address Register | 0x0000_0000 |
| **USBD_PHYCTL** | USBD_BA+0x704 | R/W | USB PHY Control Register | 0x0000_0420 |

### 5.22.7 Register Description

## Global Interrupt Status Register (USBD_GINTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_GINTSTS** | USBD_BA+0x000 | R | 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** | | **EPLIF** | **EPKIF** | **EPJIF** | **EPIIF** | **EPHIF** | **EPGIF** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **EPFIF** | **EPEIF** | **EPDIF** | **EPCIF** | **EPBIF** | **EPAIF** | **CEPIF** | **USBIF** |

| Bits | Description | |
|---|---|---|
| [31:14] | **Reserved** | Reserved. |
| [13] | **EPLIF** | **Endpoints L Interrupt**<br>When set, the corresponding Endpoint L's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [12] | **EPKIF** | **Endpoints K Interrupt**<br>When set, the corresponding Endpoint K's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [11] | **EPJIF** | **Endpoints J Interrupt**<br>When set, the corresponding Endpoint J's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [10] | **EPIIF** | **Endpoints I Interrupt**<br>When set, the corresponding Endpoint I's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [9] | **EPHIF** | **Endpoints H Interrupt**<br>When set, the corresponding Endpoint H's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |

| | | |
|---|---|---|
| [8] | **EPGIF** | **Endpoints G Interrupt**<br>When set, the corresponding Endpoint G's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [7] | **EPFIF** | **Endpoints F Interrupt**<br>When set, the corresponding Endpoint F's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [6] | **EPEIF** | **Endpoints E Interrupt**<br>When set, the corresponding Endpoint E's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [5] | **EPDIF** | **Endpoints D Interrupt**<br>When set, the corresponding Endpoint D's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [4] | **EPCIF** | **Endpoints C Interrupt**<br>When set, the corresponding Endpoint C's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [3] | **EPBIF** | **Endpoints B Interrupt**<br>When set, the corresponding Endpoint B's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [2] | **EPAIF** | **Endpoints a Interrupt**<br>When set, the corresponding Endpoint A's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [1] | **CEPIF** | **Control Endpoint Interrupt**<br>This bit conveys the interrupt status for control endpoint. When set, Control-ep's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [0] | **USBIF** | **USB Interrupt**<br>This bit conveys the interrupt status for USB specific events endpoint. When set, USB interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |

## Global Interrupt Enable Register (USBD_GINTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_GINTEN** | USBD_BA+0x008 | R/W | Global Interrupt Enable 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** | | **EPLIEN** | **EPKIEN** | **EPJIEN** | **EPIIEN** | **EPHIEN** | **EPGIENNN** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **EPFIEN** | **EPEIEN** | **EPDIEN** | **EPCIEN** | **EPBIEN** | **EPAIEN** | **CEPIEN** | **USBIEN** |

| Bits | Description | |
|---|---|---|
| [31:14] | **Reserved** | Reserved. |
| [13] | **EPLIEN** | **Interrupt Enable Control for Endpoint L**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint L<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [12] | **EPKIEN** | **Interrupt Enable Control for Endpoint K**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint K<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [11] | **EPJIEN** | **Interrupt Enable Control for Endpoint J**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint J<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [10] | **EPIIEN** | **Interrupt Enable Control for Endpoint I**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint I<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [9] | **EPHIEN** | **Interrupt Enable Control for Endpoint H**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint H<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |

| | | |
|---|---|---|
| [8] | **EPGIEN** | **Interrupt Enable Control for Endpoint G**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint G<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [7] | **EPFIEN** | **Interrupt Enable Control for Endpoint F**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint F<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [6] | **EPEIEN** | **Interrupt Enable Control for Endpoint E**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint E<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [5] | **EPDIEN** | **Interrupt Enable Control for Endpoint D**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint D<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [4] | **EPCIEN** | **Interrupt Enable Control for Endpoint C**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint C<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [3] | **EPBIEN** | **Interrupt Enable Control for Endpoint B**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint B<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [2] | **EPAIEN** | **Interrupt Enable Control for Endpoint a**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint A.<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [1] | **CEPIEN** | **Control Endpoint Interrupt Enable Control**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the control endpoint.<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [0] | **USBIEN** | **USB Interrupt Enable Control**<br>When set, this bit enables a local interrupt to be generated when a USB event occurs on the bus.<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |

## USB Bus Interrupt Status Register (USBD_BUSINTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_BUSINTSTS** | USBD_BA+0x010 | R/W | USB Bus 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** | | | | | | | **VBUSDETIF** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | **PHYCLKVLDIF** | **DMADONEIF** | **HISPDIF** | **SUSPENDIF** | **RESUMEIF** | **RSTIF** | **SOFIF** |

| Bits | Description | |
|---|---|---|
| [31:9] | **Reserved** | Reserved. |
| [8] | **VBUSDETIF** | **VBUS Detection Interrupt Status**<br>0 = No VBUS is plug-in.<br>1 = VBUS is plug-in.<br>**Note:** Write 1 to clear this bit to 0. |
| [7] | **Reserved** | Reserved. |
| [6] | **PHYCLKVLDIF** | **Usable Clock Interrupt**<br>0 = Usable clock is not available.<br>1 = Usable clock is available from the transceiver.<br>**Note:** Write 1 to clear this bit to 0. |
| [5] | **DMADONEIF** | **DMA Completion Interrupt**<br>0 = No DMA transfer over.<br>1 = DMA transfer is over.<br>**Note:** Write 1 to clear this bit to 0. |
| [4] | **HISPDIF** | **High-speed Settle**<br>0 = No valid high-speed reset protocol is detected.<br>1 = Valid high-speed reset protocol is over and the device has settled in high-speed.<br>**Note:** Write 1 to clear this bit to 0. |
| [3] | **SUSPENDIF** | **Suspend Request**<br>This bit is set as default and it has to be cleared by writing '1' before the USB reset. This bit is also set when a USB Suspend request is detected from the host.<br>0 = No USB Suspend request is detected from the host.<br>1= USB Suspend request is detected from the host.<br>**Note:** Write 1 to clear this bit to 0. |

| | | |
|---|---|---|
| [2] | **RESUMEIF** | **Resume**<br>When set, this bit indicates that a device resume has occurred.<br>0 = No device resume has occurred.<br>1 = Device resume has occurred.<br>**Note:** Write 1 to clear this bit to 0. |
| [1] | **RSTIF** | **Reset Status**<br>When set, this bit indicates that either the USB root port reset is end.<br>0 = No USB root port reset is end.<br>1 = USB root port reset is end.<br>**Note:** Write 1 to clear this bit to 0. |
| [0] | **SOFIF** | **SOF Receive Control**<br>This bit indicates when a start-of-frame packet has been received.<br>0 = No start-of-frame packet has been received.<br>1 = Start-of-frame packet has been received.<br>**Note:** Write 1 to clear this bit to 0. |

## USB Bus Interrupt Enable Register (USBD_BUSINTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_BUSINTEN** | USBD_BA+0x014 | R/W | USB Bus Interrupt Enable 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** | | | | | | | **VBUSDETIEN** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | **PHYCLKVLDIEN** | **DMADONEIEN** | **HISPDIEN** | **SUSPENDIEN** | **RESUMEIEN** | **RSTIEN** | **SOFIEN** |

| Bits | Description | |
|---|---|---|
| [31:9] | **Reserved** | Reserved. |
| [8] | **VBUSDETIEN** | **VBUS Detection Interrupt Enable Control**<br>This bit enables the VBUS floating detection interrupt.<br>0 = VBUS floating detection interrupt Disabled.<br>1 = VBUS floating detection interrupt Enabled. |
| [7] | **Reserved** | Reserved. |
| [6] | **PHYCLKVLDIEN** | **Usable Clock Interrupt**<br>This bit enables the usable clock interrupt.<br>0 = Usable clock interrupt Disabled.<br>1 = Usable clock interrupt Enabled. |
| [5] | **DMADONEIEN** | **DMA Completion Interrupt**<br>This bit enables the DMA completion interrupt<br>0 = DMA completion interrupt Disabled.<br>1 = DMA completion interrupt Enabled. |
| [4] | **HISPDIEN** | **High-speed Settle**<br>This bit enables the high-speed settle interrupt.<br>0 = High-speed settle interrupt Disabled.<br>1 = High-speed settle interrupt Enabled. |
| [3] | **SUSPENDIEN** | **Suspend Request**<br>This bit enables the Suspend interrupt.<br>0 = Suspend interrupt Disabled.<br>1 = Suspend interrupt Enabled. |

| | | |
|---|---|---|
| [2] | **RESUMEIEN** | **Resume**<br>This bit enables the Resume interrupt.<br>0 = Resume interrupt Disabled.<br>1 = Resume interrupt Enabled. |
| [1] | **RSTIEN** | **Reset Status**<br>This bit enables the USB-Reset interrupt.<br>0 = USB-Reset interrupt Disabled.<br>1 = USB-Reset interrupt Enabled. |
| [0] | **SOFIEN** | **SOF Interrupt**<br>This bit enables the SOF interrupt.<br>0 = SOF interrupt Disabled.<br>1 = SOF interrupt Enabled. |

## USB Operational Register (USBD_OPER)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_OPER** | USBD_BA+0x018 | R/W | USB Operational 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** |
| **Reserved** | | | | | **CURSPD** | **HISPDEN** | **RESUMEEN** |

| Bits | Description | |
|---|---|---|
| [31:3] | **Reserved** | Reserved. |
| [2] | **CURSPD** | **USB Current Speed**<br>0 = The device has settled in Full Speed.<br>1 = The USB device controller has settled in High-speed. |
| [1] | **HISPDEN** | **USB High-speed**<br>0 = The USB device controller to suppress the chirp-sequence during reset protocol, thereby allowing the USB device controller to settle in full-speed, even though it is connected to a USB2.0 Host.<br>1 = The USB device controller to initiate a chirp-sequence during reset protocol. |
| [0] | **RESUMEEN** | **Generate Resume**<br>0 = No Resume sequence to be initiated to the host.<br>1 = A Resume sequence to be initiated to the host if device remote wakeup is enabled. This bit is self-clearing. |

## USB Frame Count Register (USBD_FRAMECNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_FRAMECNT** | USBD_BA+0x01C | R | USB Frame Count 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** | | **FRAMECNT** | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **FRAMECNT** | | | | | **MFRAMECNT** | | |

| Bits | Description | |
|---|---|---|
| [31:14] | **Reserved** | Reserved. |
| [13:3] | **FRAMECNT** | **Frame Counter**<br>This field contains the frame count from the most recent start-of-frame packet. |
| [2:0] | **MFRAMECNT** | **Micro-frame Counter**<br>This field contains the micro-frame number for the frame number in the frame counter field. |

## USB Function Address Register (USBD_FADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_FADDR** | USBD_BA+0x020 | R/W | USB 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 Function Address**<br>This field contains the current USB address of the device. This field is cleared when a root port reset is detected. |

## USB Test Mode Register (USBD_TEST)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_TEST** | USBD_BA+0x024 | R/W | USB Test Mode 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** | | | | | **TESTMODE** | | |

| Bits | Description | |
|---|---|---|
| [31:3] | **Reserved** | Reserved. |
| [2:0] | **TESTMODE** | **Test Mode Selection**<br>000 = Normal Operation.<br>001 = Test_J.<br>010 = Test_K.<br>011 = Test_SE0_NAK.<br>100 = Test_Packet.<br>101 = Test_Force_Enable.<br>110 = Reserved.<br>111 = Reserved.<br>**Note:** This field is cleared when root port reset is detected. |

## Control Endpoint Data Buffer (USBD_CEPDAT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_CEPDAT** | USBD_BA+0x028 | R/W | Control Endpoint Data Buffer | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **DAT** | **Control-endpoint Data Buffer**<br>Control endpoint data buffer for the buffer transaction (read or write).<br>**Note:** Only word or byte access are supported. |

## Control Endpoint Control Register (USBD_CEPCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_CEPCTL** | USBD_BA+0x02C | R/W | Control Endpoint 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** | | | | **FLUSH** | **ZEROLEN** | **STALLEN** | **NAKCLR** |

| Bits | Description | |
|---|---|---|
| [31:4] | **Reserved** | Reserved. |
| [3] | **FLUSH** | **CEP-fLUSH Bit**<br>0 = No the packet buffer and its corresponding USBD_CEPDATCNT register to be cleared.<br>1 = The packet buffer and its corresponding USBD_CEPDATCNT register to be cleared. This bit is self-cleaning. |
| [2] | **ZEROLEN** | **Zero Packet Length**<br>This bit is valid for Auto Validation mode only.<br>0 = No zero length packet to the host during Data stage to an IN token.<br>1 = USB device controller can send a zero length packet to the host during Data stage to an IN token. This bit gets cleared once the zero length data packet is sent. So, the local CPU need not write again to clear this bit. |
| [1] | **STALLEN** | **Stall Enable Control**<br>When this stall bit is set, the control endpoint sends a stall handshake in response to any in or out token thereafter. This is typically used for response to invalid/unsupported requests. When this bit is being set the NAK clear bit has to be cleared at the same time since the NAK clear bit has highest priority than STALL. It is automatically cleared on receipt of a next setup-token. So, the local CPU need not write again to clear this bit.<br>0 = No sends a stall handshake in response to any in or out token thereafter.<br>1 = The control endpoint sends a stall handshake in response to any in or out token thereafter.<br>**Note:** Only when CPU writes data[1:0] is 2'b10 or 2'b00, this bit can be updated. |

| | | |
|---|---|---|
| [0] | **NAKCLR** | **No Acknowledge Control**<br>This bit plays a crucial role in any control transfer.<br>0 = The bit is being cleared by the local CPU by writing zero, the USB device controller will be responding with NAKs for the subsequent status phase. This mechanism holds the host from moving to the next request, until the local CPU is also ready to process the next request.<br>1 = This bit is set to one by the USB device controller, whenever a setup token is received. The local CPU can take its own time to finish off any house-keeping work based on the request and then clear this bit.<br>**Note:** Only when CPU writes data[1:0] is 2'b10 or 2'b00, this bit can be updated. |

## Control Endpoint Interrupt Enable(USBD_CEPINTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_CEPINTEN** | USBD_BA+0x030 | R/W | Control Endpoint Interrupt Enable | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | **BUFEMPTYIEN** | **BUFFULLIEN** | **STSDONEIEN** | **ERRIEN** | **STALLIEN** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **NAKIEN** | **RXPKIEN** | **TXPKIEN** | **PINGIEN** | **INTKIEN** | **OUTTKIEN** | **SETUPPKIEN** | **SETUPTKIEN** |

| Bits | Description | |
|---|---|---|
| [31:13] | **Reserved** | Reserved. |
| [12] | **BUFEMPTYIEN** | **Buffer Empty Interrupt**<br>0 = The buffer empty interrupt in Control Endpoint Disabled.<br>1= The buffer empty interrupt in Control Endpoint Enabled. |
| [11] | **BUFFULLIEN** | **Buffer Full Interrupt**<br>0 = The buffer full interrupt in Control Endpoint Disabled.<br>1 = The buffer full interrupt in Control Endpoint Enabled. |
| [10] | **STSDONEIEN** | **Status Completion Interrupt**<br>0 = The Status Completion interrupt in Control Endpoint Disabled.<br>1 = The Status Completion interrupt in Control Endpoint Enabled. |
| [9] | **ERRIEN** | **USB Error Interrupt**<br>0 = The USB Error interrupt in Control Endpoint Disabled.<br>1 = The USB Error interrupt in Control Endpoint Enabled. |
| [8] | **STALLIEN** | **STALL Sent Interrupt**<br>0 = The STALL sent interrupt in Control Endpoint Disabled.<br>1 = The STALL sent interrupt in Control Endpoint Enabled. |
| [7] | **NAKIEN** | **NAK Sent Interrupt**<br>0 = The NAK sent interrupt in Control Endpoint Disabled.<br>1 = The NAK sent interrupt in Control Endpoint Enabled. |
| [6] | **RXPKIEN** | **Data Packet Received Interrupt**<br>0 = The data received interrupt in Control Endpoint Disabled.<br>1 = The data received interrupt in Control Endpoint Enabled. |

| | | |
|---|---|---|
| [5] | **TXPKIEN** | **Data Packet Transmitted Interrupt**<br>0 = The data packet transmitted interrupt in Control Endpoint Disabled.<br>1 = The data packet transmitted interrupt in Control Endpoint Enabled. |
| [4] | **PINGIEN** | **Ping Token Interrupt**<br>0 = The ping token interrupt in Control Endpoint Disabled.<br>1 = The ping token interrupt Control Endpoint Enabled. |
| [3] | **INTKIEN** | **In Token Interrupt**<br>0 = The IN token interrupt in Control Endpoint Disabled.<br>1 = The IN token interrupt in Control Endpoint Enabled. |
| [2] | **OUTTKIEN** | **Out Token Interrupt**<br>0 = The OUT token interrupt in Control Endpoint Disabled.<br>1 = The OUT token interrupt in Control Endpoint Enabled. |
| [1] | **SETUPPKIEN** | **Setup Packet Interrupt**<br>0 = The SETUP packet interrupt in Control Endpoint Disabled.<br>1 = The SETUP packet interrupt in Control Endpoint Enabled. |
| [0] | **SETUPTKIEN** | **Setup Token Interrupt Enable Control**<br>0 = The SETUP token interrupt in Control Endpoint Disabled.<br>1 = The SETUP token interrupt in Control Endpoint Enabled. |

## Control Endpoint Interrupt Status (USBD_CEPINTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_CEPINTSTS** | USBD_BA+0x034 | R/W | Control Endpoint Interrupt Status | 0x0000_1800 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | **BUFEMPTYIF** | **BUFFULLIF** | **STSDONEIF** | **ERRIF** | **STALLIF** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **NAKIF** | **RXPKIF** | **TXPKIF** | **PINGIF** | **INTKIF** | **OUTTKIF** | **SETUPPKIF** | **SETUPTKIF** |

| Bits | Description | |
|---|---|---|
| [31:13] | **Reserved** | Reserved. |
| [12] | **BUFEMPTYIF** | **Buffer Empty Interrupt**<br>0 = The control-endpoint buffer is not empty.<br>1 = The control-endpoint buffer is empty.<br>**Note:** Write 1 to clear this bit to 0. |
| [11] | **BUFFULLIF** | **Buffer Full Interrupt**<br>0 = The control-endpoint buffer is not full.<br>1 = The control-endpoint buffer is full.<br>**Note:** Write 1 to clear this bit to 0. |
| [10] | **STSDONEIF** | **Status Completion Interrupt**<br>0 = Not a USB transaction has completed successfully.<br>1 = The status stage of a USB transaction has completed successfully.<br>**Note:** Write 1 to clear this bit to 0. |
| [9] | **ERRIF** | **USB Error Interrupt**<br>0 = No error had occurred during the transaction.<br>1 = An error had occurred during the transaction.<br>**Note:** Write 1 to clear this bit to 0. |
| [8] | **STALLIF** | **STALL Sent Interrupt**<br>0 = Not a stall-token is sent in response to an IN/OUT token.<br>1 = A stall-token is sent in response to an IN/OUT token.<br>**Note:** Write 1 to clear this bit to 0. |
| [7] | **NAKIF** | **NAK Sent Interrupt**<br>0 = Not a NAK-token is sent in response to an IN/OUT token.<br>1 = A NAK-token is sent in response to an IN/OUT token.<br>**Note:** Write 1 to clear this bit to 0. |

| | | |
|---|---|---|
| [6] | **RXPKIF** | **Data Packet Received Interrupt**<br>0 = Not a data packet is successfully received from the host for an OUT-token and an ACK is sent to the host.<br>1 = A data packet is successfully received from the host for an OUT-token and an ACK is sent to the host.<br>**Note:** Write 1 to clear this bit to 0. |
| [5] | **TXPKIF** | **Data Packet Transmitted Interrupt**<br>0 = Not a data packet is successfully transmitted to the host in response to an IN-token and an ACK-token is received for the same.<br>1 = A data packet is successfully transmitted to the host in response to an IN-token and an ACK-token is received for the same.<br>**Note:** Write 1 to clear this bit to 0. |
| [4] | **PINGIF** | **Ping Token Interrupt**<br>0 = The control-endpoint does not received a ping token from the host.<br>1 = The control-endpoint receives a ping token from the host.<br>**Note:** Write 1 to clear this bit to 0. |
| [3] | **INTKIF** | **In Token Interrupt**<br>0 = The control-endpoint does not received an IN token from the host.<br>1 = The control-endpoint receives an IN token from the host.<br>**Note:** Write 1 to clear this bit to 0. |
| [2] | **OUTTKIF** | **Out Token Interrupt**<br>0 = The control-endpoint does not received an OUT token from the host.<br>1 = The control-endpoint receives an OUT token from the host.<br>**Note:** Write 1 to clear this bit to 0. |
| [1] | **SETUPPKIF** | **Setup Packet Interrupt**<br>This bit must be cleared (by writing 1) before the next setup packet can be received. If the bit is not cleared, then the successive setup packets will be overwritten in the setup packet buffer.<br>0 = Not a Setup packet has been received from the host.<br>1 = A Setup packet has been received from the host.<br>**Note:** Write 1 to clear this bit to 0. |
| [0] | **SETUPTKIF** | **Setup Token Interrupt**<br>0 = Not a Setup token is received.<br>1 = A Setup token is received. Writing 1 clears this status bit<br>**Note:** Write 1 to clear this bit to 0. |

## Control Endpoint In Transfer Data Count (USBD_CEPTXCNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_CEPTXCNT** | USBD_BA+0x038 | R/W | Control Endpoint In Transfer Data Count | 0x0000_0000 |

| 31 | 30 | 29 | 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** |
| **TXCNT** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **TXCNT** | **In-transfer Data Count**<br>There is no mode selection for the control endpoint (but it operates like manual mode).The local-CPU has to fill the control-endpoint buffer with the data to be sent for an in-token and to write the count of bytes in this register. When zero is written into this field, a zero length packet is sent to the host. When the count written in the register is more than the MPS, the data sent will be of only MPS. |

## Control Endpoint Out Transfer Data Count (USBD_CEPRXCNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_CEPRXCNT** | USBD_BA+0x03C | R | Control Endpoint Out Transfer Data Count | 0x0000_0000 |

| 31 | 30 | 29 | 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** |
| **RXCNT** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **RXCNT** | **Out-transfer Data Count**<br>The USB device controller maintains the count of the data received in case of an out transfer, during the control transfer. |

## Control Endpoint Data Count (USBD_CEPDATCNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_CEPDATCNT** | USBD_BA+0x040 | R | Control Endpoint Data Count | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **DATCNT** | **Control-endpoint Data Count**<br>The USB device controller maintains the count of the data of control-endpoint. |

Setup1 & Setup0 bytes (USBD_SETUP1_0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_SETUP1_0** | USBD_BA+0x044 | R | Setup1 & Setup0 bytes | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **SETUP1** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **SETUP0** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:8] | **SETUP1** | **Setup Byte 1[15:8]**<br>This register provides byte 1 of the last setup packet received. For a Standard Device Request, the following bRequest Code information is returned.<br>00000000 = Get Status.<br>00000001 = Clear Feature.<br>00000010 = Reserved.<br>00000011 = Set Feature.<br>00000100 = Reserved.<br>00000101 = Set Address.<br>00000110 = Get Descriptor.<br>00000111 = Set Descriptor.<br>00001000 = Get Configuration.<br>00001001 = Set Configuration.<br>00001010 = Get Interface.<br>00001011 = Set Interface.<br>00001100 = Synch Frame. |

| | | |
|---|---|---|
| [7:0] | **SETUP0** | **Setup Byte 0[7:0]**<br>This register provides byte 0 of the last setup packet received. For a Standard Device Request, the following bmRequestType information is returned.<br>Bit 7(Direction):<br>0: Host to device<br>1: Device to host<br>Bit 6-5 (Type):<br>00: Standard<br>01: Class<br>10: Vendor<br>11: Reserved<br>Bit 4-0 (Recipient)<br>00000**:** Device<br>00001: Interface<br>00010: Endpoint<br>00011: Other<br>Others: Reserved |

### Setup3 & Setup2 Bytes (USBD_SETUP3_2)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_SETUP3_2** | USBD_BA+0x048 | R | Setup3 & Setup2 Bytes | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **SETUP3** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **SETUP2** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:8] | **SETUP3** | **Setup Byte 3 [15:8]**<br>This register provides byte 3 of the last setup packet received. For a Standard Device Request, the most significant byte of the wValue field is returned. |
| [7:0] | **SETUP2** | **Setup Byte 2 [7:0]**<br>This register provides byte 2 of the last setup packet received. For a Standard Device Request, the least significant byte of the wValue field is returned. |

Setup5 & Setup4 Bytes (USBD_SETUP5_4)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_SETUP5_4** | USBD_BA+0x04C | R | Setup5 & Setup4 Bytes | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **SETUP5** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **SETUP4** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:8] | **SETUP5** | **Setup Byte 5[15:8]**<br>This register provides byte 5 of the last setup packet received. For a Standard Device Request, the most significant byte of the wIndex field is returned. |
| [7:0] | **SETUP4** | **Setup Byte 4[7:0]**<br>This register provides byte 4 of the last setup packet received. For a Standard Device Request, the least significant byte of the wIndex is returned. |

Setup7 & Setup6 bytes (USBD_SETUP7_6)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_SETUP7_6** | USBD_BA+0x050 | R | Setup7 & Setup6 Bytes | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **SETUP7** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **SETUP6** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:8] | **SETUP7** | **Setup Byte 7[15:8]**<br>This register provides byte 7 of the last setup packet received. For a Standard Device Request, the most significant byte of the wLength field is returned. |
| [7:0] | **SETUP6** | **Setup Byte 6[7:0]**<br>This register provides byte 6 of the last setup packet received. For a Standard Device Request, the least significant byte of the wLength field is returned. |

## Control Endpoint RAM Start Address Register (USBD_CEPBUFSTART)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_CEPBUFSTART** | USBD_BA+0x054 | R/W | Control Endpoint RAM Start 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** | | | | **SADDR** | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **SADDR** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:12] | **Reserved** | Reserved. |
| [11:0] | **SADDR** | **Control-endpoint Start Address**<br>This is the start-address of the RAM space allocated for the control-endpoint. |

## Control Endpoint RAM End Address Register (USBD_CEPBUFEND)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_CEPBUFEND** | USBD_BA+0x058 | R/W | Control Endpoint RAM End 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** | | | | **EADDR** | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **EADDR** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:12] | **Reserved** | Reserved. |
| [11:0] | **EADDR** | **Control-endpoint End Address**<br>This is the end-address of the RAM space allocated for the control-endpoint. |

## DMA Control Status Register (USBD_DMACTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_DMACTL** | USBD_BA+0x05C | R/W | DMA Control 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** |
| **DMARST** | **SGEN** | **DMAEN** | **DMARD** | **EPNUM** | | | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7] | **DMARST** | **Reset DMA State Machine**<br>0 = No reset the DMA state machine.<br>1 = Reset the DMA state machine. |
| [6] | **SGEN** | **Scatter Gather Function Enable Control**<br>0 = Scatter gather function Disabled.<br>1 = Scatter gather function Enabled. |
| [5] | **DMAEN** | **DMA Enable Control**<br>0 = DMA function Disabled.<br>1 = DMA function Enabled. |
| [4] | **DMARD** | **DMA Operation**<br>0 = The operation is a DMA write (read from USB buffer). DMA will check endpoint data available count (USBD_EPxDATCNT) according to EPNM setting before to perform DMA write operation.<br>1 = The operation is a DMA read (write to USB buffer). |
| [3:0] | **EPNUM** | **DMA Endpoint Address Bits**<br>Used to define the Endpoint Address |

## DMA Count Register (USBD_DMACNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_DMACNT** | USBD_BA+0x060 | R/W | DMA Count Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:20] | **Reserved** | Reserved. |
| [19:0] | **DMACNT** | **DMA Transfer Count**<br>The transfer count of the DMA operation to be performed is written to this register. |

Endpoint A~L Data Register (USBD_EPADAT~ USBD_EPLDAT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_EPADAT** | USBD_BA+0x064 | R/W | Endpoint A Data Register | 0x0000_0000 |
| **USBD_EPBDAT** | USBD_BA+0x08C | R/W | Endpoint B Data Register | 0x0000_0000 |
| **USBD_EPCDAT** | USBD_BA+0x0B4 | R/W | Endpoint C Data Register | 0x0000_0000 |
| **USBD_EPDDAT** | USBD_BA+0x0DC | R/W | Endpoint D Data Register | 0x0000_0000 |
| **USBD_EPEDAT** | USBD_BA+0x104 | R/W | Endpoint E Data Register | 0x0000_0000 |
| **USBD_EPFDAT** | USBD_BA+0x12C | R/W | Endpoint F Data Register | 0x0000_0000 |
| **USBD_EPGDAT** | USBD_BA+0x154 | R/W | Endpoint G Data Register | 0x0000_0000 |
| **USBD_EPHDAT** | USBD_BA+0x17C | R/W | Endpoint H Data Register | 0x0000_0000 |
| **USBD_EPIDAT** | USBD_BA+0x1A4 | R/W | Endpoint I Data Register | 0x0000_0000 |
| **USBD_EPJDAT** | USBD_BA+0x1CC | R/W | Endpoint J Data Register | 0x0000_0000 |
| **USBD_EPKDAT** | USBD_BA+0x1F4 | R/W | Endpoint K Data Register | 0x0000_0000 |
| **USBD_EPLDAT** | USBD_BA+0x21C | R/W | Endpoint L Data Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **EPDAT** | **Endpoint A~L Data Register**<br>Endpoint A~L data buffer for the buffer transaction (read or write).<br>**Note:** Only word or byte access are supported. |

Endpoint A~L Interrupt Status Register (USBD_EPAINTSTS~ USBD_EPLINTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_EPAINTSTS** | USBD_BA+0x068 | R/W | Endpoint A Interrupt Status Register | 0x0000_0003 |
| **USBD_EPBINTSTS** | USBD_BA+0x090 | R/W | Endpoint B Interrupt Status Register | 0x0000_0003 |
| **USBD_EPCINTSTS** | USBD_BA+0x0B8 | R/W | Endpoint C Interrupt Status Register | 0x0000_0003 |
| **USBD_EPDINTSTS** | USBD_BA+0x0E0 | R/W | Endpoint D Interrupt Status Register | 0x0000_0003 |
| **USBD_EPEINTSTS** | USBD_BA+0x108 | R/W | Endpoint E Interrupt Status Register | 0x0000_0003 |
| **USBD_EPFINTSTS** | USBD_BA+0x130 | R/W | Endpoint F Interrupt Status Register | 0x0000_0003 |
| **USBD_EPGINTSTS** | USBD_BA+0x158 | R/W | Endpoint G Interrupt Status Register | 0x0000_0003 |
| **USBD_EPHINTSTS** | USBD_BA+0x180 | R/W | Endpoint H Interrupt Status Register | 0x0000_0003 |
| **USBD_EPIINTSTS** | USBD_BA+0x1A8 | R/W | Endpoint I Interrupt Status Register | 0x0000_0003 |
| **USBD_EPJINTSTS** | USBD_BA+0x1D0 | R/W | Endpoint J Interrupt Status Register | 0x0000_0003 |
| **USBD_EPKINTSTS** | USBD_BA+0x1F8 | R/W | Endpoint K Interrupt Status Register | 0x0000_0003 |
| **USBD_EPLINTSTS** | USBD_BA+0x220 | R/W | Endpoint L Interrupt Status Register | 0x0000_0003 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | **SHORTRXIF** | **ERRIF** | **NYETIF** | **STALLIF** | **NAKIF** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **PINGIF** | **INTKIF** | **OUTTKIF** | **RXPKIF** | **TXPKIF** | **SHORTTXIF** | **BUFEMPTYIF** | **BUFFULLIF** |

| Bits | Description | |
|---|---|---|
| [31:13] | **Reserved** | Reserved. |
| [12] | **SHORTRXIF** | **Bulk Out Short Packet Received**<br>0 = No bulk out short packet is received.<br>1 = Received bulk out short packet (including zero length packet).<br>**Note:** Write 1 to clear this bit to 0. |
| [11] | **ERRIF** | **ERR Sent**<br>0 = No any error in the transaction.<br>1 = There occurs any error in the transaction.<br>**Note:** Write 1 to clear this bit to 0. |

| | | |
|---|---|---|
| [10] | **NYETIF** | **NYET Sent**<br>0 = The space available in the RAM is sufficient to accommodate the next on coming data packet.<br>1 = The space available in the RAM is not sufficient to accommodate the next on coming data packet.<br>**Note:** Write 1 to clear this bit to 0. |
| [9] | **STALLIF** | **USB STALL Sent**<br>0 = The last USB packet could be accepted or provided because the endpoint was stalled, and was acknowledged with a STALL.<br>1 = The last USB packet could not be accepted or provided because the endpoint was stalled, and was acknowledged with a STALL.<br>**Note:** Write 1 to clear this bit to 0. |
| [8] | **NAKIF** | **USB NAK Sent**<br>0 = The last USB IN packet could be provided, and was acknowledged with an ACK.<br>1 = The last USB IN packet could not be provided, and was acknowledged with a NAK.<br>**Note:** Write 1 to clear this bit to 0. |
| [7] | **PINGIF** | **PING Token Interrupt**<br>0 = A Data PING token has not been received from the host.<br>1 = A Data PING token has been received from the host.<br>**Note:** Write 1 to clear this bit to 0. |
| [6] | **INTKIF** | **Data IN Token Interrupt**<br>0 = Not Data IN token has been received from the host.<br>1 = A Data IN token has been received from the host.<br>**Note:** Write 1 to clear this bit to 0. |
| [5] | **OUTTKIF** | **Data OUT Token Interrupt**<br>0 = A Data OUT token has not been received from the host.<br>1 = A Data OUT token has been received from the host. This bit also set by PING token (in high-speed only).<br>**Note:** Write 1 to clear this bit to 0. |
| [4] | **RXPKIF** | **Data Packet Received Interrupt**<br>0 = No data packet is received from the host by the endpoint.<br>1 = A data packet is received from the host by the endpoint.<br>**Note:** Write 1 to clear this bit to 0. |
| [3] | **TXPKIF** | **Data Packet Transmitted Interrupt**<br>0 = Not a data packet is transmitted from the endpoint to the host.<br>1 = A data packet is transmitted from the endpoint to the host.<br>**Note:** Write 1 to clear this bit to 0. |
| [2] | **SHORTTXIF** | **Short Packet Transferred Interrupt**<br>0 = The length of the last packet was not less than the Maximum Packet Size (EPMPS).<br>1 = The length of the last packet was less than the Maximum Packet Size (EPMPS).<br>**Note:** Write 1 to clear this bit to 0. |

| | | |
|---|---|---|
| [1] | **BUFEMPTYIF** | **Buffer Empty**<br>For an IN endpoint, a buffer is available to the local side for writing up to FIFO full of bytes.<br>0 = The endpoint buffer is not empty.<br>1 = The endpoint buffer is empty.<br>For an OUT endpoint:<br>0 = The currently selected buffer has not a count of 0.<br>1 = The currently selected buffer has a count of 0, or no buffer is available on the local side (nothing to read).<br>**Note:** This bit is read-only. |
| [0] | **BUFFULLIF** | **Buffer Full**<br>For an IN endpoint, the currently selected buffer is full, or no buffer is available to the local side for writing (no space to write). For an OUT endpoint, there is a buffer available on the local side, and there are FIFO full of bytes available to be read (entire packet is available for reading).<br>0 = The endpoint packet buffer is not full.<br>1 = The endpoint packet buffer is full.<br>**Note:** This bit is read-only. |

Endpoint A~L Interrupt Enable Control Register (USBD_EPAINTEN~ USBD_EPLINTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_EPAINTEN** | USBD_BA+0x06C | R/W | Endpoint A Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPBINTEN** | USBD_BA+0x094 | R/W | Endpoint B Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPCINTEN** | USBD_BA+0x0BC | R/W | Endpoint C Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPDINTEN** | USBD_BA+0x0E4 | R/W | Endpoint D Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPEINTEN** | USBD_BA+0x10C | R/W | Endpoint E Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPFINTEN** | USBD_BA+0x134 | R/W | Endpoint F Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPGINTEN** | USBD_BA+0x15C | R/W | Endpoint G Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPHINTEN** | USBD_BA+0x184 | R/W | Endpoint H Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPIINTEN** | USBD_BA+0x1AC | R/W | Endpoint I Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPJINTEN** | USBD_BA+0x1D4 | R/W | Endpoint J Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPKINTEN** | USBD_BA+0x1FC | R/W | Endpoint K Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPLINTEN** | USBD_BA+0x224 | R/W | Endpoint L 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** | | | **SHORTRXIEN** | **ERRIEN** | **NYETIEN** | **STALLIEN** | **NAKIEN** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **PINGIEN** | **INTKIEN** | **OUTTKIEN** | **RXPKIEN** | **TXPKIEN** | **SHORTTXIEN** | **BUFEMPTYIEN** | **BUFFULLIEN** |

| Bits | Description | |
|---|---|---|
| [31:13] | **Reserved** | Reserved. |
| [12] | **SHORTRXIEN** | **Bulk Out Short Packet Interrupt Enable Control**<br>When set, this bit enables a local interrupt to be set whenever bulk out short packet occurs on the bus for this endpoint.<br>0 = Bulk out interrupt Disabled.<br>1 = Bulk out interrupt Enabled. |

| | | |
|---|---|---|
| [11] | **ERRIEN** | **ERR Interrupt Enable Control**<br>When set, this bit enables a local interrupt to be set whenever ERR condition occurs on the bus for this endpoint.<br>0 = Error event interrupt Disabled.<br>1 = Error event interrupt Enabled. |
| [10] | **NYETIEN** | **NYET Interrupt Enable Control**<br>When set, this bit enables a local interrupt to be set whenever NYET condition occurs on the bus for this endpoint.<br>0 = NYET condition interrupt Disabled.<br>1 = NYET condition interrupt Enabled. |
| [9] | **STALLIEN** | **USB STALL Sent Interrupt Enable Control**<br>When set, this bit enables a local interrupt to be set when a stall token is sent to the host.<br>0 = STALL token interrupt Disabled.<br>1 = STALL token interrupt Enabled. |
| [8] | **NAKIEN** | **USB NAK Sent Interrupt Enable Control**<br>When set, this bit enables a local interrupt to be set when a NAK token is sent to the host.<br>0 = NAK token interrupt Disabled.<br>1 = NAK token interrupt Enabled. |
| [7] | **PINGIEN** | **PING Token Interrupt Enable Control**<br>When set, this bit enables a local interrupt to be set when a PING token has been received from the host.<br>0 = PING token interrupt Disabled.<br>1 = PING token interrupt Enabled. |
| [6] | **INTKIEN** | **Data IN Token Interrupt Enable Control**<br>When set, this bit enables a local interrupt to be set when a Data IN token has been received from the host.<br>0 = Data IN token interrupt Disabled.<br>1 = Data IN token interrupt Enabled. |
| [5] | **OUTTKIEN** | **Data OUT Token Interrupt Enable Control**<br>When set, this bit enables a local interrupt to be set when a Data OUT token has been received from the host.<br>0 = Data OUT token interrupt Disabled.<br>1 = Data OUT token interrupt Enabled. |
| [4] | **RXPKIEN** | **Data Packet Received Interrupt Enable Control**<br>When set, this bit enables a local interrupt to be set when a data packet has been transmitted to the host.<br>0 = Data packet has been transmitted to the host interrupt Disabled.<br>1 = Data packet has been transmitted to the host interrupt Enabled. |
| [3] | **TXPKIEN** | **Data Packet Transmitted Interrupt Enable Control**<br>When set, this bit enables a local interrupt to be set when a data packet has been received from the host.<br>0 = Data packet has been received from the host interrupt Disabled.<br>1 = Data packet has been received from the host interrupt Enabled. |

| | | |
|---|---|---|
| [2] | **SHORTTXIEN** | **Short Packet Transferred Interrupt Enable Control**<br>When set, this bit enables a local interrupt to be set when a short data packet has been transferred to/from the host.<br>0 = Short data packet interrupt Disabled.<br>1 = Short data packet interrupt Enabled. |
| [1] | **BUFEMPTYIEN** | **Buffer Empty Interrupt**<br>When set, this bit enables a local interrupt to be set when a buffer empty condition is detected on the bus.<br>0 = Buffer empty interrupt Disabled.<br>1 = Buffer empty interrupt Enabled. |
| [0] | **BUFFULLIEN** | **Buffer Full Interrupt**<br>When set, this bit enables a local interrupt to be set when a buffer full condition is detected on the bus.<br>0 = Buffer full interrupt Disabled.<br>1 = Buffer full interrupt Enabled. |

Endpoint A~L Data Available Count Register (USBD_EPADATCNT~ USBD_EPLDATCNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_EPADATCNT** | USBD_BA+0x070 | R | Endpoint A Data Available Count Register | 0x0000_0000 |
| **USBD_EPBDATCNT** | USBD_BA+0x098 | R | Endpoint B Data Available Count Register | 0x0000_0000 |
| **USBD_EPCDATCNT** | USBD_BA+0x0C0 | R | Endpoint C Data Available Count Register | 0x0000_0000 |
| **USBD_EPDDATCNT** | USBD_BA+0x0E8 | R | Endpoint D Data Available Count Register | 0x0000_0000 |
| **USBD_EPEDATCNT** | USBD_BA+0x110 | R | Endpoint E Data Available Count Register | 0x0000_0000 |
| **USBD_EPFDATCNT** | USBD_BA+0x138 | R | Endpoint F Data Available Count Register | 0x0000_0000 |
| **USBD_EPGDATCNT** | USBD_BA+0x160 | R | Endpoint G Data Available Count Register | 0x0000_0000 |
| **USBD_EPHDATCNT** | USBD_BA+0x188 | R | Endpoint H Data Available Count Register | 0x0000_0000 |
| **USBD_EPIDATCNT** | USBD_BA+0x1B0 | R | Endpoint I Data Available Count Register | 0x0000_0000 |
| **USBD_EPJDATCNT** | USBD_BA+0x1D8 | R | Endpoint J Data Available Count Register | 0x0000_0000 |
| **USBD_EPKDATCNT** | USBD_BA+0x200 | R | Endpoint K Data Available Count Register | 0x0000_0000 |
| **USBD_EPLDATCNT** | USBD_BA+0x228 | R | Endpoint L Data Available Count Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31] | **Reserved** | Reserved. |
| [30:16] | **DMALOOP** | **DMA Loop**<br>This register is the remaining DMA loop to complete. Each loop means 32-byte transfer. |

| | | |
|---|---|---|
| [15:0] | **DATCNT** | **Data Count**<br>For an IN endpoint (EPDIR(USBD_EPxCFG[3] is high.), this register returns the number of valid bytes in the IN endpoint packet buffer.<br>For an OUT endpoint (EPDIR(USBD_EPxCFG[3] is low.), this register returns the number of received valid bytes in the Host OUT transfer. |

Endpoint A~L Response Control Register (USBD_EPARSPCTL~ USBD_EPLRSPCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_EPARSPCTL** | USBD_BA+0x074 | R/W | Endpoint A Response Control Register | 0x0000_0000 |
| **USBD_EPBRSPCTL** | USBD_BA+0x09C | R/W | Endpoint B Response Control Register | 0x0000_0000 |
| **USBD_EPCRSPCTL** | USBD_BA+0x0C4 | R/W | Endpoint C Response Control Register | 0x0000_0000 |
| **USBD_EPDRSPCTL** | USBD_BA+0x0EC | R/W | Endpoint D Response Control Register | 0x0000_0000 |
| **USBD_EPERSPCTL** | USBD_BA+0x114 | R/W | Endpoint E Response Control Register | 0x0000_0000 |
| **USBD_EPFRSPCTL** | USBD_BA+0x13C | R/W | Endpoint F Response Control Register | 0x0000_0000 |
| **USBD_EPGRSPCTL** | USBD_BA+0x164 | R/W | Endpoint G Response Control Register | 0x0000_0000 |
| **USBD_EPHRSPCTL** | USBD_BA+0x18C | R/W | Endpoint H Response Control Register | 0x0000_0000 |
| **USBD_EPIRSPCTL** | USBD_BA+0x1B4 | R/W | Endpoint I Response Control Register | 0x0000_0000 |
| **USBD_EPJRSPCTL** | USBD_BA+0x1DC | R/W | Endpoint J Response Control Register | 0x0000_0000 |
| **USBD_EPKRSPCTL** | USBD_BA+0x204 | R/W | Endpoint K Response Control Register | 0x0000_0000 |
| **USBD_EPLRSPCTL** | USBD_BA+0x22C | R/W | Endpoint L Response 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** |
| **DISBUF** | **SHORTTXEN** | **ZEROLEN** | **HALT** | **TOGGLE** | **MODE** | | **FLUSH** |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [7] | **DISBUF** | **Buffer Disable Control**<br>This bit is used to receive unknown size OUT short packet. The received packet size is reference USBD_EPxDATCNT register.<br>0 = Buffer Not Disabled when Bulk-OUT short packet is received.<br>1 = Buffer Disabled when Bulk-OUT short packet is received. |
| [6] | **SHORTTXEN** | **Short Packet Transfer Enable**<br>This bit is applicable only in case of Auto-Validate Method. This bit is set to validate any remaining data in the buffer which is not equal to the MPS of the endpoint, and happens to be the last transfer. This bit gets cleared once the data packet is sent.<br>0 = Not validate any remaining data in the buffer which is not equal to the MPS of the endpoint.<br>1 = Validate any remaining data in the buffer which is not equal to the MPS of the endpoint. |
| [5] | **ZEROLEN** | **Zero Length**<br>This bit is used to send a zero-length packet response to an IN-token. When this bit is set, a zero packet is sent to the host on reception of an IN-token. This bit gets cleared once the zero length data packet is sent.<br>0 = A zero packet is not sent to the host on reception of an IN-token.<br>1 = A zero packet is sent to the host on reception of an IN-token. |
| [4] | **HALT** | **Endpoint Halt**<br>This bit is used to send a STALL handshake as response to the token from the host. When an Endpoint Set Feature (ep_halt) is detected by the local CPU, it must write a '1' to this bit.<br>0 = Not send a STALL handshake as response to the token from the host.<br>1 = Send a STALL handshake as response to the token from the host. |
| [3] | **TOGGLE** | **Endpoint Toggle**<br>This bit is used to clear the endpoint data toggle bit. Reading this bit returns the current state of the endpoint data toggle bit.<br>The local CPU may use this bit to initialize the end-point's toggle in case of reception of a Set Interface request or a Clear Feature (ep_halt) request from the host. Only when toggle bit is "1", this bit can be written into the inversed write data bit[3].<br>0 = Not clear the endpoint data toggle bit.<br>1 = Clear the endpoint data toggle bit. |
| [2:1] | **MODE** | **Mode Control**<br>The two bits decide the operation mode of the in-endpoint.<br>00: Auto-Validate Mode<br>01: Manual-Validate Mode<br>10: Fly Mode<br>11: Reserved<br>These bits are not valid for an out-endpoint. The auto validate mode will be activated when the reserved mode is selected. |
| [0] | **FLUSH** | **Buffer Flush**<br>Writing 1 to this bit causes the packet buffer to be flushed and the corresponding EP_AVAIL register to be cleared. This bit is self-clearing. This bit should always be written after an configuration event.<br>0 = The packet buffer is not flushed.<br>1 = The packet buffer is flushed by user. |

Endpoint A~L Maximum Packet Size Register (USBD_EPAMPS~ USBD_EPLMPS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_EPAMPS** | USBD_BA+0x078 | R/W | Endpoint A Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPBMPS** | USBD_BA+0x0A0 | R/W | Endpoint B Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPCMPS** | USBD_BA+0x0C8 | R/W | Endpoint C Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPDMPS** | USBD_BA+0x0F0 | R/W | Endpoint D Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPEMPS** | USBD_BA+0x118 | R/W | Endpoint E Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPFMPS** | USBD_BA+0x140 | R/W | Endpoint F Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPGMPS** | USBD_BA+0x168 | R/W | Endpoint G Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPHMPS** | USBD_BA+0x190 | R/W | Endpoint H Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPIMPS** | USBD_BA+0x1B8 | R/W | Endpoint I Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPJMPS** | USBD_BA+0x1E0 | R/W | Endpoint J Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPKMPS** | USBD_BA+0x208 | R/W | Endpoint K Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPLMPS** | USBD_BA+0x230 | R/W | Endpoint L Maximum Packet Size 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** | | | | | **EPMPS** | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **EPMPS** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:11] | **Reserved** | Reserved. |
| [10:0] | **EPMPS** | **Endpoint Maximum Packet Size**<br>This field determines the Maximum Packet Size of the Endpoint. |

Endpoint A~L Transfer Count Register (USBD_EPATXCNT~ USBD_EPLTXCNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_EPATXCNT** | USBD_BA+0x07C | R/W | Endpoint A Transfer Count Register | 0x0000_0000 |
| **USBD_EPBTXCNT** | USBD_BA+0x0A4 | R/W | Endpoint B Transfer Count Register | 0x0000_0000 |
| **USBD_EPCTXCNT** | USBD_BA+0x0CC | R/W | Endpoint C Transfer Count Register | 0x0000_0000 |
| **USBD_EPDTXCNT** | USBD_BA+0x0F4 | R/W | Endpoint D Transfer Count Register | 0x0000_0000 |
| **USBD_EPETXCNT** | USBD_BA+0x11C | R/W | Endpoint E Transfer Count Register | 0x0000_0000 |
| **USBD_EPFTXCNT** | USBD_BA+0x144 | R/W | Endpoint F Transfer Count Register | 0x0000_0000 |
| **USBD_EPGTXCNT** | USBD_BA+0x16C | R/W | Endpoint G Transfer Count Register | 0x0000_0000 |
| **USBD_EPHTXCNT** | USBD_BA+0x194 | R/W | Endpoint H Transfer Count Register | 0x0000_0000 |
| **USBD_EPITXCNT** | USBD_BA+0x1BC | R/W | Endpoint I Transfer Count Register | 0x0000_0000 |
| **USBD_EPJTXCNT** | USBD_BA+0x1E4 | R/W | Endpoint J Transfer Count Register | 0x0000_0000 |
| **USBD_EPKTXCNT** | USBD_BA+0x20C | R/W | Endpoint K Transfer Count Register | 0x0000_0000 |
| **USBD_EPLTXCNT** | USBD_BA+0x234 | R/W | Endpoint L Transfer Count 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** | | | | | **TXCNT** | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **TXCNT** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:11] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [10:0] | **TXCNT** | **Endpoint Transfer Count**<br>For IN endpoints, this field determines the total number of bytes to be sent to the host in case of manual validation method.<br>For OUT endpoints, this field has no effect. |

Endpoint A~L Configuration Register (USBD_EPACFG~ USBD_EPLCFG)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_EPACFG** | USBD_BA+0x080 | R/W | Endpoint A Configuration Register | 0x0000_0012 |
| **USBD_EPBCFG** | USBD_BA+0x0A8 | R/W | Endpoint B Configuration Register | 0x0000_0022 |
| **USBD_EPCCFG** | USBD_BA+0x0D0 | R/W | Endpoint C Configuration Register | 0x0000_0032 |
| **USBD_EPDCFG** | USBD_BA+0x0F8 | R/W | Endpoint D Configuration Register | 0x0000_0042 |
| **USBD_EPECFG** | USBD_BA+0x120 | R/W | Endpoint E Configuration Register | 0x0000_0052 |
| **USBD_EPFCFG** | USBD_BA+0x148 | R/W | Endpoint F Configuration Register | 0x0000_0062 |
| **USBD_EPGCFG** | USBD_BA+0x170 | R/W | Endpoint G Configuration Register | 0x0000_0072 |
| **USBD_EPHCFG** | USBD_BA+0x198 | R/W | Endpoint H Configuration Register | 0x0000_0082 |
| **USBD_EPICFG** | USBD_BA+0x1C0 | R/W | Endpoint I Configuration Register | 0x0000_0092 |
| **USBD_EPJCFG** | USBD_BA+0x1E8 | R/W | Endpoint J Configuration Register | 0x0000_00A2 |
| **USBD_EPKCFG** | USBD_BA+0x210 | R/W | Endpoint K Configuration Register | 0x0000_00B2 |
| **USBD_EPLCFG** | USBD_BA+0x238 | R/W | Endpoint L Configuration Register | 0x0000_00C2 |

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

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [7:4] | **EPNUM** | **Endpoint Number**<br>This field selects the number of the endpoint. Valid numbers 1 to 15.<br>**Note:** Do not support two endpoints have same endpoint number. |
| [3] | **EPDIR** | **Endpoint Direction**<br>0 = out-endpoint (Host OUT to Device).<br>1 = in-endpoint (Host IN to Device).<br>**Note:** A maximum of one OUT and IN endpoint is allowed for each endpoint number. |
| [2:1] | **EPTYPE** | **Endpoint Type**<br>This field selects the type of this endpoint. Endpoint 0 is forced to a Control type.<br>00 = Reserved.<br>01 = Bulk.<br>10 = Interrupt.<br>11 = Isochronous. |
| [0] | **EPEN** | **Endpoint Valid**<br>When set, this bit enables this endpoint. This bit has no effect on Endpoint 0, which is always enabled.<br>0 = The endpoint Disabled.<br>1 = The endpoint Enabled. |

Endpoint A~L RAM Start Address Register (USBD_EPABUFSTART~ USBD_EPLBUFSTART)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_EPABUFSTART** | USBD_BA+0x084 | R/W | Endpoint A RAM Start Address Register | 0x0000_0000 |
| **USBD_EPBBUFSTART** | USBD_BA+0x0AC | R/W | Endpoint B RAM Start Address Register | 0x0000_0000 |
| **USBD_EPCBUFSTART** | USBD_BA+0x0D4 | R/W | Endpoint C RAM Start Address Register | 0x0000_0000 |
| **USBD_EPDBUFSTART** | USBD_BA+0x0FC | R/W | Endpoint D RAM Start Address Register | 0x0000_0000 |
| **USBD_EPEBUFSTART** | USBD_BA+0x124 | R/W | Endpoint E RAM Start Address Register | 0x0000_0000 |
| **USBD_EPFBUFSTART** | USBD_BA+0x14C | R/W | Endpoint F RAM Start Address Register | 0x0000_0000 |
| **USBD_EPGBUFSTART** | USBD_BA+0x174 | R/W | Endpoint G RAM Start Address Register | 0x0000_0000 |
| **USBD_EPHBUFSTART** | USBD_BA+0x19C | R/W | Endpoint H RAM Start Address Register | 0x0000_0000 |
| **USBD_EPIBUFSTART** | USBD_BA+0x1C4 | R/W | Endpoint I RAM Start Address Register | 0x0000_0000 |
| **USBD_EPJBUFSTART** | USBD_BA+0x1EC | R/W | Endpoint J RAM Start Address Register | 0x0000_0000 |
| **USBD_EPKBUFSTART** | USBD_BA+0x214 | R/W | Endpoint K RAM Start Address Register | 0x0000_0000 |
| **USBD_EPLBUFSTART** | USBD_BA+0x23C | R/W | Endpoint L RAM Start 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** | | | | **SADDR** | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **SADDR** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:12] | **Reserved** | Reserved. |
| [11:0] | **SADDR** | **Endpoint Start Address**<br>This is the start-address of the RAM space allocated for the endpoint A~L. |

Endpoint A~L RAM End Address Register (USBD_EPABUFEND~ USBD_EPLBUFEND)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_EPABUFEND** | USBD_BA+0x088 | R/W | Endpoint A RAM End Address Register | 0x0000_0000 |
| **USBD_EPBBUFEND** | USBD_BA+0x0B0 | R/W | Endpoint B RAM End Address Register | 0x0000_0000 |
| **USBD_EPCBUFEND** | USBD_BA+0x0D8 | R/W | Endpoint C RAM End Address Register | 0x0000_0000 |
| **USBD_EPDBUFEND** | USBD_BA+0x100 | R/W | Endpoint D RAM End Address Register | 0x0000_0000 |
| **USBD_EPEBUFEND** | USBD_BA+0x128 | R/W | Endpoint E RAM End Address Register | 0x0000_0000 |
| **USBD_EPFBUFEND** | USBD_BA+0x150 | R/W | Endpoint F RAM End Address Register | 0x0000_0000 |
| **USBD_EPGBUFEND** | USBD_BA+0x178 | R/W | Endpoint G RAM End Address Register | 0x0000_0000 |
| **USBD_EPHBUFEND** | USBD_BA+0x1A0 | R/W | Endpoint H RAM End Address Register | 0x0000_0000 |
| **USBD_EPIBUFEND** | USBD_BA+0x1C8 | R/W | Endpoint I RAM End Address Register | 0x0000_0000 |
| **USBD_EPJBUFEND** | USBD_BA+0x1F0 | R/W | Endpoint J RAM End Address Register | 0x0000_0000 |
| **USBD_EPKBUFEND** | USBD_BA+0x218 | R/W | Endpoint K RAM End Address Register | 0x0000_0000 |
| **USBD_EPLBUFEND** | USBD_BA+0x240 | R/W | Endpoint L RAM End 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** | | | | **EADDR** | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **EADDR** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:12] | **Reserved** | Reserved. |
| [11:0] | **EADDR** | **Endpoint End Address**<br>This is the end-address of the RAM space allocated for the endpoint A~L. |

AHB Address Register (USBD_DMAADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_DMAADDR** | USBD_BA+0x700 | R/W | AHB DMA Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **DMAADDR** | **DMAADDR**<br>The register specifies the address from which the DMA has to read / write. The address must WORD (32-bit) aligned. |

## USB PHY Control Register (USBD_PHYCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_PHYCTL** | USBD_BA+0x704 | R/W | USB PHY Control Register | 0x0000_0420 |

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

| Bits | Description | |
|---|---|---|
| [31] | **VBUSDET** | **VBUS Status**<br>0 = The VBUS is not detected yet.<br>1 = The VBUS is detected. |
| [30:25] | **Reserved** | Reserved. |
| [24] | **WKEN** | **Wake-up Enable Control**<br>0 = The wake-up function Disabled.<br>1 = The wake-up function Enabled. |
| [23:10] | **Reserved** | Reserved. |
| [9] | **PHYEN** | **PHY Suspend Enable Control**<br>0 = The USB PHY is suspend.<br>1 = The USB PHY is not suspend. |
| [8] | **DPPUEN** | **DP Pull-up**<br>0 = Pull-up resistor on D+ Disabled.<br>1 = Pull-up resistor on D+ Enabled. |
| [7:0] | **Reserved** | Reserved. |

## 5.23 USB Host Controller (USBH)

### 5.23.1 Overview

The Universal Serial Bus (USB) is a fast, bi-directional, isochronous, low-cost, dynamically attachable serial interface standard intended for modem, scanners, PDAs, keyboards, mice, and digital imaging devices. The USB is a 4-wire serial cable bus that supports serial data exchange between a Host Controller and a network of peripheral devices. The attached peripherals share USB bandwidth through a host-scheduled, token-based protocol. Peripherals may be attached, configured, used, and detached, while the host and other peripherals continue operation (i.e. hot plug and unplug is supported).

### 5.23.2 Features

- Fully compliant with USB Revision 2.0 specification.
- Enhanced Host Controller Interface (EHCI) Revision 1.0 compatible.
- Open Host Controller Interface (OHCI) Revision 1.0 compatible.
- Supports high-speed (480Mbps), 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.
- Built-in DMA for real-time data transfer.

### 5.23.3 Block Diagram

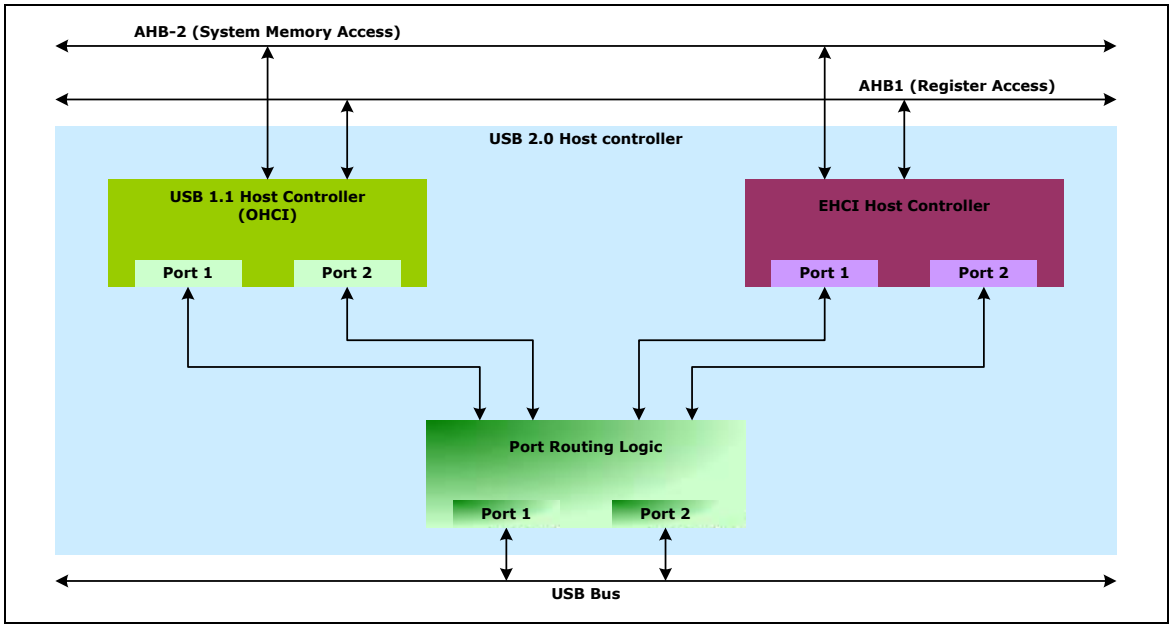



Figure 5.23-1 USB Host Controller Block Diagram

### 5.23.4 Basic Configuration

USB host clock source is derived from PLL and USB PHY. User has to set the PLL related configurations before USB host enabled. Set the USBH (CLK_HCLKEN[18]) bit to enable USB host clock and 4-bit pre-scale USB_N (CLK_DIVCTL2[11:8]) to generate proper 48 MHz clock to USB host.

In addition, USB host needs a clock from USB PHY for USB 2.0 high speed operation. User has to set SUSPEND (USBPCR0[8] and USBPCR1[8]) high to enable USB PHY. Then, user has to check if CLKVALID (USBPCR0[11]) is high before starting to use USB host controller.

### 5.23.5 Functional Description

#### 5.23.5.1 *EHCI Controller*

The EHCI is interfaced with the system through AHB interface. Whenever the CPU wants to initiate a register read or register write, it uses the AHB slave I/F signals and performs the necessary operation (register read writes). The CPU acts as a bus master, having initiated this transfer. At that time, EHCI acts as a target and responds to the transfer initiated by the system software. For example, if the CPU wants to write into one of the memory mapped registers of EHCI, it says the address and value to be written into that addressed register. EHCI targets the register by using that address and fills the register with the value specified by the software. If it is a register read, EHCI gets the value from the addressed register and puts it on the system bus.

Likewise, when the EHCI wants to perform a data transfer, it acts as a master and initiates a data transfer. At that time, the system memory acts as a bus target. EHCI, as a master can perform two types of data transfers, from EHCI to the system memory and from system memory to the EHCI. When the EHCI wants the data to be moved from the downstream USB2.0 device to the system memory, it initiates a memory write transfer by accessing the memory interfacing signals. EHCI writes the control word (write), data and data count to be moved to the system memory. The memory controller accepts the data and moves it to the memory. If the data has to be moved from memory to the downstream device, the EHCI issue a read transfer to system bus. The memory controller gives

data through the memory interfacing signals. EHCI accepts the data and moves them to the downstream device.

#### 5.23.5.2 *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.

- AHB Master

The master issues the address and data onto the bus when granted.

- AHB Slave

The configuration of the Host Controller is through the slave interface.

- List Processing

The List Processor manages the data structures from the Host Controller Driver and coordinates all activity within the Host Controller.

- Frame Management

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.

- Interrupt Processing

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.

- Host Controller Bus Master

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.

- Data Buffer

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 two external ports, Port 1 and Port 2 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.

- SIE

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.

- Root Hub

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.

### 5.23.6 Register Map

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EHCI Base Address:**<br>**EHCI_BA = 0xB000_5000**<br>**OHCI_BA = 0xB000_7000** | | | | |
| **EHCVNR** | EHCI_BA+0x000 | R | EHCI Version Number Register | 0x0095_0020 |
| **EHCSPR** | EHCI_BA+0x004 | R | EHCI Structural Parameters Register | 0x0000_0012 |
| **EHCCPR** | EHCI_BA+0x008 | R | EHCI Capability Parameters Register | 0x0000_0000 |
| **UCMDR** | EHCI_BA+0x020 | R/W | USB Command Register | 0x0008_0000 |
| **USTSR** | EHCI_BA+0x024 | R/W | USB Status Register | 0x0000_1000 |
| **UIENR** | EHCI_BA+0x028 | R/W | USB Interrupt Enable Register | 0x0000_0000 |
| **UFINDR** | EHCI_BA+0x02C | R/W | USB Frame Index Register | 0x0000_0000 |
| **UPFLBAR** | EHCI_BA+0x034 | R/W | USB Periodic Frame List Base Address Register | 0x0000_0000 |
| **UCALAR** | EHCI_BA+0x038 | R/W | USB Current Asynchronous List Address Register | 0x0000_0000 |
| **UASSTR** | EHCI_BA+0x03C | R/W | USB Asynchronous Schedule Sleep Timer Register | 0x0000_0BD6 |
| **UCFGR** | EHCI_BA+0x060 | R/W | USB Configure Flag Register | 0x0000_0000 |
| **UPSCR0** | EHCI_BA+0x064 | R/W | USB Port 0 Status and Control Register | 0x0000_2000 |
| **UPSCR1** | EHCI_BA+0x068 | R/W | USB Port 1 Status and Control Register | 0x0000_2000 |
| **USBPCR0** | EHCI_BA+0x0C4 | R/W | USB PHY 0 Control Register | 0x0000_0060 |
| **USBPCR1** | EHCI_BA+0x0C8 | R/W | USB PHY 1 Control Register | 0x0000_0020 |
| **HcRev** | OHCI_BA+0x000 | R | Host Controller Revision Register | 0x0000_0010 |
| **HcControl** | OHCI_BA+0x004 | R/W | Host Controller Control Register | 0x0000_0000 |
| **HcComSts** | OHCI_BA+0x008 | R/W | Host Controller Command Status Register | 0x0000_0000 |
| **HcIntSts** | OHCI_BA+0x00C | R/W | Host Controller Interrupt Status Register | 0x0000_0000 |
| **HcIntEn** | OHCI_BA+0x010 | R/W | Host Controller Interrupt Enable Register | 0x0000_0000 |
| **HcIntDis** | OHCI_BA+0x014 | R/W | Host Controller Interrupt Disable Register | 0x0000_0000 |
| **HcHCCA** | OHCI_BA+0x018 | R/W | Host Controller Communication Area Register | 0x0000_0000 |
| **HcPerCED** | OHCI_BA+0x01C | R/W | Host Controller Period Current ED Register | 0x0000_0000 |
| **HcCtrHED** | OHCI_BA+0x020 | R/W | Host Controller Control Head ED Register | 0x0000_0000 |
| **HcCtrCED** | OHCI_BA+0x024 | R/W | Host Controller Control Current ED Register | 0x0000_0000 |
| **HcBlkHED** | OHCI_BA+0x028 | R/W | Host Controller Bulk Head ED Register | 0x0000_0000 |
| **HcBlkCED** | OHCI_BA+0x02C | R/W | Host Controller Bulk Current ED Register | 0x0000_0000 |
| **HcDoneH** | OHCI_BA+0x030 | R/W | Host Controller Done Head Register | 0x0000_0000 |
| **HcFmIntv** | OHCI_BA+0x034 | R/W | Host Controller Frame Interval Register | 0x0000_2EDF |

| | | | | |
|---|---|---|---|---|
| **HcFmRem** | OHCI_BA+0x038 | R | Host Controller Frame Remaining Register | 0x0000_0000 |
| **HcFNum** | OHCI_BA+0x03C | R | Host Controller Frame Number Register | 0x0000_0000 |
| **HcPerSt** | OHCI_BA+0x040 | R/W | Host Controller Periodic Start Register | 0x0000_0000 |
| **HcLSTH** | OHCI_BA+0x044 | R/W | Host Controller Low Speed Threshold Register | 0x0000_0628 |
| **HcRhDeA** | OHCI_BA+0x048 | R/W | Host Controller Root Hub Descriptor A Register | 0x0100_0002 |
| **HcRhDeB** | OHCI_BA+0x04C | R/W | Host Controller Root Hub Descriptor B Register | 0x0000_0000 |
| **HcRhSts** | OHCI_BA+0x050 | R/W | Host Controller Root Hub Status Register | 0x0000_0000 |
| **HcRhPrt1** | OHCI_BA+0x054 | R/W | Host Controller Root Hub Port Status [1] | 0x0000_0000 |
| **HcRhPrt2** | OHCI_BA+0x058 | R/W | Host Controller Root Hub Port Status [2] | 0x0000_0000 |
| **OpModEn** | OHCI_BA+0x204 | R/W | USB Operational Mode Enable Register | 0X0000_0000 |

### 5.23.7 Register Description

### EHCI Version Number Register (EHCVNR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EHCVNR** | EHCI_BA+0x000 | R | EHCI Version Number Register | 0x0095_0020 |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **VERSION** | **Host Controller Interface Version Number**<br>This is a two-byte register containing a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision. |
| [15:8] | **Reserved** | Reserved. |
| [7:0] | **CRLEN** | **Capability Registers Length**<br>This register is used as an offset to add to register base to find the beginning of the Operational Register Space. |

## EHCI Structural Parameters Register (EHCSPR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EHCSPR** | EHCI_BA+0x004 | R | EHCI Structural Parameters Register | 0x0000_0012 |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:12] | **N_CC** | **Number of Companion Controller**<br>This field indicates the number of companion controllers associated with this USB 2.0 host controller.<br>A zero in this field indicates there are no companion host controllers. Port-ownership hand-off is not supported. Only high-speed devices are supported on the host controller root ports.<br>A value larger than zero in this field indicates there are companion USB 1.1 host controller(s). Port-ownership hand-offs are supported. High, Full- and Low-speed devices are supported on the host controller root ports. |
| [11:8] | **N_PCC** | **Number of Ports Per Companion Controller**<br>This field indicates the number of ports supported per companion host controller. It is used to indicate the port routing configuration to system software.<br>For example, if N_PORTS has a value of 6 and N_CC has a value of 2 then N_PCC could have a value of 3. The convention is that the first N_PCC ports are assumed to be routed to companion controller 1, the next N_PCC ports to companion controller 2, etc. In the previous example, the N_PCC could have been 4, where the first 4 are routed to companion controller 1 and the last two are routed to companion controller 2.<br>The number in this field must be consistent with N_PORTS and N_CC. |
| [7:5] | **Reserved** | Reserved. |
| [4] | **PPC** | **Port Power Control**<br>This field indicates whether the host controller implementation includes port power control. A one in this bit indicates the ports have port power switches. A zero in this bit indicates the port do not have port power stitches. The value of this field affects the functionality of the Port Power field in each port status and control register. |
| [3:0] | **N_PORTS** | **Number of Physical Downstream Ports**<br>This field specifies the number of physical downstream ports implemented on this host controller. The value of this field determines how many port registers are addressable in the Operational Register Space (see Table 2-8). Valid values are in the range of 1H to FH.<br>A zero in this field is undefined. |

### EHCI Capability Parameters Register (EHCCPR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **EHCCPR** | EHCI_BA+0x008 | R | EHCI Capability Parameters 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** |
| **EECP** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **IST** | | | | **Reserved** | **ASPC** | **PFLF** | **AC64** |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:8] | **EECP** | **EHCI Extended Capabilities Pointer (EECP)**<br>0 = No extended capabilities are implemented. |
| [7:4] | **IST** | **Isochronous Scheduling Threshold**<br>This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule.<br>When bit [7] is zero, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state. |
| [3] | **Reserved** | Reserved. |
| [2] | **ASPC** | **Asynchronous Schedule Park Capability**<br>0 = This EHCI host controller doesn't support park feature of high-speed queue heads in the Asynchronous Schedule. |
| [1] | **PFLF** | **Programmable Frame List Flag**<br>0 = System software must use a frame list length of 1024 elements with this EHCI host controller. |
| [0] | **AC64** | **64-bit Addressing Capability**<br>0 = Data structure using 32-bit address memory pointers. |

## USB Command Register (UCMDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UCMDR** | EHCI_BA+0x020 | R/W | USB Command Register | 0x0008_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **ITC** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | **IAAD** | **ASEN** | **PSEN** | **FLSZ** | | **HCRST** | **RUN** |

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:16] | **ITC** | **Interrupt Threshold Control (R/W)**<br>This field is used by system software to select the maximum rate at which the host controller will issue interrupts. The only valid values are defined below. If software writes an invalid value to this register, the results are undefined. Value Maximum Interrupt Interval<br>0x00 = Reserved.<br>0x01 = 1 micro-frame.<br>0x02 = 2 micro-frames.<br>0x04 = 4 micro-frames.<br>0x08 = 8 micro-frames (default, equates to 1 ms).<br>0x10 = 16 micro-frames (2 ms).<br>0x20 = 32 micro-frames (4 ms).<br>0x40 = 64 micro-frames (8 ms).<br>Any other value in this register yields undefined results.<br>Software modifications to this bit while HCHalted bit is equal to zero results in undefined behavior. |
| [15:7] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [6] | **IAAD** | **Interrupt on Async Advance Doorbell (R/W)**<br>This bit is used as a doorbell by software to tell the host controller to issue an interrupt the next time it advances asynchronous schedule. Software must write a 1 to this bit to ring the doorbell.<br>When the host controller has evicted all appropriate cached schedule state, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Async Advance Enable bit in the USBINTR register is a one then the host controller will assert an interrupt at the next interrupt threshold.<br>The host controller sets this bit to a zero after it has set the Interrupt on Async Advance status bit in the USBSTS register to a one.<br>Software should not write a one to this bit when the asynchronous schedule is disabled. Doing so will yield undefined results. |
| [5] | **ASEN** | **Asynchronous Schedule Enable (R/W)**<br>This bit controls whether the host controller skips processing the Asynchronous Schedule. Values mean:<br>0 = Do not process the Asynchronous Schedule.<br>1 = Use the ASYNCLISTADDR register to access the Asynchro-nous Schedule. |
| [4] | **PSEN** | **Periodic Schedule Enable (R/W)**<br>This bit controls whether the host controller skips processing the Periodic Schedule. Values mean:<br>0 = Do not process the Periodic Schedule.<br>1 = Use the PERIODICLISTBASE register to access the Periodic Schedule. |
| [3:2] | **FLSZ** | **Frame List Size (R/W or RO)**<br>This field is R/W only if Programmable Frame List Flag in the HCCPARAMS registers is set to a one. This field specifies the size of the frame list. The size the frame list controls which bits in the Frame Index Register should be used for the Frame List Current index. Values mean:<br>00 = 1024 elements (4096 bytes) Default value.<br>01 = 512 elements (2048 bytes).<br>10 = 256 elements (1024 bytes) – for resource-constrained environment.<br>11 = Reserved. |
| [1] | **HCRST** | **Host Controller Reset (HCRESET) (R/W)**<br>This control bit is used by software to reset the host controller. The effects of this on Root Hub registers are similar to a Chip Hardware Reset.<br>When software writes a one to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines, etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports.<br>All operational registers, including port registers and port state machines are set to their initial values. Port ownership reverts to the companion host controller(s), with the side effects. Software must reinitialize the host controller in order to return the host controller to an operational state.<br>This bit is set to zero by the Host Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register.<br>Software should not set this bit to a one when the HCHalted bit in the USBSTS register is a zero. Attempting to reset an actively running host controller will result in undefined behavior. |

| | | |
|---|---|---|
| [0] | **RUN** | **Run/Stop (R/W)**<br>When set to a 1, the Host Controller proceeds with execution of the schedule. The Host Controller continues execution as long as this bit is set to a 1. When this bit is set to 0, the Host Controller completes the current and any actively pipelined transactions on the USB and then halts. The Host Controller must halt within 16 micro-frames after software clears the Run bit. The HC Halted bit in the status register indicates when the Host Controller has finished its pending pipelined transactions and has entered the stopped state. Software must not write a one to this field unless the host controller is in the Halted state (i.e. HCHalted in the USBSTS register is a one). Doing so will yield undefined results.<br>0 = Stop.<br>1 = Run. |

## USB Status Register (USTSR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USTSR** | EHCI_BA+0x024 | R/W | USB Status 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** |
| **ASS** | **PSS** | **RECLA** | **HCHalted** | **Reserved** | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | **IAA** | **HSERR** | **FLR** | **PCD** | **UERRINT** | **USBINT** |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15] | **ASS** | **Asynchronous Schedule Status (RO)**<br>The bit reports the current real status of the Asynchronous Schedule. If this bit is a zero then the status of them Asynchronous Schedule is disabled. If this bit is a one then the status of the Asynchronous Schedule is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. When this bit and the Asynchronous Schedule Enable bit are the same value, the Asynchronous Schedule is either enabled (1) or disabled (0). |
| [14] | **PSS** | **Periodic Schedule Status (RO)**<br>The bit reports the current real status of the Periodic Schedule. If this bit is a zero then the status of the Periodic Schedule is disabled. If this bit is a one then the status of the Periodic Schedule is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. When this bit and the Periodic Schedule Enable bit are the same value, the Periodic Schedule is either enabled (1) or disabled (0). |
| [13] | **RECLA** | **Reclamation (RO)**<br>This is a read-only status bit, which is used to detect an empty asynchronous schedule. |
| [12] | **HCHalted** | **HCHalted (RO)**<br>This bit is a zero whenever the Run/Stop bit is a one. The Host Controller sets this bit to one after it has stopped executing as a result of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g. internal error). |
| [11:6] | **Reserved** | Reserved. |
| [5] | **IAA** | **Interrupt on Async Advance (R/WC)**<br>System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the USBCMD register. This status bit indicates the assertion of that interrupt source. |

| | | |
|---|---|---|
| [4] | **HSERR** | **Host System Error (R/WC)**<br>The Host Controller sets this bit to 1 when a serious error occurs during a host system access involving the Host Controller module. |
| [3] | **FLR** | **Frame List Rollover (R/WC)**<br>The Host Controller sets this bit to a one when the Frame List Index rolls over from its maximum value to zero. The exact value at which the rollover occurs depends on the frame list size. For example, if the frame list size (as programmed in the Frame List Size field of the USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX[13] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a one every time FRINDEX[12] toggles. |
| [2] | **PCD** | **Port Change Detect (R/WC)**<br>The Host Controller sets this bit to a one when any port for which the Port Owner bit is set to zero has a change bit transition from a zero to a one or a Force Port Resume bit transition from a zero to a one as a result of a J-K transition detected on a suspended port. This bit will also be set as a result of the Connect Status Change being set to a one after system software has relinquished ownership of a connected port by writing a one to a port's Port Owner bit.<br>This bit is allowed to be maintained in the Auxiliary power well. Alternatively, it is also acceptable that on a D3 to D0 transition of the EHCI HC device, this bit is loaded with the OR of all of the PORTSC change bits (including: Force port resume, over-current change, enable/disable change and connect status change). |
| [1] | **UERRINT** | **USB Error Interrupt (USBERRINT) (R/WC)**<br>The Host Controller sets this bit to 1 when completion of a USB transaction results in an error condition (e.g., error counter underflow). If the TD on which the error interrupt occurred also had its IOC bit set, both this bit and USBINT bit are set. |
| [0] | **USBINT** | **USB Interrupt (USBINT) (R/WC)**<br>The Host Controller sets this bit to 1 on the completion of a USB transaction, which results in the retirement of a Transfer Descriptor that had its IOC bit set.<br>The Host Controller also sets this bit to 1 when a short packet is detected (actual number of bytes received was less than the expected number of bytes). |

## USB Interrupt Enable Register (UIENR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UIENR** | EHCI_BA+0x028 | R/W | USB 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** | | **IAAEN** | **HSERREN** | **FLREN** | **PCIEN** | **UERRIEN** | **USBIEN** |

| Bits | Description | |
|---|---|---|
| [31:6] | **Reserved** | Reserved. |
| [5] | **IAAEN** | **Interrupt on Async Advance Enable**<br>When this bit is a one, and the Interrupt on Async Advance bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the Interrupt on Async Advance bit. |
| [4] | **HSERREN** | **Host System Error Enable**<br>When this bit is a one, and the Host System Error Status bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Host System Error bit. |
| [3] | **FLREN** | **Frame List Rollover Enable**<br>When this bit is a one, and the Frame List Rollover bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Frame List Rollover bit. |
| [2] | **PCIEN** | **Port Change Interrupt Enable**<br>When this bit is a one, and the Port Change Detect bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Port Change Detect bit. |
| [1] | **UERRIEN** | **USB Error Interrupt Enable**<br>When this bit is a one, and the USBERRINT bit in the USBSTS register is a one, the host t controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBERRINT bit. |
| [0] | **USBIEN** | **USB Interrupt Enable**<br>When this bit is a one, and the USBINT bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBINT bit. |

## USB Frame Index Register (UFINDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UFINDR** | EHCI_BA+0x02C | R/W | USB Frame Index 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** | | **FI** | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **FI** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:14] | **Reserved** | Reserved. |
| [13:0] | **FI** | **Frame Index**<br>The value in this register increment at the end of each time frame (e.g. micro-frame). Bits [N:3] are used for the Frame List current index. This means that each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register.<br>**FLSZ (UCMDR[3:2] Number Elements N**<br>0x0 1024 12<br>0x1 512 11<br>0x2 256 10<br>0x3 Reserved |

## USB Periodic Frame List Base Address Register (UPFLBAR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UPFLBAR** | EHCI_BA+0x034 | R/W | USB Periodic Frame List Base Address Register | 0x0000_0000 |

| 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** | | | | **Reserved** | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:12] | **BADDR** | **Base Address**<br>These bits correspond to memory address signals [31:12], respectively. |
| [11:0] | **Reserved** | Reserved. |

## USB Current Asynchronous List Address Register (UCALAR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UCALAR** | EHCI_BA+0x038 | R/W | USB Current Asynchronous List Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:5] | **LPL** | **Link Pointer Low (LPL)**<br>These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (QH). |
| [4:0] | **Reserved** | Reserved. |

## USB Asynchronous Schedule Sleep Timer Register (UASSTR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UASSTR** | EHCI_BA+0x03C | R/W | USB Asynchronous Schedule Sleep Timer Register | 0x0000_0BD6 |

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

| Bits | Description | |
|---|---|---|
| [31:11] | **Reserved** | Reserved. |
| [11:0] | **ASSTMR** | **Asynchronous Schedule Sleep Timer**<br>This field defines the AsyncSchedSleepTime of EHCI spec.<br>The asynchronous schedule sleep timer is used to control how often the host controller fetches asynchronous schedule list from system memory while the asynchronous schedule is empty.<br>The default value of this timer is 12'hBD6. Because this timer is implemented in UTMI clock (30MHz) domain, the default sleeping time will be about 100us. |

## USB Configure Flag Register (UCFGR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UCFGR** | EHCI_BA+0x060 | R/W | USB Configure 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** | | | | | | | **CF** |

| Bits | Description | |
|---|---|---|
| [31:1] | **Reserved** | Reserved. |
| [0] | **CF** | **Configure Flag (CF)**<br>Host software sets this bit as the last action in its process of configuring the Host Controller. This bit controls the default port-routing control logic. Bit values and side-effects are listed below.<br>0 = Port routing control logic default-routes each port to an implementation dependent classic host controller.<br>1 = Port routing control logic default-routes all ports to this host controller. |

## USB Port Status and Control Register (UPSCR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UPSCR0** | EHCI_BA+0x064 | R/W | USB Port 0 Status and Control Register | 0x0000_2000 |
| **UPSCR1** | EHCI_BA+0x068 | R/W | USB Port 1 Status and Control Register | 0x0000_2000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | **PTC** | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | **PO** | **PP** | **LSTS** | | **Reserved** | **PRST** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **SUSPEND** | **FPR** | **OCC** | **OCA** | **PEC** | **PE** | **CSC** | **CCS** |

| Bits | Description | |
|---|---|---|
| [31:20] | **Reserved** | Reserved. |
| [19:16] | **PTC** | **Port Test Control (R/W)**<br>When this field is zero, the port is NOT operating in a test mode. A non-zero value indicates that it is operating in test mode and the specific test mode is indicated by the specific value. The encoding of the test mode bits are (0x6 ~ 0xF are reserved):<br>Bits Test Mode<br>0x0 = Test mode not enabled.<br>0x1 = Test J_STATE.<br>0x2 = Test K_STATE.<br>0x3 = Test SE0_NAK.<br>0x4 = Test Packet.<br>0x5 = Test FORCE_ENABLE. |
| [15:14] | **Reserved** | Reserved. |
| [13] | **PO** | **Port Owner (R/W)**<br>This bit unconditionally goes to a 0b when the Configured bit in the CONFIGFLAG register makes a 0 to 1 transition. This bit unconditionally goes to 1 whenever the Configured bit is zero.<br>System software uses this field to release ownership of the port to a selected host controller (in the event that the attached device is not a high-speed device). Software writes a one to this bit when the attached device is not a high-speed device. A one in this bit means that a companion host controller owns and controls the port. |
| [12] | **PP** | **Port Power (PP)**<br>Host controller has port power control switches. This bit represents the Current setting of the switch (0 = off, 1 = on). When power is not available on a port (i.e. PP equals a 0), the port is nonfunctional and will not report attaches, detaches, etc.<br>When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitioned by the host controller from a 1 to 0 (removing power from the port). |

| | | |
|---|---|---|
| [11:10] | **LSTS** | **Line Status (RO)**<br>These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. These bits are used for detection of low-speed USB devices prior to the port reset and enable sequence. This field is valid only when the port enable bit is zero and the current connect status bit is set to a one.<br>The encoding of the bits are:<br>Bits[11:10] USB State Interpretation<br>00 = SE0 Not Low-speed device, perform EHCI reset.<br>01 = K-state Low-speed device, release ownership of port.<br>10 = J-state Not Low-speed device, perform EHCI reset.<br>11 = Undefined Not Low-speed device, perform EHCI reset.<br>This value of this field is undefined if Port Power is zero. |
| [9] | **Reserved** | Reserved. |
| [8] | **PRST** | **Port Reset (R/W)**<br>When software writes a one to this bit (from a zero), the bus reset sequence as defined in the USB Specification Revision 2.0 is started. Software writes a zero to this bit to terminate the bus reset sequence. Software must keep this bit at a one long enough to ensure the reset sequence, as specified in the USB Specification Revision 2.0, completes. **Note:** when software writes this bit to a one, it must also write a zero to the Port Enable bit.<br>Note that when software writes a zero to this bit there may be a delay before the bit status changes to a zero. The bit status will not read as a zero until after the reset has completed. If the port is in high-speed mode after reset is complete, the host controller will automatically enable this port (e.g. set the Port Enable bit to a one). A host controller must terminate the reset and stabilize the state of the port within 2 milliseconds of software transitioning this bit from a one to a zero. For example: if the port detects that the attached device is high-speed during reset, then the host controller must have the port in the enabled state within 2ms of software writing this bit to a zero.<br>The HCHalted bit in the USBSTS register should be a zero before software attempts to use this bit. The host controller may hold Port Reset asserted to a one when the HCHalted bit is a one.<br>This field is zero if Port Power is zero.<br>0 = Port is not in Reset.<br>1 = Port is in Reset. |

| | | |
|---|---|---|
| [7] | **SUSPEND** | **Suspend (R/W)**<br>Port Enabled Bit and Suspend bit of this register define the port states as follows:.<br>00 = Disable.<br>01 = Disable.<br>10 = Enable.<br>11 = Suspend.<br>When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction, if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB.<br>A write of zero to this bit is ignored by the host controller. The host controller will unconditionally set this bit to a zero when:<br>Software sets the Force Port Resume bit to a zero (from a one).<br>Software sets the Port Reset bit to a one (from a zero).<br>If host software sets this bit to a one when the port is not enabled (i.e.<br>Port enabled bit is a zero) the results are undefined.<br>This field is zero if Port Power is zero.<br>0 = Port not in suspend state.<br>1 = Port in suspend state. |
| [6] | **FPR** | **Force Port Resume (R/W)**<br>This functionality defined for manipulating this bit depends on the value of the Suspend bit. For example, if the port is not suspended (Suspend and Enabled bits are a one) and software transitions this bit to a one, then the effects on the bus are undefined.<br>Software sets this bit to a 1 to drive resume signaling. The Host Controller sets this bit to a 1 if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to a one. If software sets this bit to a one, the host controller must not set the Port Change Detect bit.<br>Note that when the EHCI controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. Software must appropriately time the Resume and set this bit to a zero when the appropriate amount of time has elapsed. Writing a zero (from one) causes the port to return to high-speed mode (forcing the bus below the port into a high-speed idle). This bit will remain a one until the port has switched to the high-speed idle. The host controller must complete this transition within 2 milliseconds of software setting this bit to a zero.<br>This field is zero if Port Power is zero.<br>0 = No resume (Kstate) detected/driven on port.<br>1 = Resume detected/driven on port. |
| [5] | **OCC** | **Over-current Change (R/WC)**<br>1 = This bit gets set to a one when there is a change to Over-current Active. Software clears this bit by writing a one to this bit position. |
| [4] | **OCA** | **Over-current Active (RO)**<br>This bit will automatically transition from a one to a zero when the over current condition is removed.<br>0 = This port does not have an over-current condition.<br>1 = This port currently has an overcurrent condition. |

| | | |
|---|---|---|
| [3] | **PEC** | **Port Enable/Disable Change (R/WC)**<br>For the root hub, this bit gets set to a one only when a port is disabled due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification for the definition of a Port Error). Software clears this bit by writing a 1 to it.<br>This field is zero if Port Power is zero.<br>0 = No change.<br>1 = Port enabled/disabled status has changed. |
| [2] | **PE** | **Port Enabled/Disabled (R/W)**<br>Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. The host controller will only set this bit to a one when the reset sequence determines that the attached device is a high-speed device.<br>Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events.<br>When the port is disabled (0b) downstream propagation of data is blocked on this port, except for reset.<br>This field is zero if Port Power is zero.<br>0 = Disable.<br>1 = Enable. |
| [1] | **CSC** | **Connect Status Change (R/W)**<br>Indicates a change has occurred in the port's Current Connect Status. The host controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be "setting" an already-set bit (i.e., the bit will remain set).Software sets this bit to 0 by writing a 1 to it.<br>This field is zero if Port Power is zero.<br>0 = No change.<br>1 = Change in Current Connect Status. |
| [0] | **CCS** | **Current Connect Status (RO)**<br>This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set.<br>This field is zero if Port Power is zero.<br>0 = No device is present.<br>1 = Device is present on port. |

## USB PHY 0 Control Register (USBPCR0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBPCR0** | EHCI_BA+0x0C4 | R/W | USB PHY 0 Control Register | 0x0000_0060 |

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

| Bits | Description | |
|---|---|---|
| [31:12] | **Reserved** | Reserved. |
| [11] | **CLKVALID** | **UTMI Clock Valid**<br>This bit is a flag to indicate if the UTMI clock from USB 2.0 PHY is ready. S/W program must prevent to write other control registers before this UTMI clock valid flag is active.<br>0 = UTMI clock is not valid.<br>1 = UTMI clock is valid. |
| [10:9] | **Reserved** | Reserved. |
| [8] | **SUSPEND** | **Suspend Assertion**<br>This bit controls the suspend mode of USB PHY 0.<br>While PHY was suspended, all circuits of PHY were powered down and outputs are tristated.<br>This bit is 1'b0 in default. This means the USB PHY 0 is suspended in default. It is necessary to set this bit 1'b1 to make USB PHY 0 leave suspend mode before doing configuration of USB host.<br>0 = USB PHY 0 was suspended.<br>1 = USB PHY 0 was not suspended. |
| [7:0] | **Reserved** | Reserved. |

### USB PHY 1 Control Register (USBPCR1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBPCR1** | EHCI_BA+0x0C8 | R/W | USB PHY 1 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** | | | | | | | **SUSPEND** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:9] | **Reserved** | Reserved. |
| [8] | **SUSPEND** | **Suspend Assertion**<br>This bit controls the suspend mode of USB PHY 1.<br>While PHY was suspended, all circuits of PHY were powered down and outputs are tristated.<br>This bit is 1'b0 in default. This means the USB PHY 0 is suspended in default. It is necessary to set this bit 1'b1 to make USB PHY 0 leave suspend mode before doing configuration of USB host.<br>0 = USB PHY 1 was suspended.<br>1 = USB PHY 1 was not suspended. |
| [7:0] | **Reserved** | Reserved. |

## Host Controller Revision Register (HcRev)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HcRev** | OHCI_BA+0x000 | R | Host Controller Revision Register | 0x0000_0010 |

| 31 | 30 | 29 | 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**<br>Indicates the Open HCI Specification revision number implemented by the Hardware. Host Controller supports 1.0 specification.<br>(X.Y = XYh). |

## Host Controller Control Register (HcControl)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HcControl** | OHCI_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** | | | | | **RWE** | **RWC** | **IR** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **HCFS** | | **BLE** | **CLE** | **IE** | **PLE** | **CBSR** | |

| Bits | Description | |
|---|---|---|
| [31:11] | **Reserved** | Reserved. |
| [10] | **RWE** | **Remote Wakeup Connected Enable**<br>If a remote wakeup signal is supported, this bit enables that operation. Since there is no remote wakeup signal supported, this bit is ignored. |
| [9] | **RWC** | **Remote Wakeup Connected**<br>This bit indicated whether the HC supports a remote wakeup signal. This implementation does not support any such signal. The bit is hard-coded to '0.' |
| [8] | **IR** | **Interrupt Routing**<br>This bit is used for interrupt routing:<br>0 = Interrupts routed to normal interrupt mechanism (INT).<br>1 = Interrupts routed to SMI. |
| [7:6] | **HCFS** | **Host Controller Functional State**<br>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:<br>00 = UsbReset.<br>01 = UsbResume.<br>10 = UsbOperational.<br>11 = UsbSuspend. |
| [5] | **BLE** | **Bulk List Enable**<br>When set this bit enables processing of the Bulk list. |
| [4] | **CLE** | **Control List Enable**<br>When set this bit enables processing of the Control list. |

| | | |
|---|---|---|
| [3] | **IE** | **Isochronous List Enable**<br>When clear, this bit disables the Isochronous List when the Periodic List is enabled (so Interrupt EDs may be serviced). While processing the Periodic List, the Host Controller will check this bit when it finds an isochronous ED. |
| [2] | **PLE** | **Periodic List Enable**<br>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. |
| [1:0] | **CBSR** | **Control Bulk Service Ratio**<br>Specifies the number of Control Endpoints serviced for every Bulk Endpoint.<br>00 = 1:1.<br>01 = 2:1.<br>10 = 3:1.<br>11 = 4:1. |

## Host Controller Command Status Register (HcComSts)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HcComSts** | OHCI_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** | | | | **OCR** | **BLF** | **CLF** | **HCR** |

| Bits | Description | |
|---|---|---|
| [31:18] | **Reserved** | Reserved. |
| [17:16] | **SOC** | **Schedule Overrun Count**<br>This field is increment every time the SchedulingOverrun bit in HcInterruptStatus is set. The count wraps from ‘11’ to ‘00.’ |
| [15:4] | **Reserved** | Reserved. |
| [3] | **OCR** | **Ownership Chang Request**<br>When set by software, this bit sets the OwnershipChange field in HcInterruptStatus. The bit is cleared by software. |
| [2] | **BLF** | **Bulk List Filled**<br>Set to indicate there is an active ED on the Bulk List. The 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. |
| [1] | **CLF** | **Control List Filled**<br>Set to indicate there is an active ED 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] | **HCR** | **Host Controller Reset**<br>This bit is set to initiate the software reset. This bit is cleared by the Host Controller, upon completed of the reset operation. |

## Host Controller Interrupt Status Register (HcIntSts)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcIntSts | OHCI_BA+0x00C | R/W | Host Controller Interrupt Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | OC | 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 | UE | RD | SF | WDH | SO |

| Bits | Description | |
|---|---|---|
| [31] | **Reserved** | Reserved. |
| [30] | **OC** | **Ownership Change**<br>This bit is set when the OwnershipChangeRequest bit of HcCommandStatus is set. |
| [29:7] | **Reserved** | Reserved. |
| [6] | **RHSC** | **Root Hub Status Change**<br>This bit is set when the content of HcRhStatus or the content of any HcRhPortStatus register has changed. |
| [5] | **FNO** | **Frame Number Overflow**<br>Set when bit 15 of HcFNum changes value. |
| [4] | **UE** | **Unrecoverable Error**<br>This event is not implemented and is hard-coded to '0.' Writes are ignored. |
| [3] | **RD** | **Resume Detected**<br>Set when Host Controller detects resume signaling on a downstream port. |
| [2] | **SF** | **Start of Frame**<br>Set when the Frame Management block signals a 'Start of Frame' event. |
| [1] | **WDH** | **Write Back Done Head**<br>Set after the Host Controller has written HcDoneHead to HccaDoneHead. |
| [0] | **SO** | **Scheduling Overrun**<br>Set when the List Processor determines a Schedule Overrun has occurred. |

## Host Controller Interrupt Enable Register (HcIntEn)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcIntEn | OHCI_BA+0x010 | R/W | Host Controller Interrupt Enable Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MIE | OC | 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 | UE | RD | SF | WDH | SO |

| Bits | Description | |
|---|---|---|
| [31] | MIE | **Master Interrupt Enable**<br>This bit is a global interrupt enable. A write of '1' allows interrupts to be enabled via the specific enable bits listed above. |
| [30] | OC | **Ownership Change Enable**<br>0 =Ignore.<br>1 = Enables interrupt generation due to Ownership Change. |
| [29:7] | Reserved | Reserved. |
| [6] | RHSC | **Root Hub Status Change Enable**<br>0 = Ignore.<br>1 = Enables interrupt generation due to Root Hub Status Change. |
| [5] | FNO | **Frame Number Overflow Enable**<br>0 = Ignore.<br>1 = Enables interrupt generation due to Frame Number Overflow. |
| [4] | UE | **Unrecoverable Error Enable**<br>This event is not implemented. All writes to this bit are ignored. |
| [3] | RD | **Resume Detected Enable**<br>0 = Ignore.<br>1 = Enables interrupt generation due to Resume Detected. |
| [2] | SF | **Start of Frame Enable**<br>0 = Ignore.<br>1 = Enables interrupt generation due to Start of Frame. |
| [1] | WDH | **Write Back Done Head Enable**<br>0 = Ignore.<br>1 = Enables interrupt generation due to Write-back Done Head. |

| | | |
|---|---|---|
| [0] | **SO** | **Scheduling Overrun Enable**<br>0 = Ignore.<br>1 = Enables interrupt generation due to Scheduling Overrun. |

## Host Controller Interrupt Disable Register (HcIntDis)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcIntDis | OHCI_BA+0x014 | R/W | Host Controller Interrupt Disable Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MIE | OC | 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 | UE | RD | SF | WDH | SO |

| Bits | Description | |
|---|---|---|
| [31] | **MIE** | **Master Interrupt Disable**<br>Global interrupt disable. A write of '1' disables all interrupts. |
| [30] | **OC** | **Ownership Change Disable**<br>0 = Ignore.<br>1 = Disables interrupt generation due to Ownership Change. |
| [29:7] | **Reserved** | Reserved. |
| [6] | **RHSC** | **Root Hub Status Change Disable**<br>0 = Ignore.<br>1 = Disables interrupt generation due to Root Hub Status Change. |
| [5] | **FNO** | **Frame Number Overflow Disable**<br>0 = Ignore.<br>1 = Disables interrupt generation due to Frame Number Overflow. |
| [4] | **UE** | **Unrecoverable Error Disable**<br>This event is not implemented. All writes to this bit are ignored. |
| [3] | **RD** | **Resume Detected Disable**<br>0 = Ignore.<br>1 = Disables interrupt generation due to Resume Detected. |
| [2] | **SF** | **Start of Frame Disable**<br>0 = Ignore.<br>1 = Disables interrupt generation due to Start of Frame. |
| [1] | **WDH** | **Write Back Done Head Disable**<br>0 = Ignore.<br>1 = Disables interrupt generation due to Write-back Done Head. |

| | | |
|---|---|---|
| [0] | **SO** | **Scheduling Overrun Disable**<br>0 = Ignore.<br>1 = Disables interrupt generation due to Scheduling Overrun. |

## Host Controller Communication Area Register (HcHCCA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HcHCCA** | OHCI_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 | |
|---|---|---|
| [31:7] | **HCCA** | **Host Controller Communication Area**<br>Pointer to HCCA base address. |
| [7:0] | **Reserved** | Reserved. |

## Host Controller Period Current ED Register (HcPerCED)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HcPerCED** | OHCI_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**<br>Pointer to the current Periodic List ED. |
| [3:0] | **Reserved** | Reserved. |

## Host Controller Control Head ED Register (HcCtrHED)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HcCtrHED** | OHCI_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**<br>Pointer to the Control List Head ED. |
| [3:0] | **Reserved** | Reserved. |

### Host Controller Control Current ED Register (HcCtrCED)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HcCtrCED** | OHCI_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**<br>Pointer to the current Control List Head ED. |
| [3:0] | **Reserved** | Reserved. |

## Host Controller Bulk Head ED Register (HcBlkHED)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HcBlkHED** | OHCI_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**<br>Pointer to the Bulk List Head ED. |
| [3:0] | **Reserved** | Reserved. |

### Host Controller Bulk Current Head ED Register (HcBlkCED)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HcBlkCED** | OHCI_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**<br>Pointer to the current Bulk List Head ED. |
| [3:0] | **Reserved** | Reserved. |

## Host Controller Done Head Register (HcDoneH)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HcDoneH** | OHCI_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**<br>Pointer to the current Done List Head ED. |
| [3:0] | **Reserved** | Reserved. |

## Host Controller Frame Interval Register (HcFmIntv)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HcFmIntv** | OHCI_BA+0x034 | R/W | Host Controller Frame Interval Register | 0x0000_2EDF |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **FIT** | **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**<br>This bit is toggled by HCD when it loads a new value into FrameInterval. |
| [30: 16] | **FSMPS** | **FS Largest Data Packet**<br>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**<br>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 (HcFmRem)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HcFmRem** | OHCI_BA+0x038 | R | Host Controller Frame Remaining Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31] | **FR** | **Frame Remaining Toggle**<br>Loaded with FrameIntervalToggle when FrameRemaining is loaded. |
| [30:14] | **Reserved** | Reserved. |
| [13:0] | **FRT** | **Frame Remaining**<br>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 FrameInterval. In addition, the counter loads when the Host Controller transitions into UsbOperational. |

## Host Controller Frame Number Register (HcFNum)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HcFNum** | OHCI_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**<br>This 16-bit incrementing counter field is incremented coincident with the loading of FrameRemaining. The count rolls over from ‘FFFFh’ to ‘0h.’ |

## Host Controller Periodic Start Register (HcPerSt)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HcPerSt** | OHCI_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**<br>This field contains a value used by the List Processor to determine where in a frame the Periodic List processing must begin. |

## Host Controller Root Hub Descriptor A Register (HcRhDeA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HcRhDeA** | OHCI_BA+0x048 | R/W | Host Controller Root Hub Descriptor A Register | 0x0100_0002 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **POTPGT** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | **NOCP** | **OCPM** | **DT** | **NPS** | **PSM** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **NDP** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **POTPGT** | **Power on to Power Good Time**<br>This field value is represented as the number of 2 ms intervals, which ensuring that the power switching is effective within 2 ms. Only bits [25:24] are implemented as R/W. The remaining bits are read only as '0'. It is not expected that these bits be written to anything other than 1h, but limited adjustment is provided. This field should be written to support system implementation. This field should always be written to a non-zero value. |
| [23:13] | **Reserved** | Reserved. |
| [12] | **NOCP** | **No over Current Protection**<br>Global over-current reporting implemented in HYDRA-2. This bit should be written to support the external system port over-current implementation.<br>0 = Over-current status is reported.<br>1 = Over-current status is not reported. |
| [11] | **OCPM** | **Over Current Protection Mode**<br>Global over-current reporting implemented in HYDRA-2. This bit should be written 0 and is only valid when NoOverCurrentProtection is cleared.<br>0 = Global Over-Current.<br>1 = Individual Over-Current. |
| [10] | **DT** | **Device Type**<br>-4is not a compound device. |
| [9] | **NPS** | **No Power Switching**<br>Global power switching implemented in HYDRA-2. This bit should be written to support the external system port power switching implementation.<br>0 = Ports are power switched.<br>1 = Ports are always powered on. |

| | | |
|---|---|---|
| [8] | **PSM** | **Power Switching Mode**<br>Global power switching mode implemented in HYDRA-2. This bit is only valid when NoPowerSwitching is cleared. This bit should be written '0'.<br>0 = Global Switching.<br>1 = Individual Switching. |
| [7:0] | **NDP** | **Number Downstream Ports**<br>NUC970 supports two downstream ports. |

## Host Controller Root Hub Descriptor B Register (HcRhDeB)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HcRhDeB** | OHCI_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** |
| **DevRemove** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **DR** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **PPCM** | **Port Power Control Mask**<br>Global-power switching. This field is only valid if NoPowerSwitching is cleared and PowerSwitchingMode is set (individual port switching). When set, the port only responds to individual port power switching commands (Set/ClearPortPower). When cleared, the port only responds to global power switching commands (Set/ClearGlobalPower).<br>0 = Device not removable<br>1 = Global-power mas..k<br>Port Bit relationship - Unimplemented ports are reserved, read/write '0'.<br>0 : Reserved<br>1 : Port 1<br>2 : Port 2<br><br>15 : Port 15 |
| [15:0] | **DR** | **Device Removable**<br>NUC970 ports default to removable devices.<br>0 = Device not removable<br>1 = Device removable.<br>Port Bit relationship<br>0 : Reserved<br>1 : Port 1<br>2 : Port 2<br>...<br>15 : Port 15<br>Unimplemented ports are reserved, read/write '0'. |

## Host Controller Root Hub Status Register (HcRhSts)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| HcRhSts | OHCI_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 Wakeup Enable**<br>Writing a '1' to this bit clears DeviceRemoteWakeupEnable. Writing a '1' has no effect. |
| [17] | **OCIC** | **Over Current Indicator Change**<br>This bit is set when OverCurrentIndicator changes. Writing a '1' clears this bit. Writing a '0' has no effect. |
| [16] | **LPSC** | **(Read) LocalPowerStatusChange**<br>Not supported. Always read '0'.<br>(Write) SetGlobalPower<br>Write a '1' issues a SetGlobalPower command to the ports. Writing a '0' has no effect. |
| [15] | **DRWE** | **(Read) DeviceRemoteWakeupEnable**<br>This bit enables ports' ConnectStatusChange as a remote wakeup event.<br>0 = disabled.<br>1 = enabled.<br>(Write) SetRemoteWakeupEnable<br>Writing a '1' sets DeviceRemoteWakeupEnable. Writing a '0' has no effect. |
| [1] | **OCI** | **Over Current Indicator**<br>This bit reflects the state of the OVRCUR pin. This field is only valid if NoOverCurrentProtection and OverCurrentProtectionMode are cleared.<br>0 = No over-current condition.<br>1 = Over-current condition. |
| [0] | **LPS** | **(Read) LocalPowerStatus**<br>Not Supported. Always read '0'.<br>(Write) ClearGlobalPower<br>Writing a '1' issues a ClearGlobalPower command to the ports. Writing a '0' has no effect. |

## Host Controller Root Hub Port Status (HcRhPrt [1: 2])

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HcRhPrt1** | OHCI_BA+0x054 | R/W | Host Controller Root Hub Port Status [1] | 0x0000_0000 |
| **HcRhPrt2** | OHCI_BA+0x058 | R/W | Host Controller Root Hub Port Status [2] | 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 | |
|---|---|---|
| [20] | **PRSC** | **Port Reset Status Change**<br>This bit indicates that the port reset signal has completed.<br>0 = Port reset is not complete.<br>1 = Port reset is complete. |
| [19] | **OCIC** | **Port over Current Indicator Change**<br>This bit is set when OverCurrentIndicator changes. Writing a '1' clears this bit. Writing a '0' has no effect. |
| [18] | **PSSC** | **Port Suspend Status Change**<br>This bit indicates the completion of the selective resume sequence for the port.<br>0 = Port is not resumed.<br>1 = Port resume is complete. |
| [17] | **PESC** | **Port Enable Status Change**<br>This bit indicates that the port has been disabled due to a hardware event (cleared PortEnableStatus).<br>0 = Port has not been disabled.<br>1 = PortEnableStatus has been cleared. |
| [16] | **CSC** | **Connect Status Change**<br>This bit indicates connect or disconnect event has been detected. Writing a '1' clears this bit. Writing a '0' has no effect.<br>0 = No connect/disconnect event.<br>1 = Hardware detection of connect/disconnect event.<br>**Note:** If DeviceRemoveable is set, this bit resets to '1'. |

| | | |
|---|---|---|
| [9] | **LSDA** | **(Read) LowSpeedDeviceAttached**<br>This bit defines the speed (and bud idle) of the attached device. It is only valid when CurrentConnectStatus is set.<br>0 = Full Speed device.<br>1 = Low Speed device.<br>(Write) ClearPortPower<br>Writing a '1' clears PortPowerStatus. Writing a '0' has no effect |
| [8] | **PPS** | **(Read) PortPowerStatus**<br>This bit reflects the power state of the port regardless of the power switching mode.<br>0 = Port power is off.<br>1 = Port power is on.<br>**Note:** If NoPowerSwitching is set, this bit is always read as '1'.<br>(Write) SetPortPower<br>Writing a '1' sets PortPowerStatus. Writing a '0' has no effect. |
| [4] | **PRS** | **(Read) PortResetStatus**<br>0 = Port reset signal is not active.<br>1 = Port reset signal is active.<br>(Write) SetPortReset<br>Writing a '1' sets PortResetStatus. Writing a '0' has no effect. |
| [3] | **POCI** | **(Read) PortOverCurrentIndicator**<br>-2 supports global over-current reporting. This bit reflects the state of the OVRCUR pin dedicated to this port. This field is only valid if NoOverCurrentProtection is cleared and OverCurrentProtectionMode is set.<br>0 = No over-current condition.<br>1 = Over-current condition.<br>(Write) ClearPortSuspend<br>Writing a '1' initiates the selective resume sequence for the port. Writing a '0' has no effect. |
| [2] | **PSS** | **(Read) PortSuspendStatus**<br>0 = Port is not suspended.<br>1 = Port is selectively suspended.<br>(Write) SetPortSuspend<br>Writing a '1' sets PortSuspendStatus. Writing a '0' has no effect. |
| [1] | **PES** | **(Read) PortEnableStatus**<br>0 = Port disabled.<br>1 = Port enabled.<br>(Write) SetPortEnable<br>Writing a '1' sets PortEnableStatus. Writing a '0' has no effect. |
| [0] | **CCS** | **(Read) CurrentConnectStatus**<br>0 = No device connected.<br>1 = Device connected.<br>**NOTE:** If DeviceRemoveable is set (not removable) this bit is always '1'.<br>(Write) ClearPortEnable<br>Writing '1' a clears PortEnableStatus. Writing a '0' has no effect. |

## USB Operational Mode Enable Register (OpModEn)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **OpModEn** | OHCI_BA+0x204 | R/W | USB Operational Mode 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** | | | | | | | **SIEPD** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | | **OCAL** | **Reserved** | **ABORT** | **DBR16** |

| Bits | Description | |
|---|---|---|
| [31:9] | **Reserved** | Reserved. |
| [8] | **SIEPD** | **SIE Pipeline Disable**<br>When set, waits for all USB bus activity to complete prior to returning completion status to the List Processor. This is a failsafe mechanism to avoid potential problems with the clk_dr transition between 1.5 MHz and 12 MHz. |
| [7:4] | **Reserved** | Reserved. |
| [3] | **OCAL** | **Over Current Active Low**<br>This bit controls the polarity of over current flag from external power IC.<br>0 = Over current flag is high active.<br>1 = Over current flag is low active. |
| [2] | **Reserved** | Reserved. |
| [1] | **ABORT** | **AHB Bus ERROR Response**<br>This bit indicates there is an ERROR response received in AHB bus.<br>0 = No ERROR response received.<br>1 = ERROR response received. |
| [0] | **DBR16** | **Data Buffer Region 16**<br>When set, the size of the data buffer region is 16 bytes. Otherwise, the size is 32 bytes. |

## 5.24 Controller Area Network (CAN)

### 5.24.1 Overview

The C_CAN consists of the CAN Core, Message RAM, Message Handler, Control Registers and Module Interface (Refer to Figure 5.24-1) 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 1MBit/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.

### 5.24.2 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

### 5.24.3 Block Diagram

The C_CAN interfaces with the AMBA APB bus. The following figure 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 ARM.

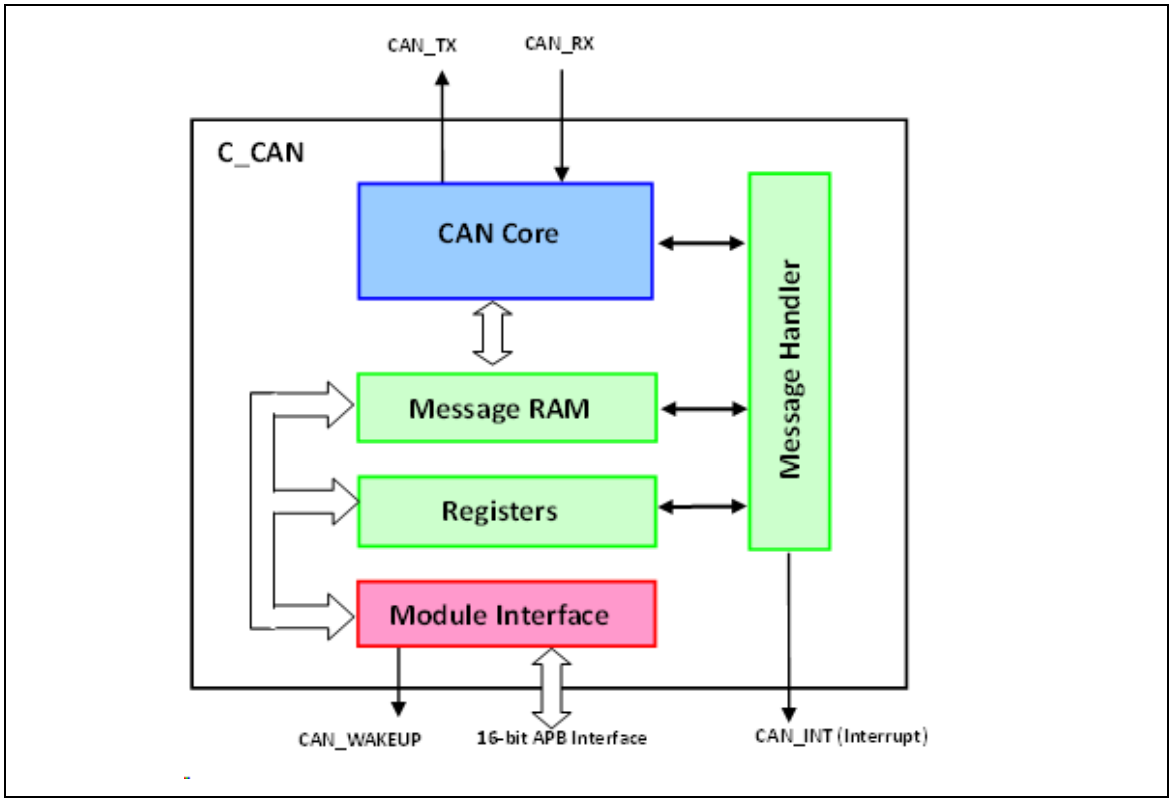



Figure 5.24-1 CAN Peripheral Block Diagram

### 5.24.4 Basic Configuration

Before using CAN functionality, it's necessary to configure I/O pins as the CAN function and enable CAN's clock.

Write 0xC to MFP_GPB10 (SYS_GPB_MFPH[11:8]) and MFP_GPB11 (SYS_GPB_MFPH[15:12]) configures pin PB.10 and PB.11 to be CAN0_RX and CAN0_TX respectively.

Write 0xC to MFP_GPH2 (SYS_GPH_MFPL[11:8]) and MFP_GPH3 (SYS_GPH_MFPL[15:12]) configures pin PH.2 and PH.3 to be CAN0_RX and CAN0_TX respectively.

Write 0xC to MFP_GPI3 (SYS_GPI_MFPL[15:12]) and MFP_GPI4 (SYS_GPI_MFPL[19:16]) configures pin PI.3 and PI.4 to be CAN0_RX and CAN0_TX respectively.

Write 0xC to MFP_GPH14 (SYS_GPH_MFPH[27:24]) and MFP_GPH15 (SYS_GPH_MFPH[31:28]) configures pin PH.14 and PH.15 to be CAN1_RX and CAN1_TX respectively.

Please note that configure PB.10, PH.2 and PI.3 to be CAN0_RX functionality in the same time or configure PB.11, PH.3 and PI.4 to be CAN0_TX functionality in the same time is prohibited.

To enable CAN's clock, please refer to register CLK_PCLKEN1. Set CAN0 (CLK_PCLKEN1[8]) high to enable CAN0 clock while set CAN1 (CLK_PCLKEN1[9]) high to enable CAN1 clock.

### 5.24.5 Functional Description

#### 5.24.5.1 *Software Initialization*

The software initialization is started by setting the Init bit in the CAN Control Register, 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 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 bits in the CAN Control Register are set.

Resetting the Init bit (by software only) finishes the software initialization. Later, the Bit Stream Processor (BSP) (see Section 5.13.6.10: 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 is set again.

#### 5.24.5.2 *CAN Message Transfer*

Once the C_CAN is initialized and Init bit 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 and eight data bytes 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 with NewDat bit 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.

#### 5.24.5.3 *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 in the CAN Control Register to one. In this operation mode, the programmer has to consider the different behavior of bits TxRqst and NewDat in the Control Registers 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.

### 5.24.6 Test Mode

Test Mode is entered by setting the Test bit in the CAN Control Register. In Test Mode, bits Tx1, Tx0, LBack, Silent and Basic in the Test Register are writeable. Bit Rx 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.

#### 5.24.6.1 *Silent Mode*

The CAN Core can be set in Silent Mode by programming the Silent bit in the Test Register 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 analysis the traffic on a CAN bus without affecting it by the transmission of dominant bits. The following figure shows the connection of signals CAN_TX and CAN_RX to the CAN Core in Silent Mode.

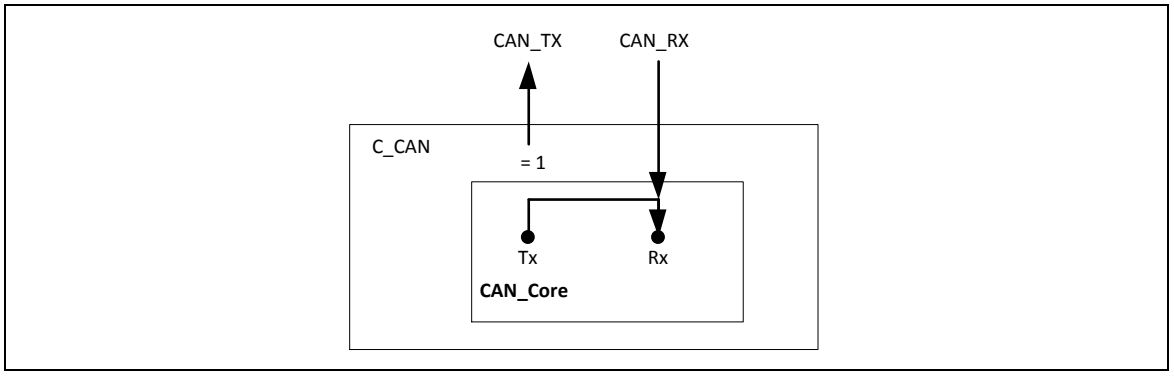



Figure 5.24-2 CAN Core in Silent Mode

5.24.6.2 *Loop Back Mode*

The CAN Core can be set in Loop Back Mode by programming the Test Register bit LBack 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 5-78 shows the connection of signals, CAN_TX and CAN_RX, to the CAN Core in Loop Back Mode.

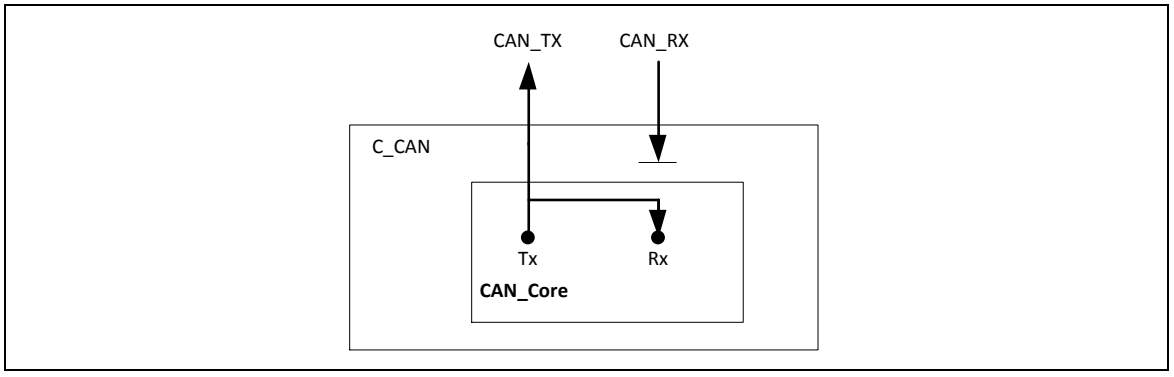



Figure 5.24-3 CAN Core in Loop Back Mode

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.

5.24.6.3 *Loop Back Combined with Silent Mode*

It is also possible to combine Loop Back Mode and Silent Mode by programming bits LBack and Silent 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. The following figure 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.

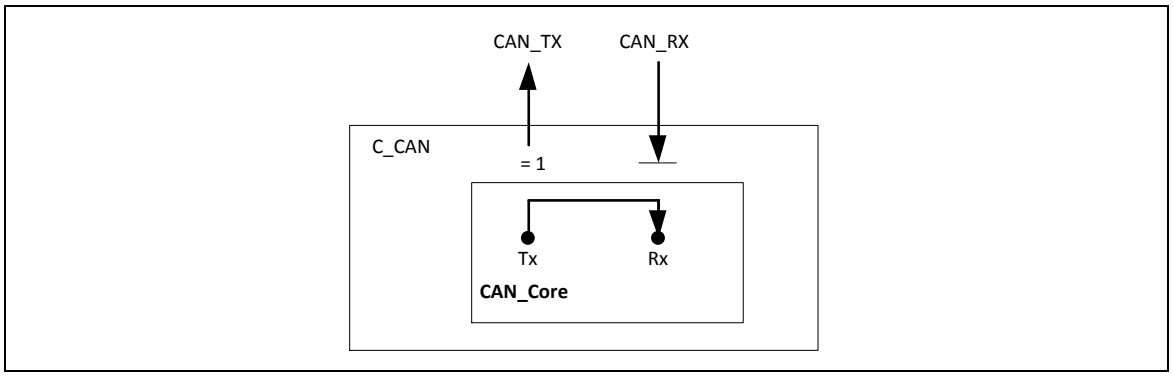



Figure 5.24-4 CAN Core in Loop Back Mode Combined with Silent Mode

5.24.6.4 *Basic Mode*

The CAN Core can be set in Basic Mode by programming the Test Register bit Basic 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 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 and MsgLst bits in the IF2 Message Control Register retain their function, DLC3-0 indicates the received DLC, and the other control bits are read as ‘0’.

5.24.6.5 *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 and Tx0 bits of the CAN Test Register.

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.

### 5.24.7 CAN Communications

#### 5.24.7.1 *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 = ‘0’) and the bit timing must be configured before the application software clears the Init bit in ter.

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 in the CAN Control Register 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.

#### 5.24.7.2 *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.

#### 5.24.7.3 *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 in the respective Command Request Register (CAN_IFn_CRR) to ‘1’. After the transfer has completed, the Busy bit is again cleared (see the following figure).

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.

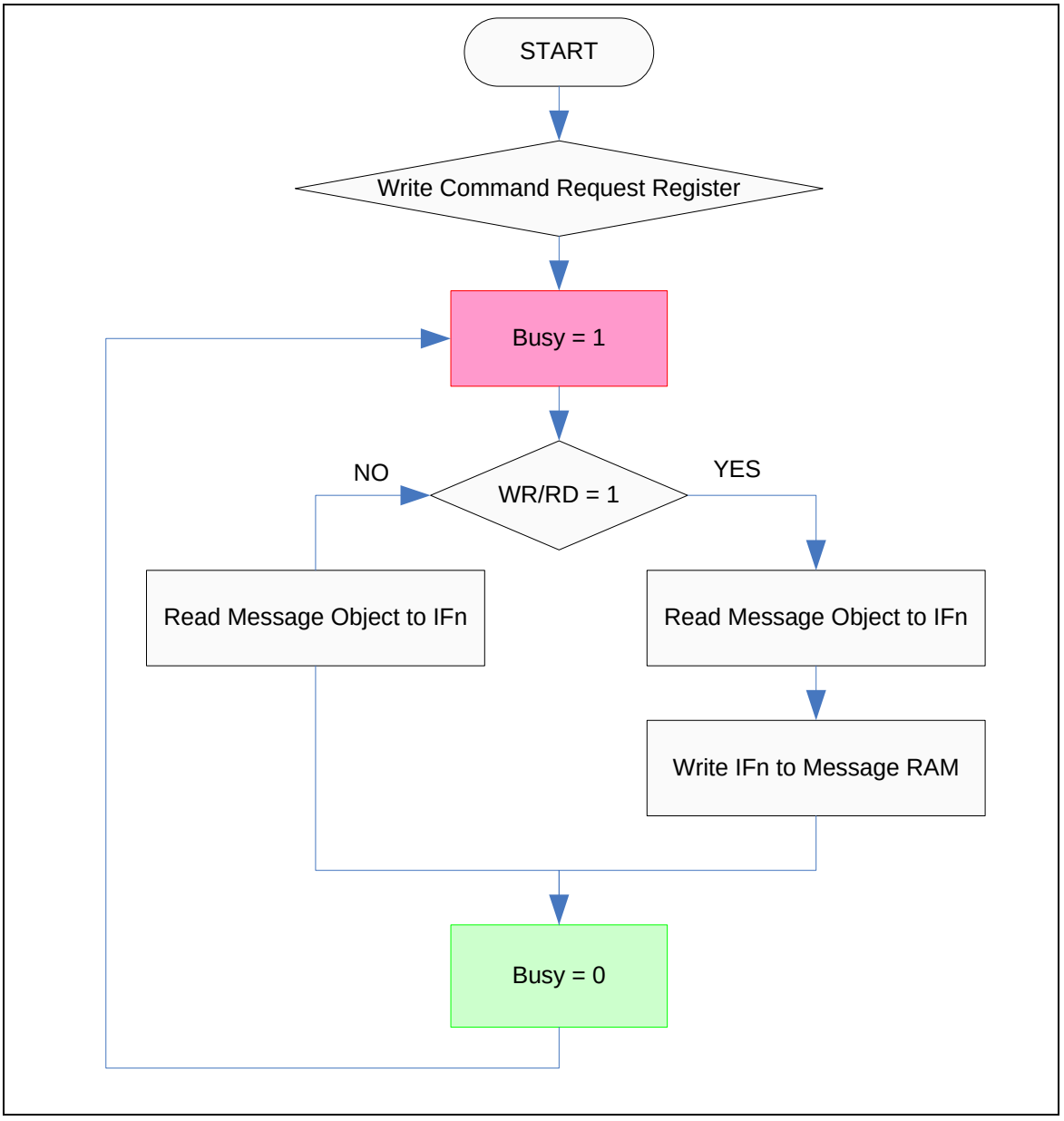



Figure 5.24-5 Data transfer between IFn Registers and Message

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.

#### 5.24.7.4 *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 bits in the Message Valid Register and TxRqst bits in the Transmission Request Register 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 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_MCR) will be reset. If TxIE bit of the Message Control register (CAN_IFn_MCR) is set, IntPnd bit 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.

#### 5.24.7.5 *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, UMask, NewDat, and EoB) 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 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 is set to indicate that the previous data (supposedly not seen by the software) is lost. If the RxIE bit is set, the IntPnd bit is set, causing the Interrupt Register to point to this Message Object.

The TxRqst bit 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 = ‘1’ (direction = transmit), RmtEn = ‘1’, UMask = ‘1’ or’0’

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.

2) Dir = ‘1’ (direction = transmit), RmtEn = ‘0’, UMask =’0’

At the reception of a matching Remote Frame, the TxRqst bit of this Message Object remains unchanged; the Remote Frame is ignored.

3) Dir = ‘1’ (direction = transmit), RmtEn = ‘0’, UMask =’1’

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 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.

#### 5.24.7.6 *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

#### 5.24.7.7 *Configuring a Transmit Object*

The following table 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 5.24-1 Initialization of a Transmit Object

**Note:** appl. = application software.

The Arbitration Register values (ID28-0 and Xtd bit) 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 is set, the IntPnd bit will be set after a successful transmission of the Message Object.

If the RmtEn bit is set, a matching received Remote Frame will cause the TxRqst bit to be set; the Remote Frame will autonomously be answered by a Data Frame.

The Data Register values (DLC3-0, Data0-7) 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=’1’) to allow groups of Remote Frames with similar identifiers to set the TxRqst bit. The Dir bit should not be masked.

#### 5.24.7.8 *Updating a Transmit Object*

The software may update the data bytes of a Transmit Object any time through the IFn Interface registers, neither MsgVal nor TxRqst 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 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.

#### 5.24.7.9 *Configuring a Receive Object*

The following table 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 5.24-2 Initialization of a Receive Object

The Arbitration Registers values (ID28-0 and Xtd bit) 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 is set, the IntPnd bit will be set when a received Data Frame is accepted and stored in the Message Object.

The Data Length Code (DLC3-0) 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=’1’) to allow groups of Data Frames with similar identifiers to be accepted. The Dir bit should not be masked in typical applications.

#### 5.24.7.10 *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 and IntPnd are cleared in the Message RAM (not in the Message Buffer).

If the Message Object uses masks for acceptance filtering, the arbitration bits shows 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 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 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.

#### 5.24.7.11 *Configuring a FIFO Buffer*

With the exception of the EoB bit, the configuration of Receive Objects belonging to a FIFO Buffer is the same as the configuration of a (single) Receive Object, see Section 5.13.6.5: 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 bits of the last Message Object of a FIFO Buffer is set to one, configuring it as the End of the Block.

#### 5.24.7.12 *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 of this Message Object is set. By setting NewDat while EoB 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 previous messages.

#### 5.24.7.13 *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 and IntPnd are reset to zero (TxRqst/NewDat = '1' and ClrIntPnd = '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.

The following figure shows how a set of Message Objects which are concatenated to a FIFO Buffer can be handled by the application software.

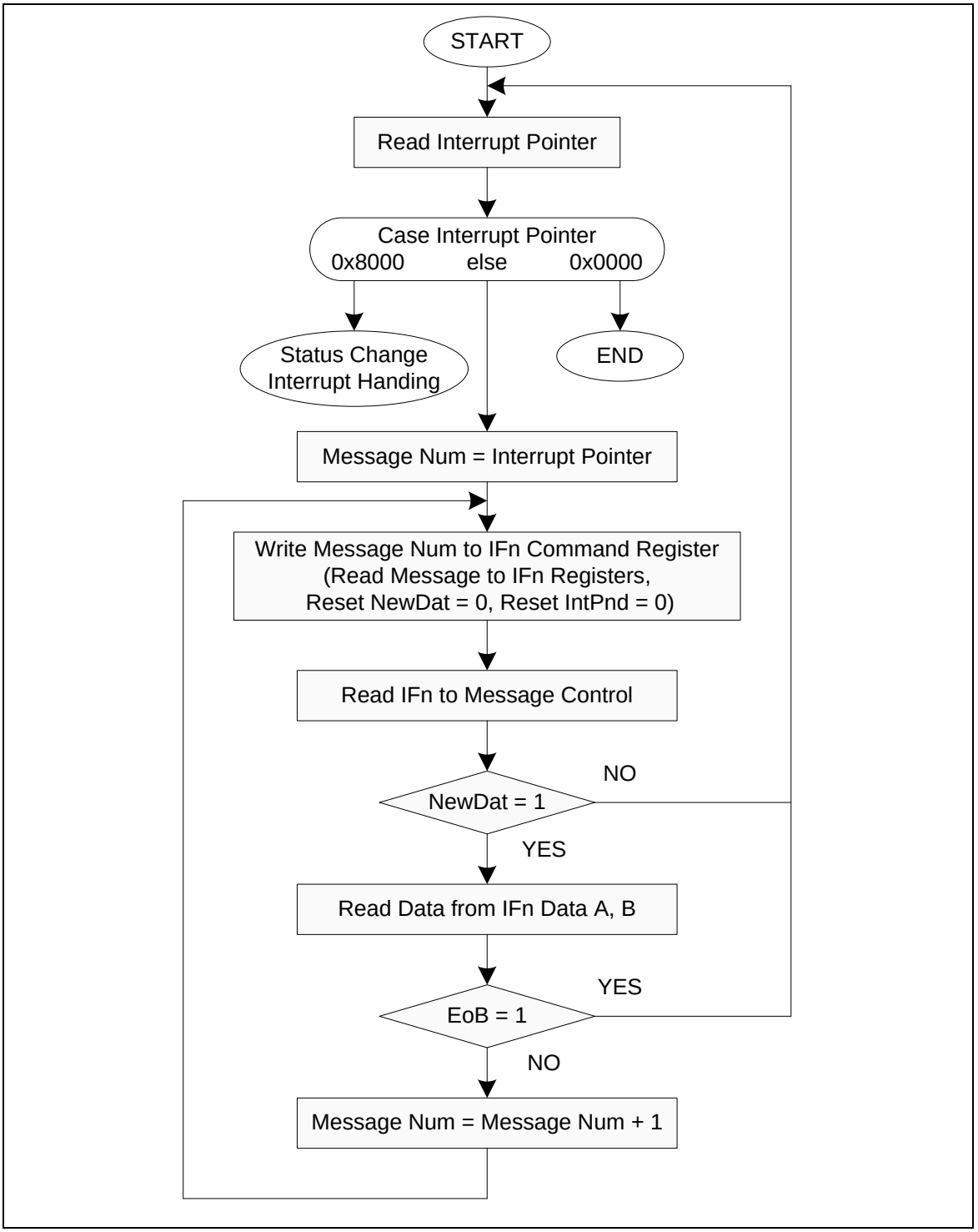



Figure 5.24-6 Application Software Handling of a FIFO Buffer

5.24.7.14 *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 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 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, TxOk and LEC, 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 and SIE in the CAN Control Register) 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 in the Command Mask Register). When IntPnd is cleared, the Interrupt Register will point to the next Message Object with a pending interrupt.

#### 5.24.7.15 *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.

#### 5.24.7.16 *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 the following figure). 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 (see Table 5.24-3). 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 of the Bit Timing Register (CAN_BTR): 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 times 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.

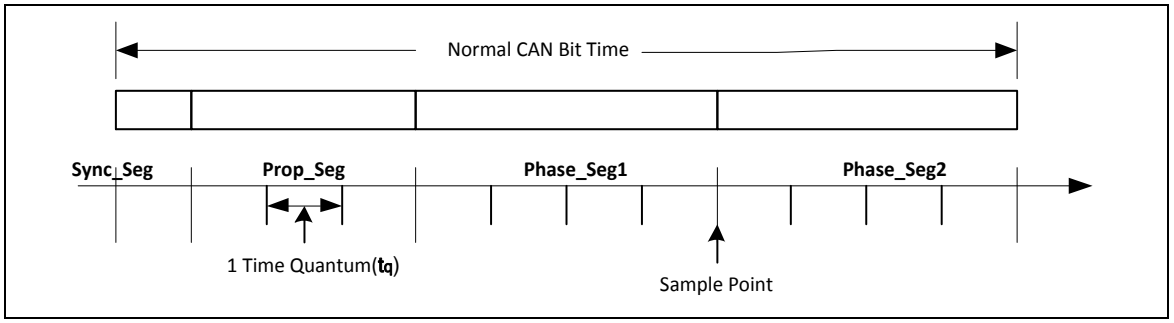



Figure 5.24-7 Bit Timing

| Parameter | Range | Remark |
|---|---|---|
| BRP | [1 .. 32] | Defines the length of the time quantum tq |
| Sync_Seg | 1 tq | Fixed length, synchronization of bus input to APB clock |
| Prop_Seg | [1.. 8] tq | Compensates for the physical delay times |
| Phase_Seg1 | [1..8] tq | Which may be lengthened temporarily by synchronization |
| Phase_Seg2 | [1.. 8] tq | Which may be shortened temporarily by synchronization |
| SJW | [1 .. 4] tq | Which may not be longer than either Phase Buffer Segment |
| This table describes the minimum programmable ranges required by the CAN protocol | | |

Table 5.24-3 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 times and the oscillator's tolerance range have to be considered.

### 5.24.7.17 *Propagation Time Segment*

This part of the bit time is used to compensate physical delay times within the network. These delay times 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 the following figure shows the phase shift and propagation times between two CAN nodes.

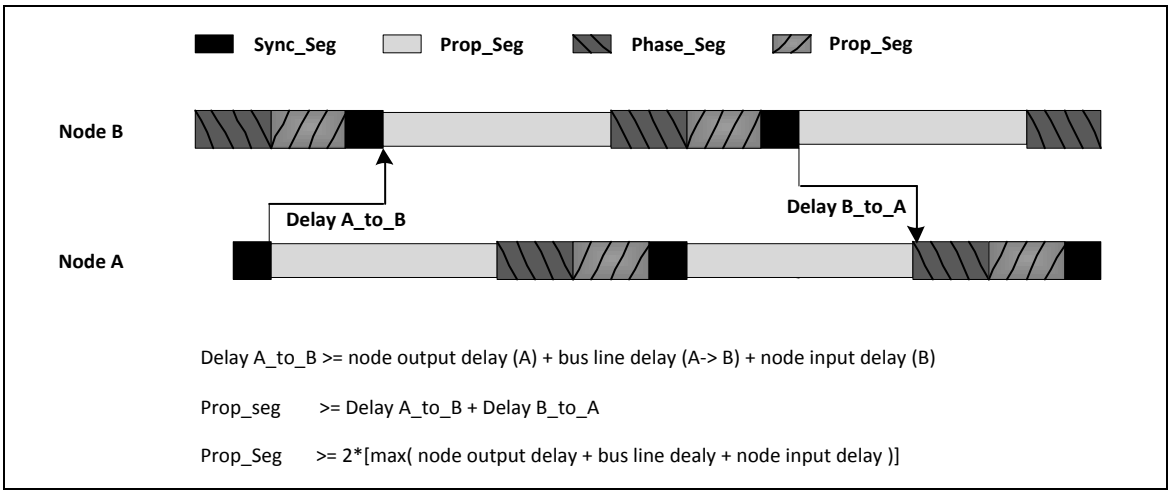



Figure 5.24-8 Propagation Time Segment

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 to 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.

#### 5.24.7.18 *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 times, 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 the following figure 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.

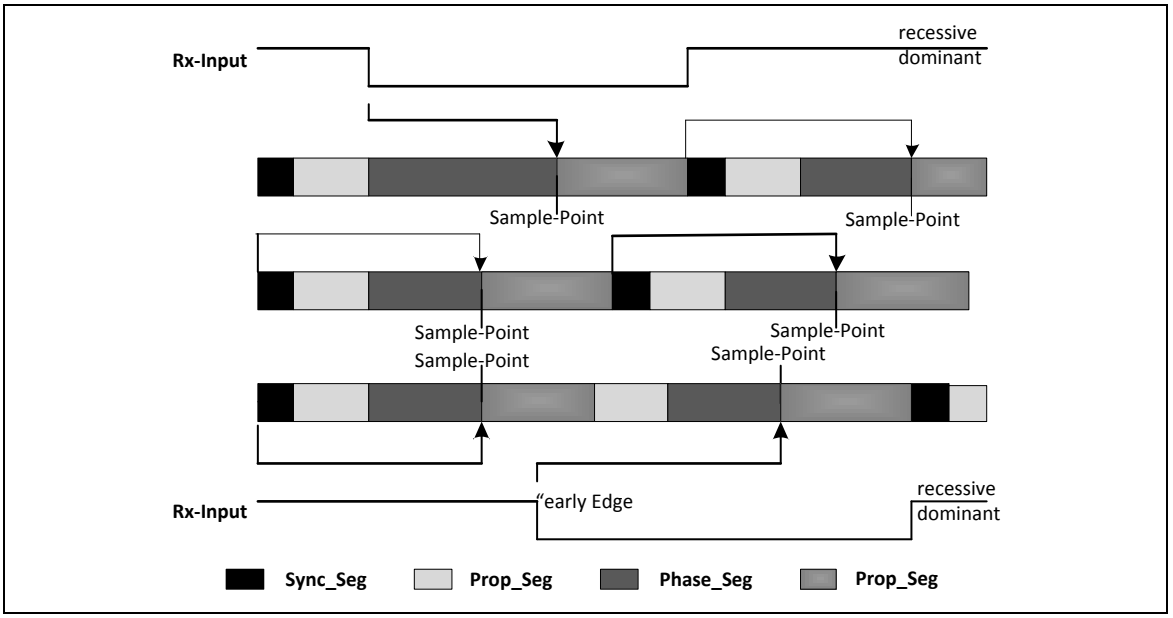



Figure 5.24-9 Synchronization on “late” and “early” Edges

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 the following figure 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.

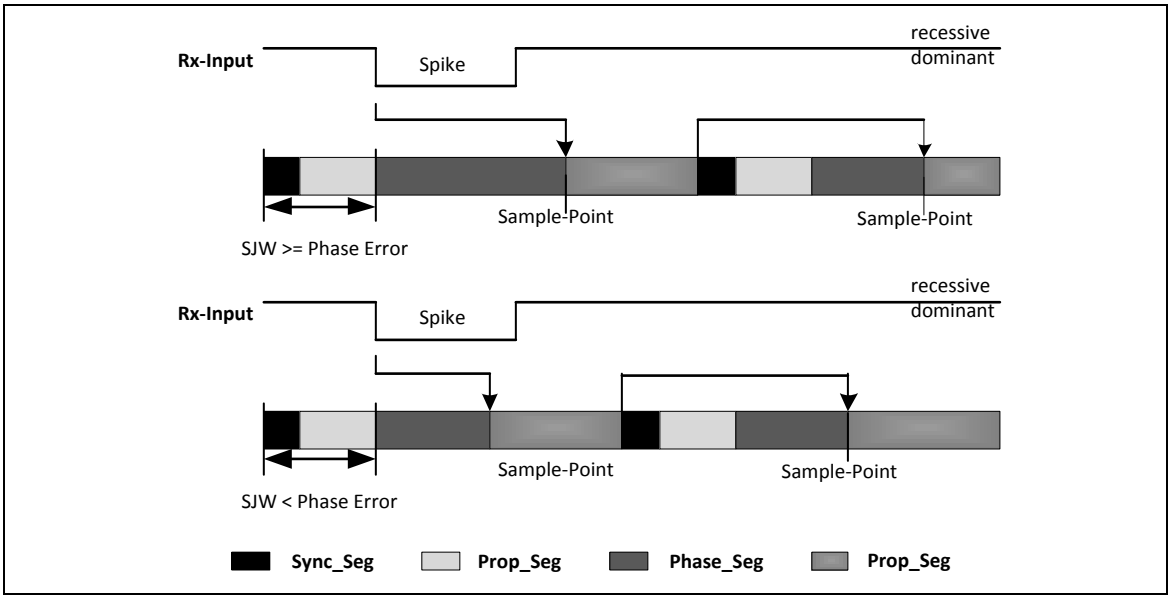



Figure 5.24-10 Filtering of Short Dominant Spikes

5.24.7.19 *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 $f_{osc}$ around the nominal frequency $f_{nom}$ is:

(1 – df) · fnom <= fosc <= (1 + df) · 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):

$$\text{I: } df <= \frac{\min(\text{Phase\_Seg1}, \text{Phase\_Seg2})}{2 * (13 * \text{bit\_time} - \text{Phase\_Seg2})}$$

$$\text{II: } df <= \frac{\text{SJW}}{20 * \text{bit\_time}}$$

**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 = 8 s) with a bus length of 40 m.

5.24.7.20 *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) is combined with Phase_Seg2 (as TSEG2) in one byte, SJW and BRP 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.

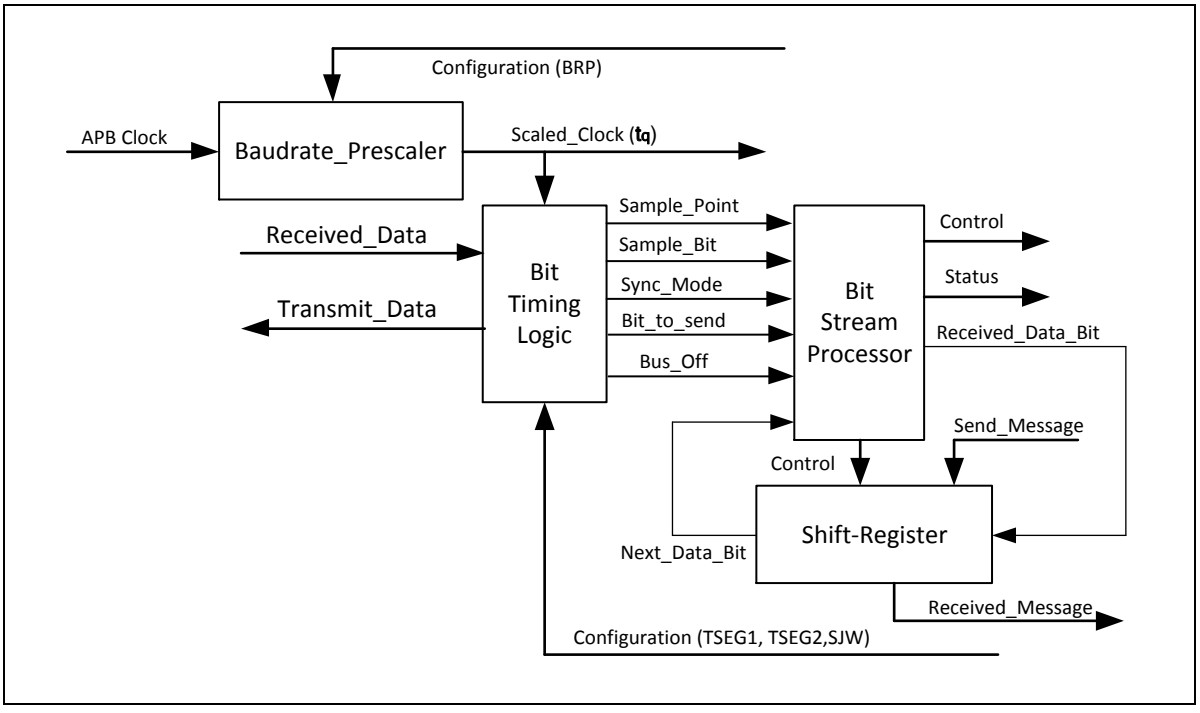



Figure 5.24-11 Structure of the CAN Core's CAN Protocol Controller

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 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.

#### 5.24.7.21 *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 $t_q$ = (Baud Rate Prescaler)/$f_{apb\_clk}$. 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 times 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 $t_q$ long (fixed), leaving (bit time – Prop_Seg – 1) $t_q$ for the two Phase Buffer Segments. If the number of remaining $t_q$ 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] $t_q$.

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 5.13.6.10.4: 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 times, 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 show 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 is 0, the bit rate is 1 MBit/s.

| | | | |
|---|---|---|---|
| $t_q$ | 100 ns | | = $t_{APB\_CLK}$ |
| delay of bus driver | 50 | ns | |
| delay of receiver circuit | 30 | ns | |
| delay of bus line (40m) | 220 | ns | |
| $t_{Prop}$ | 600 ns | | = 6 • $t_q$ |
| $t_{SJW}$ | 100 ns | | = 1 • $t_q$ |
| $t_{TSeg1}$ | 700 | ns | = $t_{Prop}$ + $t_{SJW}$ |
| $t_{TSeg2}$ | 200 | ns | = Information Processing Time + 1 • $t_q$ |
| $t_{Sync-Seg}$ | 100 | ns | = 1 • $t_q$ |
| bit time | 1000 | ns | = $t_{Sync-Seg}$ + $t_{TSeg1}$ + $t_{TSeg2}$ |
| tolerance for APB_CLK | 0.39 | % | $= \frac{Min(PB1, PB2)}{2 \times 13 \times (bit\ \ time - PB2))}$ |

$$= \frac{0.1\mu s}{2x(13x(1\mu s - 0.2\mu s))}$$

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.

Example for Bit Timing at Low Baud Rate

In this example, the frequency of APB_CLK is 2 MHz, BRP is 1, the bit rate is 100 Kbit/s.

| | | | |
|---|---|---|---|
| $t_q$ | 1 | s | $= 2 \cdot t_{APB\_CLK}$ |
| delay of bus driver | 200 | ns | |
| delay of receiver circuit | 80 | ns | |
| delay of bus line (40m) | 220 | ns | |
| $t_{Prop}$ | 1 | s | $= 1 \cdot t_q$ |
| $t_{SJW}$ | 4 | s | $= 4 \cdot t_q$ |
| $t_{TSeg1}$ | 5 | s | $= t_{Prop} + t_{SJW}$ |
| $t_{TSeg2}$ | 4 | s | = Information Processing Time + 3 • $t_q$ |
| $t_{Sync\text{-}Seg}$ | 1 | s | $= 1 \cdot t_q$ |
| bit time | 10 | s | $= t_{Sync\text{-}Seg} + t_{TSeg1} + t_{TSeg2}$ |
| 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.

### 5.24.8 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAN Base Address:**<br>**CAN0_BA = 0xB800_0000**<br>**CAN1_BA = 0xB800_4000** | | | | |
| **CAN_CON** | CANx_BA+0x00 | R/W | Control Register | 0x0000_0001 |
| **CAN_STATUS** | CANx_BA+0x04 | R/W | Status Register | 0x0000_0000 |
| **CAN_ERR** | CANx_BA+0x08 | R | Error Counter | 0x0000_0000 |
| **CAN_BTIME** | CANx_BA+0x0C | R/W | Bit Timing Register | 0x0000_2301 |
| **CAN_IIDR** | CANx_BA+0x10 | R | Interrupt Identifier Register | 0x0000_0000 |
| **CAN_TEST** | CANx_BA+0x14 | R/W | Test Register | *(1) |
| **CAN_BRPE** | CANx_BA+0x18 | R/W | BRP Extension Register | 0x0000_0000 |
| **CAN_IF1_CREQ**<br>**CAN_IF2_CREQ** | CANx_BA+0x20<br>CANx_BA+0x80 | R/W | IFn (*2) Command Request Registers | 0x0000_0001 |
| **CAN_IF1_CMASK**<br>**CAN_IF2_CMASK** | CANx_BA+0x24<br>CANx_BA+0x84 | R/W | IFn Command Mask Registers | 0x0000_0000 |
| **CAN_IF1_MASK1**<br>**CAN_IF2_MASK1** | CANx_BA+0x28<br>CANx_BA+0x88 | R/W | IFn Mask 1 Register | 0x0000_FFFF |
| **CAN_IF1_MASK2**<br>**CAN_IF2_MASK2** | CANx_BA+0x2C<br>CANx_BA+0x8C | R/W | IFn Mask 2 Register | 0x0000_FFFF |
| **CAN_IF1_ARB1**<br>**CAN_IF2_ARB1** | CANx_BA+0x30<br>CANx_BA+0x90 | R/W | IFn Arbitration 1 Register | 0x0000_0000 |
| **CAN_IF1_ARB2**<br>**CAN_IF2_ARB2** | CANx_BA+0x34<br>CANx_BA+0x94 | R/W | IFn Arbitration 2 Register | 0x0000_0000 |
| **CAN_IF1_MCON**<br>**CAN_IF2_MCON** | CANx_BA+0x38<br>CANx_BA+0x98 | R/W | IFn Message Control Registers | 0x0000_0000 |
| **CAN_IF1_DAT_An/**<br>**CAN_IF1_DAT_Bn/**<br>**CAN_IF2_DAT_An/**<br>**CAN_IF2_DAT_Bn/** | CANx_BA+0x3C~40<br>CANx_BA+0x44~48<br>CANx_BA+0x9C~A0<br>CANx_BA+0xA4~A8 | R/W | IFn Data An (*3) and Data Bn (*3) Registers<br>eg: CAN_IF1_DAT_A1 = CAN_BA+0x3Ch<br>CAN_IF1_DAT_A2 = CAN_BA+0x40h | 0x0000_0000 |
| **CAN_TXREQ1**<br>**CAN_TXREQ2** | CANx_BA+0x100<br>CANx_BA+0x104 | R | Transmission Request Registers 1 & 2 | 0x0000_0000 |
| **CAN_NDAT1**<br>**CAN_NDAT2** | CANx_BA+0x120<br>CANx_BA+0x124 | R | New Data Registers 1 & 2 | 0x0000_0000 |
| **CAN_IPND1**<br>**CAN_IPND2** | CANx_BA+0x140<br>CANx_BA+0x144 | R | Interrupt Pending Registers 1 & 2 | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **CAN_MVLD1**<br>**CAN_MVLD2** | CANx_BA+0x160<br>CANx_BA+0x164 | R | Message Valid Registers 1 & 2 | 0x0000_0000 |
| **CAN_WU_EN** | CANx_BA+0x168 | R/W | Wake-up Function Enable | 0x0000_0000 |
| **CAN_WU_STATUS** | CANx_BA+0x16C | R/W | Wake-up Function Status | 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. CANx_BA, where x = 0 or 1.

CAN Interface Reset State

After the hardware reset, the C_CAN registers hold the reset values which are given in the register description in 5.24.8

Register Map.

Additionally the busoff 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 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.

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 5.24-4 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

### 5.24.9 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 | CANx_BA+0x00 | R/W | CAN 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 |
| Test | CCE | DAR | Reserved | EIE | SIE | IE | Init |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7] | **Test** | **Test Mode Enable Control**<br>0 = Normal Operation.<br>1 = Test Mode. |
| [6] | **CCE** | **Configuration Change Enable Control**<br>0 = No write access to the Bit Timing Register.<br>1 = Write access to the Bit Timing Register (CAN_BTIME & CAN_BRP) allowed. (while Init bit =1). |
| [5] | **DAR** | **Automatic Re-transmission Disable Control**<br>0 = Automatic Retransmission of disturbed messages enabled.<br>1 = Automatic Retransmission disabled. |
| [4] | **Reserved** | Reserved. |
| [3] | **EIE** | **Error Interrupt Enable Control**<br>0 = Disabled - No Error Status Interrupt will be generated.<br>1 = Enabled - A change in the bits BOff or EWarn in the Status Register will generate an interrupt. |
| [2] | **SIE** | **Status Change Interrupt Enable Control**<br>0 = Disabled - No Status Change Interrupt will be generated.<br>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 Control**<br>0 = Disabled.<br>1 = Enabled. |
| [0] | **Init** | **Init Initialization**<br>0 = Normal Operation. |

| | | |
|---|---|---|
| | | 1 = Initialization is started. |

**Note:** The busoff recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or resetting the Init bit. If the device goes in the busoff 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 busoff 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 busoff recovery sequence.

## CAN Status Register (CAN_STATUS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CAN_STATUS | CANx_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)**<br>0 = The CAN module is not in bus-off state.<br>1 = The CAN module is in bus-off state. |
| [6] | **EWarn** | **Error Warning Status (Read Only)**<br>0 = Both error counters are below the error warning limit of 96.<br>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)**<br>0 = The CAN Core is error active.<br>1 = The CAN Core is in the error passive state as defined in the CAN Specification. |
| [4] | **RxOK** | **Received a Message Successfully**<br>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.<br>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**<br>0 = Since this bit was reset by the CPU, no message has been successfully transmitted. This bit is never reset by the CAN Core.<br>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 (Type of the Last Error to Occur on the CAN Bus)**<br>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. The following table describes the error code. |

| 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 busoff 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 busoff 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 5.24-5 Error Code

Status Interrupts

A Status Interrupt is generated by bits BOff and EWarn (Error Interrupt) or by RxOk, TxOk, and LEC (Status Change Interrupt) assumed that the corresponding enable bits in the CAN Control Register are set. A change of bit EPass 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** | CANx_BA+0x08 | R | 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[6:0]** | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **TEC[7:0]** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15] | **RP** | **Receive Error Passive**<br>0 = The Receive Error Counter is below the error passive level.<br>1 = The Receive Error Counter has reached the error passive level as defined in the CAN Specification. |
| [14:8] | **REC** | **Receive Error Counter**<br>Actual state of the Receive Error Counter. Values between 0 and 127. |
| [7:0] | **TEC** | **Transmit Error Counter**<br>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** | CANx_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**<br>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**<br>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** | **(Re)Synchronization Jump Width**<br>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. |
| [5:0] | **BRP** | **Baud Rate Prescaler**<br>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 and Init in the CAN Control Register are set.

### Interrupt Identify Register (CAN_IIDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CAN_IIDR | CANx_BA+0x10 | R | Interrupt Identifier 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 |
| IntId[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IntId[7:0] | | | | | | | |

| Bits | Description | |
|---|---|---|
| [15:0] | IntId | **Interrupt Identifier (Indicates the Source of the Interrupt)**<br>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 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.<br>The Status Interrupt has the highest priority. Among the message interrupts, the Message Object' s interrupt priority decreases with increasing message number.<br>A message interrupt is cleared by clearing the Message Object's IntPnd bit. 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 5.24-6 Source of Interrupts

## Test Register (CAN_TEST)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CAN_TEST | CANx_BA+0x14 | R/W | Test Register | 0x0000_00x0 |

| 31 | 30 | 29 | 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[1:0] | | LBack | Silent | Basic | Res | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7] | **Rx** | **Monitors the Actual Value of CAN_RX Pin (Read Only)**<br>0 = The CAN bus is dominant (CAN_RX = '0').<br>1 = The CAN bus is recessive (CAN_RX = '1'). |
| [6:5] | **Tx[1:0]** | **Tx[1:0]: Control of CAN_TX Pin**<br>00 = Reset value, CAN_TX is controlled by the CAN Core.<br>01 = Sample Point can be monitored at CAN_TX pin.<br>10 = CAN_TX pin drives a dominant ('0') value.<br>11 = CAN_TX pin drives a recessive ('1') value. |
| [4] | **LBack** | **Loop Back Mode Enable Control**<br>0 = Loop Back Mode is disabled.<br>1 = Loop Back Mode is enabled. |
| [3] | **Silent** | **Silent Mode**<br>0 = Normal operation.<br>1 = The module is in Silent Mode. |
| [2] | **Basic** | **Basic Mode**<br>0 = Basic Mode disabled.<br>1= IF1 Registers used as Tx Buffer, IF2 Registers used as Rx Buffer. |
| [1:0] | **Reserved** | Reserved. |

Reset value: 0000 0000 R000 0000 b (R:current value of RX pin)

Write access to the Test Register is enabled by setting the Test bit in the CAN Control Register. The different test functions may be combined, but Tx[1-0] "00" disturbs message transfer.

## Baud Rate Prescaler Extension REGISTER (CAN_BRPE)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAN_BRPE** | CANx_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** | **BRPE: Baud Rate Prescaler Extension**<br>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. The following table 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 |
|---|---|---|---|
| CANx_BA+0x20 | IF1 Command Request | CANx_BA+0x80 | IF2 Command Request |
| CANx_BA+0x24 | IF1 Command Mask | CANx_BA+0x84 | IF2 Command Mask |
| CANx_BA+0x28 | IF1 Mask 1 | CANx_BA+0x88 | IF2 Mask 1 |
| CANx_BA+0x2C | IF1 Mask 2 | CANx_BA+0x8C | IF2 Mask 2 |
| CANx_BA+0x30 | IF1 Arbitration 1 | CANx_BA+0x90 | IF2 Arbitration 1 |
| CANx_BA+0x34 | IF1 Arbitration 2 | CANx_BA+0x94 | IF2 Arbitration 2 |
| CANx_BA+0x38 | IF1 Message Control | CANx_BA+0x98 | IF2 Message Control |
| CANx_BA+0x3C | IF1 Data A 1 | CANx_BA+0x9C | IF2 Data A 1 |
| CANx_BA+0x40 | IF1 Data A 2 | CANx_BA+0xA0 | IF2 Data A 2 |
| CANx_BA+0x44 | IF1 Data B 1 | CANx_BA+0xA4 | IF2 Data B 1 |
| CANx_BA+0x48 | IF1 Data B 2 | CANx_BA+0xA8 | IF2 Data B 2 |

Table 5.24-7 IF1 and IF2 Message Interface Register

### IFn Command Request Register (CAN_IFn_CREQ)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CAN_IFn_CREQ | CANx_BA+0x20/0x80 | R/W | IFn Command Request 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 |
| Busy | Res | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Res | | Message Number | | | | | |

| Bits | Description | |
|---|---|---|
| [15] | **Busy** | **Busy Flag**<br>0 = Read/write action has finished.<br>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**<br>0x01-0x20: Valid Message Number, the Message Object in the Message RAM is selected for data transfer.<br>0x00: Not a valid Message Number, interpreted as 0x20.<br>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 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 | CANx_BA+0x24/0x84 | R/W | IFn Command 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 |
| WR/RD | Mask | Arb | Control | ClrIntPnd | TxRqst/ NewDat | DAT_A | DAT_B |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7] | **WR/RD** | **Write / Read**<br>0 = Read: Transfer data from the Message Object addressed by the Command Request Register into the selected Message Buffer Registers.<br>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**<br>Direction = Write.<br>0: = Mask bits unchanged.<br>1 = Transfer Identifier Mask + MDir + MXtd to Message Object.<br>Direction = Read.<br>0 = Mask bits unchanged.<br>1 = Transfer Identifier Mask + MDir + MXtd to IFn Message Buffer Register. |
| [5] | **Arb** | **Access Arbitration Bits**<br>Direction = Write.<br>0 = Arbitration bits unchanged.<br>1 = Transfer Identifier + Dir + Xtd + MsgVal to Message Object.<br>Direction = Read.<br>0 = Arbitration bits unchanged.<br>1 = Transfer Identifier + Dir + Xtd + MsgVal to IFn Message Buffer Register. |
| [4] | **Control** | **Control Access Control Bits**<br>Direction = Write.<br>0 = Control Bits unchanged. |

| | | |
|---|---|---|
| | | 1 = Transfer Control Bits to Message Object.<br>Direction = Read.<br>0 = Control Bits unchanged.<br>1 = Transfer Control Bits to IFn Message Buffer Register. |
| [3] | **ClrIntPnd** | **Clear Interrupt Pending Bit**<br>Direction = Write.<br>When writing to a Message Object, this bit is ignored.<br>Direction = Read.<br>0 = IntPnd bit remains unchanged.<br>1 = Clear IntPnd bit in the Message Object. |
| [2] | **TxRqst/NewDat** | Access Transmission Request Bit when Direction = Write.<br>0 = TxRqst bit unchanged.<br>1 = Set TxRqst bit.<br>**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.<br>Access New Data Bit when Direction = Read.<br>0 = NewDat bit remains unchanged.<br>1 = Clear NewDat bit in the Message Object.<br>**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]**<br>Direction = Write.<br>0 = Data Bytes [3:0] unchanged.<br>1 = Transfer Data Bytes [3:0] to Message Object.<br>Direction = Read.<br>0 = Data Bytes [3:0] unchanged.<br>1 = Transfer Data Bytes [3:0] to IFn Message Buffer Register. |
| [0] | **DAT_B** | **Access Data Bytes [7:4]**<br>Direction = Write.<br>0 = Data Bytes [7:4] unchanged.<br>1 = Transfer Data Bytes [7:4] to Message Object.<br>Direction = Read.<br>0 = Data Bytes [7:4] unchanged.<br>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 | CANx_BA+0x28/0x88 | 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[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Msk[7:0] | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **Msk[15:0]** | **Identifier Mask 15-0**<br>0 = The corresponding bit in the identifier of the message object cannot inhibit the match in the acceptance filtering.<br>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 | CANx_BA+0x2C/0x8C | 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[28:24] | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Msk[23:16] | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15] | **MXtd** | **Mask Extended Identifier**<br>0 = The extended identifier bit (IDE) has no effect on the acceptance filtering.<br>1 = The extended identifier bit (IDE) is used for acceptance filtering.<br>**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. For acceptance filtering, only these bits together with mask bits Msk28 to Msk18 are considered. |
| [14] | **MDir** | **Mask Message Direction**<br>0 = The message direction bit (Dir) has no effect on the acceptance filtering.<br>1 = The message direction bit (Dir) is used for acceptance filtering. |
| [13] | **Reserved** | Reserved. |
| [12:0] | **Msk[28:16]** | **Identifier Mask 28-16**<br>0 = The corresponding bit in the identifier of the message object cannot inhibit the match in the acceptance filtering.<br>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 | CANx_BA+0x30/0x90 | R/W | IFn Arbitration 1 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 |
| ID[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ID[7:0] | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **ID[15:0]** | **Message Identifier 15-0**<br>ID28 - ID0, 29-bit Identifier (“Extended Frame”).<br>ID28 - ID18, 11-bit Identifier (“Standard Frame”) |

### IFn Arbitration 2 Register (CAN_IFn_ARB2)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CAN_IFn_ARB2 | CANx_BA+0x34/0x94 | R/W | IFn Arbitration 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 |
| MsgVal | Xtd | Dir | ID[28:24] | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ID[23:16] | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15] | **MsgVal** | **Message Valid**<br>0 = The Message Object is ignored by the Message Handler.<br>1 = The Message Object is configured and should be considered by the Message Handler.<br>**Note:** The application software must reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN Control Register. This bit must also be reset before the identifier Id28-0, the control bits Xtd, Dir, or the Data Length Code DLC3-0 are modified, or if the Messages Object is no longer required. |
| [14] | **Xtd** | **Extended Identifier**<br>0 = The 11-bit (“standard”) Identifier will be used for this Message Object.<br>1 = The 29-bit (“extended”) Identifier will be used for this Message Object. |
| [13] | **Dir** | **Message Direction**<br>0 = Direction is receive.<br>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.<br>1 = Direction is transmit.<br>On TxRqst, the respective Message Object is transmitted as a Data Frame. On reception of a Remote Frame with matching identifier, the TxRqst bit of this Message Object is set (if RmtEn = one). |
| [12:0] | **ID[28:16]** | **Message Identifier 28-16**<br>ID28 - ID0, 29-bit Identifier (“Extended Frame”).<br>ID28 - ID18, 11-bit Identifier (“Standard Frame”) |

### IFn Message Control Register (CAN_IFn_MCON)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CAN_IFn_MCON | CANx_BA+0x38/0x98 | R/W | IFn Message 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 |
| NewDat | MsgLst | IntPnd | UMask | TxIE | RxIE | RmtEn | TxRqst |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EoB | Reserved | | | DLC[3:0] | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15] | **NewDat** | **New Data**<br>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.<br>1 = The Message Handler or the application software has written new data into the data portion of this Message Object. |
| [14] | **MsgLst** | Message Lost (only valid for Message Objects with direction = receive).<br>0 = No message lost since last time this bit was reset by the CPU.<br>1 = The Message Handler stored a new message into this object when NewDat was still set, the CPU has lost a message. |
| [13] | **IntPnd** | **Interrupt Pending**<br>0 = This message object is not the source of an interrupt.<br>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**<br>0 = Mask ignored.<br>1 = Use Mask (Msk28-0, MXtd, and MDir) for acceptance filtering.<br>**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 is set to one. |
| [11] | **TxIE** | **Transmit Interrupt Enable Control**<br>0 = IntPnd will be left unchanged after the successful transmission of a frame.<br>1 = IntPnd will be set after a successful transmission of a frame. |
| [10] | **RxIE** | **Receive Interrupt Enable Control**<br>0 = IntPnd will be left unchanged after a successful reception of a frame.<br>1 = IntPnd will be set after a successful reception of a frame. |
| [9] | **RmtEn** | **Remote Enable Control** |

| | | |
|---|---|---|
| | | 0 = At the reception of a Remote Frame, TxRqst is left unchanged.<br>1 = At the reception of a Remote Frame, TxRqst is set. |
| [8] | **TxRqst** | **Transmit Request**<br>0 = This Message Object is not waiting for transmission.<br>1 = The transmission of this Message Object is requested and is not yet done. |
| [7] | **EoB** | **End of Buffer**<br>0 = Message Object belongs to a FIFO Buffer and is not the last Message Object of that FIFO Buffer.<br>1 = Single Message Object or last Message Object of a FIFO Buffer.<br>**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**<br>0-8: Data Frame has 0-8 data bytes.<br>9-15: Data Frame has 8 data bytes<br>**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.<br>Data 0: 1st data byte of a CAN Data Frame<br>Data 1: 2nd data byte of a CAN Data Frame<br>Data 2: 3rd data byte of a CAN Data Frame<br>Data 3: 4th data byte of a CAN Data Frame<br>Data 4: 5th data byte of a CAN Data Frame<br>Data 5: 6th data byte of a CAN Data Frame<br>Data 6: 7th data byte of a CAN Data Frame<br>Data 7 : 8th data byte of a CAN Data Frame<br>**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 | CANx_BA+0x3C/0x9C | R/W | IFn Data A1 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 |
| 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**<br>2nd data byte of a CAN Data Frame |
| [7:0] | **Data (0)** | **Data Byte 0**<br>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 | CANx_BA+0x40/0xA0 | R/W | IFn Data A2 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 |
| 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**<br>4th data byte of CAN Data Frame |
| [7:0] | **Data (2)** | **Data Byte 2**<br>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 | CANx_BA+0x44/0xA4 | R/W | IFn Data B1 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 |
| 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**<br>6th data byte of CAN Data Frame |
| [7:0] | **Data (4)** | **Data Byte 4**<br>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 | CANx_BA+0x48/0xA8 | R/W | IFn Data B2 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 |
| 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**<br>8th data byte of CAN Data Frame. |
| [7:0] | **Data (6)** | **Data Byte 6**<br>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. The following table 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] | Data0 | Data1 | Data2 | Data3 | Data4 | Data5 | Data6 | Data7 |

Table 5.24-8 Structure of a Message Object in the Message Memory

The Arbitration Registers ID28-0, Xtd, and Dir are used to define the identifier and type of outgoing messages and are used (together with the mask registers Msk28-0, MXtd, and MDir) 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, NewDat, IntPnd, and MsgVal 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** | CANx_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** |
| **TxRqst 16-9** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **TxRqst 8-1** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **TxRqst 16-1** | **Transmission Request Bits 16-1 (of All Message Objects)**<br>0 = This Message Object is not waiting for transmission.<br>1 = The transmission of this Message Object is requested and is not yet done.<br>These bits are read only. |

### Transmission Request Register 2 (CAN_TXREQ2)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAN_TXREQ2** | CANx_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] | **TxRqst 32-17** | **Transmission Request Bits 32-17 (of All Message Objects)**<br>0 = This Message Object is not waiting for transmission.<br>1 = The transmission of this Message Object is requested and is not yet done.<br>These bits are read only. |

### 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** | CANx_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** |
| **NewData 8-1** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **NewData16-1** | **New Data Bits 16-1 (of All Message Objects)**<br>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.<br>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** | CANx_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** |
| **NewData 32-25** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **NewData 24-17** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **NewData 32-17** | **New Data Bits 32-17 (of All Message Objects)**<br>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.<br>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** | CANx_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** |
| **IntPnd 8-1** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **IntPnd16-1** | **Interrupt Pending Bits 16-1 (of All Message Objects)**<br>0 = This message object is not the source of an interrupt.<br>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** | CANx_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** |
| **IntPnd 32-25** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **IntPnd 24-17** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **IntPnd 32-17** | **Interrupt Pending Bits 32-17(of All Message Objects)**<br>0 = This message object is not the source of an interrupt.<br>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 | CANx_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 |
| MsgVal 16- 9 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MsgVal 8-1 | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **MsgVal 16-1** | **Message Valid Bits 16-1 (of All Message Objects) (Read Only)**<br>0 = This Message Object is ignored by the Message Handler.<br>1 = This Message Object is configured and should be considered by the Message Handler.<br>Ex. 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 | CANx_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 |
| MsgVal 32-25 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MsgVal 24-17 | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **MsgVal 32-17** | **Message Valid Bits 32-17 (of All Message Objects) (Read Only)**<br>0 = This Message Object is ignored by the Message Handler.<br>1 = This Message Object is configured and should be considered by the Message Handler.<br>Ex.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 | CANx_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 Control**<br>0 = The wake-up function is disable.<br>1 = The wake-up function is enable.<br>**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 | CANx_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**<br>0 = No wake-up event is occurred.<br>1 = Wake-up event is occurred.<br>**Note:** The bit can be written '0' to clear. |

## 5.25 Flash Memory Interface (FMI)

### 5.25.1 Overview

The Flash Memory Interface (FMI) of this Chip has DMA unit and FMI unit. The DMA unit provides a DMA (Direct Memory Access) function for FMI to exchange data between system memory (ex. SDRAM) and shared buffer (128 bytes), and the FMI unit control the interface of eMMC or NAND flash. The interface controller can support eMMC and NAND-type flash and the FMI is cooperated with DMAC to provide a fast data transfer between system memory and cards.

### 5.25.2 Features

- Support single DMA channel and address in non-word boundary.
- Support hardware Scatter-Gather function.
- Support 128Bytes shared buffer for data exchange between system memory and flash device. (Separate into two 64 bytes ping-pong FIFO).
- Support eMMC Flash device.
- Supports SLC and MLC NAND type Flash.
- Adjustable NAND page sizes. (512B+spare area, 2048B+spare area, 4096B+spare area and 8192B+spare area).
- Support up to 4bit/8bit/12bit/15bit/24bit hardware ECC calculation circuit to protect data communication.
- Support programmable NAND timing cycle.

### 5.25.3 Block Diagram

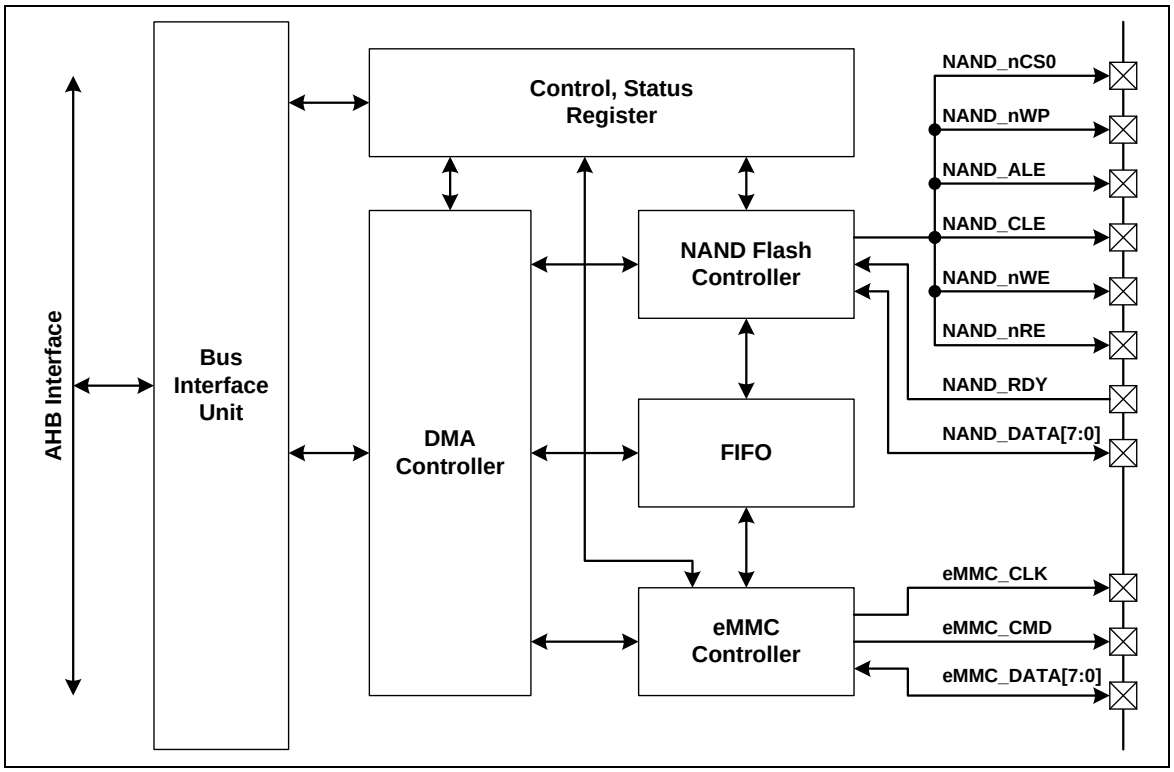



Figure 5.25-1 FMI Block Diagram

### 5.25.4 Basic Configuration

Before using Flash Memory Interface, it's necessary to configure related pins as the NAND/eMMC function and enable FMI's clock.

For NAND/eMMC related pin configuration, please refer to the register SYS_MFP_GPBL, SYS_MFP_GPCL, SYS_MFP_GPCH, SYS_MFP_GPGL, SYS_MFP_GPIL and SYS_MFP_GPIH to know how to configure related pins as the NAND/eMMC function.

Set both FMI (CLK_HCLKEN[20]) and NAND (CLK_HCLKEN[21]) high to enable clock for NAND flash controller operation while set FMI (CLK_HCLKEN[20]), NAND (CLK_HCLKEN[21]) and eMMC (CLK_HCLKEN[22]) high to enable clock for eMMC controller operation.

Here is a simple example programming flow without DMA Scatter-Gather enable.

1. Set DMACEN (FMI_DMACTL[0]) to enable DMAC.
2. Fill corresponding starting address in FMI_DMASA for FMI.
3. Enable IP to start DMA transfer.
4. Wait IP finished, DMAC doesn't need to be took care by software.

Here is a simple example programming flow with DMA Scatter-Gather enable.

1. Set DMACEN (FMI_DMACTL[0]) to enable DMAC and SG_EN (FMI_DMACTL[3]) to enable Scatter-Gather function.
2. Fill corresponding starting address of Physical Address Descriptor (PAD) table in FMI_DMASA for FMI.
3. When bit-0 of FMI_DMASA is 1, the PAD will fetch in out of order, otherwise, it's 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. Enable IP to start DMA transfer.
5. Wait IP finished, DMAC doesn't need to be took care by software.

### 5.25.5 Functional Description

#### 5.25.5.1 *DMA Controller*

The DMA Controller provides a DMA (Direct Memory Access) function for FMI controller to exchange data between system memory (ex. SDRAM) and shared buffer (128 bytes). Arbitration of DMA request between FMI is done by DMAC's bus master. User only simply fills in the starting address and enables DMAC, and then you can let DMAC to handle the data transfer automatically.

There is a 128 bytes shared buffer inside DMAC, separate into two 64 bytes ping-pong FIFO (total 128 bytes). It can provide multi-block transfers using ping-pong mechanism for FMI. When FMI is not busy, these shared buffers can be accessed directly by software.

#### 5.25.5.2 *Flash Memory Interface Controller (FMI)*

The Flash Memory Interface supports eMMC and NAND-type flash. FMI is cooperated with DMAC to provide a fast data transfer between system memory and cards. There is a single 128 bytes buffer embedded in DMAC for temporary data storage (separate into two 64 bytes ping-pong FIFO). Due to DMAC only has single channel, that means only one interface can be active at one time.

#### 5.25.5.3 *eMMC*

FMI provides an interface for eMMC flash device access. This eMMC controller supports 1-bit/4-bit data bus mode for eMMC device.

eMMC controller uses an independent clock source named eMMC_CLK as engine clock. eMMC_CLK can be completely asynchronous with system clock HCLK. In addition, eMMC clock can be changed arbitrarily by software. Note that HCLK should be faster than eMMC_CLK.

This eMMC controller can generate all types of 48-bit command to eMMC device and retrieve all types of response from eMMC device. After response in, the content of response will be stored at FMI_EMMCRESP0 and FMI_EMMCRESP1. eMMC controller will calculate CRC-7 and check its correctness for response. If CRC-7 is error, CRC_IF (FMI_EMMCINTSTS[1]) will be set and CRC7 (FMI_EMMCINTSTS[2]) will be '0'. For response R1b, user should note that after response in, eMMC device will put busy signal on data line DAT0; user has to check this status with clock polling until it became high. For response R3, CRC-7 is invalid; but eMMC controller will still calculate CRC-7 and get an error result, this error has to be ignored by software and clear CRC_IF (FMI_EMMCINTSTS[1]) flag.

This eMMC controller is composed of two state machines – command/response part and data part. For command/response part, the trigger bits are CO_EN (FMI_EMMCCTL[0]), RI_EN (FMI_EMMCCTL[1]), R2_EN (FMI_EMMCCTL[4]), CLK74_OE (FMI_EMMCCTL[5]) and CLK8_OE (FMI_EMMCCTL[6]). If all of these bits enabled by software, the execution priority will be CLK74_OE (FMI_EMMCCTL[5]), CO_EN (FMI_EMMCCTL[0]), RI_EN (FMI_EMMCCTL[1]) or R2_EN (FMI_EMMCCTL[4]), CLK8_OE(FMI_EMMCCTL[6]). Please note that RI_EN (FMI_EMMCCTL[1]) and R2_EN (FMI_EMMCCTL[4]) cannot be triggered at the same time.

For data part, there are DI_EN (FMI_EMMCCTL[2]) and DO_EN (FMI_EMMCCTL[3]) for choose. Please note that DI_EN(FMI_EMMCCTL[2]) and DO_EN (FMI_EMMCCTL[3]) cannot be triggered at the same time. If DI_EN (FMI_EMMCCTL[2]) is triggered, eMMC controller waits start bit from data line DAT0 immediately, and then get specified amount data from eMMC device. After data-in, eMMC controller will check CRC-16 correctness; if it is error, CRC_IF (FMI_EMMCINTSTS[1]) will be set and CRC16 (FMI_EMMCINTSTS[3]) will be '0'. If DO_EN (FMI_EMMCCTL[3]) is triggered, eMMC controller will wait response in finished, and then send specified amount data to eMMC device. After data-out, eMMC controller will get CRC status from eMMC device and check its correctness; it should be '010', otherwise CRC_IF (FMI_EMMCINTSTS[1]) will be set and CRCSTAT (FMI_EMMCINTSTS[6:4]) will be the value it received.

If R2_EN (FMI_EMMCCTL[4]) is triggered, eMMC controller will receive response R2 (136 bits) from eMMC device, CRC-7 and end bit will be dropped. The receiving data will be placed at DMAC's buffer, starting from address offset 0x0.

This eMMC controller also provides multiple block transfer function (change FMI_EMMCBLEN to change the block length). By using this function data transfer throughput accelerated. If CRC-7, CRC-16 or CRC status is error, eMMC controller will stop transfer and set CRC_IF (FMI_EMMCINTSTS[1]). When this situation occurred, it's necessary to set SW_RST (FMI_EMMCCTL [14]) to reset eMMC controller.

There is a hardware time-out mechanism for response in and data in inside of eMMC engine. By specifying a 24-bit time-out value at FMI_EMMCTOUT, eMMC controller will decide when to time-out the transfer.

#### 5.25.5.4 *NAND Flash*

FMI provides an interface for NAND-type Flash access. It supports 512bytes/page, 2048bytes/page, 4096bytes/page and 8192bytes/page NAND. This NAND-type Flash controller provides all required signals for NAND flash, including NAND_RDY, NAND_nCS, NAND_CLE, NAND_ALE, NAND_nWE, NAND_nRE and data pins.

The NAND Flash controller provide direct command port, address port and data port to control NAND flash signals manually. When command port written, NAND flash controller generate appropriate signal to NAND. When address port written without setting EOA (FMI_NANDADDR[31])] high, NAND flash controller generate an address cycle to NAND, but do not clear ALE until the last address cycle written through address port with setting EOA (FMI_NANDADDR[31]) high. By using this method, address cycle can be generated by software arbitrarily.

For example, if user wants to write 4 address cycles to NAND, it necessary to write 3 addresses to

address port without setting EOA (FMI_NANDADDR[31])] high, and then write the last one address to address port with setting EOA (FMI_NANDADDR[31])] high.

NAND flash controller also provides a status and an interrupt flag of NAND_RDY pin. The interrupt flag will be set only when rising edge encountered on NAND_RDY pin.

FMI support four different page size, they are 512bytes/page, 2048bytes/page, 4096bytes/page and 8192bytes/page. Use PSIZE (FMI_NANDCTL[17:16]) to select the NAND flash type. Using DMA function for data transfer could increase the performance. For different model of NAND, it's necessary to adjust the timing parameter at FMI_NANDTMCTL to meet specification of NAND flash device. Adjust timing parameter can also improve data transfer performance.

NAND flash controller equips a BCH algorithm for error recovery. The BCH algorithm can correct up to 4 bits errors, 8 bits errors, 12 bits errors, 15 bits errors or 24 bits errors. By reading ECC_FLD_IF (FMI_NANDINTSTS[2]) to check the error occurrence while by reading FMI_NANDECCES0, FMI_NANDECCES1, FMI_NANDECCES2 and FMI_NANDECCES3 to know how many errors and if those errors are correctable or not. If those errors are correctable, please read FMI_NANDECCEAx and FMI_NANDECCEDx to correct the errors manually.

For 512/2K/4K/8K Page size NAND flash with BCH algorithm, T can be t4, t8, t12, t15 or t24. Based on the page size and T setting, FMI generate different size of parity data. The number of byte for parity data in different page size and T setting listed in the table shown below. The data arrangement of redundant area is as figure shown below.

It's recommended to choose appropriate T based on NAND flash page size and redundant area size.

| BCH Algorithm | Parity (Byte)<br>512 Page Size | Parity (Byte)<br>2048 Page Size | Parity (Byte)<br>4096 Page Size | Parity (Byte)<br>8192 Page Size |
|---|---|---|---|---|
| BCH T4 | 8 | 32 | 64 | 128 |
| BCH T8 | 15 | 60 | 120 | 240 |
| BCH T12 | 23 | 92 | 184 | 368 |
| BCH T15 | 29 | 116 | 232 | 464 |
| BCH T24 | No support | 90 | 180 | 360 |

Table 5.25-1 Number of Parity (Byte) for Each BCH Algorithm

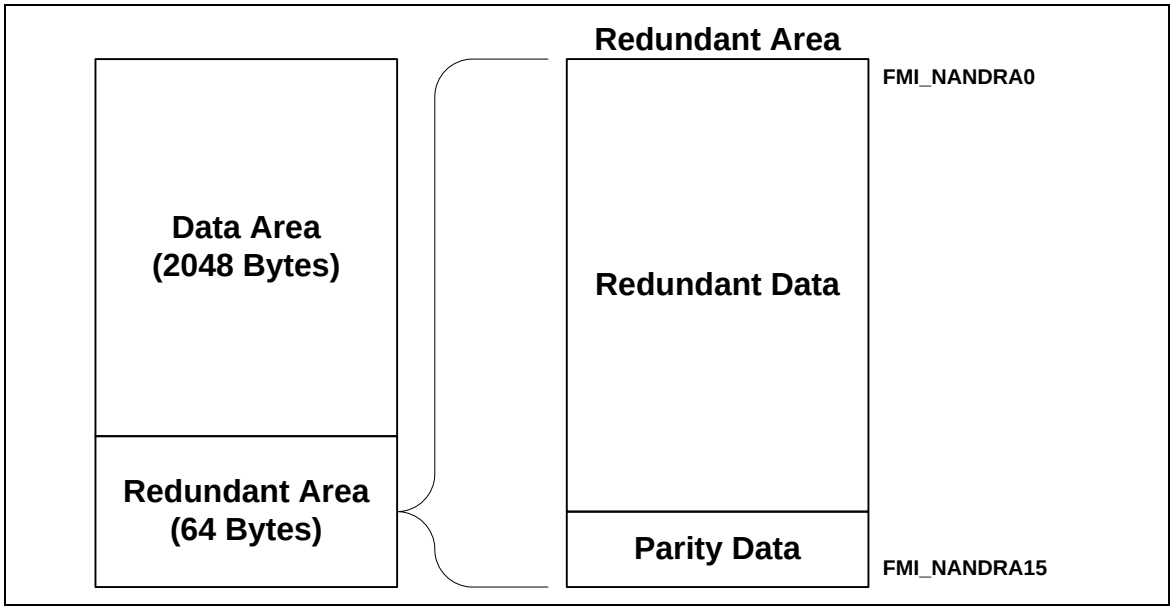



Figure 5.25-2 Data Arrangement for 2 kB Page Size NAND Flash

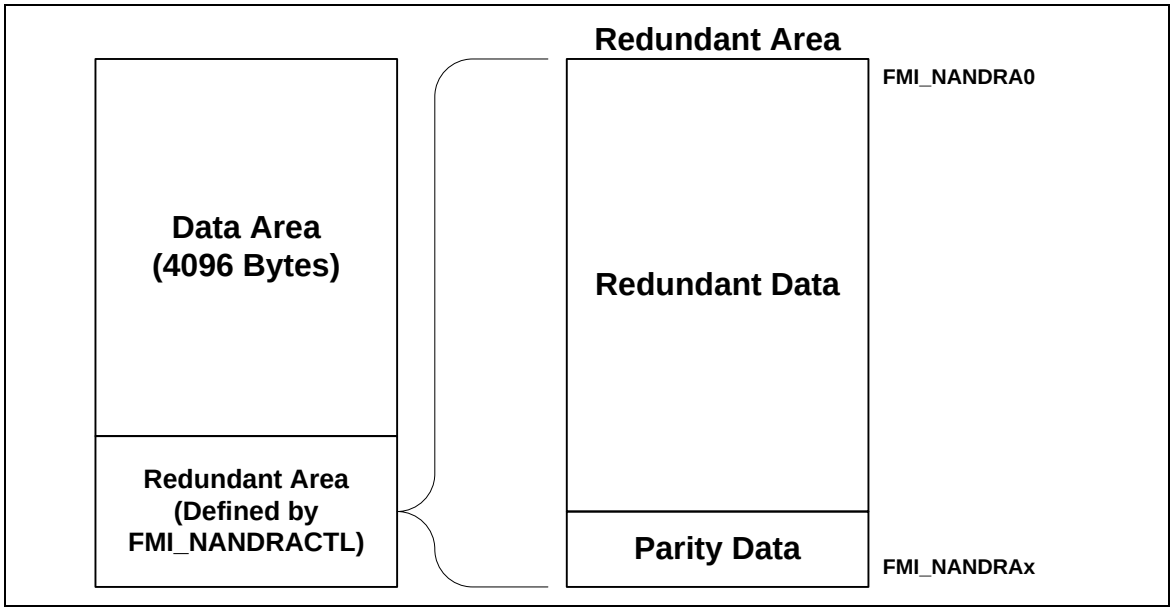



Figure 5.25-3 Data Arrangement for 4 kB Page Size NAND Flash

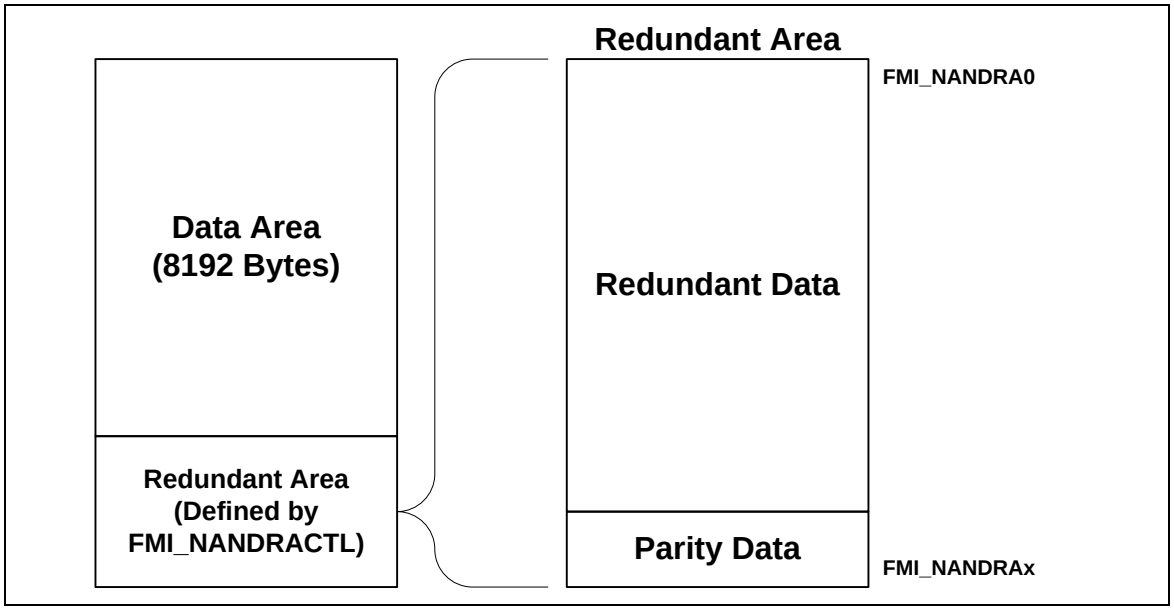



Figure 5.25-4 Data Arrangement for 8 kB Page Size NAND Flash

### 5.25.6 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI Base Address:**<br>**FMI_BA = 0xB000_D000** | | | | |
| **FMI_BUFFERn**<br>**n = 0, 1..31** | FMI_BA+0x000+0x4*n | R/W | FMI Embedded Buffer Word n<br>N = 0, 1..31 | 0x0000_0000 |
| **FMI_DMACTL** | FMI_BA+0x400 | R/W | FMI DMA Control Register | 0x0000_0000 |
| **FMI_DMASA** | FMI_BA+0x408 | R/W | FMI DMA Transfer Starting Address Register | 0x0000_0000 |
| **FMI_DMABCNT** | FMI_BA+0x40C | R | FMI DMA Transfer Byte Count Register | 0x0000_0000 |
| **FMI_DMAINTEN** | FMI_BA+0x410 | R/W | FMI DMA Interrupt Enable Register | 0x0000_0001 |
| **FMI_DMAINTSTS** | FMI_BA+0x414 | R/W | FMI DMA Interrupt Status Register | 0x0000_0000 |
| **FMI_CTL** | FMI_BA+0x800 | R/W | FMI Control Register | 0x0000_0000 |
| **FMI_INTEN** | FMI_BA+0x804 | R/W | FMI Interrupt Enable Register | 0x0000_0001 |
| **FMI_INTSTS** | FMI_BA+0x808 | R/W | FMI Interrupt Status Register | 0x0000_0000 |
| **FMI_EMMCCTL** | FMI_BA+0x820 | R/W | eMMC Control Register | 0x0101_0000 |
| **FMI_EMMCCMD** | FMI_BA+0x824 | R/W | eMMC Command Argument Register | 0x0000_0000 |
| **FMI_EMMCINTEN** | FMI_BA+0x828 | R/W | eMMC Interrupt Enable Register | 0x0000_0000 |
| **FMI_EMMCINTSTS** | FMI_BA+0x82C | R/W | eMMC Interrupt Status Register | 0x00XX_008C |
| **FMI_EMMCRESP0** | FMI_BA+0x830 | R | eMMC Receiving Response Token Register 0 | 0x0000_0000 |
| **FMI_EMMCRESP1** | FMI_BA+0x834 | R | eMMC Receiving Response Token Register 1 | 0x0000_0000 |
| **FMI_EMMCBLEN** | FMI_BA+0x838 | R/W | eMMC Block Length Register | 0x0000_01FF |
| **FMI_EMMCTOUT** | FMI_BA+0x83C | R/W | eMMC Response/Data-in Time-out Register | 0x0000_0000 |
| **FMI_NANDCTL** | FMI_BA+0x8A0 | R/W | NAND Flash Control Register | 0x1E88_0090 |
| **FMI_NANDTMCTL** | FMI_BA+0x8A4 | R/W | NAND Flash Timing Control Register | 0x0001_0105 |
| **FMI_NANDINTEN** | FMI_BA+0x8A8 | R/W | NAND Flash Interrupt Enable Register | 0x0000_0000 |
| **FMI_NANDINTSTS** | FMI_BA+0x8AC | R/W | NAND Flash Interrupt Status Register | 0x00XX_0000 |
| **FMI_NANDCMD** | FMI_BA+0x8B0 | W | NAND Flash Command Port Register | 0xXXXX_XXXX |
| **FMI_NANDADDR** | FMI_BA+0x8B4 | W | NAND Flash Address Port Register | 0xXXXX_XXXX |
| **FMI_NANDDATA** | FMI_BA+0x8B8 | R/W | NAND Flash Data Port Register | 0xXXXX_XXXX |
| **FMI_NANDRACTL** | FMI_BA+0x8BC | R/W | NAND Flash Redundant Area Control Register | 0x0000_0000 |
| **FMI_NANDECTL** | FMI_BA+0x8C0 | R/W | NAND Flash Extend Control Register | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **FMI_NANDECCES0** | FMI_BA+0x8D0 | R | NAND Flash ECC Error Status 0 Register | 0x0000_0000 |
| **FMI_NANDECCES1** | FMI_BA+0x8D4 | R | NAND Flash ECC Error Status 1 Register | 0x0000_0000 |
| **FMI_NANDECCES2** | FMI_BA+0x8D8 | R | NAND Flash ECC Error Status 2 Register | 0x0000_0000 |
| **FMI_NANDECCES3** | FMI_BA+0x8DC | R | NAND Flash ECC Error Status 3 Register | 0x0000_0000 |
| **FMI_NANDPROTA0** | FMI_BA+0x8E0 | R/W | NAND Flash Protect Region End Address 0 Register | 0x0000_0000 |
| **FMI_NANDPROTA1** | FMI_BA+0x8E4 | R/W | NAND Flash Protect Region End Address 1 Register | 0x0000_0000 |
| **FMI_NANDECCEA0** | FMI_BA+0x900 | R | NAND Flash ECC Error Byte Address 0 Register | 0x0000_0000 |
| **FMI_NANDECCEA1** | FMI_BA+0x904 | R | NAND Flash ECC Error Byte Address 1 Register | 0x0000_0000 |
| **FMI_NANDECCEA2** | FMI_BA+0x908 | R | NAND Flash ECC Error Byte Address 2 Register | 0x0000_0000 |
| **FMI_NANDECCEA3** | FMI_BA+0x90C | R | NAND Flash ECC Error Byte Address 3 Register | 0x0000_0000 |
| **FMI_NANDECCEA4** | FMI_BA+0x910 | R | NAND Flash ECC Error Byte Address 4 Register | 0x0000_0000 |
| **FMI_NANDECCEA5** | FMI_BA+0x914 | R | NAND Flash ECC Error Byte Address 5 Register | 0x0000_0000 |
| **FMI_NANDECCEA6** | FMI_BA+0x918 | R | NAND Flash ECC Error Byte Address 6 Register | 0x0000_0000 |
| **FMI_NANDECCEA7** | FMI_BA+0x91C | R | NAND Flash ECC Error Byte Address 7 Register | 0x0000_0000 |
| **FMI_NANDECCEA8** | FMI_BA+0x920 | R | NAND Flash ECC Error Byte Address 8 Register | 0x0000_0000 |
| **FMI_NANDECCEA9** | FMI_BA+0x924 | R | NAND Flash ECC Error Byte Address 9 Register | 0x0000_0000 |
| **FMI_NANDECCEA10** | FMI_BA+0x928 | R | NAND Flash ECC Error Byte Address 10 Register | 0x0000_0000 |
| **FMI_NANDECCEA11** | FMI_BA+0x92C | R | NAND Flash ECC Error Byte Address 11 Register | 0x0000_0000 |
| **FMI_NANDECCED0** | FMI_BA+0x960 | R | NAND Flash ECC Error Data Register 0 | 0x8080_8080 |
| **FMI_NANDECCED1** | FMI_BA+0x964 | R | NAND Flash ECC Error Data Register 1 | 0x8080_8080 |
| **FMI_NANDECCED2** | FMI_BA+0x968 | R | NAND Flash ECC Error Data Register 2 | 0x8080_8080 |
| **FMI_NANDECCED3** | FMI_BA+0x96C | R | NAND Flash ECC Error Data Register 3 | 0x8080_8080 |
| **FMI_NANDECCED4** | FMI_BA+0x970 | R | NAND Flash ECC Error Data Register 4 | 0x8080_8080 |

| | | | | |
|---|---|---|---|---|
| **FMI_NANDECCED5** | FMI_BA+0x974 | R | NAND Flash ECC Error Data Register 5 | 0x8080_8080 |
| **FMI_NANDRAn**<br>**n = 0, 1..117** | FMI_BA+0xA00+0x4*n | R/W | NAND Flash Redundant Area Word n<br>n = 0, 1..117 | Undefined |

### 5.25.7 Register Description

### FMI DMA Control Register (FMI_DMACTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_DMACTL** | FMI_BA+0x400 | R/W | FMI DMA 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** | | | | | | **FMI_BUSY** | **Reserved** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | | **SG_EN** | **Reserved** | **SW_RST** | **DMACEN** |

| Bits | Description | |
|---|---|---|
| [31:10] | **Reserved** | Reserved. |
| [9] | **FMI_BUSY** | **FMI DMA Transfer Is in Progress**<br>This bit indicates if FMI is granted and doing DMA transfer or not.<br>0 = FMI DMA transfer is not in progress.<br>1 = FMI DMA transfer is in progress. |
| [8:4] | **Reserved** | Reserved. |
| [3] | **SG_EN** | **Enable Scatter-gather Function for FMI**<br>Enable DMA scatter-gather function or not.<br>0 = Normal operation. DMAC will treat the starting address in FMI_DMASA as starting pointer of a single block memory.<br>1 = Enable scatter-gather operation. DMAC will treat the starting address in FMI_DMASA as a starting address of Physical Address Descriptor (PAD) table. The format of these Pads' will be described later. |
| [2] | **Reserved** | Reserved. |
| [1] | **SW_RST** | **Software Engine Reset**<br>0 = Writing 0 to this bit has no effect.<br>1 = Writing 1 to this bit will reset the internal state machine and pointers. The contents of control register will not be cleared. This bit will auto clear after few clock cycles.<br>**NOTE:** The software reset DMA region. |
| [0] | **DMACEN** | **DMAC Engine Enable**<br>Setting this bit to 1 enables DMAC's operation. If this bit is cleared, DMAC will ignore all DMA request from FMI and force Bus Master into IDLE state.<br>0 = Disable DMAC.<br>1 = Enable DMAC.<br>**NOTE:** If target abort is occurred, DMACEN will be cleared. |

### FMI DMA Transfer Starting Address Register (FMI_DMASA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_DMASA** | FMI_BA+0x408 | R/W | FMI DMA Transfer Starting Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **ADDR** | **DMA Transfer Starting Address**<br>This field indicates a 32-bit starting address of system memory (SRAM/SDRAM) for DMAC to retrieve or fill in data (for FMI engine).<br>If DMAC is not in normal mode, this field will be interpreted as a starting address of Physical Address Descriptor (PAD) table. |
| [0] | **ORDER** | **Determined to the PAD Table Fetching Is in Order or Out of Order**<br>0 = PAD table is fetched in order.<br>1 = PAD table is fetched out of order.<br>**Note:** the bit0 is valid in scatter-gather mode when SG_EN = 1. |

**NOTE:** Starting address of the SDRAM must be word aligned, for example, 0x0000_0000, 0x0000_0004, etc.

The diagram shown below describes the format of PAD table. Note that the total byte count of all Pads must be equal to the byte count filled in FMI engine.

The EOT is the End of PAD Table. The EOT should be set to 1 in the last descriptor.

The byte count has to be the multiple of 4 bytes.

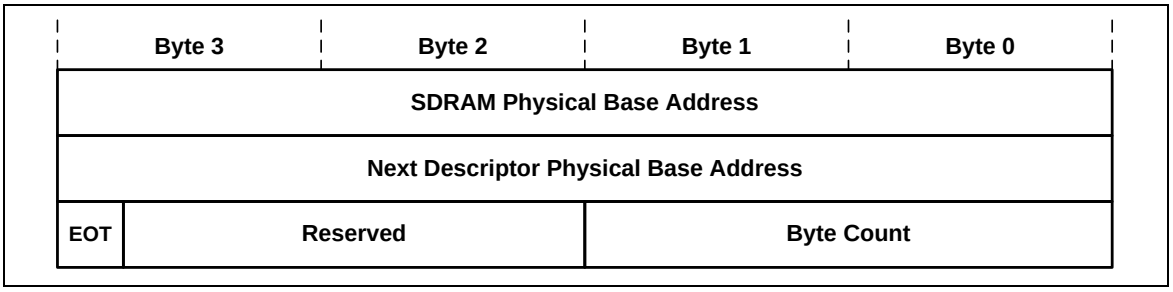



Figure 5.25-5 PAD (Physical Address Descriptor) Table Format

The diagram shown below indicates how FMI feteched the PAD tables. FMI feteched next PAD tables sequentially if ORDER (FMI_DMASA[0]) set as low. FMI fetehced next PAD tables based on the Next Descriptor Physical Base Address of PAD table if ORDER (FMI_DMASA[0]) set as high.

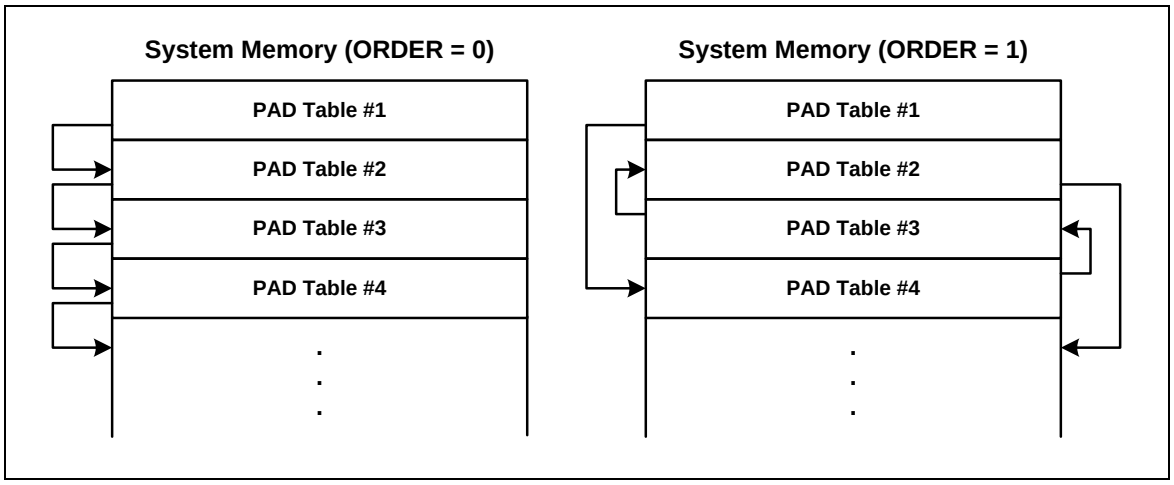



Figure 5.25-6 PAD (Physical Address Descriptor) Table Fetch Modes

## FMI DMA Transfer Byte Count Register (FMI_DMABCNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_DMABCNT** | FMI_BA+0x40C | R | FMI 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)**<br>This field indicates the remained byte count of DMAC transfer. The value of this field is valid only when FMI is busy; otherwise, it is zero. |

## FMI DMA Interrupt Enable Register (FMI_DMAINTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_DMAINTEN** | FMI_BA+0x410 | R/W | FMI DMA Interrupt Enable 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** | | | | | | **WEOT_IE** | **TABORT_IE** |

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [1] | **WEOT_IE** | **Wrong EOT Encountered Interrupt Enable**<br>0 = Disable interrupt generation when wrong EOT is encountered.<br>1 = Enable interrupt generation when wrong EOT is encountered. |
| [0] | **TABORT_IE** | **DMA Read/Write Target Abort Interrupt Enable**<br>0 = Disable target abort interrupt generation during DMA transfer.<br>1 = Enable target abort interrupt generation during DMA transfer. |

## FMI DMA Interrupt Status Register (FMI_DMAINTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_DMAINTSTS** | FMI_BA+0x414 | R/W | FMI 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** | | | | | | **WEOT_IF** | **TABORT_IF** |

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [1] | **WEOT_IF** | **Wrong EOT Encountered Interrupt Flag**<br>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 FMI), this bit will be set.<br>0 = No EOT encountered before DMA transfer finished.<br>1 = EOT encountered before DMA transfer finished.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [0] | **TABORT_IF** | **DMA Read/Write Target Abort Interrupt Flag**<br>0 = No bus ERROR response received.<br>1 = Bus ERROR response received.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |

**NOTE:** When DMAC's bus master received ERROR response, it means that target abort happened. DMAC will stop transfer and respond this event by set TABORT_IF high. Then, FMI go to IDLE state. When target abort occurred or WEOT_IF is set, it's necessary to reset FMI's DMA and related function (eMMC or NAND flash controller), and then transfer those data again.

### FMI Control Register (FMI_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_CTL** | FMI_BA+0x800 | R/W | FMI 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** | | | | **NAND_EN** | **Reserved** | **eMMC_EN** | **SW_RST** |

| Bits | Description | |
|---|---|---|
| [31:4] | **Reserved** | Reserved. |
| [3] | **NAND_EN** | **NAND Flash Functionality Enable**<br>0 = Disable NAND flash functionality of FMI.<br>1 = Enable NAND flash functionality of FMI. |
| [2] | **Reserved** | Reserved. |
| [1] | **eMMC_EN** | **eMMC Functionality Enable**<br>0 = Disable eMMC functionality of FMI.<br>1 = Enable eMMC functionality of FMI. |
| [0] | **SW_RST** | **Software Engine Reset**<br>0 = Writing 0 to this bit has no effect.<br>1 = Writing 1 to this bit will reset all FMI engines. The contents of control register will not be cleared. This bit will auto clear after few clock cycles. |

**NOTE:** Only one engine can be enabled at one time, or FMI will work abnormal.

### FMI Interrupt Enable Register (FMI_INTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_INTEN** | FMI_BA+0x804 | R/W | FMI Interrupt Enable 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** | | | | | | | **DTA_IE** |

| Bits | Description | |
|---|---|---|
| [31:1] | **Reserved** | Reserved. |
| [0] | **DTA_IE** | **DMAC READ/WRITE Target Abort Interrupt Enable**<br>0 = Disable DMAC READ/WRITE target abort interrupt generation.<br>1 = Enable DMAC READ/WRITE target abort interrupt generation. |

### FMI Interrupt Status Register (FMI_INTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_INTSTS** | FMI_BA+0x808 | R/W | FMI 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** | | | | | | | **DTA_IF** |

| Bits | Description | |
|---|---|---|
| [31:1] | **Reserved** | Reserved. |
| [0] | **DTA_IF** | **DMAC READ/WRITE Target Abort Interrupt Flag (Read Only)**<br>This bit indicates DMAC received an ERROR response from internal AHB bus during DMA read/write operation. When Target Abort is occurred, please reset all engine.<br>0 = No bus ERROR response received.<br>1 = Bus ERROR response received.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |

**NOTE:** No matter interrupt enable is turn on or not, the interrupt flag will be set when target condition is occurred.

## eMMC Control Register (FMI_EMMCCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_EMMCCTL** | FMI_BA+0x820 | R/W | eMMC Control Register | 0x0101_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | **NWR** | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **BLK_CNT** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **DBW** | **SW_RST** | **CMD_CODE** | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | **CLK8_OE** | **CLK74_OE** | **R2_EN** | **DO_EN** | **DI_EN** | **RI_EN** | **CO_EN** |

| Bits | Description | |
|---|---|---|
| [31] | **Reserved** | Reserved. |
| [30:29] | **Reserved** | Reserved. |
| [28] | **Reserved** | Reserved. |
| [27:24] | **NWR** | **NWR Parameter for Block Write Operation**<br>This value indicates the NWR parameter for data block write operation in eMMC clock counts. The actual clock cycle will beNWR+1. |
| [23:16] | **BLK_CNT** | **Block Counts to Be Transferred or Received**<br>This field contains the block counts for data-in and data-out transfer. For READ_MULTIPLE_BLOCK and WRITE_MULTIPLE_BLOCK command, using this function to accelerate data transfer and improve performance. Don't fill 0x0 to this field.<br>**Note:** For READ_MULTIPLE_BLOCK and WRITE_MULTIPLE_BLOCK command, the actual total length is BLK_CNT * (BLK_LENGTH (FMI_EMMCBLEN[10:0]) +1). |
| [15] | **DBW** | **eMMC Data Bus Width (for 1-bit / 4-bit Selection)**<br>0 = Data bus width is 1-bit.<br>1 = Data bus width is 4-bit. |
| [14] | **SW_RST** | **Software Engine Reset**<br>0 = Writing 0 to this bit has no effect.<br>1 = Writing 1 to this bit will reset the internal state machine and counters. The contents of control register will not be cleared (but RI_EN, DI_EN, DO_EN and R2_EN will be cleared). This bit will be auto cleared after few clock cycles. |
| [13:8] | **CMD_CODE** | **eMMC Command Code**<br>This register contains the eMMC command code (0x00 – 0x3F). |
| [7] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [6] | **CLK8_OE** | **Generating 8 Clock Cycles Output Enable**<br>0 = No effect. (Please use SW_RST(FMI_EMMCCTL[14]) to clear this bit.)<br>1 = Enable, eMMC host will output 8 clock cycles.<br>**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] | **CLK74_OE** | **Initial 74 Clock Cycles Output Enable**<br>0 = No effect. (Please use SW_RST(FMI_EMMCCTL[14]) to clear this bit.)<br>1 = Enable, eMMC host will output 74 clock cycles to eMMC device.<br>**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] | **R2_EN** | **Response R2 Input Enable**<br>0 = No effect. (Please use SW_RST(FMI_EMMCCTL[14]) to clear this bit.)<br>1 = Enable, eMMC host will wait to receive a response R2 from eMMC device and store the response data into DMAC's flash buffer (exclude CRC-7).<br>**NOTE:** When operation is finished, this bit will be cleared automatically, so don't write 0 to this bit (the controller will be abnormal). |
| [3] | **DO_EN** | **Data Output Enable**<br>0 = No effect. (Please use SW_RST(FMI_EMMCCTL[14]) to clear this bit.)<br>1 = Enable, eMMC host will transfer block data and the CRC-16 value to eMMC device.<br>**NOTE:** When operation is finished, this bit will be cleared automatically, so don't write 0 to this bit (the controller will be abnormal). |
| [2] | **DI_EN** | **Data Input Enable**<br>0 = No effect. (Please use SW_RST(FMI_EMMCCTL[14]) to clear this bit.)<br>1 = Enable, eMMC host will wait to receive block data and the CRC-16 value from eMMC device.<br>**NOTE:** When operation is finished, this bit will be cleared automatically, so don't write 0 to this bit (the controller will be abnormal). |
| [1] | **RI_EN** | **Response Input Enable**<br>0 = No effect. (Please use SW_RST(FMI_EMMCCTL[14]) to clear this bit.)<br>1 = Enable, eMMC host will wait to receive a response from eMMC device.<br>**NOTE:** When operation is finished, this bit will be cleared automatically, so don't write 0 to this bit (the controller will be abnormal). |
| [0] | **CO_EN** | **Command Output Enable**<br>0 = No effect. (Please use SW_RST(FMI_EMMCCTL[14]) to clear this bit.)<br>1 = Enable, eMMC host will output a command to eMMC device.<br>**NOTE:** When operation is finished, this bit will be cleared automatically, so don't write 0 to this bit (the controller will be abnormal). |

### eMMC Command Argument Register (FMI_EMMCCMD)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| FMI_EMMCCMD | FMI_BA+0x824 | R/W | eMMC Command Argument Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | CMDARG | **eMMC Command Argument**<br>This register contains a 32-bit value specifies the argument of eMMC command from host controller to eMMC device. Before trigger CO_EN (FMI_EMMCCMD[0]), it's necessary to write argument in this field. |

## eMMC Interrupt Enable Register (FMI_EMMCINTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| FMI_EMMCINTEN | FMI_BA+0x828 | R/W | eMMC 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 | | DITO_IE | RITO_IE | Reserved | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | CRC_IE | BLKD_IE |

| Bits | Description | |
|---|---|---|
| [31:14] | **Reserved** | Reserved. |
| [13] | **DITO_IE** | **Data Input Time-out Interrupt Enable**<br>Enable/Disable interrupts generation of eMMC controller when data input time-out. Time-out value is specified at FMI_EMMCTMOUT.<br>0 = Disable.<br>1 = Enable. |
| [12] | **RITO_IE** | **Response Time-out Interrupt Enable**<br>Enable/Disable interrupts generation of eMMC controller when receiving response or R2 time-out. Time-out value is specified at FMI_EMMCTMOUT.<br>0 = Disable.<br>1 = Enable. |
| [11:2] | **Reserved** | Reserved. |
| [1] | **CRC_IE** | **CRC-7, CRC-16 and CRC Status Error Interrupt Enable**<br>0 = eMMC host will not generate interrupt when CRC-7, CRC-16 and CRC status is error.<br>1 = eMMC host will generate interrupt when CRC-7, CRC-16 and CRC status is error. |
| [0] | **BLKD_IE** | **Block Transfer Done Interrupt Enable**<br>0 = eMMC host will not generate interrupt when data-in (out) transfer done.<br>1 = eMMC host will generate interrupt when data-in (out) transfer done. |

## eMMC Interrupt Status Register (FMI_EMMCINTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| FMI_EMMCINTSTS | FMI_BA+0x82C | R/W | eMMC Interrupt Status Register | 0x00XX_008C |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | DITO_IF | RITO_IF | Reserved | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DAT0 | CRCSTAT | | | CRC16 | CRC7 | CRC_IF | BLKD_IF |

| Bits | Description | |
|---|---|---|
| [31:14] | **Reserved** | Reserved. |
| [13] | **DITO_IF** | **Data Input Time-out Interrupt Flag (Read Only)**<br>This bit indicates that eMMC host counts to time-out value when receiving data (waiting start bit).<br>0 = Not time-out.<br>1 = Data input time-out.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [12] | **RITO_IF** | **Response Time-out Interrupt Flag (Read Only)**<br>This bit indicates that eMMC host counts to time-out value when receiving response or R2 (waiting start bit).<br>0 = Not time-out.<br>1 = Response time-out.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [11:8] | **Reserved** | Reserved. |
| [7] | **DAT0** | **DAT0 Pin Status of Current Selected EMMC Port (Read Only)**<br>This bit is the DAT0 pin status of current selected eMMC port. |
| [6:4] | **CRCSTAT** | **CRC Status Value of Data-out Transfer (Read Only)**<br>eMMC host will record CRC status of data-out transfer. Using this value to identify what type of error is during data-out transfer.<br>010 = Positive CRC status.<br>101 = Negative CRC status.<br>111 = eMMC device programming error occurs. |

| | | |
|---|---|---|
| [3] | **CRC16** | **CRC-16 Check Status of Data-in Transfer (Read Only)**<br>eMMC host will check CRC-16 correctness after data-in transfer.<br>0 = Fault.<br>1 = OK. |
| [2] | **CRC7** | **CRC-7 Check Status (Read Only)**<br>eMMC host will check CRC-7 correctness during each response in. If that response does not contain CRC-7 information (ex. R3), please disable CRC_IE (FMI_EMMCINTEN[1]) and ignore this bit.<br>0 = Fault.<br>1 = OK. |
| [1] | **CRC_IF** | **CRC-7, CRC-16 and CRC Status Error Interrupt Flag (Read Only)**<br>This bit indicates that eMMC host has occurred CRC error during response in, data-in or data-out (CRC status error) transfer. When CRC error is occurred, it's necessary to reset eMMC engine.<br>Some response (ex. R3) doesn't have CRC-7 information with it. However, eMMC host will still calculate CRC-7, get CRC error and set this flag. In this condition, please ignore CRC error and clears this bit manually.<br>0 = No CRC error is occurred.<br>1 = CRC error is occurred.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [0] | **BLKD_IF** | **Block Transfer Done Interrupt Flag (Read Only)**<br>This bit indicates that eMMC host has finished all data-in or data-out block transfer. If there is a CRC-16 error or incorrect CRC status during multiple block data transfer, the transfer will be broken and this bit will also be set.<br>0 = Not finished yet.<br>1 = Done.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |

### eMMC Receiving Response Token Register 0 (FMI_EMMCRESP0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_EMMCRESP0** | FMI_BA+0x830 | R | eMMC Receiving Response Token Register 0 | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **RESPONSE** | **eMMC Receiving Response Token 0**<br>eMMC host controller will receive a response token for getting a reply from eMMC device when RI_EN (FMI_EMMCCTL[1]) is set. This field contains response bit 47-16 of the response token. |

### eMMC Receiving Response Token Register 1 (FMI_EMMCRESP1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_EMMCRESP1** | FMI_BA+0x834 | R | eMMC 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** |
| **RESPONSE** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **RESPONSE** | **eMMC Receiving Response Token 1**<br>eMMC host controller will receive a response token for getting a reply from eMMC device when RI_EN (FMI_EMMCCTL[1]) is set. This register contains the bit 15-8 of the response token. |

### eMMC Block Length Register (FMI_EMMCBLEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| FMI_EMMCBLEN | FMI_BA+0x838 | R/W | eMMC 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 | | | | | BLK_LENGTH | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BLK_LENGTH | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:11] | **Reserved** | Reserved. |
| [10:0] | **BLK_LENGTH** | **eMMC Block Length in Byte Unit**<br>An 11-bit value specifies the eMMC transfer byte count of a block. The actual byte count is equal to BLK_LENGTH+1.<br>Note : The default eMMC block length is 512 bytes |

## eMMC Response/Data-in Time-out Register (FMI_EMMCTOUT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| FMI_EMMCTOUT | FMI_BA+0x83C | R/W | eMMC 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 |
| TIMEOUT | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TIMEOUT | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TIMEOUT | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **TIMEOUT** | **eMMC Response/Data-in Time-out Value**<br>A 24-bit value specifies the time-out counts of response and data input. eMMC host controller will wait start bit of response or data-in until this value reached. The time period is depended on eMMC engine clock frequency. Do not write a small number into this field, or you may never get response or data due to time-out.<br>**NOTE:** Fill 0x0 into this field will disable hardware time-out function. |

## NAND Flash Control Register (FMI_NANDCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| FMI_NANDCTL | FMI_BA+0x8A0 | R/W | NAND Flash Control Register | 0x1E88_0090 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | CS1 | CS0 | Reserved |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ECC_EN | BCH_TSEL | | | | | PSIZE | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | SRAM_INT | PROT_3BEN |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ECC_CHK | Reserved | PROT_REGION_EN | REDUN_AUTO_WEN | REDUN_REN | DWR_EN | DRD_EN | SW_RST |

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26] | **CS1** | **NAND Flash Chip Select 1 Enable**<br>0 = Chip select 1 enable.<br>1 = Chip select 1 disable. |
| [25] | **CS0** | **NAND Flash Chip Select 0 Enable**<br>0 = Chip select 0 enable.<br>1 = Chip select 0 disable. |
| [24] | **Reserved** | Reserved. |
| [23] | **ECC_EN** | **ECC Algorithm Enable**<br>This field is used to select the ECC algorithm for data protecting. The BCH algorithm can correct 4 or 8 or 12 or 15 or 24 bits.<br>0 = Disable BCH code encode/decode.<br>1 = Enable BCH code encode/decode.<br>**Note:** If disable ECC_EN and when read data from NAND, NAND controller will ignore its ECC check result. When write data to NAND, NAND controller will write out 0xFF to every parity field.<br>**Note:** The ECC algorithm only protects data area and hardware ECC parity code in default. By setting PROT_3BEN (FMI_NANDCTL[8]) high, the first 3 bytes of redundant data are also protected by ECC algorithm. |

| | | |
|---|---|---|
| [22:18] | **BCH_TSEL** | **BCH Correct Bit Selection**<br>This field is used to select BCH correct bits for data protecting. For BCH algorithm, T can be 4 or 8 or 12 or 15 or 24 for choosing (correct 4 or 8 or 12 or 15 or 24 bits).<br>00001 = Using BCH T24 to encode/decode (T24).(1024 Bytes per block)<br>00010 = Using BCH T4 to encode/decode (T4).<br>00100 = Using BCH T8 to encode/decode (T8).<br>01000 = Using BCH T12 to encode/decode (T12).<br>10000 = Using BCH T15 to encode/decode (T15). |
| [17:16] | **PSIZE** | **Page Size of NAND**<br>This bit indicates the page size of NAND. There are four page sizes for choose, 512bytes/page, 2048bytes/page, 4096bytes/page and 8192bytes/page. Before setting PSIZE register, user must set BCH_TSEL register at first.<br>00 = Page size is 512bytes/page.<br>01 = Page size is 2048bytes/page.<br>10 = Page size is 4096bytes/page.<br>11 = Page size is 8192bytes/page. |
| [15:10] | **Reserved** | Reserved. |
| [9] | **SRAM_INT** | **SRAM Initial**<br>0 = Writing 0 to this bit has no effect.<br>1 = Writing 1 to this bit will reset the internal FMI_NANDRA0~FMI_NANDRA1 to 0xFFFF_FFFF.<br>The contents of control register will not be cleared. This bit will be auto cleared after few clock cycles. |
| [8] | **PROT_3BEN** | **Protect_3Byte Software Data Enable**<br>The ECC algorithm only protects data area and hardware ECC parity code. User can choose to protect software redundant data first 3 bytes by setting this bit high.<br>0 = Software redundant data is not protected by ECC algorithm.<br>1 = Software redundant data first 3 bytes protected by ECC algorithm. |
| [7] | **ECC_CHK** | **None Used Field ECC Check After Read Page Data**<br>0 = Disable. NAND controller will always check ECC result for each field, no matter it is used or not.<br>1 = Enable. NAND controller will check 1's count for byte 2, 3 of redundant data of the ECC in each field. If count value is greater than 8, NAND controller will treat this field as none used field; otherwise, it's used. If that field is none used field, NAND controller will ignore its ECC check result. |
| [6] | **Reserved** | Reserved. |
| [5] | **PROT_REGION_EN** | **Protect Region Enable**<br>This field is used to protect NAND Flash region from address 0 to address {FMI_NANDPRTOA1, FMI_NANDPROTA0} not be written.<br>0 = Disable.<br>1 = Enable. |
| [4] | **REDUN_AUTO_WEN** | **Redundant Area Auto Write Enable**<br>This field is used to auto write redundant data out to NAND Flash. The redundant data area is dependent on FMI_NANDRACTL register.<br>0 = Disable auto write redundant data out to NAND flash.<br>1 = Enable auto write redundant data out to NAND flash. |

| | | |
|---|---|---|
| [3] | **REDUN_REN** | **Redundant Area Read Enable**<br>This bit enables NAND controller to transfer redundant data from NAND Flash into FMI_NANDRA, the data size is dependent on FMI_NANDRACTL register.<br>0 = No effect.<br>1 = Enable read redundant data transfer.<br>**NOTE:** When transfer completed, this bit will be cleared automatically. |
| [2] | **DWR_EN** | **DMA Write Data Enable**<br>This bit enables NAND controller to transfer data (1 page) from DMAC's embedded frame buffer into NAND Flash or NAND type flash.<br>0 = No effect.<br>1 = Enable DMA write data transfer.<br>**NOTE:** When DMA transfer completed, this bit will be cleared automatically. |
| [1] | **DRD_EN** | **DMA Read Data Enable**<br>This bit enables NAND controller to transfer data (1 page) from NAND Flash or NAND type flash into DMAC's embedded frame buffer.<br>0 = No effect.<br>1 = Enable DMA read data transfer.<br>**NOTE:** When DMA transfer completed, this bit will be cleared automatically. |
| [0] | **SW_RST** | **Software Engine Reset**<br>0 = Writing 0 to this bit has no effect.<br>1 = Writing 1 to this bit will reset the internal state machine and counters (include DWR_EN (FMI_NANDCTL[2]) and DRD_EN (FMI_NANDCTL[1])). The contents of control register will not be cleared. This bit will be auto cleared after few clock cycles. |

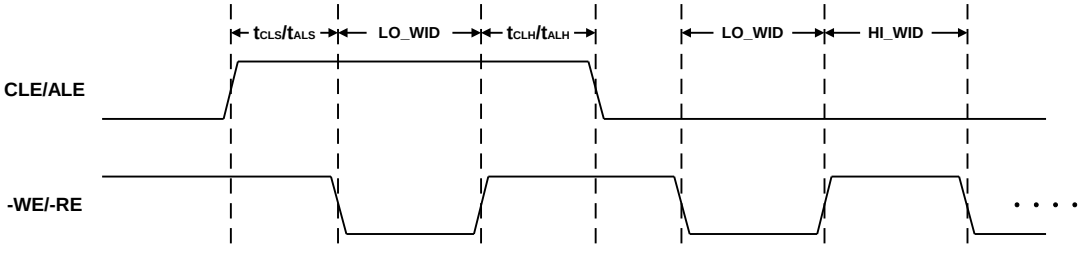
## NAND Flash Timing Control Register (FMI_NANDTMCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| FMI_NANDTMCTL | FMI_BA+0x8A4 | R/W | NAND Flash Timing Control Register | 0x0001_0105 |

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

| Bits | Description | |
|---|---|---|
| [31:23] | **Reserved** | Reserved. |
| [22:16] | **CALE_SH** | **CLE/ALE Setup/Hold Time**<br>This field controls the CLE/ALE setup/hold time to –WE.<br>The setup/hold time can be calculated using following equation:<br>tCLS = (CALE_SH+1)*TAHB.<br>tCLH = ((CALE_SH*2)+2)*TAHB.<br>tALS = (CALE_SH+1)*TAHB.<br>tALH = ((CALE_SH*2)+2)*TAHB. |
| [15:8] | **HI_WID** | **Read/Write Enable Signal High Pulse Width**<br>This field controls the high pulse width of signals –RE and –WE while H/W mode page access is enabled. The pulse width is a multiple of period of AHB bus clock. ( The actual width time will be [clock period*(HI_WID+1)] ) |
| [7:0] | **LO_WID** | **Read/Write Enable Signal Low Pulse Width**<br>This field controls the low pulse width of signals –RE and –WE while H/W mode page access is enabled. The pulse width is a multiple of period of AHB bus clock. ( The actual width time will be [clock period*(LO_WID+1)] ) |

**NOTE1:** The reset value calculated based on 100MHz AHB Clock.



Timing Controlled by FMI_NANDTMCTL Register

## NAND Flash Interrupt Enable Register (FMI_NANDINTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDINTEN** | FMI_BA+0x8A8 | R/W | NAND Flash 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** | | | | **RB1_IE** | **RB0_IE** | **Reserved** | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | | **PROT_REGION_WR_IE** | **ECC_FLD_IE** | **Reserved** | **DMA_IE** |

| Bits | Description | |
|---|---|---|
| [31:12] | **Reserved** | Reserved. |
| [11] | **RB1_IE** | **Ready/-Busy 1 Rising Edge Detect Interrupt Enable**<br>0 = Disable R/-B rising edge detect interrupt generation.<br>1 = Enable R/-B rising edge detect interrupt generation. |
| [10] | **RB0_IE** | **Ready/-Busy Rising Edge Detect Interrupt Enable**<br>0 = Disable R/-B rising edge detect interrupt generation.<br>1 = Enable R/-B rising edge detect interrupt generation. |
| [9:4] | **Reserved** | Reserved. |
| [3] | **PROT_REGION_WR_IE** | **Protect Region Write Detect Interrupt Enable**<br>0=Disable interrupt generation for detect writing to NAND Flash's protect region.<br>1=Enable interrupt generation for detect writing to NAND Flash's protect region. |
| [2] | **ECC_FLD_IE** | **ECC Field Check Error Interrupt Enable**<br>This bit can check the ECC error on each field (512bytes) of data transfer. Enable this bit to detect error and do error correction.<br>0 = Disable.<br>1 = Enable. |
| [1] | **Reserved** | Reserved. |
| [0] | **DMA_IE** | **DMA Read/Write Data Complete Interrupt Enable**<br>0 = Disable DMA read/write data complete interrupt generation.<br>1 = Enable DMA read/write data complete interrupt generation. |

## NAND Flash Interrupt Status Register (FMI_NANDINTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDINTSTS** | FMI_BA+0x8AC | R/W | NAND Flash Interrupt Status Register | 0x00XX_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | **RB1_Status** | **RB0_Status** | **Reserved** | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | **RB1_IF** | **RB0_IF** | **Reserved** | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | | **PROT_REGION_WR_IF** | **ECC_FLD_IF** | **Reserved** | **DMA_IF** |

| Bits | Description | |
|---|---|---|
| [31:20] | **Reserved** | Reserved. |
| [19] | **RB1_Status** | **Ready/-Busy 1 Pin Status (Read Only)**<br>This bit reflects the Ready/-Busy pin status of NAND Flash. |
| [18] | **RB0_Status** | **Ready/-Busy 0 Pin Status (Read Only)**<br>This bit reflects the Ready/-Busy pin status of NAND Flash. |
| [17:12] | **Reserved** | Reserved. |
| [11] | **RB1_IF** | **Ready/-Busy 1 Rising Edge Detect Interrupt Flag (Read Only)**<br>0 = R/-B rising edge is not detected.<br>1 = R/-B rising edge is detected.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [10] | **RB0_IF** | **Ready/-Busy 0 Rising Edge Detect Interrupt Flag (Read Only)**<br>0 = R/-B rising edge is not detected.<br>1 = R/-B rising edge is detected.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [9:4] | **Reserved** | Reserved. |
| [3] | **PROT_REGION_WR_IF** | **Protect Region Write Detect Interrupt Flag (Read Only)**<br>0 = Writing to NAND Flash's protect region is not detected.<br>1 = Writing to NAND Flash's protect region is detected.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |

| | | |
|---|---|---|
| [2] | **ECC_FLD_IF** | **ECC Field Check Error Interrupt Flag (Read Only)**<br>This bit can check the ECC error on each field (512bytes) of data transfer. Read this bit to check if the error occurred.<br>0 = No occurrence of ECC error.<br>1 = Occurrence of ECC error.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [1] | **Reserved** | Reserved. |
| [0] | **DMA_IF** | **DMA Read/Write Data Complete Interrupt Flag (Read Only)**<br>0 = DMA read/write transfer is not finished yet.<br>1 = DMA read/write transfer is done.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |

## NAND Flash Command Port Register (FMI_NANDCMD)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDCMD** | FMI_BA+0x8B0 | W | NAND Flash Command Port 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** |
| **COMMAND** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **COMMAND** | **NAND Flash Command Port**<br>When CPU writes to this port, FMI will send a command to NAND Flash. |

## NAND Flash Address Port Register (FMI_NANDADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDADDR** | FMI_BA+0x8B4 | W | NAND Flash Address Port Register | 0xXXXX_XXXX |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **EOA** | **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** |
| **ADDRESS** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31] | **EOA** | **End of Address**<br>Writing this bit to indicate if this address is the last one or not. By writing address port with this bit low, NAND flash controller will set ALE pin to active (HIGH). After the last address is written (with this bit set high), NAND flash controller will set ALE pin to inactive (LOW).<br>0 = Not the last address cycle.<br>1 = The last one address cycle. |
| [30:8] | **Reserved** | Reserved. |
| [7:0] | **ADDRESS** | **NAND Flash Address Port**<br>By writing this port, NAND flash control will send an address to NAND Flash. |

## NAND Flash Data Port Register (FMI_NANDDATA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDDATA** | FMI_BA+0x8B8 | R/W | NAND Flash Data Port 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** |
| **DATA** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **DATA** | **NAND Flash Data Port**<br>CPU can access NAND's memory array through this data port. When CPU WRITE, the lower 8-bit data from CPU will appear on the data bus of NAND controller. When CPU READ, NAND controller will get 8-bit data from data bus. |

## NAND Flash Redundant Area Control Register (FMI_NANDRACTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDRACTL** | FMI_BA+0x8BC | R/W | NAND Flash Redundant Area Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **MECC** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **MECC** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | | | **RA128EN** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **RA128EN** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **MECC** | **Mask ECC During Write Page Data**<br>These 16 bits registers indicate NAND controller to write out ECC parity or just 0xFF for each field (every 512 bytes) the real parity data will be write out to FMI_NANDRAx.<br>0x00 = Do not mask the ECC parity for each field.<br>0x01 = Mask ECC parity and write out FF to NAND ECC parity for 512 Bytes page size or 2K/4K/8K page size first 512 field.<br>0x02 = Mask ECC parity and write out FF to NAND ECC parity for 512 Bytes page size or 2K/4K/8K page size second 512 field.<br>0xxx = Mask ECC parity and write out FF to NAND ECC parity for 512 Bytes page size or 2K/4K/8K page size each 512 field. |
| [15:9] | **Reserved** | Reserved. |
| [8:0] | **RA128EN** | **Redundant Area 128 Byte Enable**<br>These bits indicate NAND flash extended redundant area.<br>If PSIZE (FMI_NANDCTL[17:16]) = 2'b00, this field will be set 0x10 (16bytes) automatically.<br>If PSIZE (FMI_NANDCTL[17:16]) = 2'b01, this field will be set 0x40 (64bytes) automatically.<br>If PSIZE (FMI_NANDCTL[17:16]) = 2'b10, this field will be set 0x80 (128 bytes) automatically.<br>If PSIZE (FMI_NANDCTL[17:16]) = 2'b11, this field will be set 0x100 (256bytes) automatically.<br>**Note:** The REA128EN must be 4 byte aligned, so bit1 and bit0 can't be filled 1 to it.<br>The maximum redundant area of the controller is 472Bytes. |

## NAND Flash Extend Control Regsiter (FMI_NANDECTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDECTL** | FMI_BA+0x8C0 | R/W | NAND Flash Extend 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** | | | | | | | **WP** |

| Bits | Description | |
|---|---|---|
| [31:1] | **Reserved** | Reserved. |
| [0] | **WP** | **NAND Flash Write Protect Control (Low Active)**<br>Set this bit low to make NAND_nWP functional pin low to prevent the write to NAND flash device.<br>0 = NAND flash is write-protected and is not writeable.<br>1 = NAND flash is not write-protected and is writeable. |

## NAND Flash ECC Error Status 0 Register (FMI_NANDECCES0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| FMI_NANDECCES0 | FMI_BA+0x8D0 | R | NAND Flash ECC Error Status 0 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | F4_ECNT | | | | | F4_STAT | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | F3_ECNT | | | | | F3_STAT | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | F2_ECNT | | | | | F2_STAT | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | F1_ECNT | | | | | F1_STAT | |

| Bits | Description | |
|---|---|---|
| [31] | **Reserved** | Reserved. |
| [30:26] | **F4_ECNT** | **Error Count of ECC Field 4**<br>This field contains the error counts after ECC correct calculation of Field 4. For this ECC core (BCH algorithm), only when F4_STAT equals to 0x01, the value in this field is meaningful. F4_ECNT means how many errors depending on which ECC is used. |
| [25:24] | **F4_STAT** | **ECC Status of Field 4**<br>This field contains the ECC correction status (BCH algorithm) of ECC-field 4.<br>00 = No error.<br>01 = Correctable error.<br>10 = Uncorrectable error.<br>11 = Reserved. |
| [23] | **Reserved** | Reserved. |
| [22:18] | **F3_ECNT** | **Error Count of ECC Field 3**<br>This field contains the error counts after ECC correct calculation of Field 3. For this ECC core (BCH algorithm), only when F3_STAT equals to 0x01, the value in this field is meaningful. F3_ECNT means how many errors depending on which ECC is used. |
| [17:16] | **F3_STAT** | **ECC Status of Field 3**<br>This field contains the ECC correction status (BCH algorithm) of ECC-field 3.<br>00 = No error.<br>01 = Correctable error.<br>10 = Uncorrectable error.<br>11 = Reserved. |
| [15] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [14:10] | **F2_ECNT** | **Error Count of ECC Field 2**<br>This field contains the error counts after ECC correct calculation of Field 2. For this ECC core (BCH algorithm), only when F2_STAT equals to 0x01, the value in this field is meaningful. F2_ECNT means how many errors depending on which ECC is used. |
| [9:8] | **F2_STAT** | **ECC Status of Field 2**<br>This field contains the ECC correction status (BCH algorithm) of ECC-field 2.<br>00 = No error.<br>01 = Correctable error.<br>10 = Uncorrectable error.<br>11 = Reserved. |
| [7] | **Reserved** | Reserved. |
| [6:2] | **F1_ECNT** | **Error Count of ECC Field 1**<br>This field contains the error counts after ECC correct calculation of Field 1. For this ECC core (BCH algorithm), only when F1_STAT equals to 0x01, the value in this field is meaningful. F1_ECNT means how many errors depending on which ECC is used. |
| [1:0] | **F1_STAT** | **ECC Status of Field 1**<br>This field contains the ECC correction status (BCH algorithm) of ECC-field 1.<br>00 = No error.<br>01 = Correctable error.<br>10 = Uncorrectable error.<br>11 = Reserved. |

## NAND Flash ECC Error Status 1 Register (FMI_NANDECCES1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| FMI_NANDECCES1 | FMI_BA+0x8D4 | R | NAND Flash ECC Error Status 1 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | F8_ECNT | | | | | F8_STAT | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | F7_ECNT | | | | | F7_STAT | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | F6_ECNT | | | | | F6_STAT | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | F5_ECNT | | | | | F5_STAT | |

| Bits | Description | |
|---|---|---|
| [31] | **Reserved** | Reserved. |
| [30:26] | **F8_ECNT** | **Error Count of ECC Field 8**<br>This field contains the error counts after ECC correct calculation of Field 8. For this ECC core (BCH algorithm), only when F8_STAT equals to 0x01, the value in this field is meaningful. F8_ECNT means how many errors depending on which ECC is used. |
| [25:24] | **F8_STAT** | **ECC Status of Field 8**<br>This field contains the ECC correction status (BCH algorithm) of ECC-field 8.<br>00 = No error.<br>01 = Correctable error.<br>10 = Uncorrectable error.<br>11 = Reserved. |
| [23] | **Reserved** | Reserved. |
| [22:18] | **F7_ECNT** | **Error Count of ECC Field 7**<br>This field contains the error counts after ECC correct calculation of Field 7. For this ECC core (BCH algorithm), only when F7_STAT equals to 0x01, the value in this field is meaningful. F7_ECNT means how many errors depending on which ECC is used. |
| [17:16] | **F7_STAT** | **ECC Status of Field 7**<br>This field contains the ECC correction status (BCH algorithm) of ECC-field 7.<br>00 = No error.<br>01 = Correctable error.<br>10 = Uncorrectable error.<br>11 = Reserved. |
| [15] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [14:10] | **F6_ECNT** | **Error Count of ECC Field 6**<br>This field contains the error counts after ECC correct calculation of Field 6. For this ECC core (BCH algorithm), only when F6_STAT equals to 0x01, the value in this field is meaningful. F6_ECNT means how many errors depending on which ECC is used. |
| [9:8] | **F6_STAT** | **ECC Status of Field 6**<br>This field contains the ECC correction status (BCH algorithm) of ECC-field 6.<br>00 = No error.<br>01 = Correctable error.<br>10 = Uncorrectable error.<br>11 = Reserved. |
| [7] | **Reserved** | Reserved. |
| [6:2] | **F5_ECNT** | **Error Count of ECC Field 5**<br>This field contains the error counts after ECC correct calculation of Field 5. For this ECC core (BCH algorithm), only when F5_STAT equals to 0x01, the value in this field is meaningful. F5_ECNT means how many errors depending on which ECC is used. |
| [1:0] | **F5_STAT** | **ECC Status of Field 5**<br>This field contains the ECC correction status (BCH algorithm) of ECC-field5.<br>00 = No error.<br>01 = Correctable error.<br>10 = Uncorrectable error.<br>11 = Reserved. |

## NAND Flash ECC Error Status 2 Register (FMI_NANDECCES2)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| FMI_NANDECCES2 | FMI_BA+0x8D8 | R | NAND Flash ECC Error Status 2 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | F12_ECNT | | | | | F12_STAT | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | F11_ECNT | | | | | F11_STAT | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | F10_ECNT | | | | | F10_STAT | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | F9_ECNT | | | | | F9_STAT | |

| Bits | Description | |
|---|---|---|
| [31] | **Reserved** | Reserved. |
| [30:26] | **F12_ECNT** | **Error Count of ECC Field 12**<br>This field contains the error counts after ECC correct calculation of Field 12. For this ECC core (BCH algorithm), only when F12_STAT equals to 0x01, the value in this field is meaningful. F12_ECNT means how many errors depending on which ECC is used. |
| [25:24] | **F12_STAT** | **ECC Status of Field 12**<br>This field contains the ECC correction status (BCH algorithm) of ECC-field 12.<br>00 = No error.<br>01 = Correctable error.<br>10 = Uncorrectable error.<br>11 = Reserved. |
| [23] | **Reserved** | Reserved. |
| [22:18] | **F11_ECNT** | **Error Count of ECC Field 11**<br>This field contains the error counts after ECC correct calculation of Field 11. For this ECC core (BCH algorithm), only when F11_STAT equals to 0x01, the value in this field is meaningful. F11_ECNT means how many errors depending on which ECC is used. |
| [17:16] | **F11_STAT** | **ECC Status of Field 11**<br>This field contains the ECC correction status (BCH algorithm) of ECC-field 11.<br>00 = No error.<br>01 = Correctable error.<br>10 = Uncorrectable error.<br>11 = Reserved. |
| [15] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [14:10] | **F10_ECNT** | **Error Count of ECC Field 10**<br>This field contains the error counts after ECC correct calculation of Field 10. For this ECC core (BCH algorithm), only when F10_STAT equals to 0x01, the value in this field is meaningful. F10_ECNT means how many errors depending on which ECC is used. |
| [9:8] | **F10_STAT** | **ECC Status of Field 10**<br>This field contains the ECC correction status (BCH algorithm) of ECC-field 10.<br>00 = No error.<br>01 = Correctable error.<br>10 = Uncorrectable error.<br>11 = Reserved. |
| [7] | **Reserved** | Reserved. |
| [6:2] | **F9_ECNT** | **Error Count of ECC Field 9**<br>This field contains the error counts after ECC correct calculation of Field 9. For this ECC core (BCH algorithm), only when F9_STAT equals to 0x01, the value in this field is meaningful. F9_ECNT means how many errors depending on which ECC is used. |
| [1:0] | **F9_STAT** | **ECC Status of Field 9**<br>This field contains the ECC correction status (BCH algorithm) of ECC-field 9.<br>00 = No error.<br>01 = Correctable error.<br>10 = Uncorrectable error.<br>11 = Reserved. |

## NAND Flash ECC Error Status 3 Register (FMI_NANDECCES3)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| FMI_NANDECCES3 | FMI_BA+0x8DC | R | NAND Flash ECC Error Status 3 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | F16_ECNT | | | | | F16_STAT | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | F15_ECNT | | | | | F15_STAT | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | F14_ECNT | | | | | F14_STAT | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | F13_ECNT | | | | | F13_STAT | |

| Bits | Description | |
|---|---|---|
| [31] | **Reserved** | Reserved. |
| [30:26] | **F16_ECNT** | **Error Count of ECC Field 16**<br>This field contains the error counts after ECC correct calculation of Field 16. For this ECC core (BCH algorithm), only when F16_STAT equals to 0x01, the value in this field is meaningful. F16_ECNT means how many errors depending on which ECC is used. |
| [25:24] | **F16_STAT** | **ECC Status of Field 16**<br>This field contains the ECC correction status (BCH algorithm) of ECC-field 16.<br>00 = No error.<br>01 = Correctable error.<br>10 = Uncorrectable error.<br>11 = Reserved. |
| [23] | **Reserved** | Reserved. |
| [22:18] | **F15_ECNT** | **Error Count of ECC Field 15**<br>This field contains the error counts after ECC correct calculation of Field 15. For this ECC core (BCH algorithm), only when F15_STAT equals to 0x01, the value in this field is meaningful. F15_ECNT means how many errors depending on which ECC is used. |
| [17:16] | **F15_STAT** | **ECC Status of Field 15**<br>This field contains the ECC correction status (BCH algorithm) of ECC-field 15.<br>00 = No error.<br>01 = Correctable error.<br>10 = Uncorrectable error.<br>11 = Reserved. |
| [15] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [14:10] | **F14_ECNT** | **Error Count of ECC Field 14**<br>This field contains the error counts after ECC correct calculation of Field 14. For this ECC core (BCH algorithm), only when F14_STAT equals to 0x01, the value in this field is meaningful. F14_ECNT means how many errors depending on which ECC is used. |
| [9:8] | **F14_STAT** | **ECC Status of Field 14**<br>This field contains the ECC correction status (BCH algorithm) of ECC-field 14.<br>00 = No error.<br>01 = Correctable error.<br>10 = Uncorrectable error.<br>11 = Reserved. |
| [7] | **Reserved** | Reserved. |
| [6:2] | **F13_ECNT** | **Error Count of ECC Field 13**<br>This field contains the error counts after ECC correct calculation of Field 13. For this ECC core (BCH algorithm), only when F13_STAT equals to 0x01, the value in this field is meaningful. F13_ECNT means how many errors depending on which ECC is used. |
| [1:0] | **F13_STAT** | **ECC Status of Field 13**<br>This field contains the ECC correction status (BCH algorithm) of ECC-field 13.<br>00 = No error.<br>01 = Correctable error.<br>10 = Uncorrectable error.<br>11 = Reserved. |

### NAND Flash Protect Region End Address 0 Register (FMI_NANDPROTA0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDPROTA0** | FMI_BA+0x8E0 | R/W | NAND Flash Protect Region End Address 0 Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **ADDR** | **NAND Flash Protect End Address Register 0**<br>By setting register FMI_NANDPROTA0, FMI_NANDPROTA1 and enable PROT_REGION_EN (FMI_NANDCTL[5]), the NAND Flash from address 0 to address {FMI_NANDPRTOA1, FMI_NANDPROTA0} region will be write protect. |

## NAND Flash Protect Region End Address 1 Register (FMI_NANDPROTA1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDPROTA1** | FMI_BA+0x8E4 | R/W | NAND Flash Protect Region End Address 1 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** |
| **ADDR** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **ADDR** | **NAND Flash Protect End Address Register 1**<br>By setting register FMI_NANDPRTOA0, FMI_NANDPROTA1 and enable PROT_REGION_EN (FMI_NANDCTL[5]), the NAND Flash from address 0 to address {FMI_NANDPRTOA1, FMI_NANDPROTA0} region will be write protect. |

### NAND Flash ECC Error Byte Address 0 Register (FMI_NANDECCEA0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDECCEA0** | FMI_BA+0x900 | R | NAND Flash ECC Error Byte Address 0 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | **ERR_ADDR1** | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **ERR_ADDR1** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | **ERR_ADDR0** | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ERR_ADDR0** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **ERR_ADDR1** | **ECC Error Address First Field of Error 1**<br>This field contains an 11-bit ECC error address 1 of first field. If it is a correctable error, please read the error data, ERR_DATA1 (FMI_NANDECCED0[15:8]), to correct this error. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **ERR_ADDR0** | **ECC Error Address First Field of Error 0**<br>This field contains an 11-bit ECC error address 0 of first field. If it is a correctable error, please read the error data, ERR_DATA0 (FMI_NANDECCED0[7:0]), to correct this error. |

## NAND Flash ECC Error Byte Address 1 Register (FMI_NANDECCEA1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDECCEA1** | FMI_BA+0x904 | R | NAND Flash ECC Error Byte Address 1 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | **ERR_ADDR3** | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **ERR_ADDR3** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | **ERR_ADDR2** | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ERR_ADDR2** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **ERR_ADDR3** | **ECC Error Address First Field of Error 3**<br>This field contains an 11-bit ECC error address 3 of first field. If it is a correctable error, please read the error data, ERR_DATA3 (FMI_NANDECCED0[31:24]), to correct this error. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **ERR_ADDR2** | **ECC Error Address First Field of Error 2**<br>This field contains an 11-bit ECC error address 2 of first field. If it is a correctable error, please read the error data, ERR_DATA2 (FMI_NANDECCED0[23:16]), to correct this error. |

## NAND Flash ECC Error Byte Address 2 Register (FMI_NANDECCEA2)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDECCEA2** | FMI_BA+0x908 | R | NAND Flash ECC Error Byte Address 2 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | **ERR_ADDR5** | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **ERR_ADDR5** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | **ERR_ADDR4** | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ERR_ADDR4** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **ERR_ADDR5** | **ECC Error Address First Field of Error 5**<br>This field contains an 11-bit ECC error address 5 of first field. If it is a correctable error, please read the error data, ERR_DATA5 (FMI_NANDECCED1[15:8]), to correct this error. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **ERR_ADDR4** | **ECC Error Address First Field of Error 4**<br>This field contains an 11-bit ECC error address 4 of first field. If it is a correctable error, please read the error data, ERR_DATA4 (FMI_NANDECCED1[7:0]), to correct this error. |

## NAND Flash ECC Error Byte Address 3 Register (FMI_NANDECCEA3)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDECCEA3** | FMI_BA+0x90C | R | NAND Flash ECC Error Byte Address 3 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | **ERR_ADD7** | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **ERR_ADDR7** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | **ERR_ADDR6** | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ERR_ADDR6** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **ERR_ADDR7** | **ECC Error Address First Field of Error 7**<br>This field contains an 11-bit ECC error address 7 of first field. If it is a correctable error, please read the error data, ERR_DATA7 (FMI_NANDECCED1[31:24]), to correct this error. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **ERR_ADDR6** | **ECC Error Address First Field of Error 6**<br>This field contains an 11-bit ECC error address 6 of first field. If it is a correctable error, please read the error data, ERR_DATA6 (FMI_NANDECCED1[23:16]), to correct this error. |

## NAND Flash ECC Error Byte Address 4 Register (FMI_NANDECCEA4)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDECCEA4** | FMI_BA+0x910 | R | NAND Flash ECC Error Byte Address 4 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | **ERR_ADDR9** | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **ERR_ADDR9** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | **ERR_ADDR8** | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ERR_ADDR8** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **ERR_ADDR9** | **ECC Error Address First Field of Error 9**<br>This field contains an 11-bit ECC error address 9 of first field. If it is a correctable error, please read the error data, ERR_DATA9 (FMI_NANDECCED2[15:8]), to correct this error. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **ERR_ADDR8** | **ECC Error Address First Field of Error 8**<br>This field contains an 11-bit ECC error address 8 of first field. If it is a correctable error, please read the error data, ERR_DATA8 (FMI_NANDECCED2[7:0]), to correct this error. |

## NAND Flash ECC Error Byte Address 5 Register (FMI_NANDECCEA5)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDECCEA5** | FMI_BA+0x914 | R | NAND Flash ECC Error Byte Address 5 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | **ERR_ADDR11** | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **ERR_ADDR11** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | **ERR_ADDR10** | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ERR_ADDR10** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **ERR_ADDR11** | **ECC Error Address First Field of Error 11**<br>This field contains an 11-bit ECC error address 11 of first field. If it is a correctable error, please read the error data, ERR_DATA11 (FMI_NANDECCED2[31:24]), to correct this error. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **ERR_ADDR10** | **ECC Error Address First Field of Error 10**<br>This field contains an 11-bit ECC error address 10 of first field. If it is a correctable error, please read the error data, ERR_DATA10 (FMI_NANDECCED2[23:16]), to correct this error. |

## NAND Flash ECC Error Byte Address 6 Register (FMI_NANDECCEA6)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDECCEA6** | FMI_BA+0x918 | R | NAND Flash ECC Error Byte Address 6 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | **ERR_ADDR13** | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **ERR_ADDR13** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | **ERR_ADDR12** | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ERR_ADDR12** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **ERR_ADDR13** | **ECC Error Address First Field of Error 13**<br>This field contains an 11-bit ECC error address 13 of first field. If it is a correctable error, please read the error data, ERR_DATA13 (FMI_NANDECCED3[15:8]), to correct this error. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **ERR_ADDR12** | **ECC Error Address First Field of Error 12**<br>This field contains an 11-bit ECC error address 12 of first field. If it is a correctable error, please read the error data, ERR_DATA12 (FMI_NANDECCED3[7:0]), to correct this error. |

## NAND Flash ECC Error Byte Address 7 Register (FMI_NANDECCEA7)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDECCEA7** | FMI_BA+0x91C | R | NAND Flash ECC Error Byte Address 7 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | **ERR_ADDR15** | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **ERR_ADDR15** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | **ERR_ADDR14** | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ERR_ADDR14** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **ERR_ADDR15** | **ECC Error Address First Field of Error 15**<br>This field contains an 11-bit ECC error address 15 of first field. If it is a correctable error, please read the error data, ERR_DATA15 (FMI_NANDECCED3[31:24]), to correct this error. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **ERR_ADDR14** | **ECC Error Address First Field of Error 14**<br>This field contains an 11-bit ECC error address 14 of first field. If it is a correctable error, please read the error data, ERR_DATA14 (FMI_NANDECCED3[23:16]), to correct this error. |

### NAND Flash ECC Error Byte Address 8 Register (FMI_NANDECCEA8)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDECCEA8** | FMI_BA+0x920 | R | NAND Flash ECC Error Byte Address 8 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | **ERR_ADDR17** | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **ERR_ADDR17** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | **ERR_ADDR16** | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ERR_ADDR16** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **ERR_ADDR17** | **ECC Error Address First Field of Error 17**<br>This field contains an 11-bit ECC error address 17 of first field. If it is a correctable error, please read the error data, ERR_DATA17 (FMI_NANDECCED4[15:8]), to correct this error. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **ERR_ADDR16** | **ECC Error Address First Field of Error 16**<br>This field contains an 11-bit ECC error address 16 of first field. If it is a correctable error, please read the error data, ERR_DATA16 (FMI_NANDECCED4[7:0]), to correct this error. |

## NAND Flash ECC Error Byte Address 9 Register (FMI_NANDECCEA9)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDECCEA9** | FMI_BA+0x924 | R | NAND Flash ECC Error Byte Address 9 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | **ERR_ADDR19** | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **ERR_ADDR19** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | **ERR_ADDR18** | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ERR_ADDR18** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **ERR_ADDR19** | **ECC Error Address First Field of Error 19**<br>This field contains an 11-bit ECC error address 19 of first field. If it is a correctable error, please read the error data, ERR_DATA19 (FMI_NANDECCED4[31:24]), to correct this error. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **ERR_ADDR18** | **ECC Error Address First Field of Error 18**<br>This field contains an 11-bit ECC error address 18 of first field. If it is a correctable error, please read the error data, ERR_DATA18 (FMI_NANDECCED4[23:16]), to correct this error. |

## NAND Flash ECC Error Byte Address 10 Register (FMI_NANDECCEA10)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDECCEA10** | FMI_BA+0x928 | R | NAND Flash ECC Error Byte Address 10 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | **ERR_ADDR21** | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **ERR_ADDR21** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | **ERR_ADDR20** | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ERR_ADDR20** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **ERR_ADDR21** | **ECC Error Address First Field of Error 21**<br>This field contains an 11-bit ECC error address 21 of first field. If it is a correctable error, please read the error data, ERR_DATA21 (FMI_NANDECCED5[15:8]), to correct this error. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **ERR_ADDR20** | **ECC Error Address First Field of Error 20**<br>This field contains an 11-bit ECC error address 20 of first field. If it is a correctable error, please read the error data, ERR_DATA20 (FMI_NANDECCED5[7:0]), to correct this error. |

## NAND Flash ECC Error Byte Address 11 Register (FMI_NANDECCEA11)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDECCEA11** | FMI_BA+0x92C | R | NAND Flash ECC Error Byte Address 11 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | **ERR_ADDR23** | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **ERR_ADDR23** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | **ERR_ADDR22** | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ERR_ADDR22** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **ERR_ADDR23** | **ECC Error Address First Field of Error 23**<br>This field contains an 11-bit ECC error address 23 of first field. If it is a correctable error, please read the error data, ERR_DATA23 (FMI_NANDECCED5[31:24]), to correct this error. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **ERR_ADDR22** | **ECC Error Address First Field of Error 22**<br>This field contains an 11-bit ECC error address 22 of first field. If it is a correctable error, please read the error data, ERR_DATA22 (FMI_NANDECCED5[23:16]), to correct this error. |

## NAND Flash ECC Error Data Register 0 (FMI_NANDECCED0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDECCED0** | FMI_BA+0x960 | R | NAND Flash ECC Error Data Register 0 | 0x8080_8080 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **ERR_DATA3** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **ERR_DATA2** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **ERR_DATA1** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ERR_DATA0** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **ERR_DATA3** | **ECC Error Data of First Field 3**<br>This field contains an 8-bit BCH ECC error data 3 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR3 (FMI_NANDECCEA1[26:16]), and then the result will be the correct data. |
| [23:16] | **ERR_DATA2** | **ECC Error Data of First Field 2**<br>This field contains an 8-bit BCH ECC error data 2 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR2 (FMI_NANDECCEA1[10:0]), and then the result will be the correct data. |
| [15:8] | **ERR_DATA1** | **ECC Error Data of First Field 1**<br>This field contains an 8-bit BCH ECC error data 1 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR1 (FMI_NANDECCEA0[26:0]), and then the result will be the correct data. |
| [7:0] | **ERR_DATA0** | **ECC Error Data of First Field 0**<br>This field contains an 8-bit BCH ECC error data 0 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR0 (FMI_ANNDECCEA0[10:0]), and then the result will be the correct data. |

## NAND Flash ECC Error Data Register 1 (FMI_NANDECCED1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| FMI_NANDECCED1 | FMI_BA+0x964 | R | NAND Flash ECC Error Data Register 1 | 0x8080_8080 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| ERR_DATA7 | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ERR_DATA6 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| ERR_DATA5 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ERR_DATA4 | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | ERR_DATA7 | **ECC Error Data of First Field 7**<br>This field contains an 8-bit BCH ECC error data 7 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR7 (FMI_NANDECCEA3[26:16]), and then the result will be the correct data. |
| [23:16] | ERR_DATA6 | **ECC Error Data of First Field 6**<br>This field contains an 8-bit BCH ECC error data 6 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR6 (FMI_NANDECCEA3[10:0]), and then the result will be the correct data. |
| [15:8] | ERR_DATA5 | **ECC Error Data of First Field 5**<br>This field contains an 8-bit BCH ECC error data 5 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR5 (FMI_NANDECCEA2[26:16]), and then the result will be the correct data. |
| [7:0] | ERR_DATA4 | **ECC Error Data of First Field 4**<br>This field contains an 8-bit BCH ECC error data 4 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR4 (FMI_NANDECCEA2[10:0]), and then the result will be the correct data. |

## NAND Flash ECC Error Data Register 2 (FMI_NANDECCED2)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDECCED2** | FMI_BA+0x968 | R | NAND Flash ECC Error Data Register 2 | 0x8080_8080 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **ERR_DATA11** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **ERR_DATA10** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **ERR_DATA9** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ERR_DATA8** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **ERR_DATA11** | **ECC Error Data of First Field 11**<br>This field contains an 8-bit BCH ECC error data 11 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR11 (FMI_NANDECCEA5[26:16]), and then the result will be the correct data. |
| [23:16] | **ERR_DATA10** | **ECC Error Data of First Field 10**<br>This field contains an 8-bit BCH ECC error data 10 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR10 (FMI_NANDECCEA5[10:0]), and then the result will be the correct data. |
| [15:8] | **ERR_DATA9** | **ECC Error Data of First Field 9**<br>This field contains an 8-bit BCH ECC error data 9 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR9 (FMI_NANDECCEA4[26:16]), and then the result will be the correct data. |
| [7:0] | **ERR_DATA8** | **ECC Error Data of First Field 8**<br>This field contains an 8-bit BCH ECC error data 8 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR8 (FMI_NANDECCEA4[10:0]), and then the result will be the correct data. |

## NAND Flash ECC Error Data Register 3 (FMI_NANDECCED3)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDECCED3** | FMI_BA+0x96C | R | NAND Flash ECC Error Data Register 3 | 0x8080_8080 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **ERR_DATA15** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **ERR_DATA14** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **ERR_DATA13** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ERR_DATA12** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **ERR_DATA15** | **ECC Error Data of First Field 15**<br>This field contains an 8-bit BCH ECC error data 15 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR15 (FMI_NANDECCEA7[26:16]), and then the result will be the correct data. |
| [23:16] | **ERR_DATA14** | **ECC Error Data of First Field 14**<br>This field contains an 8-bit BCH ECC error data 14 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR14 (FMI_NANDECCEA7[10:0]), and then the result will be the correct data. |
| [15:8] | **ERR_DATA13** | **ECC Error Data of First Field 13**<br>This field contains an 8-bit BCH ECC error data 13 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR13 (FMI_NANDECCEA6[26:16]), and then the result will be the correct data. |
| [7:0] | **ERR_DATA12** | **ECC Error Data of First Field 12**<br>This field contains an 8-bit BCH ECC error data 12 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR12 (FMI_NANDECCEA6[10:0]), and then the result will be the correct data. |

## NAND Flash ECC Error Data Register 4 (FMI_NANDECCED4)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDECCED4** | FMI_BA+0x970 | R | NAND Flash ECC Error Data Register 4 | 0x8080_8080 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **ERR_DATA19** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **ERR_DATA18** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **ERR_DATA17** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ERR_DATA16** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **ERR_DATA19** | **ECC Error Data of First Field 19**<br>This field contains an 8-bit BCH ECC error data 19 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR19 (FMI_NANDECCEA9[26:16]), and then the result will be the correct data. |
| [23:16] | **ERR_DATA18** | **ECC Error Data of First Field 18**<br>This field contains an 8-bit BCH ECC error data 18 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR18 (FMI_NANDECCEA9[10:0]), and then the result will be the correct data. |
| [15:8] | **ERR_DATA17** | **ECC Error Data of First Field 17**<br>This field contains an 8-bit BCH ECC error data 17 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR17 (FMI_NANDECCEA8[26:16]), and then the result will be the correct data. |
| [7:0] | **ERR_DATA16** | **ECC Error Data of First Field 16**<br>This field contains an 8-bit BCH ECC error data 16 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR16 (FMI_NANDECCEA8[10:0]), and then the result will be the correct data. |

## NAND Flash ECC Error Data Register 5 (FMI_NANDECCED5)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **FMI_NANDECCED5** | FMI_BA+0x974 | R | NAND Flash ECC Error Data Register 5 | 0x8080_8080 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **ERR_DATA23** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **ERR_DATA22** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **ERR_DATA21** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ERR_DATA20** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **ERR_DATA23** | **ECC Error Data of First Field 23**<br>This field contains an 8-bit BCH ECC error data 23 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR23 (FMI_NANDECCEA11[26:16]), and then the result will be the correct data. |
| [23:16] | **ERR_DATA22** | **ECC Error Data of First Field 22**<br>This field contains an 8-bit BCH ECC error data 22 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR22 (FMI_NANDECCEA11[10:0]), and then the result will be the correct data. |
| [15:8] | **ERR_DATA21** | **ECC Error Data of First Field 21**<br>This field contains an 8-bit BCH ECC error data 21 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR21 (FMI_NANDECCEA10[26:16]), and then the result will be the correct data. |
| [7:0] | **ERR_DATA20** | **ECC Error Data of First Field 20**<br>This field contains an 8-bit BCH ECC error data 20 of first field. If it is a correctable error, please read out the error data in this field and doing bitwise XOR with received data locating at address ERR_ADDR20 (FMI_NANDECCEA10[10:0]), and then the result will be the correct data. |

## 5.26 Secure Digital Host Controller (SDH)

### 5.26.1 Overview

The Secure-Digital Card Host Controller (SDH) equips 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/SDIO. The SDH controller supports SD/SDHC/SDIO card and cooperates with DMAC to provide a fast data transfer between system memory and cards.

### 5.26.2 Features

- Supports single DMA channel
- Supports hardware Scatter-Gather functionality.
- Supports 128 Bytes shared buffer for data exchange between system memory and cards.
- Supports SD, SDHC and SDIO card.

### 5.26.3 Block Diagram

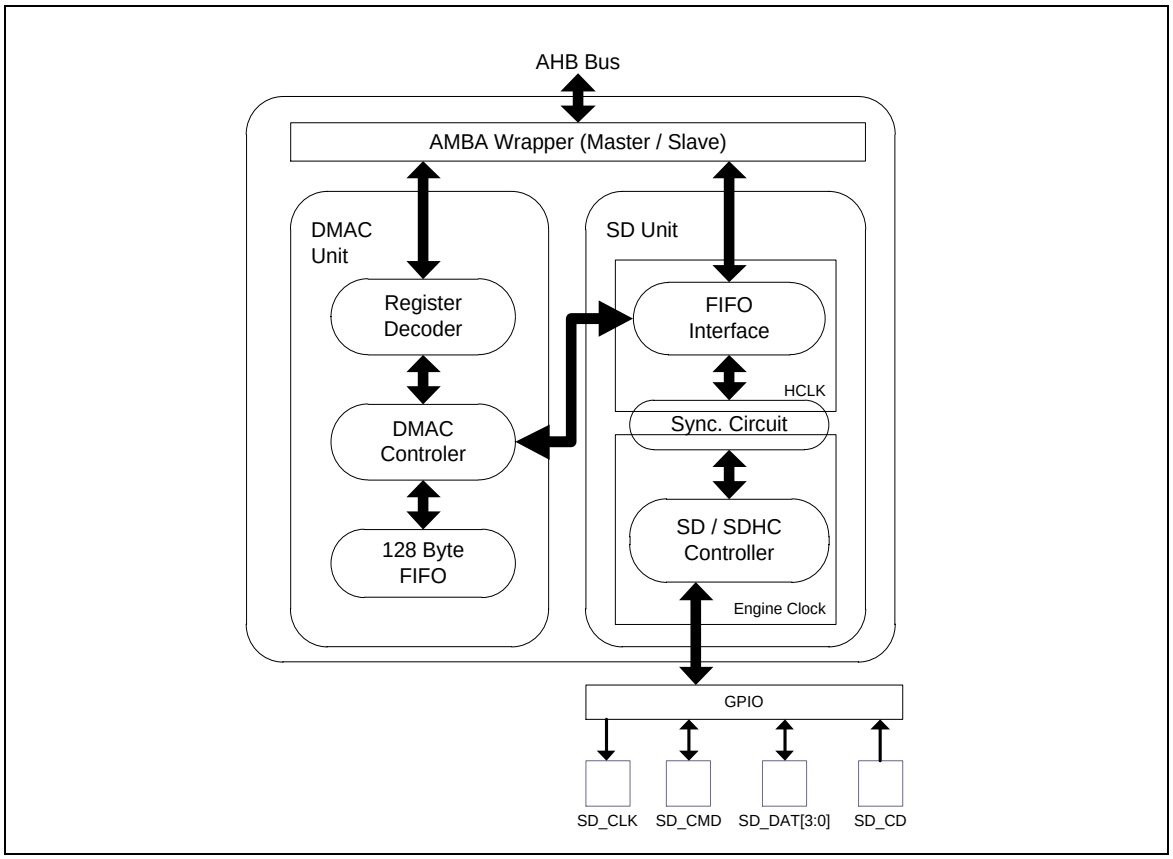



Figure 1. SD Card Host Controller Block Diagram

### 5.26.4 Basic Configuration

Before using SD host controller, it's necessary to configure related pins as the SDH function and enable SDH's clock.

For SD host related pin configuration, please refer to the register SYS_MFP_GPDL, SYS_MFP_GPEL, SYS_MFP_GPEH, SYS_MFP_GPHL and SYS_MFP_GPHH to know how to configure related pins as the SD host function.

Set SDH (CLK_HCLKEN[30]) high to turn on clock for SD host controller. In addition, it's necessary to configure SDH_S (CLK_DIVCTL9[4:0]) and SDH_N (CLK_DIVCTL9[15:8]) to generate SD_CLK to SD, SDHC or SDIO card properly.

#### 5.26.4.1 *Scatter-Gather DMA Disabled*

1. Set DMAEN (SDH_DMACTL[0]) to enable DMA.
2. Fill corresponding starting address in SDH_DMASA.
3. Enable IP to start DMA transfer.
4. Wait IP finished, software doesn't need to take care of DMA.

#### 5.26.4.2 *Scatter-Gather DMA Enabled*

1. Set DMAEN (SDH_DMACTL[0]) to enable DMACand SGEN (SDH_DMACTL[3]) to enable Scatter-Gather function.
2. Fill corresponding starting address of Physical Address Descriptor (PAD) table in SDH_DMASA.
3. When bit-0 of SDH_DMASA is 1, the PAD will fetch in out of order, otherwise, it's 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. Enable IP to start DMA transfer.
5. Wait IP finished, software doesn't need to take care of DMA.

### 5.26.5 Functional Description

#### 5.26.5.1 *DMA Controller*

The DMA Controller provides a DMA (Direct Memory Access) function for SDH to exchange data between system memory (ex. SDRAM) and shared buffer (128 bytes). After filling in the starting address and enables DMA, DMA would handle the data transfer automatically.

There is a 128 bytes shared buffer inside DMA. This 128 bytes buffer is directly accessible when SDH is not in busy.

#### 5.26.5.2 *Secure-Digital (SD)*

SDH provides an interface for SD/SDHC/SDIO/MMC card access. This SDH controller provides two SD ports –port0 and port1. Each port can provide 1-bit/4-bit data bus mode, card detect function and SDIO interrupt.

SDH controller uses an independent clock source named SDCLK as engine clock. SDCLK can be completely asynchronous with system clock HCLK, SDCLK is changeable. However the HCLK should be faster than SDCLK.

This SDH controller can generate all types of 48-bit command to SD card and retrieve all types of response from SD card. After response in, the content of response will be stored at SDH_RESP0 and SDH_RESP1. SD controller will calculate CRC-7 and check its correctness for response. If CRC-7 is error, CRC_IF (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, CRC-7 is invalid; but SD controller will still calculate CRC-7 and get an error result, software should ignore this error and clear CRC_IF (SDH_INTSTS[1]) flag.

This SD controller is composed of two state machines they are command/response part and data part.

For command/response part, the trigger bits are CO_EN (SDH_CTL[0]), RI_EN (SDH_CTL[1]), R2_EN (SDH_CTL[4]), CLK74_OE (SDH_CTL[5]) and CLK8_OE (SDH_CTL[6]). If all of these bits enabled, the execution priority will be CLK74_OE (SDH_CTL[5]), CO_EN (SDH_CTL[0]), RI_EN (SDH_CTL[1])/ R2_EN (SDH_CTL[4]), and then CLK8_OE (SDH_CTL[6]). Please note that RI_EN (SDH_CTL[1]) and R2_EN (SDH_CTL[4]) can't be triggered at the same time.

For data part, there are DI_EN (SDH_CTL[2]) and DO_EN (SDH_CTL[3]) for choose. Every time, only one of them could be triggered. If DI_EN (SDH_CTL[2]) is triggered, SD controller waits start bit from data line DAT0 immediately, and then get specified amount data from SD card. After data-in, SD controller will check CRC-16 correctness; if it is error, CRC_IF (SDH_INTSTS[1]) will be set and CRC16 (SDH_INTSTS[3]) will be '0'. If DO_EN (SDH_CTL[3]) is triggered, SD controller will wait response in finished, and then send specified amount data to SD card. After data-out, SD controller will get CRC status from SD card and check its correctness; it should be '010', otherwise CRC_IF (SDH_INTSTS[1]) will be set and CRCSTAT (SDH_INTSTS[6:4]) will be the value it received.

If R2_EN (SDH_CTL[4]) is triggered, SD controller will receive response R2 (136 bits) from SD card, CRC-7 and end bit will be dropped. The receiving data will be placed at DMA's buffer, starting from address offset 0x0.

This SD controller also provides multiple block transfer function (change BLKLEN (SDH_BLEN[10:0]) to change the block length) to accelerate data transfer throughput. If CRC-7, CRC-16 or CRC status is error, SD controller will stop transfer and set CRC_IF (SDH_INTSTS[1]) high. In this situation, the SDH has to be reset.

There is a hardware time-out mechanism for response in and data in inside SD engine. Specify a 24-bit time-out value at TMOUT (SDH_TMOUT[23:0]), and then SDH controller will decide when to time-out based on this value.

### 5.26.6 Registers Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH Base Address:**<br>**SDH_BA = 0xB000_C000** | | | | |
| **SDH_FB_n**<br>**n = 0,1…31** | SDH_BA+0x000 + 0x4 * n | R/W | SD Host Embedded Buffer Word n<br>n = 0,1…31 | 0x0000_0000 |
| **SDH_DMACTL** | SDH_BA+0x400 | R/W | SD Host DMA Control and Status Register | 0x0000_0000 |
| **SDH_DMASA** | SDH_BA+0x408 | R/W | SD Host DMA Transfer Starting Address Register | 0x0000_0000 |
| **SDH_DMABCNT** | SDH_BA+0x40C | R | SD Host DMA Transfer Byte Count Register | 0x0000_0000 |
| **SDH_DMAINTEN** | SDH_BA+0x410 | R/W | SD Host DMA Interrupt Enable Register | 0x0000_0001 |
| **SDH_DMAINTSTS** | SDH_BA+0x414 | R/W | SD Host DMA Interrupt Status Register | 0x0000_0000 |
| **SDH_GCTL** | SDH_BA+0x800 | R/W | SD Host Global Control and Status Register | 0x0000_0000 |
| **SDH_GINTEN** | SDH_BA+0x804 | R/W | SD Host Global Interrupt Control Register | 0x0000_0001 |
| **SDH_GINTSTS** | SDH_BA+0x808 | R/W | SD Host Global Interrupt Status Register | 0x0000_0000 |
| **SDH_CTL** | SDH_BA+0x820 | R/W | SD Host Control and Status Register | 0x0101_0000 |
| **SDH_CMD** | SDH_BA+0x824 | R/W | SD Host Command Argument Register | 0x0000_0000 |
| **SDH_INTEN** | SDH_BA+0x828 | R/W | SD Host Interrupt Enable Register | 0x0000_0A00 |
| **SDH_INTSTS** | SDH_BA+0x82C | R/W | SD Host Interrupt Status Register | 0x000X_008C |
| **SDH_RESP0** | SDH_BA+0x830 | R | SD Host Receiving Response Token Register 0 | 0x0000_0000 |
| **SDH_RESP1** | SDH_BA+0x834 | R | SD Host Receiving Response Token Register 1 | 0x0000_0000 |
| **SDH_BLEN** | SDH_BA+0x838 | R/W | SD Host Block Length Register | 0x0000_01FF |
| **SDH_TMOUT** | SDH_BA+0x83C | R/W | SD Host Response/Data-in Time-out Register | 0x0000_0000 |
| **SDH_ECTL** | SDH_BA+0x840 | R/W | SD Host Extend Control Register | 0x0000_0003 |

### 5.26.7 Register Description

## SD Host DMA Control and Status Register (SDH_DMACTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_DMACTL** | SDH_BA+0x400 | R/W | SD Host 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**<br>This bit indicates if SD Host is granted and doing DMA transfer or not.<br>0 = DMA transfer is not in progress.<br>1 = DMA transfer is in progress. |
| [8:4] | **Reserved** | Reserved. |
| [3] | **SGEN** | **Scatter-gather Function Enable**<br>0 = Scatter-gather function Disabled (DMA will treat the starting address in SDH_DMASA as starting pointer of a single block memory).<br>1 = Scatter-gather function Enabled (DMA will treat the starting address in SDH_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**<br>0 = No effect.<br>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.<br>**Note:** The software reset DMA related registers. |
| [0] | **DMAEN** | **DMAC Engine Enable**<br>0 = DMA Disabled.<br>1 = DMA Enabled.<br>If this bit is cleared, DMA will ignore all requests from SD host and force bus master into IDLE state.<br>**Note:** If target abort is occurred, DMAEN will be cleared automatically. |

### SD Host DMA Transfer Starting Address Register (SDH_DMASA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDH_DMASA | SDH_BA+0x408 | R/W | SD Host DMA Transfer Starting Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **ADDR** | **DMA Transfer Starting Address**<br>This field indicates a 32-bit starting address of system memory (SRAM/SDRAM) for DMA to retrieve or fill in data.<br>If DMA is not in normal mode, this field will be interpreted as a starting address of Physical Address Descriptor (PAD) table. |
| [0] | **ORDER** | **Determined to the PAD Table Fetching Is in Order or Out of Order**<br>0 = PAD table is fetched in order.<br>1 = PAD table is fetched out of order.<br>**Note:** the bit 0 is valid in scatter-gather mode when SGEN (SDH_DMACTL[3]) = 1. |

**NOTE:** Starting address of the SDRAM must be word aligned, for example, 0x0000_0000, 0x0000_0004, etc.

The diagram shown below describes the format of PAD table. Note that the total byte count of all Pads must be equal to the byte count filled in FMI engine.

The EOT is the End of PAD Table. The EOT should be set to 1 in the last descriptor.

The byte count has to be the multiple of 4 bytes.

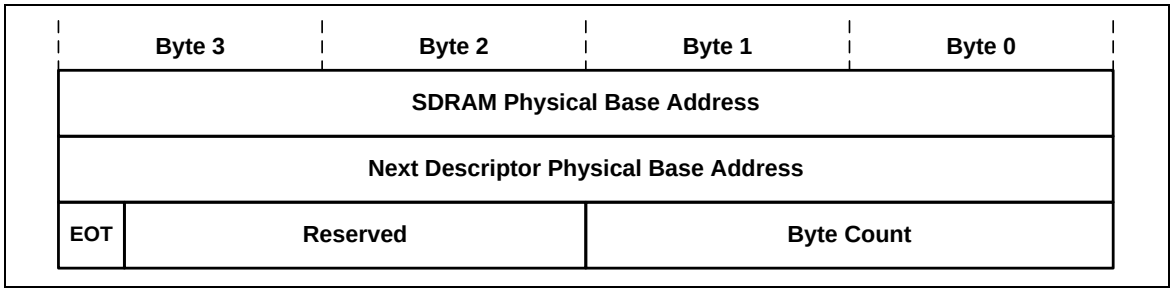



Figure 5.26-1 PAD (Physical Address Descriptor) Table Format

The diagram shown below indicates how SDH feteched the PAD tables. SDH feteched next PAD

tables sequentially if ORDER (SDH_DMASA[0]) set as low. SDH fetehced next PAD tables based on the Next Descriptor Physical Base Address of PAD table if ORDER (SDH_DMASA[0]) set as high.

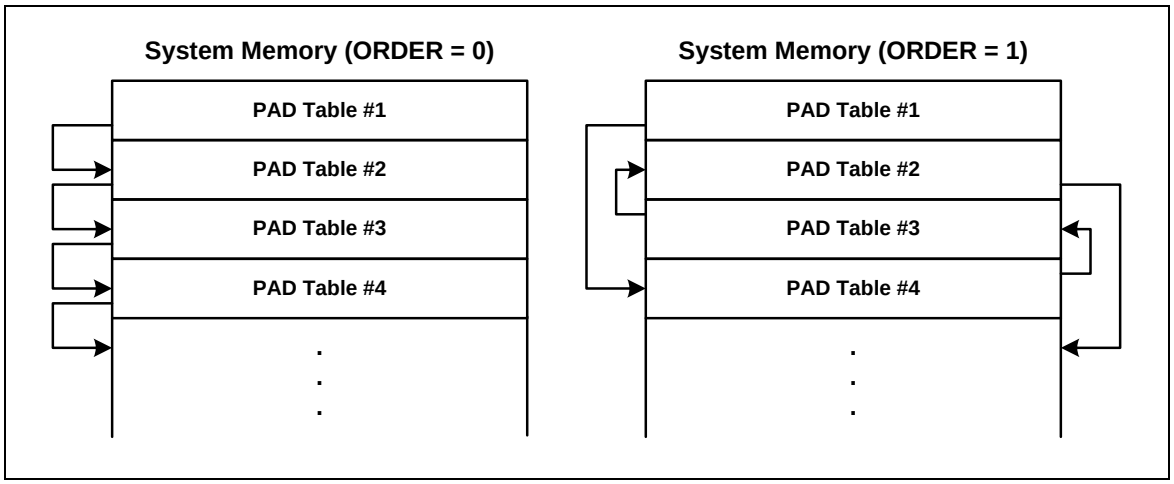



Figure 5.26-2 PAD (Physical Address Descriptor) Table Fetch Modes

## SD Host DMA Transfer Byte Count Register (SDH_DMABCNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_DMABCNT** | SDH_BA+0x40C | R | SD Host 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)**<br>This field indicates the remained byte count of DMA transfer. The value of this field is valid only when FMI is busy; otherwise, it is zero. |

## SD Host DMA Interrupt Enable Register (SDH_DMAINTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_DMAINTEN** | SDH_BA+0x410 | R/W | SD Host DMA Interrupt Enable 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** | | | | | | **WEOT_IE** | **TABORT_IE** |

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [1] | **WEOT_IE** | **Wrong EOT (End of Transfer) Encountered Interrupt Enable**<br>0 = Interrupt generation Disabled when wrong EOT (End of Transfer) is encountered.<br>1 = Interrupt generation Enabled when wrong EOT (End of Transfer) is encountered. |
| [0] | **TABORT_IE** | **DMA Read/Write Target Abort Interrupt Enable**<br>0 = Target abort interrupt generation Disabled during DMA transfer.<br>1 = Target abort interrupt generation Enabled during DMA transfer. |

### SD Host DMA Interrupt Status Register (SDH_DMAINTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDH_DMAINTSTS | SDH_BA+0x414 | R/W | SD Host 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 | | | | | | WEOT_IF | TABORT_IF |

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [1] | **WEOT_IF** | **Wrong EOT Encountered Interrupt Flag**<br>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 FMI), this bit will be set.<br>0 = No EOT encountered before DMA transfer finished.<br>1 = EOT encountered before DMA transfer finished.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [0] | **TABORT_IF** | **DMA Read/Write Target Abort Interrupt Flag**<br>0 = No bus ERROR response received.<br>1 = Bus ERROR response received.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |

**NOTE:** When DMA's bus master received ERROR response, it means that target abort is happened. DMA will stop transfer and respond this event to software and then SD host goes to IDLE state. When target abort occurred or WEOT_IF is set, software must reset DMA and IP, and then transfer those data again.

## SD Host Global Control and Status Register (SDH_GCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDH_GCTL | SDH_BA+0x800 | R/W | SD Host 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**<br>0 = SD host functionality Disabled.<br>1 = SD host functionality Enabled. |
| [0] | **GCTLRST** | **Software Engine Reset**<br>0 = Writing 0 to this bit has no effect.<br>1 = Writing 1 to this bit to reset all SD host. The contents of control register will not be cleared. This bit will auto clear after few clock cycles. |

## SD Host Global Interrupt Control Register (SDH_GINTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_GINTEN** | SDH_BA+0x804 | R/W | SD Host 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** | | | | | | | **DTA_IE** |

| Bits | Description | |
|---|---|---|
| [31:1] | **Reserved** | Reserved. |
| [0] | **DTA_IE** | **DMA READ/WRITE Target Abort Interrupt Enable**<br>0 = DMAC READ/WRITE target abort interrupt generation Disabled.<br>1 = DMAC READ/WRITE target abort interrupt generation Enabled. |

### SD Host Global Interrupt Status Register (SDH_GINTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_GINTSTS** | SDH_BA+0x808 | R/W | SD Host 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** | | | | | | | **DTA_IF** |

| Bits | Description | |
|---|---|---|
| [31:1] | **Reserved** | Reserved. |
| [0] | **DTA_IF** | **DMAC READ/WRITE Target Abort Interrupt Flag (Read Only)**<br>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.<br>0 = No bus ERROR response received.<br>1 = Bus ERROR response received.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |

**NOTE:** No matter interrupt enable is turn on or not, the interrupt flag will be set when target condition is occurred.

## SD Host Control and Status Register (SDH_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDH_CTL | SDH_BA+0x820 | R/W | SD Host Control and Status Register | 0x0101_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CLK_KEEP1 | SDPORT | | Reserved | SDNWR | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BLK_CNT | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| DBW | SW_RST | CMD_CODE | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CLK_KEEP0 | CLK8_OE | CLK74_OE | R2_EN | DO_EN | DI_EN | RI_EN | CO_EN |

| Bits | Description | |
|---|---|---|
| [31] | **CLK_KEEP1** | **SD Host Port 1 Clock Keep Running Enable**<br>0 = SD host port 1 clock generation controlled by SD host automatically.<br>1 = SD host port 1 clock always keeps free running. |
| [30:29] | **SDPORT** | **SD Port Selection**<br>00 = SD host port 0 is selected.<br>01 = SD host port 1 is selected.<br>10 = Reserved.<br>11 = Reserved. |
| [28] | **Reserved** | Reserved. |
| [27:24] | **SDNWR** | **NWR Parameter for Block Write Operation**<br>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] | **BLK_CNT** | **Block Counts to Be Transferred or Received**<br>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. Don't fill 0x0 to this field.<br>**Note:** For READ_MULTIPLE_BLOCK and WRITE_MULTIPLE_BLOCK command, the actual total length is BLK_CNT * (BLKLEN (SDH_BLEN[10:0]) +1). |
| [15] | **DBW** | **SD Data Bus Width (for 1-bit / 4-bit Selection)**<br>0 = Data bus width is 1-bit.<br>1 = Data bus width is 4-bit. |
| [14] | **SW_RST** | **Software Engine Reset**<br>0 = Writing 0 to this bit has no effect.<br>1 = Writing 1 to this bit will reset the internal state machine and counters. The contents of control register will not be cleared (but RI_EN (SDH_CTL[1]), DI_EN (SDH_CTL[2]), DO_EN (SDH_CTL[3]) and R2_EN (SDH_CTL[4]) will be cleared). This bit will be auto cleared after few clock cycles. |

| | | |
|---|---|---|
| [13:8] | **CMD_CODE** | **SD Command Code**<br>This register contains the SD command code (0x00 – 0x3F). |
| [7] | **CLK_KEEP0** | **SD Host Port 0 Clock Keep Running Enable**<br>0 = SD host port 0 clock generation controlled by SD host automatically.<br>1 = SD host port 0 clock always keeps free running. |
| [6] | **CLK8_OE** | **Generating 8 Clock Cycles Output Enable**<br>0 = No effect. (Please use SW_RST (SDH_CTL[14]) to clear this bit.)<br>1 = Enable, SD host will output 8 clock cycles.<br>**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] | **CLK74_OE** | **Initial 74 Clock Cycles Output Enable**<br>0 = No effect. (Please use SW_RST (SDH_CTL[14]) to clear this bit.)<br>1 = Enable, SD host will output 74 clock cycles to SD card.<br>**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] | **R2_EN** | **Response R2 Input Enable**<br>0 = No effect. (Please use SW_RST (SDH_CTL[14]) to clear this bit.)<br>1 = Enable, SD host will wait to receive a response R2 from SD card and store the response data into DMAC's flash buffer (exclude CRC-7).<br>**NOTE:** When operation is finished, this bit will be cleared automatically, so don't write 0 to this bit (the controller will be abnormal). |
| [3] | **DO_EN** | **Data Output Enable**<br>0 = No effect. (Please use SW_RST (SDH_CTL[14]) to clear this bit.)<br>1 = Enable, SD host will transfer block data and the CRC-16 value to SD card.<br>**NOTE:** When operation is finished, this bit will be cleared automatically, so don't write 0 to this bit (the controller will be abnormal). |
| [2] | **DI_EN** | **Data Input Enable**<br>0 = No effect. (Please use SW_RST (SDH_CTL[14]) to clear this bit.)<br>1 = Enable, SD host will wait to receive block data and the CRC-16 value from SD card.<br>**NOTE:** When operation is finished, this bit will be cleared automatically, so don't write 0 to this bit (the controller will be abnormal). |
| [1] | **RI_EN** | **Response Input Enable**<br>0 = No effect. (Please use SW_RST (SDH_CTL[14]) to clear this bit.)<br>1 = Enable, SD host will wait to receive a response from SD card.<br>**NOTE:** When operation is finished, this bit will be cleared automatically, so don't write 0 to this bit (the controller will be abnormal). |
| [0] | **CO_EN** | **Command Output Enable**<br>0 = No effect. (Please use SW_RST (SDH_CTL[14]) to clear this bit.)<br>1 = Enable, SD host will output a command to SD card.<br>**NOTE:** When operation is finished, this bit will be cleared automatically, so don't write 0 to this bit (the controller will be abnormal). |

## SD Host Command Argument Register (SDH_CMD)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_CMD** | SDH_BA+0x824 | R/W | SD Host Command Argument Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **SD_CMD_ARG** | **SD Command Argument**<br>This register contains a 32-bit value specifies the argument of SD command from host controller to SD card. Before trigger CO_EN (SDH_CTL[0]), software should fill argument in this field. |

## SD Host Interrupt Enable Register (SDH_INTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_INTEN** | SDH_BA+0x828 | R/W | SD Host Interrupt Enable Register | 0x0000_0A00 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **CD1SRC** | **CD0SRC** | **Reserved** | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | **DITO_IE** | **RITO_IE** | **SDIO1_IE** | **SDIO0_IE** | **CD1_IE** | **CD0_IE** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | | | | **CRC_IE** | **BLKD_IE** |

| Bits | Description | |
|---|---|---|
| [31] | **CD1SRC** | **SD Port 1 Card Detect Source Selection**<br>0 = From SD port 1 data bit 3, the pin SD1_DAT3.<br>Host need clock to got data on pin SD1_DAT3. Please make sure CLK_KEEP1 (SDH_CTL[31]) is 1 to generate free running clock for SD1_DAT3 pin.<br>1 = From SD port 1 card detection pin, the pin SD1_nCD. |
| [30] | **CD0SRC** | **SD Port 0 Card Detect Source Selection**<br>0 = From SD port 0 data bit 3, the pin SD0_DAT3.<br>Host need clock to got data on pin SD0_DAT3. Please make sure CLK_KEEP0 (SDH_CTL[7]) is 1 to generate free running clock for SD0_DAT3 pin.<br>1 = From SD port 0 card detection pin, the pin SD0_nCD. |
| [29:14] | **Reserved** | Reserved. |
| [13] | **DITO_IE** | **Data Input Time-out Interrupt Enable**<br>Enable/Disable interrupts generation of SD controller when data input time-out. Time-out value is specified at TMOUT (SDH_TMOUT[23:0]).<br>0 = Disable.<br>1 = Enable. |
| [12] | **RITO_IE** | **Response Time-out Interrupt Enable**<br>Enable/Disable interrupts generation of SD controller when receiving response or R2 time-out. Time-out value is specified at (SDH_TMOUT[23:0]).<br>0 = Disable.<br>1 = Enable. |
| [11] | **SDIO1_IE** | **SDIO Interrupt Enable for Port 1**<br>Enable/Disable interrupts generation of SD host when SDIO card 1 issue an interrupt via pin SD1_DAT1 to host.<br>0 = Disable.<br>1 = Enable. |

| | | |
|---|---|---|
| [10] | **SDIO0_IE** | **SDIO Interrupt Enable for Port 0**<br>Enable/Disable interrupts generation of SD host when SDIO card 0 issue an interrupt via pin SD0_DAT1 to host.<br>0 = Disable.<br>1 = Enable. |
| [9] | **CD1_IE** | **SD1 Card Detection Interrupt Enable**<br>Enable/Disable interrupts generation of SD controller when card 1 is inserted or removed.<br>0 = Disable.<br>1 = Enable. |
| [8] | **CD0_IE** | **SD0 Card Detection Interrupt Enable**<br>Enable/Disable interrupts generation of SD controller when card 0 is inserted or removed.<br>0 = Disable.<br>1 = Enable. |
| [7:2] | **Reserved** | Reserved. |
| [1] | **CRC_IE** | **CRC-7, CRC-16 and CRC Status Error Interrupt Enable**<br>0 = SD host will not generate interrupt when CRC-7, CRC-16 and CRC status is error.<br>1 = SD host will generate interrupt when CRC-7, CRC-16 and CRC status is error. |
| [0] | **BLKD_IE** | **Block Transfer Done Interrupt Enable**<br>0 = SD host will not generate interrupt when data-in (out) transfer done.<br>1 = SD host will generate interrupt when data-in (out) transfer done. |

## SD Host Interrupt Status Register (SDH_INTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDH_INTSTS | SDH_BA+0x82C | R/W | SD Host Interrupt Status Register | 0x000X_008C |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | SD1DAT1 | SD0DAT1 | CDPS1 | CDPS0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | DITO_IF | RITO_IF | SDIO1_IF | SDIO0_IF | CD1_IF | CD0_IF |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SDDAT0 | CRCSTAT | | | CRC16 | CRC7 | CRC_IF | BLKD_IF |

| Bits | Description | |
|---|---|---|
| [31:20] | **Reserved** | Reserved. |
| [19] | **SD1DAT1** | **DAT1 Pin Status of SD1 (Read Only)**<br>This bit reflects the DAT1 pin status of SD port 1. |
| [18] | **SD0DAT1** | **DAT1 Pin Status of SD0 (Read Only)**<br>This bit reflects the DAT1 pin status of SD port 0. |
| [17] | **CDPS1** | **Card Detect Status of SD1 (Read Only)**<br>This bit is the card detect pin status of SD1, and it is using for card detection. When there is a card inserted in or removed from SD1, software should check this bit to confirm if there is really a card insertion or remove.<br>If CD1SRC (SDH_INTEN[31]) = 0, select SD1_DAT3 for card detect.<br>0 = card removed.<br>1 = card inserted.<br>If CD1SRC (SDH_INTEN[31]) = 1, select SD1_nCD for card detect.<br>0 = card inserted.<br>1 = card removed. |
| [16] | **CDPS0** | **Card Detect Status of SD0 (Read Only)**<br>This bit is the card detect pin status of SD0, and it is using for card detection. When there is a card inserted in or removed from SD0, software should check this bit to confirm if there is really a card insertion or remove.<br>If CD0SRC (SDH_INTEN[30]) = 0 to select SD0_DAT3 for card detect.<br>0 = card removed.<br>1 = card inserted.<br>If CD0SRC (SDH_INTEN[30]) = 1 to select SD0_nCD for card detect.<br>0 = card inserted.<br>1 = card removed. |
| [15:14] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [13] | **DITO_IF** | **Data Input Time-out Interrupt Flag (Read Only)**<br>This bit indicates that SD host counts to time-out value when receiving data (waiting start bit).<br>0 = Not time-out.<br>1 = Data input time-out.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [12] | **RITO_IF** | **Response Time-out Interrupt Flag (Read Only)**<br>This bit indicates that SD host counts to time-out value when receiving response or R2 (waiting start bit).<br>0 = Not time-out.<br>1 = Response time-out.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [11] | **SDIO1_IF** | **SDIO 1 Interrupt Flag (Read Only)**<br>This bit indicates that SDIO card 1 issues an interrupt to host. This interrupt is designed to level sensitive. Before clear it, turn off SDIO1_IE (SDH_INTEN[11]) first.<br>0 = No interrupt is issued by SDIO card 1.<br>1 = an interrupt is issued by SDIO card 1.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [10] | **SDIO0_IF** | **SDIO 0 Interrupt Flag (Read Only)**<br>This bit indicates that SDIO card 0 issues an interrupt to host. This interrupt is designed to level sensitive. Before clear it, turn off SDIO0_IE (SDH_INTEN[10]) first.<br>0 = No interrupt is issued by SDIO card 0.<br>1 = an interrupt is issued by SDIO card 0.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [9] | **CD1_IF** | **SD0 Card Detection Interrupt Flag (Read Only)**<br>This bit indicates that SD card 1 is inserted or removed. Only when CD1_IE (SDH_INTEN[9]) is set to 1, this bit is active.<br>0 = No card is inserted or removed.<br>1 = There is a card inserted in or removed from SD1.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [8] | **CD0_IF** | **SD0 Card Detection Interrupt Flag (Read Only)**<br>This bit indicates that SD card 0 is inserted or removed. Only when CD1_IE (SDH_INTEN[8]) is set to 1, this bit is active.<br>0 = No card is inserted or removed.<br>1 = There is a card inserted in or removed from SD0.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [7] | **SDDAT0** | **DAT0 Pin Status of Current Selected SD Port (Read Only)**<br>This bit is the DAT0 pin status of current selected SD port. |
| [6:4] | **CRCSTAT** | **CRC Status Value of Data-out Transfer (Read Only)**<br>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.<br>010 = Positive CRC status.<br>101 = Negative CRC status.<br>111 = SD card programming error occurs. |

| | | |
|---|---|---|
| [3] | **CRC16** | **CRC-16 Check Status of Data-in Transfer (Read Only)**<br>SD host will check CRC-16 correctness after data-in transfer.<br>0 = Fault.<br>1 = OK. |
| [2] | **CRC7** | **CRC-7 Check Status (Read Only)**<br>SD host will check CRC-7 correctness during each response in. If that response does not contain CRC-7 information (ex. R3), then software should turn off CRC_IE (SDH_INTEN[1]) and ignore this bit.<br>0 = Fault.<br>1 = OK. |
| [1] | **CRC_IF** | **CRC-7, CRC-16 and CRC Status Error Interrupt Flag (Read Only)**<br>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 CRC-7 information with it; SD host will still calculate CRC-7, get CRC error and set this flag. In this condition, software should ignore CRC error and clears this bit manually.<br>0 = No CRC error is occurred.<br>1 = CRC error is occurred.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |
| [0] | **BLKD_IF** | **Block Transfer Done Interrupt Flag (Read Only)**<br>This bit indicates that SD host has finished all data-in or data-out block transfer. If there is a CRC-16 error or incorrect CRC status during multiple block data transfer, the transfer will be broken and this bit will also be set.<br>0 = Not finished yet.<br>1 = Done.<br>**NOTE:** This bit is read only, but can be cleared by writing '1' to it. |

### SD Host Receiving Response Token Register 0 (SDH_RESP0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_RESP0** | SDH_BA+0x830 | R | SD Host Receiving Response Token Register 0 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **SD_RSP_TK0** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **SD_RSP_TK0** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **SD_RSP_TK0** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **SD_RSP_TK0** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:0] | **SD_RSP_TK0** | **SD Receiving Response Token 0**<br>SD host controller will receive a response token for getting a reply from SD card when RI_EN (SDH_CTL[1]) is set. This field contains response bit 47-16 of the response token. |

## SD Host Receiving Response Token Register 1 (SDH_RESP1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_RESP1** | SDH_BA+0x834 | R | SD Host 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** |
| **SD_RSP_TK1** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [7:0] | **SD_RSP_TK1** | **SD Receiving Response Token 1**<br>SD host controller will receive a response token for getting a reply from SD card when RI_EN (SDH_CTL[1]) is set. This register contains the bit 15-8 of the response token. |

## SD Host Block Length Register (SDH_BLEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_BLEN** | SDH_BA+0x838 | R/W | SD Host 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 | |
|---|---|---|
| [31:11] | **Reserved** | Reserved. |
| [10:0] | **BLKLEN** | **SD Block Length in Byte Unit**<br>An 11-bit value specifies the SD transfer byte count of a block. The actual byte count is equal to BLKLEN+1.<br>**Note :** The default SD block length is 512 bytes |

## SD Host Response/Data-in Time-out Register (SDH_TMOUT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_TMOUT** | SDH_BA+0x83C | R/W | SD Host 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** |
| **TMOUT** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **TMOUT** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **TMOUT** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **TMOUT** | **SD Response/Data-in Time-out Value**<br>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 is depended 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.<br>**NOTE:** Fill 0x0 into this field will disable hardware time-out function. |

## SD Host Extend Control Register (SDH_ECTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDH_ECTL | SDH_BA+0x840 | R/W | SD Host Extend Control Register | 0x0000_0003 |

| 31 | 30 | 29 | 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 | | | | | | PWROFF1 | PWROFF0 |

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [1] | **PWROFF1** | **SD Port 1 Power Disable**<br>Set this bit low to make the SD1_nPWR functional pin low to turn on the power of SD port 1.<br>0 = SD port 1 power Enabled.<br>1 = SD port 1 power Disabled. |
| [0] | **PWROFF0** | **SD Port 0 Power Disable**<br>Set this bit low to make the SD0_nPWR functional pin low to turn on the power of SD port 0.<br>0 = SD port 0 power Enabled.<br>1 = SD port 1 power Disabled. |

## 5.27 Cryptographic Accelerator (CRYPTO)

### 5.27.1 Overview

The Crypto (Cryptographic Accelerator) includes a secure pseudo random number generator (PRNG) core and supports AES, DES/TDES, SHA and HMAC algorithms.

The PRNG core supports 64 bits, 128 bits, 192 bits, and 256 bits random number generation.

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, and CBC-CS3 mode.

The DES/TDES accelerator is an implementation fully compliant with the DES and Triple DES encryption/decryption algorithm. The DES/TDES accelerator supports ECB, CBC, CFB, OFB, and CTR mode.

The SHA accelerator is an implementation fully compliant with the SHA-160, SHA-224, SHA-256, SHA-384, and SHA-512 and corresponding HMAC algorithms.

### 5.27.2 Features

- PRNG
  - Supports 64 bits, 128 bits , 192 bits, and 256 bits random number generation
- 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 mode
  - Supports key expander
- DES
  - Supports FIPS 46-3
  - Supports both encryption and decryption
  - Supports ECB, CBC, CFB, OFB, and CTR mode
- TDES
  - Supports FIPS NIST 800-67
  - Implemented according to the X9.52 standard
  - Supports two keys or three keys mode
  - Supports both encryption and decryption
  - Supports ECB, CBC, CFB, OFB, and CTR mode
- SHA
  - Supports FIPS NIST 180, 180-2
  - Supports SHA-160, SHA-224, SHA-256, SHA-384, and SHA-512

- HMAC
  - Supports FIPS NIST 180, 180-2
  - Supports HMAC-SHA-160, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512

### 5.27.3 Block Diagram

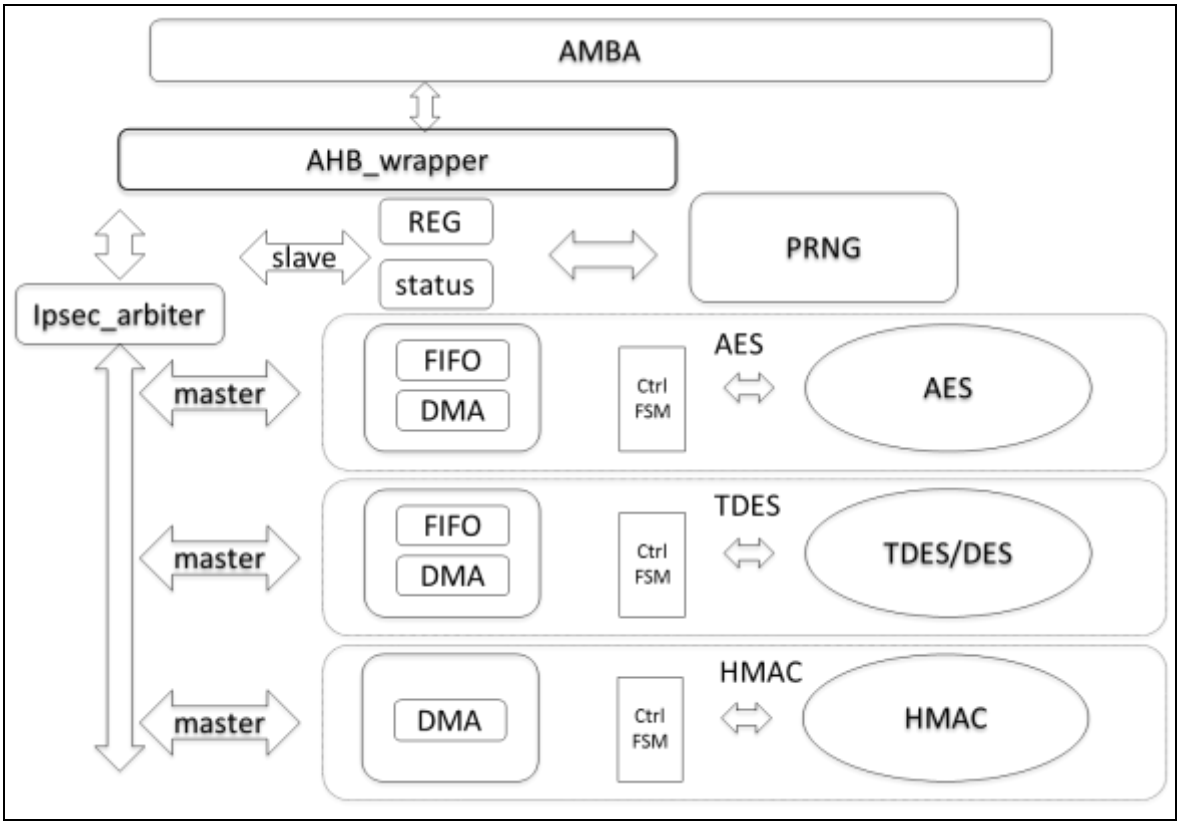



Figure 5.27-1 Cryptographic Accelerator Block Diagram

### 5.27.4 Basic Configuration

Before using cryptographic engine, it's necessary to enable clock of cryptographic engine. Set CRYPTO (CLK_HCLKEN[23]) high to enable clock for cryptographic engine operation.

### 5.27.5 Functional description

The cryptographic accelerator includes a secure pseudo random number generator (PRNG) core and supports AES, DES/TDES, SHA, and HMAC 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 64 bits, 128 bits, 192 bits, and 256 bits random number generation configured by KEYSZ.
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, and CBC-CS3 mode. 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 DES/TDES accelerator is a fully compliant implementation of the DES and Triple DES encryption/decryption algorithm. The DES/TDES accelerator supports ECB, CBC, CFB, OFB, and CTR mode. The DES/TDES accelerator also supports the DMA function to reduce the CPU intervention. Only two burst lengths, four words and eight words, are supported.
4. The SHA/HMAC accelerator is a fully compliant implementation of the SHA-160, SHA-224, SHA-256, SHA-384, SHA-512, and corresponding HMAC algorithm. 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.

Software can control the data flow by enabling the CRPT_INTEN, and monitor the accelerator status by checking the CRPT_INTSTS.

The cryptographic accelerator supports the following features to enhance the performance.

1. **DMA mode:** Once DMA source address register, destination address register, and byte count register are configurated 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 four hardware DMA channels for AES engine, four hardware DMA channels for DES/TDES engine, and one hardware DMA channel for SHA/HMAC engine.
2. **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.
3. **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 was feeding to cryptographic engine via writing to data input register.
4. **Channel Expansion mode:** In this mode, several virtual channels in one of four DMA channels are feasible in AES or DES/TDES mode. The total channel number can exceed the limit of four DMA channels. The intermediate data from feedback registers (CRPT_AES_FDBCKx, CRPT_TDES_FDBCKH, and CRPT_TDES_FDBCKL) should be stored temporarily in data SRAM. And switch 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 (CRPT_AESn_IVx, CRPY_TDESx_IVH, and CRPT_TDESx_IVL) for the accelerator to continue the operation with the original configuration setting. Note that, in ECB mode, there is no need to move the intermediate data from feedback registers to IV.

5.27.5.1 *PRNG (Pseudo Random number Generator)*

The PRNG block diagram is depicted below. The core supports 64 bits, 128 bits, 192 bits, and 256 bits random number generation configured by KEYSZ(CRPT_PRNG_CTL[3:2] ).

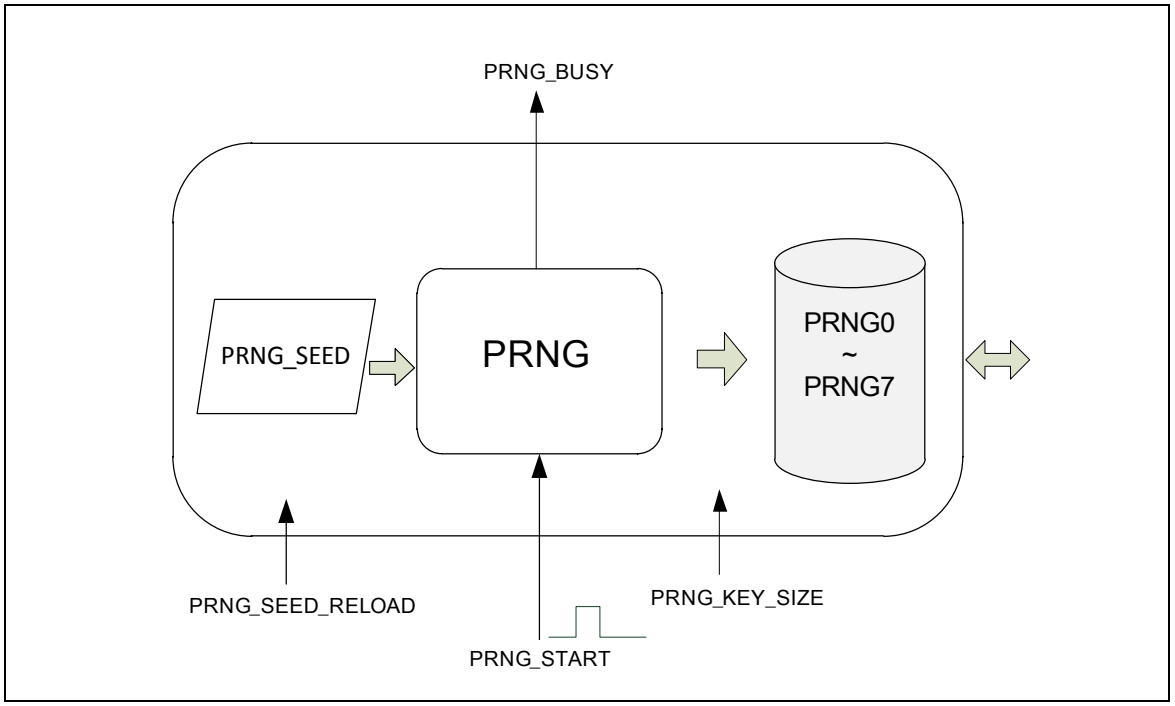



Figure 5.27-2 PRNG Function Diagram

Program steps to get the pseudo random number are depicted below.

1. Check the BUSY(CRPT_PRNG_CTL[8]) until it comes to 0.
2. Initialize PRNG parameters. Configure KEYSZ (CRPT_PRNG_CTL[3:2]), and write a random seed to CRPT_PRNG _SEED. Note that CRPT_PRNG_SEED should be initialized since it's not initialized as the chip powers up.
3. Configure PRNG control register CRPT_PRNG_CTL.
4. Software checks BUSY(CRPT_PRNG_CTL[8]) until it comes to 0, or waits for the PRNG done interrupt (must enable the corresponding interrupt enable register). Then software can read the output random numbers from CRPT_PRNG_KEY0 ~ CRPT_PRNG_KEY7.

5.27.5.2 *AES (Advanced Encryption Standard)*

**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's 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.

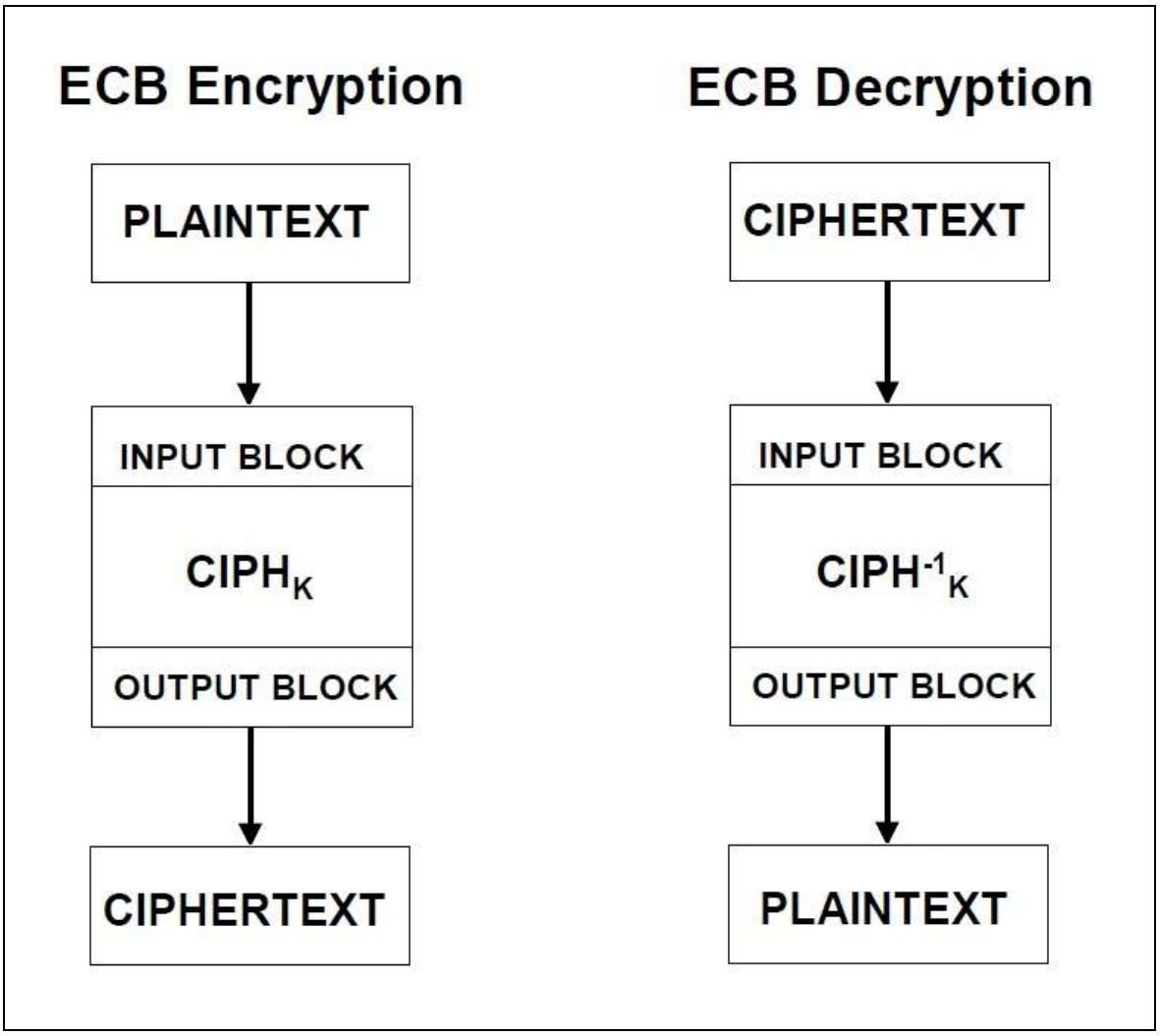



Figure 5.27-3 Electronic Codebook Mode

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.

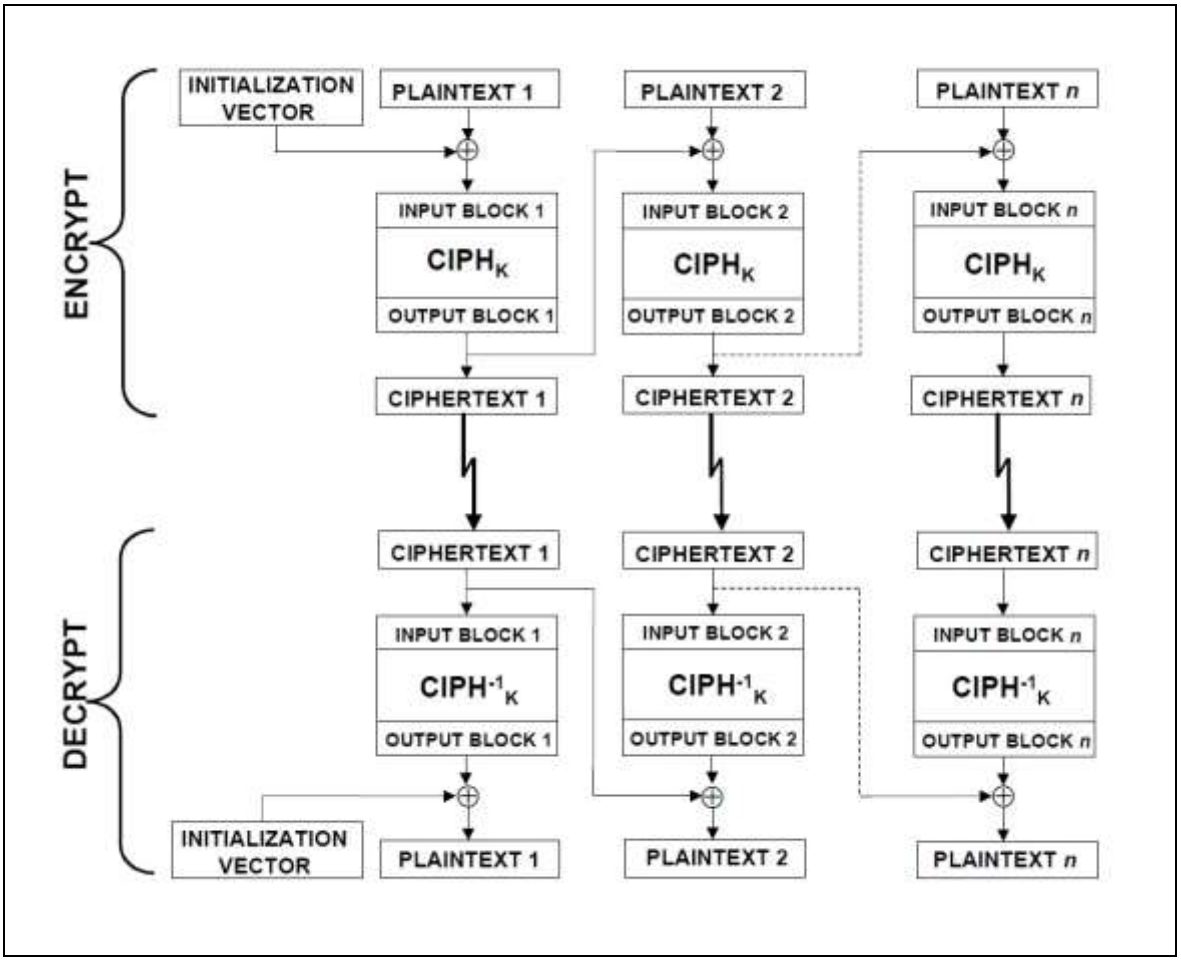



Figure 5.27-4 Cipher Block Chaining Mode

**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 need not be secret, but it must be unpredictable.

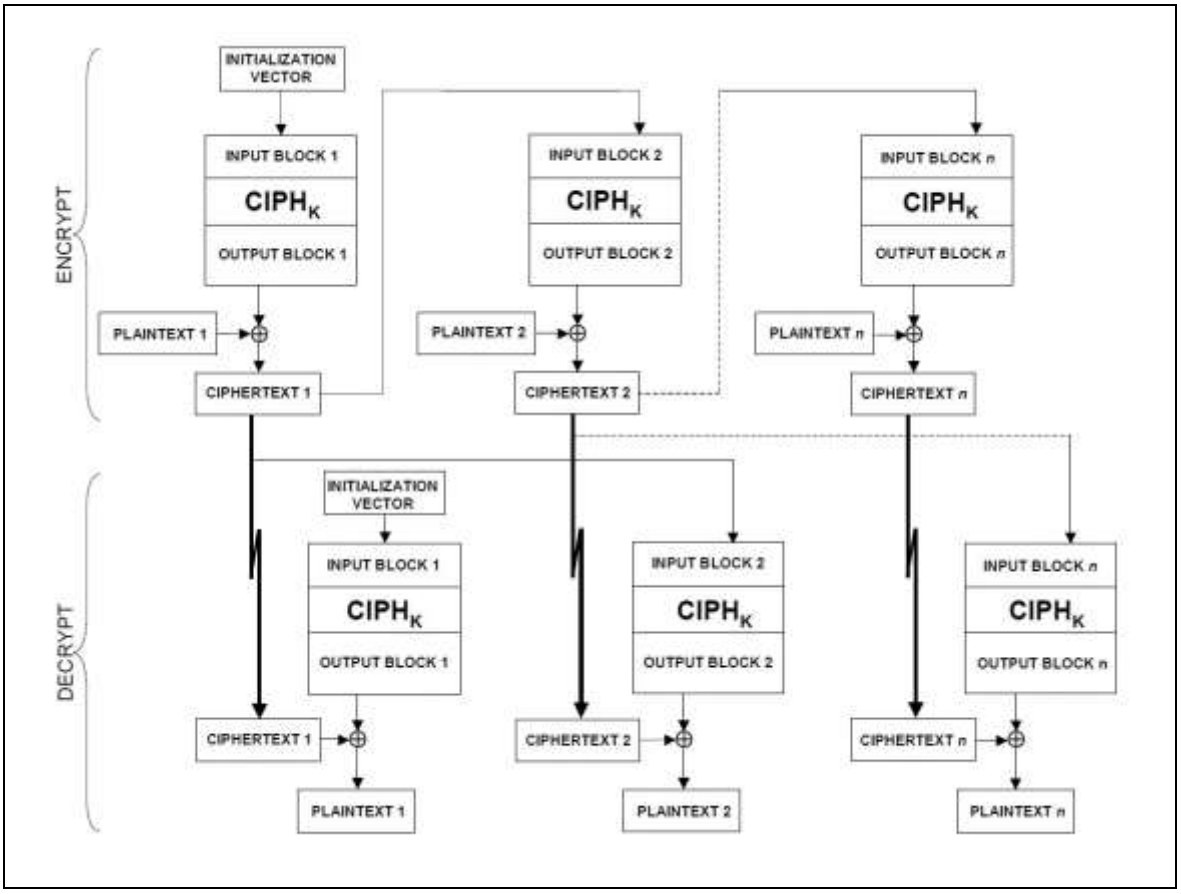



Figure 5.27-5 Cipher Feedback Mode

**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 be 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.

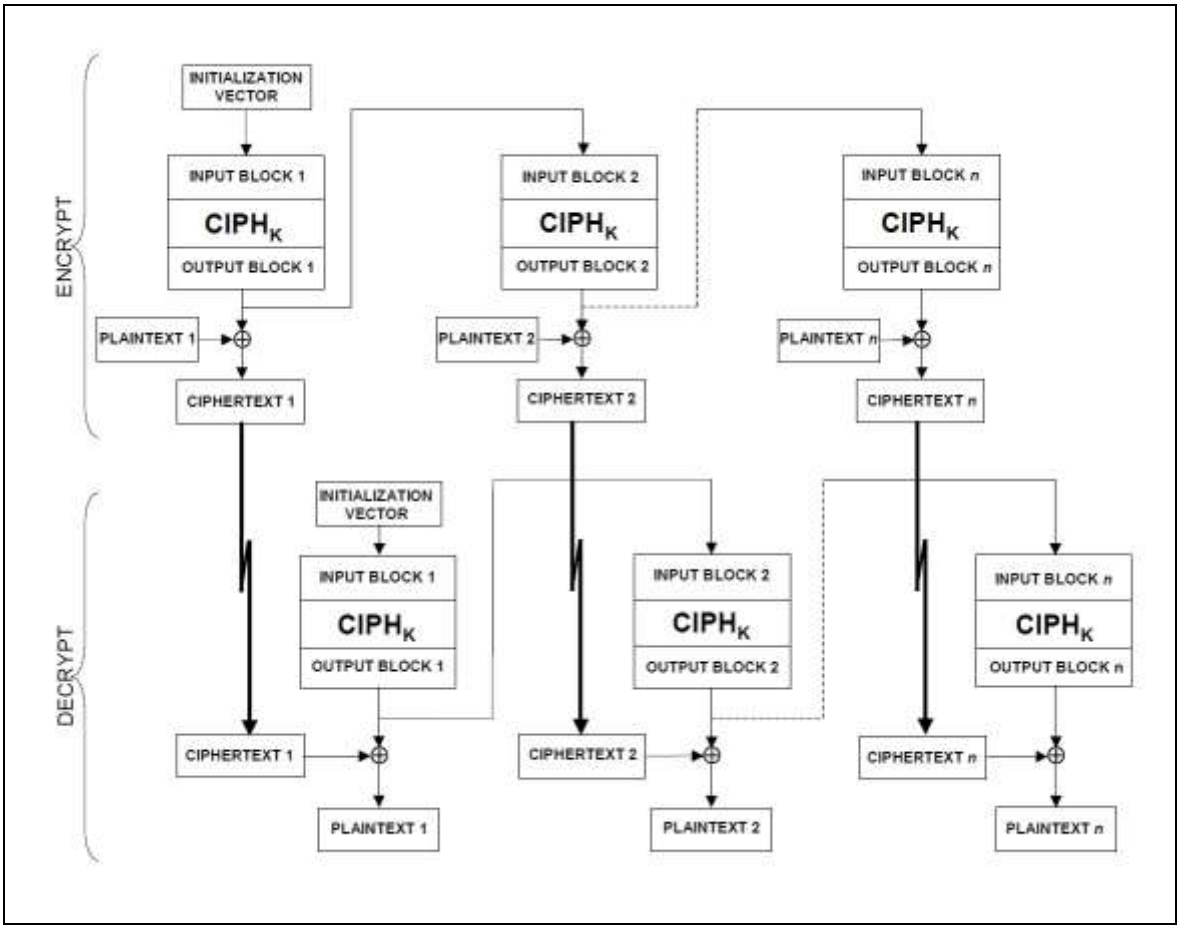



Figure 5.27-6 Output Feedback Mode

**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.

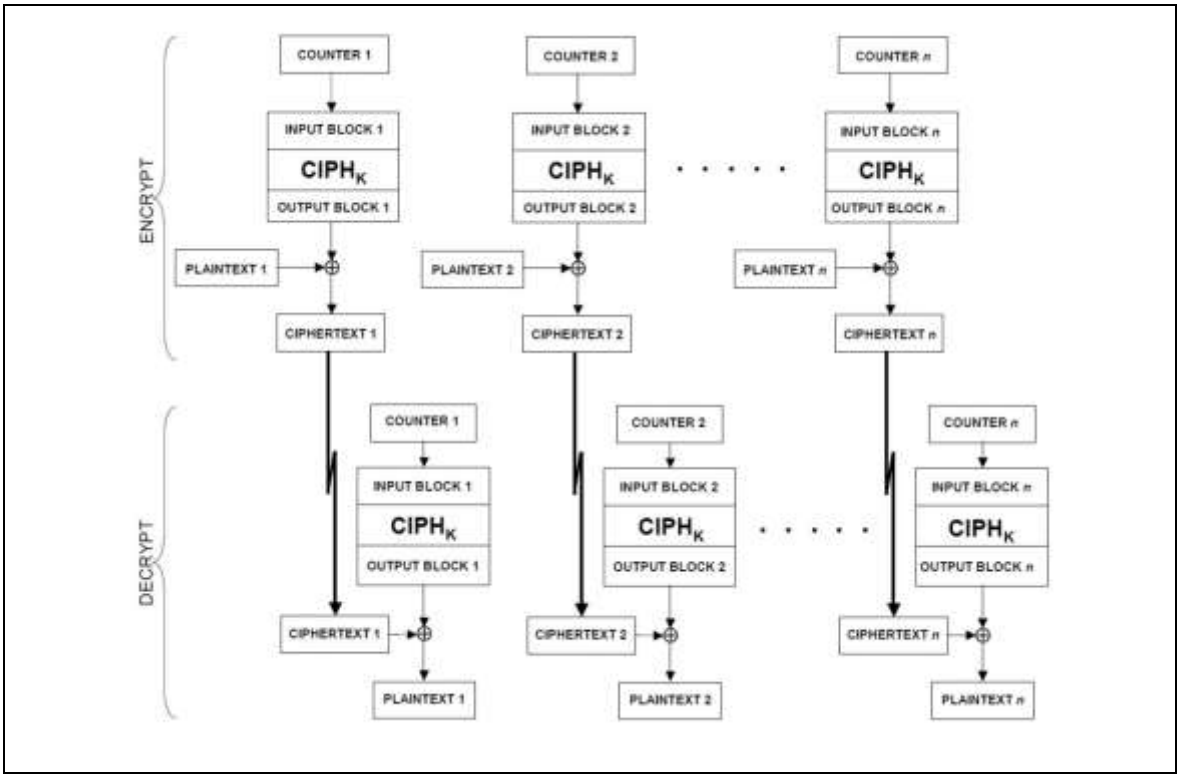



Figure 5.27-7 Counter Mode

**CBC Ciphertext-Stealing 1 Mode (CBC-CS1):**

The figure below 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 $P_n$.

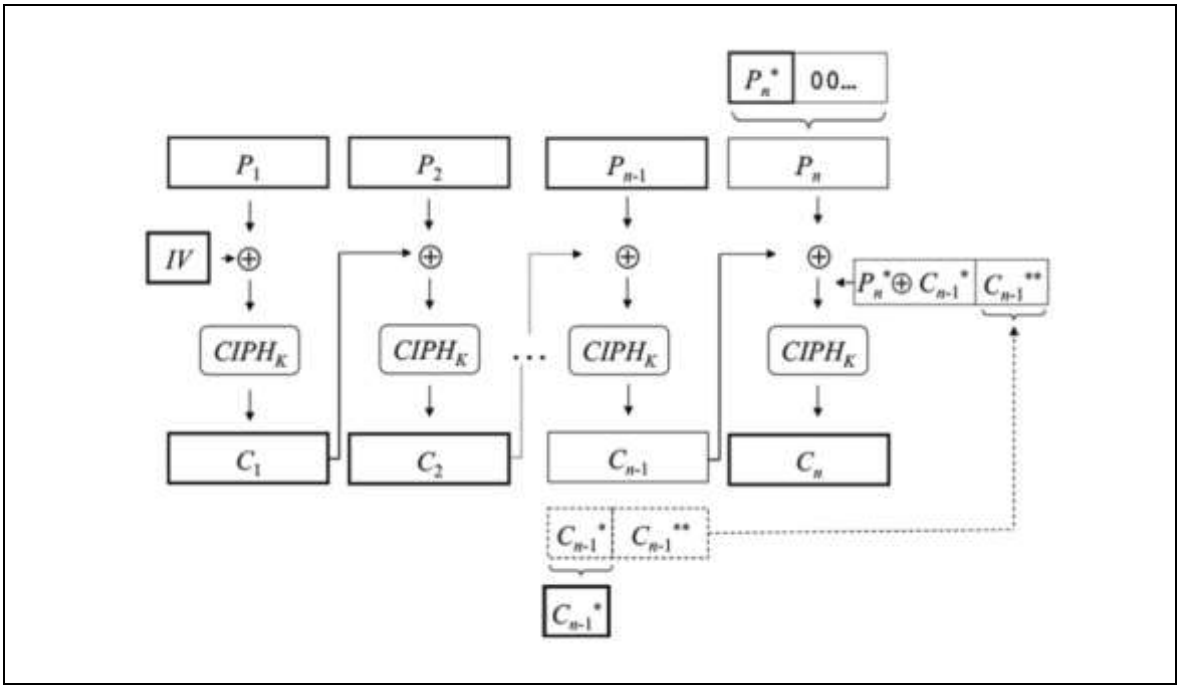



Figure 5.27-8 CBC-CS1 Encryption

The figure below illustrates the CBC-CS1-Decrypt algorithm for the case that $C_{n-1}^*$ is a partial block.

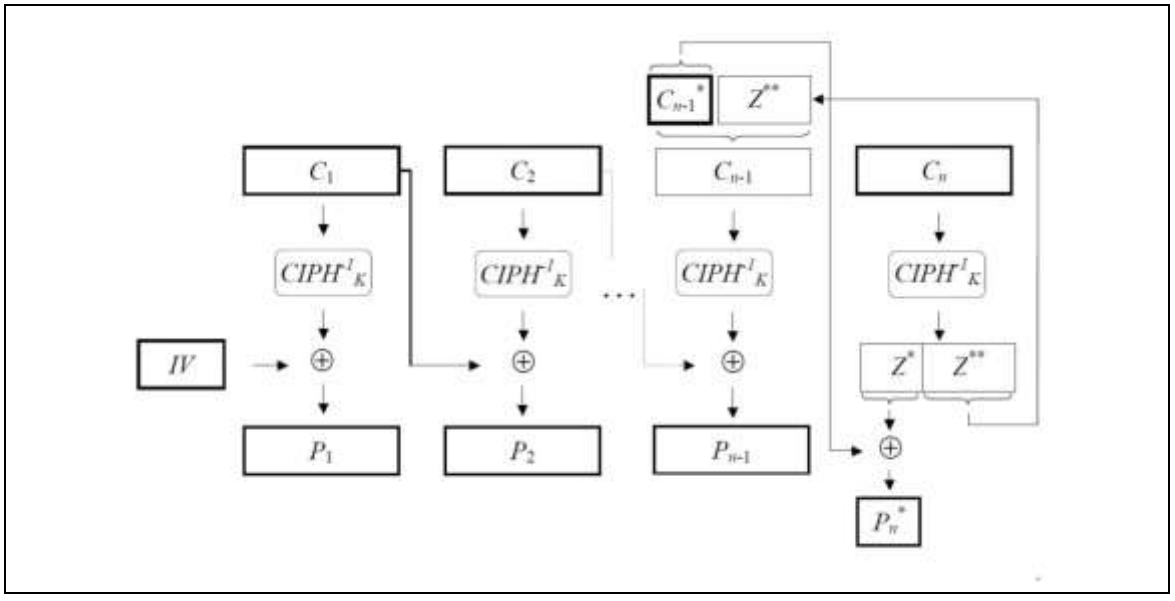



Figure 5.27-9 CBC-CS1 Decryption

**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.

Refer to the following programming steps for how to program the AES related registers.

**AES DMA mode programming flow:**

1. Write 1 to AESIEN (CRPT_INTEN[0]) to enable AES interrupt.
2. Select one from four DMA channels.
3. Program AES key to registers CRPT_AESn_KEY0 ~ CRPT_AESn_KEY7. (where n is the selected channel number)
4. Program initial vectors to registers CRPT_AESn_IV0 ~ CRPT_AESn_IV3.
5. Program DMA source address to register CRPT_AESn_SADDR.
6. Program DMA destination address to register CRPT_AESn_DADDR.
7. Program DMA byte count to register CRPT_AESn_CNT.
8. Configure AES control register CRPT_AES_CTL for channel selection, encryption/decryption, operational mode, DMA mode, key size, and DMA input/output swap.
9. Write input data to DMA source address with selected DMA byte count.
10. Write 1 to START(CRPT_AES_CTL[0]) to start AES encryption/decryption.
11. Waits for the AES interrupt flag AESIF (CRPT_INTSTS[0]) be set.
12. Read output data from DMA destination address with selected DMA byte count.
13. Repeat step 9 to step 12 until all data processed.

**AES Non-DMA mode programming flow:**

1. Write 1 to AESIEN (CRPT_INTEN[0]) to enable AES interrupt.
2. Program AES key to register CRPT_AESn_KEY0 ~ CRPT_AESn_KEY7. (where n is the selected channel number)
3. Program initial vectors to register CRPT_AESn_IV0 ~ CRPT_AESn_IV3.
4. Configure AES control register (CRPT_AES_CTL) for channel select, encryption/decryption, operational mode, and key size.
5. Write 1 to START(CRPT_AES_CTL[0]) to start AES encryption/decryption.
6. Polling INBUFFULL(CRPT_AES_STS[9]) and OUTBUFEMPTY(CRPT_AES_STS[16]). If INBUFFULL(CRPT_AES_STS[9]) is 0, write 32 bits input data to CRPT_AES_DATIN. If OUTBUFEMPTY(CRPT_AES_STS[16]) is 0, read 32 bits data from CRPT_AES_DATOUT.
7. Repeat step 6 until 128 bits data (16 bytes) are written to and read from AES engine.
8. Write 1 to DMALAST(CRPT_AES_CTL[5]).
9. Repeat steps 6 to step 8 until all data processed.

### 5.27.5.3 *DES/TDES (Data Encryption Standard / Triple DES)*

FIPS 46-3 specifies two cryptographic algorithms, the Data Encryption Standard(DES) and the Triple Data Encryption Algorithm (TDEA). The cryptographic accelerator supports FIPS 46-3, both encryption and decryption, and ECB, CBC, CFB, OFB and CTR modes.

**TDES DMA mode programming flow:**

1. Write 1 to TDESIEN (CRPT_INTEN[8]) to enable TDES interrupt.

2. Check the TDES engine is in idle state, i.e., BUSY(CRPT_TDES_STS [0]) is 0.
3. Program TDES key to registers CRPT_TDESn_KEY1H, CRPT_TDESn_KEY1L, CRPT_TDESn_KEY2H, CRPT_TDESn_KEY2L, CRPT_TDESn_KEY3H, and CRPT_TDESn_KEY3L. (where n is the selected channel number)
4. Program initial vector to registers CRPT_TDESn_IVH and CRPT_TDESn_IVL.
5. Program DMA source address to register CRPT_TDESn_SADDR.
6. Program DMA destination address to register CRPT_TDESn_DADDR.
7. Program DMA byte count to register CRPT_TDESn_CNT.
8. Configure TDES control register CRPT_TDES_CTL for channel selection, encryption/decryption, operational mode, DMA mode, TDES keys, and DMA input/output swap.
9. Write input data to DMA source address with selected DMA byte count.
10. Write 1 to START(CRPT_TDES_CTL[0]) to start TDES encryption/decryption.
11. Waits for the TDES interrupt flag TDESIF (CRPT_INTSTS[8]) be set.
12. Read output data from DMA destination address with selected DMA byte count.
13. Repeat step 9 to step 12 until all data processed.

**TDES Non-DMA mode programming flow:**

1. Write 1 to TDESIEN (CRPT_INTEN[8]) to enable TDES interrupt.
2. Check the TDES engine is in idle state, i.e., BUSY(CRPT_TDES_STS[0]) is 0.
3. Program TDES key to registers CRPT_TDESn_KEY1H, CRPT_TDESn_KEY1L, CRPT_TDESn_KEY2H, CRPT_TDESn_KEY2L, CRPT_TDESn_KEY3H, and CRPT_TDESn_KEY3L. (where n is the selected channel number)
4. Program initial vector to registers CRPT_TDESn_IVH and CRPT_TDESn_IVL.
5. Configure TDES control register CRPT_TDES_CTL for channel selection, encryption/decryption, operational mode, and TDES keys.
6. Write 1 to START(CRPT_TDES_CTL[0]) to start TDES encryption/decryption.
7. Polling INBUFFULL(CRPT_TDES_STS[9]) and OUTBUFEMPTY(CRPT_TDES_STS[16]). If INBUFFULL(CRPT_TDES_STS[9]) is 0, write 32 bits input data to CRPT_TDES_DATIN. If OUTBUFEMPTY(CRPT_TDES_STS[16]) is 0, read 32 bits data from CRPT_TDES_DATOUT.
8. Repeat step 7 until 64 bits data (8 bytes) are written to and read from TDES engine.
9. Write 1 to DMALAST(CRPT_TDES_CTL[5]).
10. Repeat steps 7 to step 9 until all data processed.

#### 5.27.5.4 *SHA (Secure Hash Algorithm)*

The Secure Hash Algorithm is a family of cryptographic hash functions published by the National Institute of Standards and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS).

User can refer to the following steps to understand how to program the SHA related registers.

**SHA DMA mode programming flow:**

1. Write 1 to HMACIEN(CRPT_INTEN[24]) to enable SHA/HMAC interrupt.
2. Configure SHA/HMAC control register CRPT_HMAC_CTL for SHA/HMAC engine input/output data swap, DMA mode, and SHA operation mode. Clear HMACEN(CRPT_HMAC_CTL[4]) to select SHA mode.
3. Program DMA source address to register CRPT_HMAC_SADDR.
4. Program DMA byte count to register CRPT_HMAC_DMACNT.
5. Write input data to DMA source address with selected DMA byte count.
6. Write 1 to START(CRPT_HMAC_CTL[0]) to start SHA encryption.
7. Waits for the SHA interrupt flag HMACIF(CRPT_INTSTS[24]) be set.

8. Read output digest (SHA160: CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST4, SHA224: CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST6, SHA256: CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST7, SHA384: CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST11, SHA512: CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST15).

**SHA Non-DMA mode programming flow:**

1. Configure SHA/HMAC control register CRPT_HMAC_CTL for SHA/HMAC engine input/output data swap and SHA operation mode. Clear HMACEN(CRPT_HMAC_CTL[4]) to select SHA mode.
2. If it's the last input word, set DMALAST(CRPT_HMAC_CTL[5]).
3. Write 1 to START(CRPT_HMAC_CTL[0]) to start SHA encryption.
4. Waits for the SHA data input request DATINREQ(CRPT_HMAC_STS[16]) be set.
5. Write one word of input data to CRPT_HMAC_DATIN.
6. Repeat step 2 to 5 until all inut words are written into SHA engine.
7. Waits for the BUSY (CRPT_HMAC_STS[0]) be cleared.
8. Read output digest (SHA160: CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST4, SHA224: CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST6, SHA256: CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST7, SHA384: CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST11, SHA512: CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST15).

#### 5.27.5.5 *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-1, may be used in the calculation of an HMAC; the resulting MAC algorithm is termed HMAC-SHA1 accordingly.

User can refer to the following steps to understand how to program the HMAC related registers.

**HMAC DMA mode programming flow:**

1. Write 1 to HMACIEN(CRPT_INTEN[24]) to enable HMAC interrupt.
2. Configure SHA/HMAC control register CRPT_HMAC_CTL for HMAC engine input/output data swap, DMA mode, and HMAC operation mode. Set HMACEN(CRPT_HMAC_CTL[4]) to select HMAC mode.
3. Program DMA source address to register CRPT_HMAC_SADDR.
4. Program DMA byte count to register CRPT_HMAC_DMACNT.
5. Write input data to DMA source address with selected DMA byte count.
6. Write 1 to START(CRPT_HMAC_CTL[0]) to start HMAC encryption.
7. Waits for the HMAC interrupt flag HMACIF(CRPT_INTSTS[24]) be set.
8. Read output digest (HMAC-SHA160: CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST4, HMAC-SHA224: CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST6, HMAC-SHA256: CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST7, HMAC-SHA384: CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST11, HMAC-SHA512: CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST15).

**HMAC Non-DMA mode programming flow:**

1. Configure SHA/HMAC control register CRPT_HMAC_CTL for SHA/HMAC engine input/output data swap and HMAC operation mode. Set HMACEN(CRPT_HMAC_CTL[4]) to select HMAC mode.

2. If it's the last input word, set DMALAST(CRPT_HMAC_CTL[5]).
3. Write 1 to START(CRPT_HMAC_CTL[0]) to start HMAC encryption.
4. Waits for the HMAC data input request DATINREQ(CRPT_HMAC_STS[16]) be set.
5. Write one word of input data to CRPT_HMAC_DATIN.
6. Repeat step 2 to 5 until all inut words are written into SHA engine.
7. Waits for the BUSY (CRPT_HMAC_STS[0]) be cleared.
8. Read output digest (HMAC-SHA160: CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST4, HMAC-SHA224: CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST6, HMAC-SHA256: CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST7, HMAC-SHA384: CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST11, HMAC-SHA512: CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST15).

### 5.27.6 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRYP Base Address:**<br>**CRYP_BA = 0xB000_C000** | | | | |
| **CRPT_INTEN** | CRYP_BA+0x000 | R/W | Crypto Interrupt Enable Control Register | 0x0000_0000 |
| **CRPT_INTSTS** | CRYP_BA+0x004 | R/W | Crypto Interrupt Flag | 0x0000_0000 |
| **CRPT_PRNG_CTL** | CRYP_BA+0x008 | R/W | PRNG Control Register | 0x0000_0000 |
| **CRPT_PRNG_SEED** | CRYP_BA+0x00C | W | Seed for PRNG | Undefined |
| **CRPT_PRNG_KEY0** | CRYP_BA+0x010 | R | PRNG Generated Key0 | Undefined |
| **CRPT_PRNG_KEY1** | CRYP_BA+0x014 | R | PRNG Generated Key1 | Undefined |
| **CRPT_PRNG_KEY2** | CRYP_BA+0x018 | R | PRNG Generated Key2 | Undefined |
| **CRPT_PRNG_KEY3** | CRYP_BA+0x01C | R | PRNG Generated Key3 | Undefined |
| **CRPT_PRNG_KEY4** | CRYP_BA+0x020 | R | PRNG Generated Key4 | Undefined |
| **CRPT_PRNG_KEY5** | CRYP_BA+0x024 | R | PRNG Generated Key5 | Undefined |
| **CRPT_PRNG_KEY6** | CRYP_BA+0x028 | R | PRNG Generated Key6 | Undefined |
| **CRPT_PRNG_KEY7** | CRYP_BA+0x02C | R | PRNG Generated Key7 | Undefined |
| **CRPT_AES_FDBCK0** | CRYP_BA+0x050 | R | AES Engine Output Feedback Data after Cryptographic Operation | 0x0000_0000 |
| **CRPT_AES_FDBCK1** | CRYP_BA+0x054 | R | AES Engine Output Feedback Data after Cryptographic Operation | 0x0000_0000 |
| **CRPT_AES_FDBCK2** | CRYP_BA+0x058 | R | AES Engine Output Feedback Data after Cryptographic Operation | 0x0000_0000 |
| **CRPT_AES_FDBCK3** | CRYP_BA+0x05C | R | AES Engine Output Feedback Data after Cryptographic Operation | 0x0000_0000 |
| **CRPT_TDES_FDBCKH** | CRYP_BA+0x060 | R | TDES/DES Engine Output Feedback High Word Data after Cryptographic Operation | 0x0000_0000 |
| **CRPT_TDES_FDBCKL** | CRYP_BA+0x064 | R | TDES/DES Engine Output Feedback Low Word Data after Cryptographic Operation | 0x0000_0000 |
| **CRPT_AES_CTL** | CRYP_BA+0x100 | R/W | AES Control Register | 0x0000_0000 |
| **CRPT_AES_STS** | CRYP_BA+0x104 | R | AES Engine Flag | 0x0001_0100 |
| **CRPT_AES_DATIN** | CRYP_BA+0x108 | R/W | AES Engine Data Input Port Register | 0x0000_0000 |
| **CRPT_AES_DATOUT** | CRYP_BA+0x10C | R | AES Engine Data Output Port Register | 0x0000_0000 |
| **CRPT_AES0_KEY0** | CRYP_BA+0x110 | R/W | AES Key Word 0 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_KEY1** | CRYP_BA+0x114 | R/W | AES Key Word 1 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_KEY2** | CRYP_BA+0x118 | R/W | AES Key Word 2 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_KEY3** | CRYP_BA+0x11C | R/W | AES Key Word 3 Register for Channel 0 | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **CRPT_AES0_KEY4** | CRYP_BA+0x120 | R/W | AES Key Word 4 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_KEY5** | CRYP_BA+0x124 | R/W | AES Key Word 5 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_KEY6** | CRYP_BA+0x128 | R/W | AES Key Word 6 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_KEY7** | CRYP_BA+0x12C | R/W | AES Key Word 7 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_IV0** | CRYP_BA+0x130 | R/W | AES Initial Vector Word 0 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_IV1** | CRYP_BA+0x134 | R/W | AES Initial Vector Word 1 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_IV2** | CRYP_BA+0x138 | R/W | AES Initial Vector Word 2 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_IV3** | CRYP_BA+0x13C | R/W | AES Initial Vector Word 3 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_SADDR** | CRYP_BA+0x140 | R/W | AES DMA Source Address Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_DADDR** | CRYP_BA+0x144 | R/W | AES DMA Destination Address Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_CNT** | CRYP_BA+0x148 | R/W | AES Byte Count Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES1_KEY0** | CRYP_BA+0x14C | R/W | AES Key Word 0 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_KEY1** | CRYP_BA+0x150 | R/W | AES Key Word 1 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_KEY2** | CRYP_BA+0x154 | R/W | AES Key Word 2 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_KEY3** | CRYP_BA+0x158 | R/W | AES Key Word 3 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_KEY4** | CRYP_BA+0x15C | R/W | AES Key Word 4 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_KEY5** | CRYP_BA+0x160 | R/W | AES Key Word 5 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_KEY6** | CRYP_BA+0x164 | R/W | AES Key Word 6 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_KEY7** | CRYP_BA+0x168 | R/W | AES Key Word 7 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_IV0** | CRYP_BA+0x16C | R/W | AES Initial Vector Word 0 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_IV1** | CRYP_BA+0x170 | R/W | AES Initial Vector Word 1 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_IV2** | CRYP_BA+0x174 | R/W | AES Initial Vector Word 2 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_IV3** | CRYP_BA+0x178 | R/W | AES Initial Vector Word 3 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_SADDR** | CRYP_BA+0x17C | R/W | AES DMA Source Address Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_DADDR** | CRYP_BA+0x180 | R/W | AES DMA Destination Address Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_CNT** | CRYP_BA+0x184 | R/W | AES Byte Count Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES2_KEY0** | CRYP_BA+0x188 | R/W | AES Key Word 0 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_KEY1** | CRYP_BA+0x18C | R/W | AES Key Word 1 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_KEY2** | CRYP_BA+0x190 | R/W | AES Key Word 2 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_KEY3** | CRYP_BA+0x194 | R/W | AES Key Word 3 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_KEY4** | CRYP_BA+0x198 | R/W | AES Key Word 4 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_KEY5** | CRYP_BA+0x19C | R/W | AES Key Word 5 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_KEY6** | CRYP_BA+0x1A0 | R/W | AES Key Word 6 Register for Channel 2 | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **CRPT_AES2_KEY7** | CRYP_BA+0x1A4 | R/W | AES Key Word 7 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_IV0** | CRYP_BA+0x1A8 | R/W | AES Initial Vector Word 0 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_IV1** | CRYP_BA+0x1AC | R/W | AES Initial Vector Word 1 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_IV2** | CRYP_BA+0x1B0 | R/W | AES Initial Vector Word 2 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_IV3** | CRYP_BA+0x1B4 | R/W | AES Initial Vector Word 3 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_SADDR** | CRYP_BA+0x1B8 | R/W | AES DMA Source Address Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_DADDR** | CRYP_BA+0x1BC | R/W | AES DMA Destination Address Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_CNT** | CRYP_BA+0x1C0 | R/W | AES Byte Count Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES3_KEY0** | CRYP_BA+0x1C4 | R/W | AES Key Word 0 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_KEY1** | CRYP_BA+0x1C8 | R/W | AES Key Word 1 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_KEY2** | CRYP_BA+0x1CC | R/W | AES Key Word 2 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_KEY3** | CRYP_BA+0x1D0 | R/W | AES Key Word 3 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_KEY4** | CRYP_BA+0x1D4 | R/W | AES Key Word 4 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_KEY5** | CRYP_BA+0x1D8 | R/W | AES Key Word 5 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_KEY6** | CRYP_BA+0x1DC | R/W | AES Key Word 6 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_KEY7** | CRYP_BA+0x1E0 | R/W | AES Key Word 7 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_IV0** | CRYP_BA+0x1E4 | R/W | AES Initial Vector Word 0 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_IV1** | CRYP_BA+0x1E8 | R/W | AES Initial Vector Word 1 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_IV2** | CRYP_BA+0x1EC | R/W | AES Initial Vector Word 2 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_IV3** | CRYP_BA+0x1F0 | R/W | AES Initial Vector Word 3 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_SADDR** | CRYP_BA+0x1F4 | R/W | AES DMA Source Address Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_DADDR** | CRYP_BA+0x1F8 | R/W | AES DMA Destination Address Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_CNT** | CRYP_BA+0x1FC | R/W | AES Byte Count Register for Channel 3 | 0x0000_0000 |
| **CRPT_TDES_CTL** | CRYP_BA+0x200 | R/W | TDES/DES Control Register | 0x0000_0000 |
| **CRPT_TDES_STS** | CRYP_BA+0x204 | R | TDES/DES Engine Flag | 0x0001_0100 |
| **CRPT_TDES0_KEY1H** | CRYP_BA+0x208 | R/W | TDES/DES Key 1 High Word Register for Channel 0 | 0x0000_0000 |
| **CRPT_TDES0_KEY1L** | CRYP_BA+0x20C | R/W | TDES/DES Key 1 Low Word Register for Channel 0 | 0x0000_0000 |
| **CRPT_TDES0_KEY2H** | CRYP_BA+0x210 | R/W | TDES Key 2 High Word Register for Channel 0 | 0x0000_0000 |
| **CRPT_TDES0_KEY2L** | CRYP_BA+0x214 | R/W | TDES Key 2 Low Word Register for Channel 0 | 0x0000_0000 |
| **CRPT_TDES0_KEY3H** | CRYP_BA+0x218 | R/W | TDES Key 3 High Word Register for Channel 0 | 0x0000_0000 |
| **CRPT_TDES0_KEY3L** | CRYP_BA+0x21C | R/W | TDES Key 3 Low Word Register for Channel 0 | 0x0000_0000 |
| **CRPT_TDES0_IVH** | CRYP_BA+0x220 | R/W | TDES/DES Initial Vector High Word Register for Channel 0 | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **CRPT_TDES0_IVL** | CRYP_BA+0x224 | R/W | TDES/DES Initial Vector Low Word Register for Channel 0 | 0x0000_0000 |
| **CRPT_TDES0_SADDR** | CRYP_BA+0x228 | R/W | TDES/DES DMA Source Address Register for Channel 0 | 0x0000_0000 |
| **CRPT_TDES0_DADDR** | CRYP_BA+0x22C | R/W | TDES/DES DMA Destination Address Register for Channel 0 | 0x0000_0000 |
| **CRPT_TDES0_CNT** | CRYP_BA+0x230 | R/W | TDES/DES Byte Count Register for Channel 0 | 0x0000_0000 |
| **CRPT_TDES_DATIN** | CRYP_BA+0x234 | R/W | TDES/DES Engine Input data Word Register | 0x0000_0000 |
| **CRPT_TDES_DATOUT** | CRYP_BA+0x238 | R | TDES/DES Engine Output data Word Register | 0x0000_0000 |
| **CRPT_TDES1_KEY1H** | CRYP_BA+0x248 | R/W | TDES/DES Key 1 High Word Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES1_KEY1L** | CRYP_BA+0x24C | R/W | TDES/DES Key 1 Low Word Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES1_KEY2H** | CRYP_BA+0x250 | R/W | TDES Key 2 High Word Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES1_KEY2L** | CRYP_BA+0x254 | R/W | TDES Key 2 Low Word Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES1_KEY3H** | CRYP_BA+0x258 | R/W | TDES Key 3 High Word Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES1_KEY3L** | CRYP_BA+0x25C | R/W | TDES Key 3 Low Word Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES1_IVH** | CRYP_BA+0x260 | R/W | TDES/DES Initial Vector High Word Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES1_IVL** | CRYP_BA+0x264 | R/W | TDES/DES Initial Vector Low Word Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES1_SADDR** | CRYP_BA+0x268 | R/W | TDES/DES DMA Source Address Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES1_DADDR** | CRYP_BA+0x26C | R/W | TDES/DES DMA Destination Address Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES1_CNT** | CRYP_BA+0x270 | R/W | TDES/DES Byte Count Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES2_KEY1H** | CRYP_BA+0x288 | R/W | TDES/DES Key 1 High Word Register for Channel 2 | 0x0000_0000 |
| **CRPT_TDES2_KEY1L** | CRYP_BA+0x28C | R/W | TDES/DES Key 1 Low Word Register for Channel 2 | 0x0000_0000 |
| **CRPT_TDES2_KEY2H** | CRYP_BA+0x290 | R/W | TDES Key 2 High Word Register for Channel 2 | 0x0000_0000 |
| **CRPT_TDES2_KEY2L** | CRYP_BA+0x294 | R/W | TDES Key 2 Low Word Register for Channel 2 | 0x0000_0000 |
| **CRPT_TDES2_KEY3H** | CRYP_BA+0x298 | R/W | TDES Key 3 High Word Register for Channel 2 | 0x0000_0000 |
| **CRPT_TDES2_KEY3L** | CRYP_BA+0x29C | R/W | TDES Key 3 Low Word Register for Channel 2 | 0x0000_0000 |
| **CRPT_TDES2_IVH** | CRYP_BA+0x2A0 | R/W | TDES/DES Initial Vector High Word Register for Channel 2 | 0x0000_0000 |
| **CRPT_TDES2_IVL** | CRYP_BA+0x2A4 | R/W | TDES/DES Initial Vector Low Word Register for Channel 2 | 0x0000_0000 |
| **CRPT_TDES2_SADDR** | CRYP_BA+0x2A8 | R/W | TDES/DES DMA Source Address Register for Channel 2 | 0x0000_0000 |
| **CRPT_TDES2_DADDR** | CRYP_BA+0x2AC | R/W | TDES/DES DMA Destination Address Register for Channel 2 | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **CRPT_TDES2_CNT** | CRYP_BA+0x2B0 | R/W | TDES/DES Byte Count Register for Channel 2 | 0x0000_0000 |
| **CRPT_TDES3_KEY1H** | CRYP_BA+0x2C8 | R/W | TDES/DES Key 1 High Word Register for Channel 3 | 0x0000_0000 |
| **CRPT_TDES3_KEY1L** | CRYP_BA+0x2CC | R/W | TDES/DES Key 1 Low Word Register for Channel 3 | 0x0000_0000 |
| **CRPT_TDES3_KEY2H** | CRYP_BA+0x2D0 | R/W | TDES Key 2 High Word Register for Channel 3 | 0x0000_0000 |
| **CRPT_TDES3_KEY2L** | CRYP_BA+0x2D4 | R/W | TDES Key 2 Low Word Register for Channel 3 | 0x0000_0000 |
| **CRPT_TDES3_KEY3H** | CRYP_BA+0x2D8 | R/W | TDES Key 3 High Word Register for Channel 3 | 0x0000_0000 |
| **CRPT_TDES3_KEY3L** | CRYP_BA+0x2DC | R/W | TDES Key 3 Low Word Register for Channel 3 | 0x0000_0000 |
| **CRPT_TDES3_IVH** | CRYP_BA+0x2E0 | R/W | TDES/DES Initial Vector High Word Register for Channel 3 | 0x0000_0000 |
| **CRPT_TDES3_IVL** | CRYP_BA+0x2E4 | R/W | TDES/DES Initial Vector Low Word Register for Channel 3 | 0x0000_0000 |
| **CRPT_TDES3_SADDR** | CRYP_BA+0x2E8 | R/W | TDES/DES DMA Source Address Register for Channel 3 | 0x0000_0000 |
| **CRPT_TDES3_DADDR** | CRYP_BA+0x2EC | R/W | TDES/DES DMA Destination Address Register for Channel 3 | 0x0000_0000 |
| **CRPT_TDES3_CNT** | CRYP_BA+0x2F0 | R/W | TDES/DES Byte Count Register for Channel 3 | 0x0000_0000 |
| **CRPT_HMAC_CTL** | CRYP_BA+0x300 | R/W | SHA/HMAC Control Register | 0x0000_0000 |
| **CRPT_HMAC_STS** | CRYP_BA+0x304 | R | SHA/HMAC Status Flag | 0x0000_0000 |
| **CRPT_HMAC_DGST0** | CRYP_BA+0x308 | R | SHA/HMAC Digest Message 0 | 0x0000_0000 |
| **CRPT_HMAC_DGST1** | CRYP_BA+0x30C | R | SHA/HMAC Digest Message 1 | 0x0000_0000 |
| **CRPT_HMAC_DGST2** | CRYP_BA+0x310 | R | SHA/HMAC Digest Message 2 | 0x0000_0000 |
| **CRPT_HMAC_DGST3** | CRYP_BA+0x314 | R | SHA/HMAC Digest Message 3 | 0x0000_0000 |
| **CRPT_HMAC_DGST4** | CRYP_BA+0x318 | R | SHA/HMAC Digest Message 4 | 0x0000_0000 |
| **CRPT_HMAC_DGST5** | CRYP_BA+0x31C | R | SHA/HMAC Digest Message 5 | 0x0000_0000 |
| **CRPT_HMAC_DGST6** | CRYP_BA+0x320 | R | SHA/HMAC Digest Message 6 | 0x0000_0000 |
| **CRPT_HMAC_DGST7** | CRYP_BA+0x324 | R | SHA/HMAC Digest Message 7 | 0x0000_0000 |
| **CRPT_HMAC_DGST8** | CRYP_BA+0x328 | R | SHA/HMAC Digest Message 8 | 0x0000_0000 |
| **CRPT_HMAC_DGST9** | CRYP_BA+0x32C | R | SHA/HMAC Digest Message 9 | 0x0000_0000 |
| **CRPT_HMAC_DGST10** | CRYP_BA+0x330 | R | SHA/HMAC Digest Message 10 | 0x0000_0000 |
| **CRPT_HMAC_DGST11** | CRYP_BA+0x334 | R | SHA/HMAC Digest Message 11 | 0x0000_0000 |
| **CRPT_HMAC_DGST12** | CRYP_BA+0x338 | R | SHA/HMAC Digest Message 12 | 0x0000_0000 |
| **CRPT_HMAC_DGST13** | CRYP_BA+0x33C | R | SHA/HMAC Digest Message 13 | 0x0000_0000 |
| **CRPT_HMAC_DGST14** | CRYP_BA+0x340 | R | SHA/HMAC Digest Message 14 | 0x0000_0000 |
| **CRPT_HMAC_DGST15** | CRYP_BA+0x344 | R | SHA/HMAC Digest Message 15 | 0x0000_0000 |
| **CRPT_HMAC_KEYCNT** | CRYP_BA+0x348 | R/W | SHA/HMAC Key Byte Count Register | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **CRPT_HMAC_SADDR** | CRYP_BA+0x34C | R/W | SHA/HMAC DMA Source Address Register | 0x0000_0000 |
| **CRPT_HMAC_DMACNT** | CRYP_BA+0x350 | R/W | SHA/HMAC Byte Count Register | 0x0000_0000 |
| **CRPT_HMAC_DATIN** | CRYP_BA+0x354 | R/W | SHA/HMAC Engine Non-DMA Mode Data Input Port Register | 0x0000_0000 |

### 5.27.7 Register Description

5.27.7.1 *Crypto Register*

## CRYPTO Interrupt Enable Control Register (CRPT_INTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_INTEN** | CRYP_BA+0x000 | R/W | Crypto Interrupt Enable Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | **HMACEIEN** | **HMACIEN** |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | **PRNGIEN** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | | **TDESEIEN** | **TDESIEN** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | | | | **AESEIEN** | **AESIEN** |

| Bits | Description | |
|---|---|---|
| [31:26] | **Reserved** | Reserved. |
| [25] | **HMACEIEN** | **SHA/HMAC Error Interrupt Enable Control**<br>0 = SHA/HMAC error interrupt flag Disabled.<br>1 = SHA/HMAC error interrupt flag Enabled. |
| [24] | **HMACIEN** | **SHA/HMAC Interrupt Enable Control**<br>0 = SHA/HMAC interrupt Disabled.<br>1 = SHA/HMAC interrupt Enabled.<br>In DMA mode, an interrupt will be triggered when amount of data set in SHA_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. |
| [23:17] | **Reserved** | Reserved. |
| [16] | **PRNGIEN** | **PRNG Interrupt Enable Control**<br>0 = PRNG interrupt Disabled.<br>1 = PRNG interrupt Enabled. |
| [15:10] | **Reserved** | Reserved. |
| [9] | **TDESEIEN** | **TDES/DES Error Flag Enable Control**<br>0 = TDES/DES error interrupt flag Disabled.<br>1 = TDES/DES error interrupt flag Enabled. |
| [8] | **TDESIEN** | **TDES/DES Interrupt Enable Control**<br>0 = TDES/DES interrupt Disabled.<br>1 = TDES/DES interrupt Enabled.<br>In DMA mode, an interrupt will be triggered when amount of data set in TDES_DMA_CNT is fed into the TDES engine.<br>In Non-DMA mode, an interrupt will be triggered when the TDES engine finishes the operation. |
| [7:2] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [1] | **AESEIEN** | **AES Error Flag Enable Control**<br>0 = AES error interrupt flag Disabled.<br>1 = AES error interrupt flag Enabled. |
| [0] | **AESIEN** | **AES Interrupt Enable Control**<br>0 = AES interrupt Disabled.<br>1 = AES interrupt Enabled.<br>In DMA mode, an interrupt will be triggered when amount of data set in AES_DMA_CNT is fed into the AES engine.<br>In Non-DMA mode, an interrupt will be triggered when the AES engine finishes the operation. |

## CRYPTO Interrupt Flag Register (CRPT_INTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CRPT_INTSTS | CRYP_BA+0x004 | R/W | Crypto Interrupt Flag | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | HMACEIF | HMACIF |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | PRNGIF |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | TDESEIF | TDESIF |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | AESEIF | AESIF |

| Bits | Description | |
|---|---|---|
| [31:26] | **Reserved** | Reserved. |
| [25] | **HMACEIF** | **SHA/HMAC Error Flag**<br>This register includes operating and setting error. The detail flag is shown in SHA _FLAG register.<br>This bit is cleared by writing 1, and it has no effect by writing 0.<br>0 = No SHA/HMAC error.<br>1 = SHA/HMAC error interrupt. |
| [24] | **HMACIF** | **SHA/HMAC Finish Interrupt Flag**<br>This bit is cleared by writing 1, and it has no effect by writing 0.<br>0 = No SHA/HMAC interrupt.<br>1 = SHA/HMAC operation done interrupt. |
| [23:17] | **Reserved** | Reserved. |
| [16] | **PRNGIF** | **PRNG Finish Interrupt Flag**<br>This bit is cleared by writing 1, and it has no effect by writing 0.<br>0 = No PRNG interrupt.<br>1 = PRNG key generation done interrupt. |
| [15:10] | **Reserved** | Reserved. |
| [9] | **TDESEIF** | **TDES/DES Error Flag**<br>This bit includes the operating and setting error. The detailed flag is shown in the TDES _FLAG register. This includes operating and setting error.<br>This bit is cleared by writing 1, and it has no effect by writing 0.<br>0 = No TDES/DES error.<br>1 = TDES/DES encryption/decryption error interrupt. |

| | | |
|---|---|---|
| [8] | **TDESIF** | **TDES/DES Finish Interrupt Flag**<br>This bit is cleared by writing 1, and it has no effect by writing 0.<br>0 = No TDES/DES interrupt.<br>1 = TDES/DES encryption/decryption done interrupt. |
| [7:2] | **Reserved** | Reserved. |
| [1] | **AESEIF** | **AES Error Flag**<br>This bit is cleared by writing 1, and it has no effect by writing 0.<br>0 = No AES error.<br>1 = AES encryption/decryption done interrupt. |
| [0] | **AESIF** | **AES Finish Interrupt Flag**<br>This bit is cleared by writing 1, and it has no effect by writing 0.<br>0 = No AES interrupt.<br>1 = AES encryption/decryption done interrupt. |

5.27.7.2 *PRNG Register*

## PRNG Control Register (CRPT_PRNG_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CRPT_PRNG_CTL | CRYP_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 |
| Reserved | | | | KEYSZ | | SEEDRLD | START |

| Bits | Description | |
|---|---|---|
| [31:9] | **Reserved** | Reserved. |
| [8] | **BUSY** | **PRNG Busy (Read Only)**<br>0 = PRNG engine is idle.<br>1 = Indicate that the PRNG engine is generating CRPT_PRNG_KEYx. |
| [7:4] | **Reserved** | Reserved. |
| [3:2] | **KEYSZ** | **PRNG Generate Key Size**<br>00 = 64 bits.<br>01 = 128 bits.<br>10 = 192 bits.<br>11 = 256 bits. |
| [1] | **SEEDRLD** | **Reload New Seed for PRNG Engine**<br>0 = Generating key based on the current seed.<br>1 = Reload new seed. |
| [0] | **START** | **Start PRNG Engine**<br>0 = Stop PRNG engine.<br>1 = Generate new key and store the new key to register CRPT_PRNG_KEYx , which will be cleared when the new key is generated. |

### PRNG Seed Register (CRPT_PRNG_SEED)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CRPT_PRNG_SEED | CRYP_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)**<br>The bits store the seed for PRNG engine. |

## PRNG Key x Register (CRPT_PRNG_KEYx)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_PRNG_KEY0** | CRYP_BA+0x010 | R | PRNG Generated Key0 | Undefined |
| **CRPT_PRNG_KEY1** | CRYP_BA+0x014 | R | PRNG Generated Key1 | Undefined |
| **CRPT_PRNG_KEY2** | CRYP_BA+0x018 | R | PRNG Generated Key2 | Undefined |
| **CRPT_PRNG_KEY3** | CRYP_BA+0x01C | R | PRNG Generated Key3 | Undefined |
| **CRPT_PRNG_KEY4** | CRYP_BA+0x020 | R | PRNG Generated Key4 | Undefined |
| **CRPT_PRNG_KEY5** | CRYP_BA+0x024 | R | PRNG Generated Key5 | Undefined |
| **CRPT_PRNG_KEY6** | CRYP_BA+0x028 | R | PRNG Generated Key6 | Undefined |
| **CRPT_PRNG_KEY7** | CRYP_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)**<br>The bits store the key that is generated by PRNG. |

5.27.7.3 *AES Register*

## AES Control Register (CRPT_AES_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CRPT_AES_CTL | CRYP_BA+0x100 | R/W | AES Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| KEYPRT | KEYUNPRT | | | | | CHANNEL | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| INSWAP | OUTSWAP | Reserved | | | | | ENCRPT |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| OPMODE | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DMAEN | DMACSCAD | DMALAST | EXTKEY | KEYSZ | | STOP | START |

| Bits | Description | |
|---|---|---|
| [31] | **KEYPRT** | **Protect Key**<br>Read as a flag to reflect KEYPRT.<br>0 = No effect.<br>1 = Protect the content of the AES key from reading. The return value for reading CRPT_AESn_KEYx is not the content of the registers CRPT_AESn_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**<br>Writing 0 to CRPT_AES_CTL[31] and “10110” to CRPT_AES_CTL[30:26] is to unprotect the AES key.<br>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. |
| [25:24] | **CHANNEL** | **AES Engine Working Channel**<br>00 = Current control register setting is for channel 0.<br>01 = Current control register setting is for channel 1.<br>10 = Current control register setting is for channel 2.<br>11 = Current control register setting is for channel 3. |
| [23] | **INSWAP** | **AES Engine Input Data Swap**<br>0 = Keep the original order.<br>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**<br>0 = Keep the original order.<br>1 = The order that CPU outputs data from the accelerator will be changed from {byte3, byte2, byte1, byte0} to {byte0, byte1, byte2, byte3}. |

| | | |
|---|---|---|
| [16] | **ENCRPT** | **AES Encryption/Decryption**<br>0 = AES engine executes decryption operation.<br>1 = AES engine executes encryption operation. |
| [15:8] | **OPMODE** | **AES Engine Operation Modes**<br>0x00 = ECB (Electronic Codebook Mode)<br>0x01 = CBC (Cipher Block Chaining Mode).<br>0x02 = CFB (Cipher Feedback Mode).<br>0x03 = OFB (Output Feedback Mode).<br>0x04 = CTR (Counter Mode).<br>0x10 = CBC-CS1 (CBC Ciphertext-Stealing 1 Mode).<br>0x11 = CBC-CS2 (CBC Ciphertext-Stealing 2 Mode).<br>0x12 = CBC-CS3 (CBC Ciphertext-Stealing 3 Mode). |
| [7] | **DMAEN** | **AES Engine DMA Enable Control**<br>0 = AES DMA engine Disabled.<br>The AES engine operates in Non-DMA mode, and gets data from the port CRPT_AES_DATIN.<br>1 = AES_DMA engine Enabled.<br>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**<br>0 = DMA cascade function Disabled.<br>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**<br>In DMA mode, this bit must be set as beginning the last DMA cascade round.<br>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.<br>This bit is always 0 when it's read back. Must be written again once START is triggered. |
| [4] | **EXTKEY** | **External Key**<br>0 = AES key is from CRPT_AESn_KEYx.<br>1 = AES uses external key.<br>If the AES accelerator is operating and the corresponding flag BUSY is 1, updating this register has no effect. |
| [3:2] | **KEYSZ** | **AES Key Size**<br>This bit defines three different key size for AES operation.<br>2'b00 = 128 bits key.<br>2'b01 = 192 bits key.<br>2'b10 = 256 bits key.<br>2'b11 = Reserved.<br>If the AES accelerator is operating and the corresponding flag BUSY is 1, updating this register has no effect. |

| | | |
|---|---|---|
| [1] | **STOP** | **AES Engine Stop**<br>0 = No effect.<br>1 = Stop AES engine.<br>**Note:** This bit is always 0 when it's read back. |
| [0] | **START** | **AES Engine Start**<br>0 = No effect.<br>1 = Start AES engine. BUSY flag will be set.<br>**Note:** This bit is always 0 when it's read back. |

| DMA Judge Priority Of Input And Output Data | FSM | Actions |
|---|---|---|
| 1 | idle | idle |
| 2 | R16 | Outbuffer is full, so it reads 16 words out. |
| 3 | W16 | Inbuffer is empty, and the amount of input data is more than 16 words. It writes 16 words to Inputbuffer |
| 4 | W8 | The amount of data in Inbuffer is less than 8 words, and the amount of input data is more than 8 words.<br>It writes 8 words to Inputbuffer |
| 5 | R8 | The amount of output data in output buffer is more than 8 words. It reads 8 words out. |
| 6 | W4 | The amount of data in Inbuffer is less than 12 words, and the amount of input data is more than 4 words.<br>It writes 4 words to Inputbuffer |
| 7 | R4 | The amount of output data in output buffer is more than 4 words. It reads 4 words out. |
| 8 | W2 | The amount of data in Inbuffer is less than 14 words, and the amount of input data is more than 2 words.<br>It writes 2 words to Inputbuffer |
| 9 | W1 | The amount of data in Inbuffer is less than 15 words, and the amount of input data is 1 word.<br>It writes 1 word to Inputbuffer |
| 10 | R2 | The amount of data in Outbuffer is not empty, and the amount of output data is more 2 words.<br>It reads 2 words out. |
| 11 | R1 | The amount of data in Outbuffer is not empty, and the amount of output data is 1 word.<br>It reads 1 word out. |

### AES Status Flag Register (CRPT_AES_STS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CRPT_AES_STS | CRYP_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 | | | 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 | |
|---|---|---|
| [31:21] | **Reserved** | Reserved. |
| [20] | **BUSERR** | **AES DMA Access Bus Error Flag**<br>0 = No error.<br>1 = Bus error will stop DMA operation and AES engine. |
| [19] | **Reserved** | Reserved. |
| [18] | **OUTBUFERR** | **AES Out Buffer Error Flag**<br>0 = No error.<br>1 = Error happens during getting the result from AES engine. |
| [17] | **OUTBUFFULL** | **AES Out Buffer Full Flag**<br>0 = AES output buffer is not full.<br>1 = AES output buffer is full, and software needs to get data from CRPT_AES_DATOUT. Otherwise, the AES engine will be pending since the output buffer is full. |
| [16] | **OUTBUFEMPTY** | **AES Out Buffer Empty**<br>0 = AES output buffer is not empty. There are some valid data kept in output buffer.<br>1 = AES output buffer is empty. Software cannot get data from CRPT_AES_DATOUT. Otherwise, the flag OUTBUFERR will be set to 1 since the output buffer is empty. |
| [15:13] | **Reserved** | Reserved. |
| [12] | **CNTERR** | **CRPT_AESn_CNT Setting Error**<br>0 = No error in CRPT_AESn_CNT setting.<br>1 = CRPT_AESn_CNT is not a multiply of 16 in ECB, CBC, CFB, OFB, and CTR mode. |
| [11] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [10] | **INBUFERR** | **AES Input Buffer Error Flag**<br>0 = No error.<br>1 = Error happens during feeding data to the AES engine. |
| [9] | **INBUFFULL** | **AES Input Buffer Full Flag**<br>0 = AES input buffer is not full. Software can feed the data into the AES engine.<br>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**<br>0 = There are some data in input buffer waiting for the AES engine to process.<br>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**<br>0 = The AES engine is idle or finished.<br>1 = The AES engine is under processing. |

### AES Data Input Port Register (CRPT_AES_DATIN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_AES_DATIN** | CRYP_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**<br>CPU feeds data to AES engine through this port by checking CRPT_AES_STS. Feed data as INBUFFULL is 0. |

### AES Data Output Port Register (CRPT_AES_DATOUT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_AES_DATOUT** | CRYP_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**<br>CPU gets results from the AES engine through this port by checking CRPT_AES_STS. Get data as OUTBUFEMPTY is 0. |

### AES Key Word x Register (CRPT_AES0_KEYx, CRPT_AES1_KEYx, CRPT_AES2_KEYx, CRPT_AES3_KEYx)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_AES0_KEY0** | CRYP_BA+0x110 | R/W | AES Key Word 0 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_KEY1** | CRYP_BA+0x114 | R/W | AES Key Word 1 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_KEY2** | CRYP_BA+0x118 | R/W | AES Key Word 2 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_KEY3** | CRYP_BA+0x11C | R/W | AES Key Word 3 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_KEY4** | CRYP_BA+0x120 | R/W | AES Key Word 4 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_KEY5** | CRYP_BA+0x124 | R/W | AES Key Word 5 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_KEY6** | CRYP_BA+0x128 | R/W | AES Key Word 6 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_KEY7** | CRYP_BA+0x12C | R/W | AES Key Word 7 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES1_KEY0** | CRYP_BA+0x14C | R/W | AES Key Word 0 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_KEY1** | CRYP_BA+0x150 | R/W | AES Key Word 1 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_KEY2** | CRYP_BA+0x154 | R/W | AES Key Word 2 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_KEY3** | CRYP_BA+0x158 | R/W | AES Key Word 3 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_KEY4** | CRYP_BA+0x15C | R/W | AES Key Word 4 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_KEY5** | CRYP_BA+0x160 | R/W | AES Key Word 5 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_KEY6** | CRYP_BA+0x164 | R/W | AES Key Word 6 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_KEY7** | CRYP_BA+0x168 | R/W | AES Key Word 7 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES2_KEY0** | CRYP_BA+0x188 | R/W | AES Key Word 0 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_KEY1** | CRYP_BA+0x18C | R/W | AES Key Word 1 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_KEY2** | CRYP_BA+0x190 | R/W | AES Key Word 2 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_KEY3** | CRYP_BA+0x194 | R/W | AES Key Word 3 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_KEY4** | CRYP_BA+0x198 | R/W | AES Key Word 4 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_KEY5** | CRYP_BA+0x19C | R/W | AES Key Word 5 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_KEY6** | CRYP_BA+0x1A0 | R/W | AES Key Word 6 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_KEY7** | CRYP_BA+0x1A4 | R/W | AES Key Word 7 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES3_KEY0** | CRYP_BA+0x1C4 | R/W | AES Key Word 0 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_KEY1** | CRYP_BA+0x1C8 | R/W | AES Key Word 1 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_KEY2** | CRYP_BA+0x1CC | R/W | AES Key Word 2 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_KEY3** | CRYP_BA+0x1D0 | R/W | AES Key Word 3 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_KEY4** | CRYP_BA+0x1D4 | R/W | AES Key Word 4 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_KEY5** | CRYP_BA+0x1D8 | R/W | AES Key Word 5 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_KEY6** | CRYP_BA+0x1DC | R/W | AES Key Word 6 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_KEY7** | CRYP_BA+0x1E0 | R/W | AES Key Word 7 Register for Channel 3 | 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** | **CRPT_AESn_KEYx**<br>The KEY keeps the security key for AES operation.<br>n = 0, 1..3.<br>x = 0, 1..7.<br>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. {CRPT_AESn_KEY3, CRPT_AESn_KEY2, CRPT_AESn_KEY1, CRPT_AESn_KEY0} stores the 128-bit security key for AES operation. {CRPT_AESn_KEY5, CRPT_AESn_KEY4, CRPT_AESn_KEY3, CRPT_AESn_KEY2, CRPT_AESn_KEY1, CRPT_AESn_KEY0} stores the 192-bit security key for AES operation. {CRPT_AESn_KEY7, CRPT_AESn_KEY6, CRPT_AESn_KEY5, CRPT_AESn_KEY4, CRPT_AESn_KEY3, CRPT_AESn_KEY2, CRPT_AESn_KEY1, CRPT_AESn_KEY0} stores the 256-bit security key for AES operation. |

## AES Initial Vector Word x Register (CRPT_AES0_IVx, CRPT_AES1_IVx, CRPT_AES2_IVx, CRPT_AES3_IVx)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_AES0_IV0** | CRYP_BA+0x130 | R/W | AES Initial Vector Word 0 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_IV1** | CRYP_BA+0x134 | R/W | AES Initial Vector Word 1 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_IV2** | CRYP_BA+0x138 | R/W | AES Initial Vector Word 2 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES0_IV3** | CRYP_BA+0x13C | R/W | AES Initial Vector Word 3 Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES1_IV0** | CRYP_BA+0x16C | R/W | AES Initial Vector Word 0 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_IV1** | CRYP_BA+0x170 | R/W | AES Initial Vector Word 1 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_IV2** | CRYP_BA+0x174 | R/W | AES Initial Vector Word 2 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES1_IV3** | CRYP_BA+0x178 | R/W | AES Initial Vector Word 3 Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES2_IV0** | CRYP_BA+0x1A8 | R/W | AES Initial Vector Word 0 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_IV1** | CRYP_BA+0x1AC | R/W | AES Initial Vector Word 1 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_IV2** | CRYP_BA+0x1B0 | R/W | AES Initial Vector Word 2 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES2_IV3** | CRYP_BA+0x1B4 | R/W | AES Initial Vector Word 3 Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES3_IV0** | CRYP_BA+0x1E4 | R/W | AES Initial Vector Word 0 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_IV1** | CRYP_BA+0x1E8 | R/W | AES Initial Vector Word 1 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_IV2** | CRYP_BA+0x1EC | R/W | AES Initial Vector Word 2 Register for Channel 3 | 0x0000_0000 |
| **CRPT_AES3_IV3** | CRYP_BA+0x1F0 | R/W | AES Initial Vector Word 3 Register for Channel 3 | 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**<br>n = 0, 1..3.<br>x = 0, 1..3.<br>Four initial vectors (CRPT_AESn_IV0, CRPT_AESn_IV1, CRPT_AESn_IV2, and CRPT_AESn_IV3) are for AES operating in CBC, CFB, and OFB mode. Four registers (CRPT_AESn_IV0, CRPT_AESn_IV1, CRPT_AESn_IV2, and CRPT_AESn_IV3) act as Nonce counter when the AES engine is operating in CTR mode. |

## AES DMA Source Address Register (CRPT_AES0_SADDR, CRPT_AES1_SADDR, CRPT_AES2_SADDR, CRPT_AES3_SADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_AES0_SADDR** | CRYP_BA+0x140 | R/W | AES DMA Source Address Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES1_SADDR** | CRYP_BA+0x17C | R/W | AES DMA Source Address Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES2_SADDR** | CRYP_BA+0x1B8 | R/W | AES DMA Source Address Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES3_SADDR** | CRYP_BA+0x1F4 | R/W | AES DMA Source Address Register for Channel 3 | 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**<br>The AES accelerator supports DMA function to transfer the plain text between system memory 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 from system memory 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.<br>SADDR can be read and written. Writing to SADDR while the AES accelerator is operating doesn't 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.<br>In DMA mode, software can update the next CRPT_AESn_SADDR before triggering START.<br>The value of CRPT_AESn_SADDR and CRPT_AESn_DADDR can be the same. |

## AES DMA Destination Address Register (CRPT_AES0_DADDR, CRPT_AES1_DADDR, CRPT_AES2_DADDR, CRPT_AES3_DADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_AES0_DADDR** | CRYP_BA+0x144 | R/W | AES DMA Destination Address Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES1_DADDR** | CRYP_BA+0x180 | R/W | AES DMA Destination Address Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES2_DADDR** | CRYP_BA+0x1BC | R/W | AES DMA Destination Address Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES3_DADDR** | CRYP_BA+0x1F8 | R/W | AES DMA Destination Address Register for Channel 3 | 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**<br>The AES accelerator supports DMA function to transfer the cipher text between system memory 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 back to system memory 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.<br>DADDR can be read and written. Writing to DADDR while the AES accelerator is operating doesn't 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.<br>In DMA mode, software can update the next CRPT_AESn_DADDR before triggering START.<br>The value of CRPT_AESn_SADDR and CRPT_AESn_DADDR can be the same. |

## AES Byte Count Register (CRPT_AES0_CNT, CRPT_AES1_CNT, CRPT_AES2_CNT, CRPT_AES3_CNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_AES0_CNT** | CRYP_BA+0x148 | R/W | AES Byte Count Register for Channel 0 | 0x0000_0000 |
| **CRPT_AES1_CNT** | CRYP_BA+0x184 | R/W | AES Byte Count Register for Channel 1 | 0x0000_0000 |
| **CRPT_AES2_CNT** | CRYP_BA+0x1C0 | R/W | AES Byte Count Register for Channel 2 | 0x0000_0000 |
| **CRPT_AES3_CNT** | CRYP_BA+0x1FC | R/W | AES Byte Count Register for Channel 3 | 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**<br>The CRPT_AESn_CNT keeps the byte count of source text that is for the AES engine operating in DMA mode. The CRPT_AESn_CNT is 32-bit and the maximum of byte count is 4G bytes.<br>CRPT_AESn_CNT can be read and written. Writing to CRPT_AESn_CNT while the AES accelerator is operating doesn't affect the current AES operation. But the value of CRPT_AESn_CNT will be updated later on. Consequently, software can prepare the byte count of data for the next AES operation.<br>According to CBC-CS1, CBC-CS2, and CBC-CS3 standard, the count of operation data must be at least one block. Operations that are less than one block will output unexpected result.<br>In Non-DMA ECB, CBC, CFB, OFB, and CTR mode, CRPT_AESn_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, CRPT_AESn_CNT must be set as byte count for the last two blocks of data before feeding in the last two blocks of data. |

### AES Feedback x Register (CRPT_AES_FDBCKx)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_AES_FDBCK0** | CRYP_BA+0x050 | R | AES Engine Output Feedback Data after Cryptographic Operation | 0x0000_0000 |
| **CRPT_AES_FDBCK1** | CRYP_BA+0x054 | R | AES Engine Output Feedback Data after Cryptographic Operation | 0x0000_0000 |
| **CRPT_AES_FDBCK2** | CRYP_BA+0x058 | R | AES Engine Output Feedback Data after Cryptographic Operation | 0x0000_0000 |
| **CRPT_AES_FDBCK3** | CRYP_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**<br>The feedback value is 128 bits in size.<br>The AES engine uses the data from CRPT_AES_FDBCKx as the data inputted to CRPT_AESn_IVx for the next block in DMA cascade mode.<br>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 this register in the same channel operation, and then continue the operation with the original setting. |

5.27.7.4 *TDES/DES Register*

## TDES/DES Control Register (CRPT_TDES_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CRPT_TDES_CTL | CRYP_BA+0x200 | R/W | TDES/DES Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| KEYPRT | KEYUNPRT | | | | | CHANNEL | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| INSWAP | OUTSWAP | BLKSWAP | Reserved | | | | ENCRPT |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | OPMODE | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DMAEN | DMACSCAD | DMALAST | Reserved | 3KEYS | TMODE | STOP | START |

| Bits | Description | |
|---|---|---|
| [31] | **KEYPRT** | **Protect Key**<br>Read as a flag to reflect KEYPRT.<br>0 = No effect.<br>1 = This bit is to protect the content of TDES key from reading. The return value for reading CRPT_ TDESn_KEYxH/L is not the content in the registers CRPT_ TDESn_KEYxH/L. Once it is set, it can be cleared by asserting KEYUNPRT. The key content would be cleared as well. |
| [30:26] | **KEYUNPRT** | **Unprotect Key**<br>Writing 0 to CRPT_TDES_CTL [31] and “10110” to CRPT_TDES_CTL [30:26] is to unprotect TDES key.<br>The KEYUNPRT can be read and written. When it is written as the TDES engine is operating, BUSY flag is 1, there would be no effect on KEYUNPRT. |
| [25:24] | **CHANNEL** | **TDES/DES Engine Working Channel**<br>00 = Current control register setting is for channel 0.<br>01 = Current control register setting is for channel 1.<br>10 = Current control register setting is for channel 2.<br>11 = Current control register setting is for channel 3. |
| [23] | **INSWAP** | **TDES/DES Engine Input Data Swap**<br>0 = Keep the original order.<br>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** | **TDES/DES Engine Output Data Swap**<br>0 = Keep the original order.<br>1 = The order that CPU outputs data from the accelerator will be changed from {byte3, byte2, byte1, byte0} to {byte0, byte1, byte2, byte3}. |

| | | |
|---|---|---|
| [21] | **BLKSWAP** | **TDES/DES Engine Block Double Word Endian Swap**<br>0 = Keep the original order, e.g. **{WORD_H, WORD_L}**.<br>1 = When this bit is set to 1, the TDES engine would exchange high and low word in the sequence **{WORD_L, WORD_H}.** |
| [20:17] | **Reserved** | Reserved. |
| [16] | **ENCRPT** | **TDES/DES Encryption/Decryption**<br>0 = TDES engine executes decryption operation.<br>1 = TDES engine executes encryption operation. |
| [15:11] | **Reserved** | Reserved. |
| [10:8] | **OPMODE** | **TDES/DES Engine Operation Mode**<br>0x00 = ECB (Electronic Codebook Mode).<br>0x01 = CBC (Cipher Block Chaining Mode).<br>0x02 = CFB (Cipher Feedback Mode).<br>0x03 = OFB (Output Feedback Mode).<br>0x04 = CTR (Counter Mode).<br>Others = CTR (Counter Mode). |
| [7] | **DMAEN** | **TDES/DES Engine DMA Enable Control**<br>0 = TDES_DMA engine Disabled.<br>TDES engine operates in Non-DMA mode, and get data from the port CRPT_TDES_DATIN.<br>1 = TDES_DMA engine Enabled.<br>TDES engine operates in DMA mode, and data movement from/to the engine is done by DMA logic. |
| [6] | **DMACSCAD** | **TDES/DES Engine DMA with Cascade Mode**<br>0 = DMA cascade function Disabled.<br>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** | **TDES/DES Engine Start for the Last Block**<br>In DMA mode, this bit must be set as beginning the last DMA cascade round.<br>In Non-DMA mode, this bit must be set as feeding in last block of data. |
| [4] | **Reserved** | Reserved. |
| [3] | **3KEYS** | **TDES/DES Key Number**<br>0 = Select KEY1 and KEY2 in TDES/DES engine.<br>1 = Triple keys in TDES/DES engine Enabled. |
| [2] | **TMODE** | **TDES/DES Engine Operating Mode**<br>0 = Set DES mode for TDES/DES engine.<br>1 = Set Triple DES mode for TDES/DES engine. |
| [1] | **STOP** | **TDES/DES Engine Stop**<br>0 = No effect.<br>1 = Stop TDES/DES engine.<br>**Note:** The bit is always 0 when it's read back. |

| | | |
|---|---|---|
| [0] | **START** | **TDES/DES Engine Start**<br>0 = No effect.<br>1 = Start TDES/DES engine. The flag BUSY would be set.<br>**Note:** The bit is always 0 when it's read back. |

### TDES/DES Status Flag Register (CRPT_TDES0_STS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CRPT_TDES_STS | CRYP_BA+0x204 | R | TDES/DES Engine Flag | 0x0001_0100 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | BUSERR | Reserved | OUTBUFERR | OUTBUFFULL | OUTBUFEMPTY |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | INBUFERR | INBUFFULL | INBUFEMPTY |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | BUSY |

| Bits | Description | |
|---|---|---|
| [31:21] | **Reserved** | Reserved. |
| [20] | **BUSERR** | **TDES/DES DMA Access Bus Error Flag**<br>0 = No error.<br>1 = Bus error will stop DMA operation and TDES/DES engine. |
| [19] | **Reserved** | Reserved. |
| [18] | **OUTBUFERR** | **TDES/DES Out Buffer Error Flag**<br>0 = No error.<br>1 = Error happens during getting test result from TDES/DES engine. |
| [17] | **OUTBUFFULL** | **TDES/DES Output Buffer Full Flag**<br>0 = TDES/DES output buffer is not full.<br>1 = TDES/DES output buffer is full, and software needs to get data from TDES_DATA_OUT. Otherwise, the TDES/DES engine will be pending since output buffer is full. |
| [16] | **OUTBUFEMPTY** | **TDES/DES Output Buffer Empty Flag**<br>0 = TDES/DES output buffer is not empty. There are some valid data kept in output buffer.<br>1 = TDES/DES output buffer is empty, Software cannot get data from TDES_DATA_OUT. Otherwise the flag OUTBUFERR will be set to 1, since output buffer is empty. |
| [15:11] | **Reserved** | Reserved. |
| [10] | **INBUFERR** | **TDES/DES in Buffer Error Flag**<br>0 = No error.<br>1 = Error happens during feeding data to the TDES/DES engine. |

| | | |
|---|---|---|
| [9] | **INBUFFULL** | **TDES/DES in Buffer Full Flag**<br>0 = TDES/DES input buffer is not full. Software can feed the data into the TDES/DES engine.<br>1 = TDES input buffer is full. Software cannot feed data to the TDES/DES engine. Otherwise, the flag INBUFERR will be set to 1. |
| [8] | **INBUFEMPTY** | **TDES/DES in Buffer Empty**<br>0 = There are some data in input buffer waiting for the TDES/DES engine to process.<br>1 = TDES/DES input buffer is empty. Software needs to feed data to the TDES/DES engine. Otherwise, the TDES/DES engine will be pending to wait for input data. |
| [7:1] | **Reserved** | Reserved. |
| [0] | **BUSY** | **TDES/DES Engine Busy**<br>0 = TDES/DES engine is idle or finished.<br>1 = TDES/DES engine is under processing. |

## TDES/DES Key 1, 2, 3 High/Low Word Register (TDES_KEY1H/L, TDES_KEY2H/L, TDES_KEY3H/L)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_TDES0_KEY1H** | CRYP_BA+0x208 | R/W | TDES/DES Key 1 High Word Register for Channel 0 | 0x0000_0000 |
| **CRPT_TDES0_KEY1L** | CRYP_BA+0x20C | R/W | TDES/DES Key 1 Low Word Register for Channel 0 | 0x0000_0000 |
| **CRPT_TDES0_KEY2H** | CRYP_BA+0x210 | R/W | TDES Key 2 High Word Register for Channel 0 | 0x0000_0000 |
| **CRPT_TDES0_KEY2L** | CRYP_BA+0x214 | R/W | TDES Key 2 Low Word Register for Channel 0 | 0x0000_0000 |
| **CRPT_TDES0_KEY3H** | CRYP_BA+0x218 | R/W | TDES Key 3 High Word Register for Channel 0 | 0x0000_0000 |
| **CRPT_TDES0_KEY3L** | CRYP_BA+0x21C | R/W | TDES Key 3 Low Word Register for Channel 0 | 0x0000_0000 |
| **CRPT_TDES1_KEY1H** | CRYP_BA+0x248 | R/W | TDES/DES Key 1 High Word Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES1_KEY1L** | CRYP_BA+0x24C | R/W | TDES/DES Key 1 Low Word Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES1_KEY2H** | CRYP_BA+0x250 | R/W | TDES Key 2 High Word Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES1_KEY2L** | CRYP_BA+0x254 | R/W | TDES Key 2 Low Word Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES1_KEY3H** | CRYP_BA+0x258 | R/W | TDES Key 3 High Word Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES1_KEY3L** | CRYP_BA+0x25C | R/W | TDES Key 3 Low Word Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES2_KEY1H** | CRYP_BA+0x288 | R/W | TDES/DES Key 1 High Word Register for Channel 2 | 0x0000_0000 |
| **CRPT_TDES2_KEY1L** | CRYP_BA+0x28C | R/W | TDES/DES Key 1 Low Word Register for Channel 2 | 0x0000_0000 |
| **CRPT_TDES2_KEY2H** | CRYP_BA+0x290 | R/W | TDES Key 2 High Word Register for Channel 2 | 0x0000_0000 |
| **CRPT_TDES2_KEY2L** | CRYP_BA+0x294 | R/W | TDES Key 2 Low Word Register for Channel 2 | 0x0000_0000 |
| **CRPT_TDES2_KEY3H** | CRYP_BA+0x298 | R/W | TDES Key 3 High Word Register for Channel 2 | 0x0000_0000 |
| **CRPT_TDES2_KEY3L** | CRYP_BA+0x29C | R/W | TDES Key 3 Low Word Register for Channel 2 | 0x0000_0000 |
| **CRPT_TDES3_KEY1H** | CRYP_BA+0x2C8 | R/W | TDES/DES Key 1 High Word Register for Channel 3 | 0x0000_0000 |
| **CRPT_TDES3_KEY1L** | CRYP_BA+0x2CC | R/W | TDES/DES Key 1 Low Word Register for Channel 3 | 0x0000_0000 |
| **CRPT_TDES3_KEY2H** | CRYP_BA+0x2D0 | R/W | TDES Key 2 High Word Register for Channel 3 | 0x0000_0000 |
| **CRPT_TDES3_KEY2L** | CRYP_BA+0x2D4 | R/W | TDES Key 2 Low Word Register for Channel 3 | 0x0000_0000 |
| **CRPT_TDES3_KEY3H** | CRYP_BA+0x2D8 | R/W | TDES Key 3 High Word Register for Channel 3 | 0x0000_0000 |
| **CRPT_TDES3_KEY3L** | CRYP_BA+0x2DC | R/W | TDES Key 3 Low Word Register for Channel 3 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **KEYH/KEYL** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **KEYH/KEYL** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **KEYH/KEYL** | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| KEYH/KEYL | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:0] | **KEYH/KEYL** | **TDES/DES Key High/Low Word**<br>The key registers for TDES/DES algorithm calculation<br>The security key for the TDES/DES accelerator is 64 bits. Thus, it needs two 32-bit registers to store a security key. The register CRPT_TDESn_KEYxH is used to keep the bit [63:32] of security key for the TDES/DES operation, while the register CRPT_TDESn_KEYxL is used to keep the bit [31:0]. |

### TDES/DES IV High/Low Word Register (CRPT_TDES0_IVH/L, CRPT_TDES1_IVH/L, CRPT_TDES2_IVH/L, CRPT_TDES3_IVH/L)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_TDES0_IVH** | CRYP_BA+0x220 | R/W | TDES/DES Initial Vector High Word Register for Channel 0 | 0x0000_0000 |
| **CRPT_TDES0_IVL** | CRYP_BA+0x224 | R/W | TDES/DES Initial Vector Low Word Register for Channel 0 | 0x0000_0000 |
| **CRPT_TDES1_IVH** | CRYP_BA+0x260 | R/W | TDES/DES Initial Vector High Word Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES1_IVL** | CRYP_BA+0x264 | R/W | TDES/DES Initial Vector Low Word Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES2_IVH** | CRYP_BA+0x2A0 | R/W | TDES/DES Initial Vector High Word Register for Channel 2 | 0x0000_0000 |
| **CRPT_TDES2_IVL** | CRYP_BA+0x2A4 | R/W | TDES/DES Initial Vector Low Word Register for Channel 2 | 0x0000_0000 |
| **CRPT_TDES3_IVH** | CRYP_BA+0x2E0 | R/W | TDES/DES Initial Vector High Word Register for Channel 3 | 0x0000_0000 |
| **CRPT_TDES3_IVL** | CRYP_BA+0x2E4 | R/W | TDES/DES Initial Vector Low Word Register for Channel 3 | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **IVH/IVL** | **TDES/DES Initial Vector High/Low Word**<br>Initial vector (IV) is for TDES/DES engine in CBC, CFB, and OFB mode. IV is Nonce counter for TDES/DES engine in CTR mode. |

## TDES/DES DMA Source Address Register (CRPT_TDES0_SADDR, CRPT_TDES1_SADDR, CRPT_TDES2_SADDR, CRPT_TDES3_SADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_TDES0_SADDR** | CRYP_BA+0x228 | R/W | TDES/DES DMA Source Address Register for Channel 0 | 0x0000_0000 |
| **CRPT_TDES1_SADDR** | CRYP_BA+0x268 | R/W | TDES/DES DMA Source Address Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES2_SADDR** | CRYP_BA+0x2A8 | R/W | TDES/DES DMA Source Address Register for Channel 2 | 0x0000_0000 |
| **CRPT_TDES3_SADDR** | CRYP_BA+0x2E8 | R/W | TDES/DES DMA Source Address Register for Channel 3 | 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** | **TDES/DES DMA Source Address**<br>The TDES/DES accelerator supports DMA function to transfer the plain text between system memory and embedded FIFO. The CRPT_TDESn_SADDR keeps the source address of the data buffer where the source text is stored. Based on the source address, the TDES/DES accelerator can read the plain text from system memory and do TDES/DES operation. The start of source address should be located at word boundary. In other words, bit 1 and 0 of CRPT_TDESn_SADDR are ignored.<br>CRPT_TDESn_SADDR can be read and written. Writing to CRPT_TDESn_SADDR while the TDES/DES accelerator is operating doesn't affect the current TDES/DES operation. But the value of CRPT_TDESn_SADDR will be updated later on. Consequently, software can prepare the DMA source address for the next TDES/DES operation.<br>In DMA mode, software can update the next CRPT_TDESn_SADDR before triggering START.<br>CRPT_TDESn_SADDR and CRPT_TDESn_DADDR can be the same in the value. |

## TDES/DES DMA Destination Address Register (CRPT_TDES0_DADDR, CRPT_TDES1_DADDR, CRPT_TDES2_DADDR, CRPT_TDES3_DADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_TDES0_DADDR** | CRYP_BA+0x22C | R/W | TDES/DES DMA Destination Address Register for Channel 0 | 0x0000_0000 |
| **CRPT_TDES1_DADDR** | CRYP_BA+0x26C | R/W | TDES/DES DMA Destination Address Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES2_DADDR** | CRYP_BA+0x2AC | R/W | TDES/DES DMA Destination Address Register for Channel 2 | 0x0000_0000 |
| **CRPT_TDES3_DADDR** | CRYP_BA+0x2EC | R/W | TDES/DES DMA Destination Address Register for Channel 3 | 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] | **TDES_DADR** | **TDES/DES DMA Destination Address**<br>The TDES/DES accelerator supports DMA function to transfer the cipher text between system memory and embedded FIFO. The CRPT_TDESn_DADDR keeps the destination address of the data buffer where the engine output's text will be stored. Based on the destination address, the TDES/DES accelerator can write the cipher text back to system memory after the TDES/DES operation is finished. The start of destination address should be located at word boundary. In other words, bit 1 and 0 of CRPT_TDESn_DADDR are ignored.<br>CRPT_TDESn_DADDR can be read and written. Writing to CRPT_TDESn_DADDR while the TDES/DES accelerator is operating doesn't affect the current TDES/DES operation. But the value of CRPT_TDESn_DADDR will be updated later on. Consequently, software can prepare the destination address for the next TDES/DES operation.<br>In DMA mode, software can update the next CRPT_TDESn_DADDR before triggering START.<br>CRPT_TDESn_SADDR and CRPT_TDESn_DADDR can be the same in the value. |

## TDES/DES Block Count Register (CRPT_TDES0_CNT, CRPT_TDES1_CNT, CRPT_TDES2_CNT, CRPT_TDES3_CNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_TDES0_CNT** | CRYP_BA+0x230 | R/W | TDES/DES Byte Count Register for Channel 0 | 0x0000_0000 |
| **CRPT_TDES1_CNT** | CRYP_BA+0x270 | R/W | TDES/DES Byte Count Register for Channel 1 | 0x0000_0000 |
| **CRPT_TDES2_CNT** | CRYP_BA+0x2B0 | R/W | TDES/DES Byte Count Register for Channel 2 | 0x0000_0000 |
| **CRPT_TDES3_CNT** | CRYP_BA+0x2F0 | R/W | TDES/DES Byte Count Register for Channel 3 | 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** | **TDES/DES Byte Count**<br>The CRPT_TDESn_CNT keeps the byte count of source text that is for the TDES/DES engine operating in DMA mode. The CRPT_TDESn_CNT is 32-bit and the maximum of byte count is 4G bytes.<br>CRPT_TDESn_CNT can be read and written. Writing to CRPT_TDESn_CNT while the TDES/DES accelerator is operating doesn't affect the current TDES/DES operation. But the value of CRPT_TDESn_CNT will be updated later on. Consequently, software can prepare the byte count of data for the next TDES /DES operation.<br>In Non-DMA ECB, CBC, CFB, OFB, and CTR mode, CRPT_TDESn_CNT must be set as byte count for the last block of data before feeding in the last block of data. |

### TDES/DES Data Input Port Register (CRPT_TDES_DATIN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_TDES_DATIN** | CRYP_BA+0x234 | R/W | TDES/DES Engine Input data Word 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** | **TDES/DES Engine Input Port**<br>CPU feeds data to TDES/DES engine through this port by checking CRPT_TDES_STS. Feed data as INBUFFULL is 0. |

### TDES/DES Data Output Port Register (CRPT_TDES_DATOUT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_TDES_DATOUT** | CRYP_BA+0x238 | R | TDES/DES Engine Output data Word 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** | **TDES/DES Engine Output Port**<br>CPU gets result from the TDES/DES engine through this port by checking CRPT_TDES_STS. Get data as OUTBUFEMPTY is 0. |

## TDES/DES Feedback x Register (CRPT_TDES_FDBCKx)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_TDES_FDBCKH** | CRYP_BA+0x060 | R | TDES/DES Engine Output Feedback High Word Data after Cryptographic Operation | 0x0000_0000 |
| **CRPT_TDES_FDBCKL** | CRYP_BA+0x064 | R | TDES/DES Engine Output Feedback Low Word 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** | **TDES/DES Feedback**<br>The feedback value is 64 bits in size.<br>The TDES/DES engine uses the data from {CRPT_TDES_FDBCKH, CRPT_TDES_FDBCKL} as the data inputted to {CRPT_TDESn_IVH, CRPT_TDESn_IVL} for the next block in DMA cascade mode. The feedback register is for CBC, CFB, and OFB mode.<br>TDES/DES 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 this register in the same channel operation. Then can continue the operation with the original setting. |

5.27.7.5 *SHA/HMAC Register*

## SHA/HMAC Control Register (CRPT_HMAC_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CRPT_HMAC_CTL | CRYP_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 | Reserved | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CMPEN | Reserved | | | | OPMODE | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DMAEN | Reserved | DMALAST | HMACEN | Reserved | | STOP | START |

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23] | **INSWAP** | **SHA/HMAC Engine Input Data Swap**<br>0 = Keep the original order.<br>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**<br>0 = Keep the original order.<br>1 = The order that CPU feeds data to the accelerator will be changed from {byte3, byte2, byte1, byte0} to {byte0, byte1, byte2, byte3}. |
| [21:16] | **Reserved** | Reserved. |
| [15] | **CMPEN** | **Compare SHA/HMAC Output Digest with MTP Key**<br>0 = Don't compare with MTP key.<br>1 = The SHA/HMAC output digest would be compared with MTP key. Since MTP is 256 bits in size, HMAC digest comparing is not supported in HMAC-SHA-384 and HMAC-SHA-512. |
| [14:11] | **Reserved** | Reserved. |
| [10:8] | **OPMODE** | **SHA/HMAC Engine Operation Modes**<br>0x0xx: SHA160<br>0x100: SHA256<br>0x101: SHA224<br>0x110: SHA512<br>0x111: SHA384<br>These bits can be read and written. But writing to them wouldn't take effect as BUSY is 1. |

| | | |
|---|---|---|
| [7] | **DMAEN** | **SHA/HMAC Engine DMA Enable Control**<br>0 = SHA/HMAC DMA engine Disabled.<br>SHA/HMAC engine operates in Non-DMA mode, and gets data from the port CRPT_HMAC_DATIN.<br>1 = SHA/HMAC DMA engine Enabled.<br>SHA/HMAC engine operates in DMA mode, and data movement from/to the engine is done by DMA logic. |
| [6] | **Reserved** | Reserved. |
| [5] | **DMALAST** | **SHA/HMAC Last Block**<br>In DMA mode, this bit must be set as beginning the last DMA cascade round.<br>In Non-DMA mode, this bit must be set as feeding in last byte of data. |
| [4] | **HMACEN** | **HMAC_SHA Engine Operating Mode**<br>0 = execute SHA function.<br>1 = execute HMAC function. |
| [3:2] | **Reserved** | Reserved. |
| [1] | **STOP** | **SHA/HMAC Engine Stop**<br>0 = No effect.<br>1 = Stop SHA/HMAC engine.<br>This bit is always 0 when it's read back. |
| [0] | **START** | **SHA/HMAC Engine Start**<br>0 = No effect.<br>1 = Start SHA/HMAC engine. BUSY flag will be set.<br>This bit is always 0 when it's read back. |

## SHA/HMAC Status Register (CRPT_HMAC_STS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CRPT_HMAC_STS | CRYP_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 |
| CMPSTS | Reserved | | | | | | DMAERR |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | DMABUSY | BUSY |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [16] | **DATINREQ** | **SHA/HMAC Non-dMA Mode Data Input Request**<br>0 = No effect.<br>1 = Request SHA/HMAC Non-DMA mode data input. |
| [15] | **CMPSTS** | **SHA/HMAC Output Digest Compare Result with MTP Key**<br>0 = SHA/HMAC output digest doesn't match MTP key.<br>1 = SHA/HMAC output digest matches MTP key. |
| [14:9] | **Reserved** | Reserved. |
| [8] | **DMAERR** | **SHA/HMAC Engine DMA Error Flag**<br>0 = Show the SHA/HMAC engine access normal.<br>1 = Show the SHA/HMAC engine access error. |
| [7:2] | **Reserved** | Reserved. |
| [1] | **DMABUSY** | **SHA/HMAC Engine DMA Busy Flag**<br>0 = SHA/HMAC DMA engine is idle or finished.<br>1 = SHA/HMAC DMA engine is busy. |
| [0] | **BUSY** | **SHA/HMAC Engine Busy**<br>0 = SHA/HMAC engine is idle or finished.<br>1 = SHA/HMAC engine is busy. |

### SHA/HMAC Outputs Digest Word Register (CRPT_HMAC_DGSTx)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_HMAC_DGST0** | CRYP_BA+0x308 | R | SHA/HMAC Digest Message 0 | 0x0000_0000 |
| **CRPT_HMAC_DGST1** | CRYP_BA+0x30C | R | SHA/HMAC Digest Message 1 | 0x0000_0000 |
| **CRPT_HMAC_DGST2** | CRYP_BA+0x310 | R | SHA/HMAC Digest Message 2 | 0x0000_0000 |
| **CRPT_HMAC_DGST3** | CRYP_BA+0x314 | R | SHA/HMAC Digest Message 3 | 0x0000_0000 |
| **CRPT_HMAC_DGST4** | CRYP_BA+0x318 | R | SHA/HMAC Digest Message 4 | 0x0000_0000 |
| **CRPT_HMAC_DGST5** | CRYP_BA+0x31C | R | SHA/HMAC Digest Message 5 | 0x0000_0000 |
| **CRPT_HMAC_DGST6** | CRYP_BA+0x320 | R | SHA/HMAC Digest Message 6 | 0x0000_0000 |
| **CRPT_HMAC_DGST7** | CRYP_BA+0x324 | R | SHA/HMAC Digest Message 7 | 0x0000_0000 |
| **CRPT_HMAC_DGST8** | CRYP_BA+0x328 | R | SHA/HMAC Digest Message 8 | 0x0000_0000 |
| **CRPT_HMAC_DGST9** | CRYP_BA+0x32C | R | SHA/HMAC Digest Message 9 | 0x0000_0000 |
| **CRPT_HMAC_DGST10** | CRYP_BA+0x330 | R | SHA/HMAC Digest Message 10 | 0x0000_0000 |
| **CRPT_HMAC_DGST11** | CRYP_BA+0x334 | R | SHA/HMAC Digest Message 11 | 0x0000_0000 |
| **CRPT_HMAC_DGST12** | CRYP_BA+0x338 | R | SHA/HMAC Digest Message 12 | 0x0000_0000 |
| **CRPT_HMAC_DGST13** | CRYP_BA+0x33C | R | SHA/HMAC Digest Message 13 | 0x0000_0000 |
| **CRPT_HMAC_DGST14** | CRYP_BA+0x340 | R | SHA/HMAC Digest Message 14 | 0x0000_0000 |
| **CRPT_HMAC_DGST15** | CRYP_BA+0x344 | R | SHA/HMAC Digest Message 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 Digest Message Output Register**<br>For SHA-160, the digest is stored in CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST4.<br>For SHA-224, the digest is stored in CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST6.<br>For SHA-256, the digest is stored in CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST7.<br>For SHA-384, the digest is stored in CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST11.<br>For SHA-512, the digest is stored in CRPT_HMAC_DGST0 ~ CRPT_HMAC_DGST15. |

### SHA/HMAC Key Byte Count Register (CRPT_HMAC_KEYCNT )

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CRPT_HMAC_KEYCNT | CRYP_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**<br>The CRPT_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.<br>Writing to the register CRPT_HMAC_KEYCNT as the SHA/HMAC accelerator operating doesn't affect the current SHA/HMAC operation. But the value of CRPT_SHA _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 (CRPT_HMAC_SADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_HMAC_SADDR** | CRYP_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**<br>The SHA/HMAC accelerator supports DMA function to transfer the plain text between system memory and embedded FIFO. The CRPT_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 system memory and do SHA/HMAC operation. The start of source address should be located at word boundary. In other words, bit 1 and 0 of CRPT_HMAC_SADDR are ignored.<br>CRPT_HMAC_SADDR can be read and written. Writing to CRPT_HMAC_SADDR while the SHA/HMAC accelerator is operating doesn't affect the current SHA/HMAC operation. But the value of CRPT_HMAC_SADDR will be updated later on. Consequently, software can prepare the DMA source address for the next SHA/HMAC operation.<br>In DMA mode, software can update the next CRPT_HMAC_SADDR before triggering START.<br>CRPT_HMAC_SADDR and CRPT_HMAC_DADDR can be the same in the value. |

### SHA/HMAC Byte Count Register (CRPT_HMAC_DMACNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_HMAC_DMACNT** | CRYP_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**<br>The CRPT_HMAC_DMACNT keeps the byte count of source text that is for the SHA/HMAC engine operating in DMA mode. The CRPT_HMAC_DMACNT is 32-bit and the maximum of byte count is 4G bytes.<br>CRPT_HMAC_DMACNT can be read and written. Writing to CRPT_HMAC_DMACNT while the SHA/HMAC accelerator is operating doesn't affect the current SHA/HMAC operation. But the value of CRPT_HMAC_DMACNT will be updated later on. Consequently, software can prepare the byte count of data for the next SHA/HMAC operation.<br>In Non-DMA mode, CRPT_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 (CRPT_HMAC_DATIN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRPT_HMAC_DATIN** | CRYP_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**<br>CPU feeds data to SHA/HMAC engine through this port by checking CRPT_HMAC_STS. Feed data as DATINREQ is 1. |

## 5.28 2D Graphic Engine (GE2D)

### 5.28.1 Overview

A 32-bit 2D Graphics Engine (GE2D) is specially designed to improve the performance of graphic processing. It can accelerate the operation of individual GUI functions such as BitBLTs and Bresenham Line Draw to operate at all pixel depths including 8/16/32 bit-per-pixel.

A pixel is the smallest addressable screen element as defined in Microsoft Windows, and lines and pictures are made up by a variety of pixels. GE2D is used to speed up graphic performance in pixel data moving and line drawing, as well as to accelerate almost all computer graphic Boolean operations by eliminating the CPU overhead. Meanwhile, the functions of rotation and scaling down are implemented for some special applications. In image scaling down function, both programmable horizontal and vertical N/M scaling down factors are provided for resizing the image. For the 2D rotation, it can rotate left or right 45, 90 or 180 degrees, and also supports the flip/flop, mirror or up-side-down pictures.

### 5.28.2 Features

- Support 2D Bit Block Transfer (BitBLT) functions defined in Microsoft GDI
- Support Host BLT
- Support Pattern BLT
- Support Color/Font Expanding BLT
- Support Transparent BLT
- Support Tile BLT
- Support Block Move BLT
- Support Copy File BLT
- Support Color/Font Expansion
- Support Rectangle Fill
- Support RGB332/RGB565/RGB888 data format.
- Support fore/background colors and all Microsoft 256 ternary raster-operation codes (ROP)
- Support both inside and outside clipping function
- Support alpha-blending for source/destination picture overlaying
- Support fast Bresenham line drawing algorithm to draw solid/textured line
- Support rectangular border and frame drawing
- Support picture re-sizing
- Support down-scaling from 1/255 to 254/255
- Support up-scaling from 1 to 1.996 (1+254/255)
- Support object rotation with different degree
- Support L45 (45 degree left rotation) and L90 (90 degree left rotation)
- Support R45 (45 degree right rotation) and R90 (90 degree right rotation)
- Support M180 (mirror/flop)

- Support F180 (up-side-down (flip) and X180 (180 degree rotation)

### 5.28.3 Block Diagram

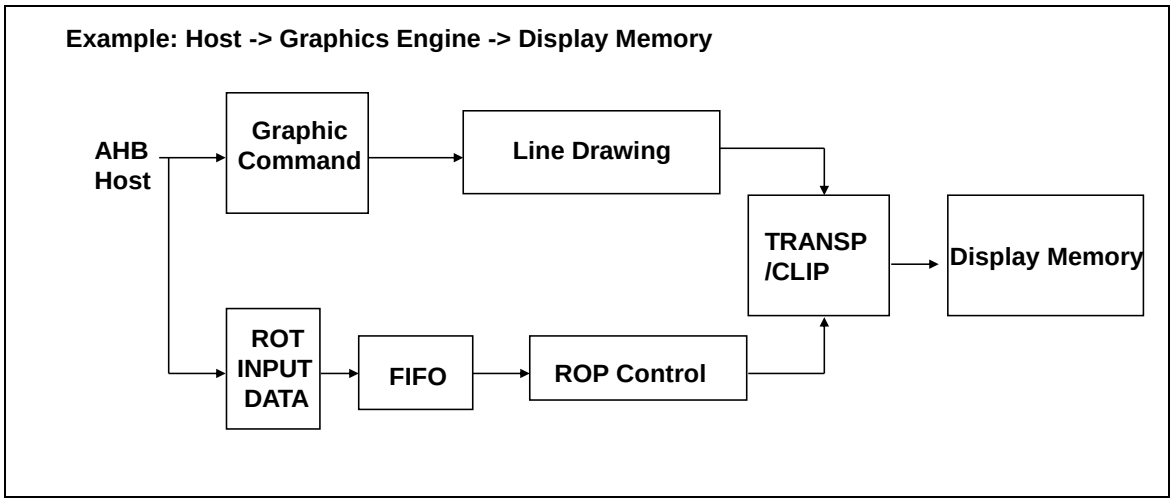



Figure 5.28-1 2-D GE Engine Architecture

### 5.28.4 Basic Configuration

Before using 2D graphic engine, it's necessary to enable clock of 2D graphic engine. Set GE2D (CLK_HCLKEN[28]) high to enable clock for 2D graphic engine operation.

### 5.28.5 Functional Description

5.28.5.1 *Bit Block Transfer (BitBLT)*

BitBLT is the operation that accelerates transfers of data between regions of display memory, or between system memory and display memory. The BitBLT engine operates on three pixel maps (operands): source, pattern, and destination, with all 256 possible raster operations (ROPs). The graphic engine can support several kinds of BitBLT including HostBLT, Pattern BLT, Color/Font Expanding BLT, Transparent BLT, Color/Font Expansion, and Rectangle Fill, etc.

The source map data may reside in display memory or system memory, both can be color or monochrome. The 8×8 pattern map data may reside in display memory when it is color data, or may come from internal pattern register when it is monochrome data. The destination map data must reside in display memory. The resultant destination data generated by the engine is normally written back to the display memory, or it may be read back by the CPU.

5.28.5.2 *HostBLT (Between System Memory and Display Memory)*

The HostBLT is used to transfer data quickly between system memory and display memory. To transfer data quickly from system memory to display memory, the host may set up a host write BLT to select system memory as source data. Data may be color or monochrome. In monochrome data case for color/font expansion, all ones in the source are expanded to a pixel of foreground color and all

zeros are either expanded to a pixel of background color or transparent. Note that color/font expansion can be performed using X/Y addressing only.

To transfer data quickly from display memory to system memory, the host may set up a host read BLT to let the resultant destination data go to the system memory. Note that only source copy operation (SRCCOPY) is available in a host read BLT.

HostBLT is executed through eight 32-bit MMIO data ports for bit block data transfer. The host must perform 32-bit word-aligned accesses when writing data to, or reading data from the Graphics Engine.

#### 5.28.5.3 *Pattern BLT*

The Pattern BLT is used to accelerate filling of an area of arbitrary size with a repeating pattern in color or mono, and any raster operation may be involved if necessary. The pattern size is 8×8 pixels, chosen for compatibility with Microsoft Windows. Note that all patterns are always aligned to the top-left corner of display memory, but not to the destination area, and the pattern source must be aligned on a boundary, which is equal to the size of the destination.

The 8×8 pattern may be color or monochrome. In the color case, the host first writes the desired pattern to off-screen memory of the frame buffer in a linear fashion. When BitBLT involving pattern is executed, the Graphics Engine will fetch the corresponding row pattern from the off-screen memory and then repetitively copies it to the destination area.

In the monochrome case, the host first writes the desired mono pattern to the internal 8-byte pattern register. The Graphics Engine then repetitively copies it to the destination area, with all ones in the pattern being expanded to a pixel of foreground color and all zeros being either expanded to a pixel of background color or transparent. Note that only X/Y addressing can be used for mono Pattern BLT.

#### 5.28.5.4 *Color/Font Expanding BLT*

To further accelerate the color/font expanding operation, the host may first write the monochrome bitmap to off-screen memory of the frame buffer, then sets up a Color/Font Expanding BLT. In this kind of BitBLT, the Graphics Engine fetches the monochrome source data from off-screen memory of the frame buffer, expands it to pixels, and then writes them to the desired destination area. All ones in the source are expanded to a pixel of foreground color and all zeros are either expanded to a pixel of background color or transparent. Note that X/Y addressing fashion is always used in Color/Font Expanding BLT.

#### 5.28.5.5 *Transparent BLT*

During a BitBLT operation, a certain area of destination may be transparent with the rest being opaque. Monochrome and color transparency are all supported in this chip. In monochrome transparency, either source or pattern data may control the transparency of BitBLT, with all ones being expanded to a pixel of foreground color and all zeros being transparent.

In color transparency, source and destination transparency are both supported, with only one being in effect at a time. Transparent source colors (a color, or color space) do not overwrite the background destination, while only transparent destination colors may be overwritten by the source.

#### 5.28.5.6 *Color/Font Expansion*

Color/Font Expansion is used to expand a monochrome data to a full depth color pixel (including 8/16/32 bit-per-pixel color) with transparency, which greatly accelerates the rendering of text, icons, and other monochrome source objects.

Color/Font Expansion may be implemented using either host write BLT or Color/Font Expanding BLT described above. Note that color/font expansion and mono pattern use the same background and foreground color.

#### 5.28.5.7 *Rectangle Fill*

A fixed-color rectangle may be filled with either foreground color or background color by using Rectangle Fill. No source area is required. All BitBLT operations are available normally.

#### 5.28.5.8 *Raster Operation (ROP)*

The GE supports all Microsoft 256 ternary raster-operation codes. These ternary raster-operation codes define how the BitBLT combines the bits in a source map with the bits in a brush or pattern map, and the bits in the destination map.

Raster operation is always active during BitBLT and must be loaded with appropriate value.

#### 5.28.5.9 *BitBLT Direction*

The BitBLT direction indicates the direction in which the X, Y address is stepped across the rectangle. It also defines the starting corner of the transfer. This is significant if the destination rectangle overlaps the source rectangle. One must be certain that the operation progresses so that the source area is not overwritten prior to being used.

BitBLT direction is controlled as shown in the following diagram.

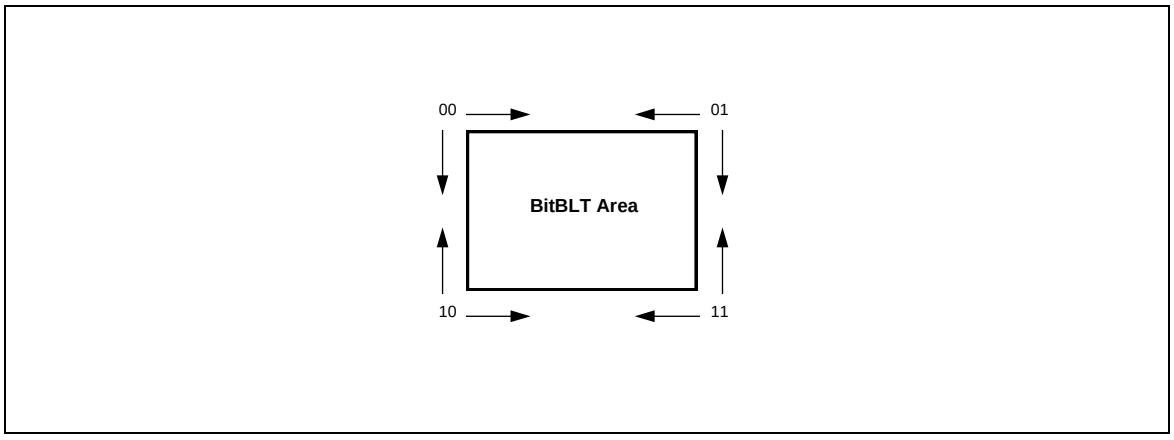



Figure 5.28-2 BitBLT Direction

#### 5.28.5.10 *Linear and X/Y Addressing*

All BitBLT operations may be performed using linear or X/Y addressing. In linear addressing all source start address, destination start address, dimension X, source pitch, and destination pitch are expressed in bytes.

In X/Y addressing, all source start X, destination start X, and dimension X are expressed in pixels. Source pitch and destination pitch are the same in X/Y addressing which is defined with three choices: 160, 320, and above 640.

5.28.5.11 *Auto Destination Update*

When the Auto Update bit is set to 1, the destination start X, Y (address) registers are automatically updated at the end of each BitBLT operation to indicate to the right of the top-right corner of the previous destination area. This is especially usefully at improving Font Expanding BLT where, at most of the time, only source start X, Y (address) registers are needed to update for each BitBLT.

5.28.5.12 *Alpha Blending*

Alpha blending is supported for graphic overlaying. Two 8-bit alpha-depth blending factors are supported for overlaying by properly programming the alpha blending factors Ks and Kd in Miscellaneous Control register. Ks specify the 8-bit alpha value of source stream, and Kd specifies the 8-bit alpha value of destination stream respectively. Note that Ks + Kd would be <= 256.

The blending equation is: [Ps × Ks + Pd × Kd]/256, where Ps means the source stream pixels and Pd means the destination stream pixels.

5.28.5.13 *Bresenham Line Draw*

The Bresenham line drawing algorithm is used to draw a pixel wide solid or textured line from screen coordinates x1, y1 to x2, y2. To draw a solid line, the foreground color is used to specify color of the line. To draw a textured line, a 16-bit line style is used to specify the pattern of line, with all ones in the style being expanded to a pixel of foreground color and all zeros being either expanded to a pixel of background color or transparent.

The Bresenham line drawing algorithm operates with all parameters normalized to the first octant (octant 0). A 3-bit octant code is specified as shown in the following Figure.

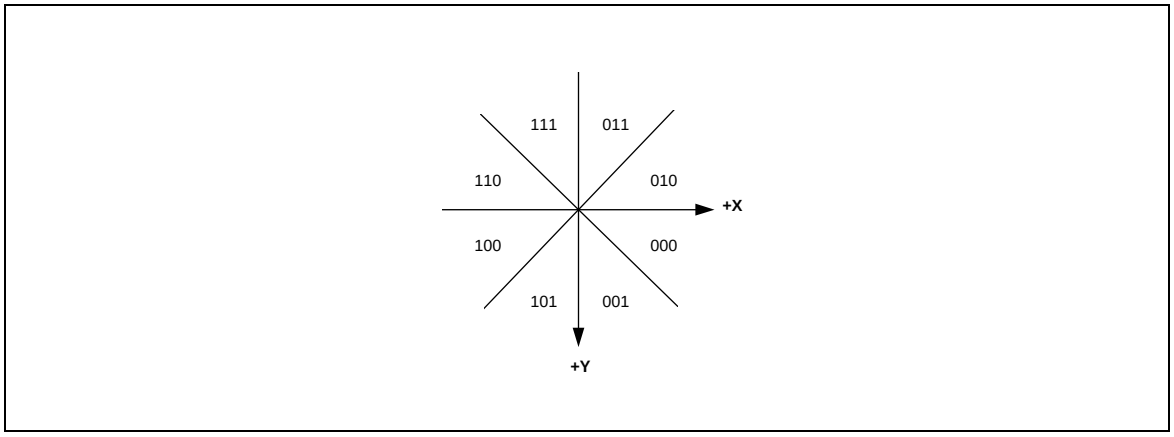



Figure 5.28-3 Bresenham Line Drawing Octant

5.28.5.14 *Bresenham Line Draw Octant Encoding*

In order to avoid drawing the endpoints of poly-lines twice, this chip provides a function that inhibits the drawing of the last pixel of the line, and this function is provided for the Bresenham Line Draw. The Bresenham Line Draw operation may be either a draw operation, or mere a move operation. On completion of the Bresenham Line Draw operation, destination start X, Y normally points at the last pixel of the line by setting Auto Destination Update to be 1. It may also points at the original position of the line when Auto Destination Update is 0. Note that the Bresenham Line Draw operation is available only in X/Y addressing mode.

5.28.5.15 *Move/Draw*

When this bit is 0, the current X, Y pointers will be moved but without any pixels being drawn. When this bit is 1, the pixels should be drawn.

5.28.5.16 *Clipping*

The clipping function supports clipped drawing writes inside or outside of any rectangular region in display memory during Graphics Engine operation. This chip supports both rectangle clipping for BitBLTs and line clipping for Bresenham Line. When enabled, the clipping function simply masks writes within or outside of the clipping window. Note that the clipping function is available only in X/Y addressing mode.

5.28.5.17 *Rotation and Scaling Up/Down*

The main function is to support the rotation or scaling up/down in any rectangular region in display memory during Graphics Engine operation. For the 2D rotation, it can rotate left or right 45, 90 or 180 degrees, and it also supports the flip/flop, mirror or up-side-down pictures. Just as the rectangle clipping for BitBLTs and line clipping for Bresenham Line, when a clip flag is enabled, the clipping function simply masks writes inside or outside the clipping window. Users can trun on the rectangle clipping functions for their different and special view effect. In image scale up/down function, both programmable horizontal and vertical N/M scaling up/down factors are provided for resizing the image. In order to scale up (1+N/M) or scale down (N/M), the value of N must be equal or less than M. Scale up supports for 1.0 ~ 1.996.

Only the source color controlled transparency is supported during the rotation or scaling size processes. Transparent source colors (a color, or color space) can overwrite or not overwrite the background destination.

The graphic engine rotation control operates with all parameters with reference to the first octant (octant 0). A 3-bit octant code is specified as shown in following figure.

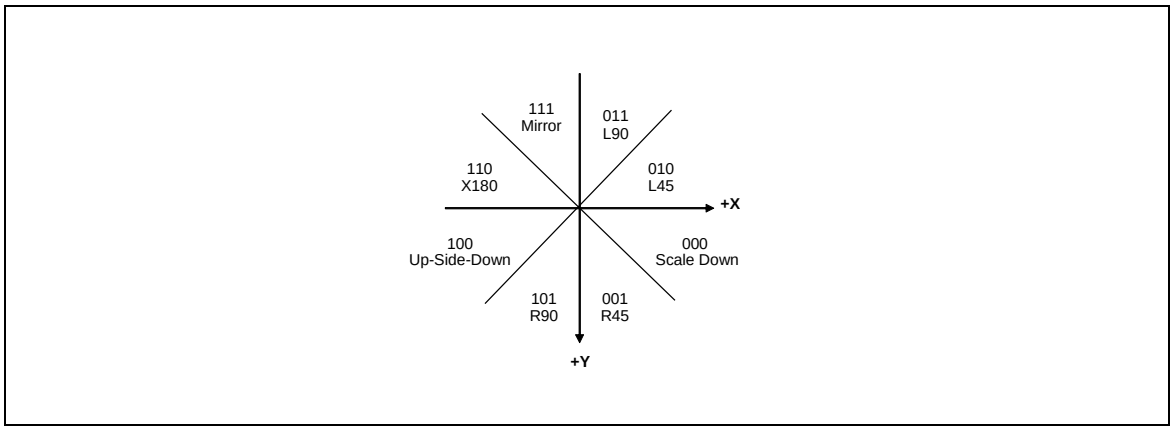



Figure 5.28-4 Rotation Operation Octant

### 5.28.6 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 |
|---|---|---|---|---|
| **GE2D Base Address:**<br>**GE2D_BA = 0xB000_B000** | | | | |
| **GE2D_TRG** | GE2D_BA+0x000 | R/W | Graphic Engine Trigger Control Register | 0x0000_0000 |
| **GE2D_XYSORG** | GE2D_BA+0x004 | R/W | Graphic Engine XY Mode Source Origin Starting Address Register | 0x0000_0000 |
| **GE2D_TCNTVHSF** | GE2D_BA+0x008 | R/W | Graphic Engine Tile Count or Vertical Horizontal Scale Factor Register | 0x0000_0000 |
| **GE2D_XYRRP** | GE2D_BA+0x00C | R/W | Graphic Engine XY Mode Rotate Reference Pixel Coordinate Register | 0x0000_0000 |
| **GE2D_INTSTS** | GE2D_BA+0x010 | R/W | Graphic Engine Interrupt Status Register | 0x0000_0000 |
| **GE2D_PATSA** | GE2D_BA+0x014 | R/W | Graphic Engine Pattern Location Starting Address Register | 0x0000_0000 |
| **GE2D_BETSC** | GE2D_BA+0x018 | R/W | Graphic Engine Bresenham Error Term Stepping Constant Register | 0x0000_0000 |
| **GE2D_BIEPC** | GE2D_BA+0x01C | R/W | Graphic Engine Bresenham Initial Error Term, Pixel Count Register | 0x0000_0000 |
| **GE2D_CTL** | GE2D_BA+0x020 | R/W | Graphic Engine Control Register | 0x0000_0000 |
| **GE2D_BGCOLR** | GE2D_BA+0x024 | R/W | Graphic Engine Background Color Register | 0x0000_0000 |
| **GE2D_FGCOLR** | GE2D_BA+0x028 | R/W | Graphic Engine Foreground Color Register | 0x0000_0000 |
| **GE2D_TRNSCOLR** | GE2D_BA+0x02C | R/W | Graphic Engine Transparency Color Register | 0x0000_0000 |
| **GE2D_TCMSK** | GE2D_BA+0x030 | R/W | Graphic Engine Transparency Color Mask Register | 0x0000_0000 |
| **GE2D_XYDORG** | GE2D_BA+0x034 | R/W | Graphic Engine XY Mode Display Memory Origin Starting Address Register | 0x0000_0000 |
| **GE2D_SDPITCH** | GE2D_BA+0x038 | R/W | Graphic Engine Source Destination Pitch Register | 0x0000_0000 |
| **GE2D_SRCSPA** | GE2D_BA+0x03C | R/W | Graphic Engine Source Start Pixel or Address Register | 0x0000_0000 |
| **GE2D_DSTSPA** | GE2D_BA+0x040 | R/W | Graphic Engine Destination Start Pixel or Address Register | 0x0000_0000 |
| **GE2D_RTGLSZ** | GE2D_BA+0x044 | R/W | Graphic Engine Rectangle Size Register | 0x0000_0000 |
| **GE2D_CLPBTL** | GE2D_BA+0x048 | R/W | Graphic Engine Clipping Boundary Top Left Register | 0x0000_0000 |
| **GE2D_CLPBBR** | GE2D_BA+0x04C | R/W | Graphic Engine Clipping Boundary Bottom Right Register | 0x0000_0000 |
| **GE2D_PTNA** | GE2D_BA+0x050 | R/W | Graphic Engine Pattern Group A Register | 0x0000_0000 |
| **GE2D_PTNB** | GE2D_BA+0x054 | R/W | Graphic Engine Pattern Group B Register | 0x0000_0000 |
| **GE2D_WRPLNMSK** | GE2D_BA+0x058 | R/W | Graphic Engine Write Plane Mask Register | 0x0000_0000 |
| **GE2D_MISCTL** | GE2D_BA+0x05C | R/W | Graphic Engine Miscellaneous Control Register | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **GE2D_HSTBLTDP0** | GE2D_BA+0x060 | R/W | Graphic Engine HostBLT Data Port 0 Register | 0x0000_0000 |
| **GE2D_HSTBLTDP1** | GE2D_BA+0x064 | R/W | Graphic Engine HostBLT Data Port 1 Register | 0x0000_0000 |
| **GE2D_HSTBLTDP2** | GE2D_BA+0x068 | R/W | Graphic Engine HostBLT Data Port 2 Register | 0x0000_0000 |
| **GE2D_HSTBLTDP3** | GE2D_BA+0x06C | R/W | Graphic Engine HostBLT Data Port 3 Register | 0x0000_0000 |
| **GE2D_HSTBLTDP4** | GE2D_BA+0x070 | R/W | Graphic Engine HostBLT Data Port 4 Register | 0x0000_0000 |
| **GE2D_HSTBLTDP5** | GE2D_BA+0x074 | R/W | Graphic Engine HostBLT Data Port 5 Register | 0x0000_0000 |
| **GE2D_HSTBLTDP6** | GE2D_BA+0x078 | R/W | Graphic Engine HostBLT Data Port 6 Register | 0x0000_0000 |
| **GE2D_HSTBLTDP7** | GE2D_BA+0x07C | R/W | Graphic Engine HostBLT Data Port 7 Register | 0x0000_0000 |

### 5.28.7 Register Description

### Graphic Engine Trigger Control Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_TRG** | GE2D_BA+0x000 | R/W | Graphic Engine 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** | | | | | | | **GO** |

| Bits | Description | |
|---|---|---|
| [31:1] | **Reserved** | Reserved. |
| [0] | **GO** | **Trigger Graphics Engine Acceleration**<br>0 = No acceleration or the acceleration is finished.<br>1 = Start GE acceleration, it will automatically be cleared when job completed. |

## Graphic Engine XY Mode Source Origin Starting Address Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_XYSORG** | GE2D_BA+0x004 | R/W | Graphic Engine XY Mode Source Origin Starting Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:28 | **Reserved** | Reserved. |
| [27:0] | **XYSORGSA** | **X/Y Mode Source Origin Starting Address (Byte Unit)**<br>This 28-bit byte address specifies the starting address of an object or a picture in the display memory when addressed by X/Y mode. This value is specified by bytes and should be 16K word (64K bytes) boundary. That is, the bits 15-0 are ignored. |

## Graphic Engine Tile Count or Vertical Horizontal Scale Factor Regsiter

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_TCNTVHSF** | GE2D_BA+0x008 | R/W | Graphic Engine Tile Count or Vertical Horizontal Scale Factor Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **VSF_N** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **VSF_M** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **TILE_X/HSF_N** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **TILE_Y/HSF_M** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **VSF_N** | **Vertical N Scaling Factor**<br>An 8-bit value specifies the numerator part (VSF_N) of the vertical scaling factor in graphic engine. The output image height will be equal to the input image height * VSF_N/VSF_M. The value of VSF_N must be equal or less than VSF_M. |
| [23:16] | **VSF_M** | **Vertical M Scaling Factor**<br>An 8-bit value specifies the denominator part (VSF_M) of the vertical scaling factor in graphic engine. The output image height will be equal to the input image height * VSF_N/VSF_M. The value of VSF_N must be equal or less than VSF_M. |
| [15:8] | **TILE_X/ HSF_N** | **Tile Count for Y-axis/Horizontal N Scaling Factor**<br>If AU (GE2D_CTL[10]) is high, GE2D is operating in TileBLT mode and this 8-bit field indicates the number of tiles in Y-axis.<br>If AU (GE2D_CTL[10]) is low, this 8-bit value specifies the numerator part (HSF_N) of the horizontal scaling factor in graphic engine. The output image width will be equal to the input image width * HSF_N/HSF_M. The value of HSF_N must be equal or less than HSF_M. |
| [7:0] | **TILE_Y/ HSF_M** | **Tile Count for X-axis /Horizontal M Scaling Factor**<br>If AU (GE2D_CTL[10]) is high, GE2D is operating in TileBLT mode and this 8-bit field indicates the number of tiles in X-axis.<br>If AU (GE2D_CTL[10]) is low, this 8-bit value specifies the denominator part (HSF_M) of the horizontal scaling factor in graphic engine. The output image width will be equal to the input image width * HSF_N/HSF_M. The value of HSF_N must be equal or less than HSF_M. |

## Graphic Engine XY Mode Rotate Reference Pixel Coordinate Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_XYRRP** | GE2D_BA+0x00C | R/W | Graphic Engine XY Mode Rotate Reference Pixel Coordinate Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **RRPY** | **Rotate Reference Pixel Y Coordinate**<br>For Rotation in X/Y addressing, this field specifies the reference pixel Y coordinate. |
| [16:11] | **Reserved** | Reserved. |
| [10:0] | **RRPX** | **Rotate Reference Pixel X Coordinate**<br>For Rotation in X/Y addressing, this field specifies the reference pixel X coordinate. |

### Graphic Engine Interrupt Status Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_INTSTS** | GE2D_BA+0x010 | R/W | Graphic Engine 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** | | | | | | | **GE2DIF** |

| Bits | Description | |
|---|---|---|
| [31:1] | **Reserved** | Reserved. |
| [0] | **GE2DIF** | **GE2D Interrupt Flag**<br>The GE2DIF high indicates that GE2D completed the operation. If this bit is high and GE2DIEN (GE2D_CTL[17]) is also high, GE2D would generate a interrupt to CPU.<br>Write 1 to clear this bit to zero.<br>0 = No interrupt.<br>1 = Interrupt occurred. |

## Graphic Engine Pattern Location Starting Address Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_PATSA** | GE2D_BA+0x014 | R/W | Graphic Engine Pattern Location Starting Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27:0] | **PATSA** | **Pattern Location Start Address**<br>This 28-bit address specifies the starting address of the off-screen memory where an 8×8 pixel pattern stored for BitBLT operation.<br>This value must be programmed on an M-byte boundary. M=8*8*BPP/8 bytes, where BPP=8/16/32. |

## Graphic Engine Bresenham Error Term Stepping Constant Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_BETSC** | GE2D_BA+0x018 | R/W | Graphic Engine Bresenham Error Term Stepping Constant Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [29:16] | **DIAGINC** | **Diagonal Error Term Increment**<br>For Bresenham line draw, this register specifies the constant to be added to the Error Term for diagonal stepping (Error > 0). The initial value is (2 ∗ (delta Y - delta X)) after normalization to first octant. |
| [13:0] | **AXIALINC** | **Axial Error Term Increment**<br>For Bresenham line draw, this register specifies the constant to be added to the Error Term for axial stepping (Error < 0). The initial value is (2 ∗ delta Y) after normalization to first octant. |

## Graphic Engine Bresenham Initial Error Term, Pixel Count Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_BIEPC** | GE2D_BA+0x01C | R/W | Graphic Engine Bresenham Initial Error Term, Pixel Count Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [29:16] | **INIT** | **Initial Error Term**<br>For Bresenham line draw, this register specifies the initial Error Term. The initial value is (2 * (delta Y) - delta X) after normalization to first octant. |
| [10:0] | **LINEPC** | **Line Pixel Count Major -1**<br>For Bresenham line draw, this register specifies the pixel count of major axis |

## Graphic Engine Control Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| GE2D_CTL | GE2D_BA+0x020 | R/W | Graphic Engine Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| ROP | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CMD | | APABLDEN | LNSTLEN | BLNMD | LPDM/SCLMD | GE2DIEN | ADDRMD |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TRANSMD | | MTS | CTS | CTP | AU | CLPEN | CLPCTL |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SRCDT | SRCDS | | PATDT | DRAWDIR | | | DSTDATD |

| Bits | Description | |
|---|---|---|
| [31:24] | **ROP** | **ROP (Raster Operation Code)**<br>It supports all Microsoft 256 Raster Operation Codes. Each raster operation code is an 8-bit value that represents the result of the Boolean operation on pre-defined pattern, source, and destination. |
| [23:22] | **CMD** | **Graphics Engine Command**<br>00 = No operation.<br>01 = BitBLT acceleration.<br>10 = Bresenham Line Draw acceleration.<br>11 = Rectangle Border drawing. |
| [21] | **APABLDEN** | **Alpha Blending Enable**<br>0 = Alpha blending disabled.<br>1 = Alpha blending enabled. |
| [20] | **LNSTLEN** | **Line Style Enable**<br>0 = Line style disabled.<br>1 = Line style enabled. |
| [19] | **BLNMD** | **Bresenham Line Move/Draw**<br>0 = Move.<br>1 = Draw. |

| | | |
|---|---|---|
| [18] | **LPDM/SCLMD** | **Last Pixel Draw/Move or Scale Up/Down Rectangular Object**<br>If GE2D operates in Bresenham Line Draw acceleration mode (CMD (GE2D_CTL[23:22]) = 00), this bit defines the operation for last pixel of Bresenham line drawing.<br>0 = Last pixel of Bresenham line will be drawn.<br>1 = Last pixel of Bresenham line will not be drawn.<br><br>If GE2D doesn't operate in Bresenham Line Draw acceleration mode, this bit defines the scale up/down for rectangular object.<br>0 = Scaling down object.<br>1 = Scaling up object. |
| [17] | **GE2DIEN** | **GE2D Interrupt Enable**<br>0 = GE2D interrupt generation disabled.<br>1 = GE2D interrupt generation enabled. |
| [16] | **ADDRMD** | **Graphics Engine Addressing Mode**<br>0 = Linear addressing mode.<br>1 = X/Y addressing mode. |
| [15:14] | **TRANSMD** | **GE Transparency Mode**<br>00 = Disabled.<br>01 = Mono transparency.<br>10 = Color transparency.<br>11 = Reserved. |
| [13] | **MTS** | **Mono Transparency Select**<br>0 = Source.<br>1 = Pattern. |
| [12] | **CTS** | **Color Transparency Select**<br>0 = Source pixels control transparency.<br>1 = Destination pixels control transparency. |
| [11] | **CTP** | **Color Transparency Polarity**<br>0 = Matching pixels are transparent.<br>1 = Matching pixels are opaque. |
| [10] | **AU** | **Auto Update**<br>0 = Auto update disabled.<br>1 = Auto update enabled and destination X, Y register is automatically updated at the end of each BitBLT operation. |
| [9] | **CLPEN** | **Clipping Enable**<br>0 = Clipping disabled.<br>1 = Clipping enabled. |
| [8] | **CLPCTL** | **Clipping Control**<br>0 = Only pixels inside the clipping rectangle are drawn.<br>1 = Only pixels outside the clipping rectangle are drawn. |
| [7] | **SRCDT** | **Source Data Type**<br>0 = Color.<br>1 = Mono.<br>**Note:** Source and pattern data are not allowed to be both mono format. |

| | | |
|---|---|---|
| [6:5] | **SRCDS** | **Source Data Select**<br>00 = Display memory.<br>01 = System memory.<br>10 = GE background color.<br>11 = GE foreground color. |
| [4] | **PATDT** | **Pattern Data Type**<br>0 = Color (from display memory).<br>1 = Mono (from internal pattern registers). |
| [3:1] | **DRAWDIR** | **Drawing Direction**<br>It determines the drawing directions for BitBLT, Bresenham line, and Rotate.<br>For BitBLT operation,<br>000 = Right-down.<br>001 = Right-down.<br>010 = Left-down.<br>011 = Left-down.<br>100 = Right-up.<br>101 = Right-up.<br>110 = Left-up.<br>111 = Left-up.<br><br>For Bresenham line drawing operation,<br>000 = + X, + Y, $\|DX\| \geq \|DY\|$.<br>001 = + X, + Y, $\|DX\| < \|DY\|$.<br>010 = + X, – Y, $\|DX\| \geq \|DY\|$.<br>011 = + X, – Y, $\|DX\| < \|DY\|$.<br>100 = – X, + Y, $\|DX\| \geq \|DY\|$.<br>101 = – X, + Y, $\|DX\| < \|DY\|$.<br>110 = – X, – Y, $\|DX\| \geq \|DY\|$.<br>111 = – X, – Y, $\|DX\| < \|DY\|$.<br><br>For Rotate operation,<br>000 = Scaling Down.<br>001 = Rotate right 45°.<br>010 = Rotate left 45°.<br>011 = Rotate left 90°.<br>100 = Up-Side-Down.<br>101 = Rotate right 90°.<br>110 = Rotate 180°.<br>111 = Mirror or Flop. |
| [0] | **DSTDATD** | **Destination Data Direction**<br>0 = Display memory.<br>1 = System memory. |

## Graphic Engine Background Color Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_BGCOLR** | GE2D_BA+0x024 | R/W | Graphic Engine Background Color Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **BGCOLR** | **Graphics Engine Background Color**<br>These bits specify the background color for Graphics Engine. Only the corresponding number of bits-per-pixel in the display mode is required in the register.<br>In RGB 8:8:8 color mode, BGCOLR[23:16] is used to store the value of red, BGCOLR[15:8] is used to store the value of green and BGCOLR[7:0] is used to store the value of blue. |

## Graphic Engine Foreground Color Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_FGCOLR** | GE2D_BA+0x028 | R/W | Graphic Engine Foreground Color Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **FGCOLR** | **Graphics Engine Foreground Color**<br>These bits specify the foreground color for Graphics Engine. Only the corresponding number of bits-per-pixel in the display mode is required in the register.<br>In RGB 8:8:8 color mode, FGCOLR[23:16] is used to store the value of red, FGCOLR[15:8] is used to store the value of green and FGCOLR[7:0] is used to store the value of blue. |

## Graphic Engine Transparency Color Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_TRNSCOLR** | GE2D_BA+0x02C | R/W | Graphic Engine Transparency Color Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **TRNSCOLR** | **Graphics Engine Transparency Color**<br>These bits specify the transparency color for Graphics Engine. Only the corresponding number of bits-per-pixel in the display mode is required in the register.<br>In RGB 8:8:8 color mode, TRNSCOLR[23:16] is used to store the value of red, TRNSCOLR[15:8] is used to store the value of green and TRNSCOLR[7:0] is used to store the value of blue. |

## Graphic Engine Transparency Color Mask Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_TCMSK** | GE2D_BA+0x030 | R/W | Graphic Engine Transparency Color Mask Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **TCMSK** | **Transparency Color Mask**<br>These bits specify a mask for use in comparison against the transparency color. Only the corresponding number of bits-per-pixel in the display mode is required in the register. |

## Graphic Engine XY Mode Display Memory Origin Starting Address Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_XYDORG** | GE2D_BA+0x034 | R/W | Graphic Engine XY Mode Display Memory Origin Starting Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27:0] | **XYDORGSA** | **X/Y Mode Display Memory Origin Starting Address**<br>This 28-bit byte address specifies the starting address of an object or a picture in the display memory when addressed by X/Y mode. This value is specified by bytes. |

## Graphic Engine Source Destination Pitch Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_SDPITCH** | GE2D_BA+0x038 | R/W | Graphic Engine Source Destination Pitch Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:29] | **Reserved** | Reserved. |
| [28:16] | **DSTPITCH** | **Destination Pitch**<br>This 13-bit register specifies the destination pitch in bytes in linear addressing mode, and in X/Y addressing by pixels. |
| [15:13] | **Reserved** | Reserved. |
| [12:0] | **SRCPITCH** | **Source Pitch**<br>This 13-bit register specifies the source pitch in bytes in linear addressing mode, and in X/Y addressing by pixels. |

## Graphic Engine Source Start Pixel or Address Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_SRCSPA** | GE2D_BA+0x03C | R/W | Graphic Engine Source Start Pixel or Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27:0] | **SRCSPA** | **Source Start Pixel/Address**<br>If ADDRMD (GE2D_CTL[16]) is low, GE2D operates at linear addressing mode. In this mode, this 28-bit address indicates the source starting address.<br>If ADDRMD (GE2D_CTL[16]) is high, GE2D operates at X/Y addressing mode. In this mode, SRCSPA[26:16] indicates the Y coordinate of source start pixel while SRCSPA[10:0] indicates the X coordinate of source start pixel. In this mode, SRCSPA[27] and SRCSPA[15:11] are reserved. |

## Graphic Engine Destination Start Pixel or Address Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_DSTSPA** | GE2D_BA+0x040 | R/W | Graphic Engine Destination Start Pixel or Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27:0] | **DSTSPA** | **Destination Start Pixel/Address**<br>If ADDRMD (GE2D_CTL[16]) is low, GE2D operates at linear addressing mode. In this mode, this 28-bit address indicates the destination starting address.<br>If ADDRMD (GE2D_CTL[16]) is high, GE2D operates at X/Y addressing mode. In this mode, DSTSPA[26:16] indicates the Y coordinate of source start pixel while DSTSPA[10:0] indicates the X coordinate of source start pixel. In this mode, DSTSPA[27] and DSTSPA[15:11] are reserved. |

## Graphic Engine Rectangle Size Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_RTGLSZ** | GE2D_BA+0x044 | R/W | Graphic Engine Rectangle Size Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **HEIGHT** | **Rectangle Height**<br>If ADDRMD (GE2D_CTL[16]) is low, GE2D operates at linear addressing mode and this field indicates the height of rectangle by byte.<br>If ADDRMD (GE2D_CTL[16]) is high, GE2D operates at X/Y addressing mode and this field indicates the height of rectangle by pixel. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **WIDTH** | **Rectangle Width**<br>If ADDRMD (GE2D_CTL[16]) is low, GE2D operates at linear addressing mode and this field indicates the width of rectangle by byte.<br>If ADDRMD (GE2D_CTL[16]) is high, GE2D operates at X/Y addressing mode and this field indicates the width of rectangle by pixel. |

## Graphic Engine Clipping Boundary Top Left Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_CLPBTL** | GE2D_BA+0x048 | R/W | Graphic Engine Clipping Boundary Top Left Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **CLPTOP** | **Clipping Boundary Top**<br>This register specifies the top of the clipping rectangle by pixel. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **CLPLFT** | **Clipping Boundary Left**<br>This register specifies the left limit of the clipping rectangle pixel. |

## Graphic Engine Clipping Boundary Bottom Right Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_CLPBBR** | GE2D_BA+0x04C | R/W | Graphic Engine Clipping Boundary Bottom Right Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **CLPBOT** | **Clipping Boundary Bottom**<br>This register specifies the bottom of the clipping rectangle by pixel. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **CLPRGT** | **Clipping Boundary Right**<br>This register specifies the right limit of the clipping rectangle pixel. |

## Graphic Engine Pattern Group A Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_PTNA** | GE2D_BA+0x050 | R/W | Graphic Engine Pattern Group A Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **PTNLNE3** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **PTNLNE2** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **PTNLNE 1** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **PTNLNE 0** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **PTNLNE3** | **Pattern Line 3**<br>When pattern is monochrome, this field is the line 3 of the 8×8 pattern. |
| [23:16] | **PTNLNE2** | **Pattern Line 2**<br>When pattern is monochrome, this field is the line 2 of the 8×8 pattern. |
| [15:8] | **PTNLNE1** | **Pattern Line 1**<br>When pattern is monochrome, this field is the line 1 of the 8×8 pattern. |
| [7:0] | **PTNLNE0** | **Pattern Line 0**<br>When pattern is monochrome, this field is the line 0 of the 8×8 pattern. |

## Graphic Engine Pattern Group B Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_PTNB** | GE2D_BA+0x054 | R/W | Graphic Engine Pattern Group B Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **PTNLNE7** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **PTNLNE6** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **PTNLNE5** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **PTNLNE4** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **PTNLNE7** | **Pattern Line 7**<br>When pattern is monochrome, this field is the line 7 of the 8×8 pattern. |
| [23:16] | **PTNLNE6** | **Pattern Line 6**<br>When pattern is monochrome, this field is the line 6 of the 8×8 pattern. |
| [15:8] | **PTNLNE5** | **Pattern Line 5**<br>When pattern is monochrome, this field is the line 5 of the 8×8 pattern. |
| [7:0] | **PTNLNE4** | **Pattern Line 4**<br>When pattern is monochrome, this field is the line 4 of the 8×8 pattern. |

## Graphic Engine Write Plane Mask Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_WRPLNMSK** | GE2D_BA+0x058 | R/W | Graphic Engine Write Plane Mask Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **WRPLNMSK** | **Write Plane Mask**<br>These bits specify which bits within each pixel are subject to update by the Graphics Engine. A one enable writing to the corresponding bit plane and a 0 inhibits writing to the corresponding bit plane. Only the corresponding number of bits-per-pixel in the display mode is required in the register. |

## Graphic Engine Miscellaneous Control Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| GE2D_MISCTL | GE2D_BA+0x05C | R/W | Graphic Engine Miscellaneous Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| LNEPTN/ABLDFCT | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| LNEPTN/ABLDFCT | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| FIFOSTS | | | | EMPTY | FULL | BLTSTS | BUSY |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GE2DRST | FIFORST | BPP | | BLTMD | BLTTYP | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **LNEPTN/ABLDFCT** | **Line Style Pattern or Alpha Blending Factor**<br>In Bresenham line drawing mode, this field indicates the 16-bit line style pattern.<br>In BitBLT mode, this field indicates the alpha blending factor. The bit [31:24] indicates the alpha blending factor Ks for source data while bit [23:16] indicates the alpha blending factor Kd for destination date. |
| [15:12] | **FIFOSTS** | **GE2D FIFO Counter Status**<br>This field shows how many FIFO entries occupied by valid data.<br>The value “0000” indicates the FIFO is empty while value “1000” indicates the FIFO is full. |
| [11] | **EMPTY** | **FIFO Empty Status**<br>0 = FIFO is not empty.<br>1 = FIFO is empty. |
| [10] | **FULL** | **FIFO Full Status**<br>0 = FIFO is not full.<br>1 = FIFO is full. |
| [9] | **BLTSTS** | **GE2D BitBLT Operation Complete Status**<br>0 = No BitBLT operation or BitBLT operation didn’t complete yet.<br>1 = BitBLT operation completed. |
| [8] | **BUSY** | **GE2D Operation Status**<br>0 = GE2D is ready for new operation.<br>1 = GE2D is busy. |
| [7] | **GE2DRST** | **GE2D Reset**<br>0 = No operation.<br>1 = Reset GE2D. |

| | | |
|---|---|---|
| [6] | **FIFORST** | **FIFO Reset**<br>0 = No operation.<br>1 = Reset FIFO. |
| [5:4] | **BPP** | **Graphics Engine Pixel Depth**<br>00 = 8-bit.<br>01 = 16-bit.<br>10 = 32-bit.<br>11 = reserved. |
| [3] | **BLTMD** | **BitBLT Mode**<br>0 = BitBLT type is according to GE2D_CTL control bits.<br>1 = BitBLT type follows BLTTYP (GE2D_MISCTL[2:0]) setting. |
| [2:0] | **BLTTYP** | **BitBLT Type Setting**<br>000 = HostBLT (write mode).<br>001 = HostBLT (read mode).<br>010 = SolidFillBLT.<br>011 = PatternBLT.<br>100 = BlockMoveBLT.<br>101 = Color/Font Expansion BLT.<br>110 = Monochrome Transparent BLT.<br>111 = Color Transparent BLT. |

## Graphic Engine HostBLT Data Port Register

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GE2D_HSTBLTDP0** | GE2D_BA+0x060 | R/W | Graphic Engine HostBLT Data Port 0 Register | 0x0000_0000 |
| **GE2D_HSTBLTDP1** | GE2D_BA+0x064 | R/W | Graphic Engine HostBLT Data Port 1 Register | 0x0000_0000 |
| **GE2D_HSTBLTDP2** | GE2D_BA+0x068 | R/W | Graphic Engine HostBLT Data Port 2 Register | 0x0000_0000 |
| **GE2D_HSTBLTDP3** | GE2D_BA+0x06C | R/W | Graphic Engine HostBLT Data Port 3 Register | 0x0000_0000 |
| **GE2D_HSTBLTDP4** | GE2D_BA+0x070 | R/W | Graphic Engine HostBLT Data Port 4 Register | 0x0000_0000 |
| **GE2D_HSTBLTDP5** | GE2D_BA+0x074 | R/W | Graphic Engine HostBLT Data Port 5 Register | 0x0000_0000 |
| **GE2D_HSTBLTDP6** | GE2D_BA+0x078 | R/W | Graphic Engine HostBLT Data Port 6 Register | 0x0000_0000 |
| **GE2D_HSTBLTDP7** | GE2D_BA+0x07C | R/W | Graphic Engine HostBLT Data Port 7 Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **HSTBLTDP** | **HostBLT Data Port**<br>This is a 32-bit MMIO (Memory Mapping I/O) data port to be accessed by CPU for HostBLT operation. |

## 5.29 JPEG Codec (JPEG)

### 5.29.1 Overview

The JPEG Codec supports Baseline Sequential Mode JPEG still image compression and decompression that is fully compliant with ISO/IEC International Standard 10918-1 (T.81). The features and capability of the JPEG codec are listed below.

### 5.29.2 Features

If image data input or output by planar format (PLANAR_ON (JITCR[15])= 1), the features are as following:

- Support to encode interleaved YCbCr 4:2:2/4:2:0 and gray-level (Y only) format image
- Support to decode interleaved YCbCr 4:4:4/4:2:2/4:2:0/4:1:1 and gray-level (Y only) format image
- Support to decode YCbCr 4:2:2 transpose format
- The encoded JPEG bit-stream format is fully compatible with JFIF and EXIF standards
- Support arbitrary width and height image encode and decode (up to 8192x8192)
- Support three programmable quantization-tables
- Support standard default Huffman-table and programmable Huffman-table for decode
- Support arbitrarily 1X~8X image up-scaling function for encode mode
- Support down-scaling function for encode and decode modes(Thumbnail/Primary)
- Support specified window decode mode
- Support quantization-table adjustment for bit-rate and quality control in encode mode
- Support rotate function in encode mode

If image data input or output by packet format (PLANAR_ON (JITCR[15])= 0), the feature are as following:

- Support to encode interleaved YUYV format input image, output bitstream 4:2:2 and 4:2:0 format
- Support to decode interleaved YCbCr 4:4:4/4:2:2/4:2:0 format image
- Support decoded output image YUYV422, RGB555, RGB565, RGB888 format (ORDER= 1).
- The encoded JPEG bit-stream format is fully compatible with JFIF and EXIF standards
- Support arbitrary width and height image encode and decode(Primary Only)
- Support three programmable quantization-tables
- Support standard default Huffman-table and programmable Huffman-table for decode
- Support arbitrarily 1X~8X image up-scaling function for encode mode
- Support down-scaling function 1X~ 16X for Y422 and Y420, 1X~ 8X for Y444 for decode mode
- Support specified window decode mode
- Support quantization-table adjustment for bit-rate and quality control in encode mode

### 5.29.3 Block Diagram

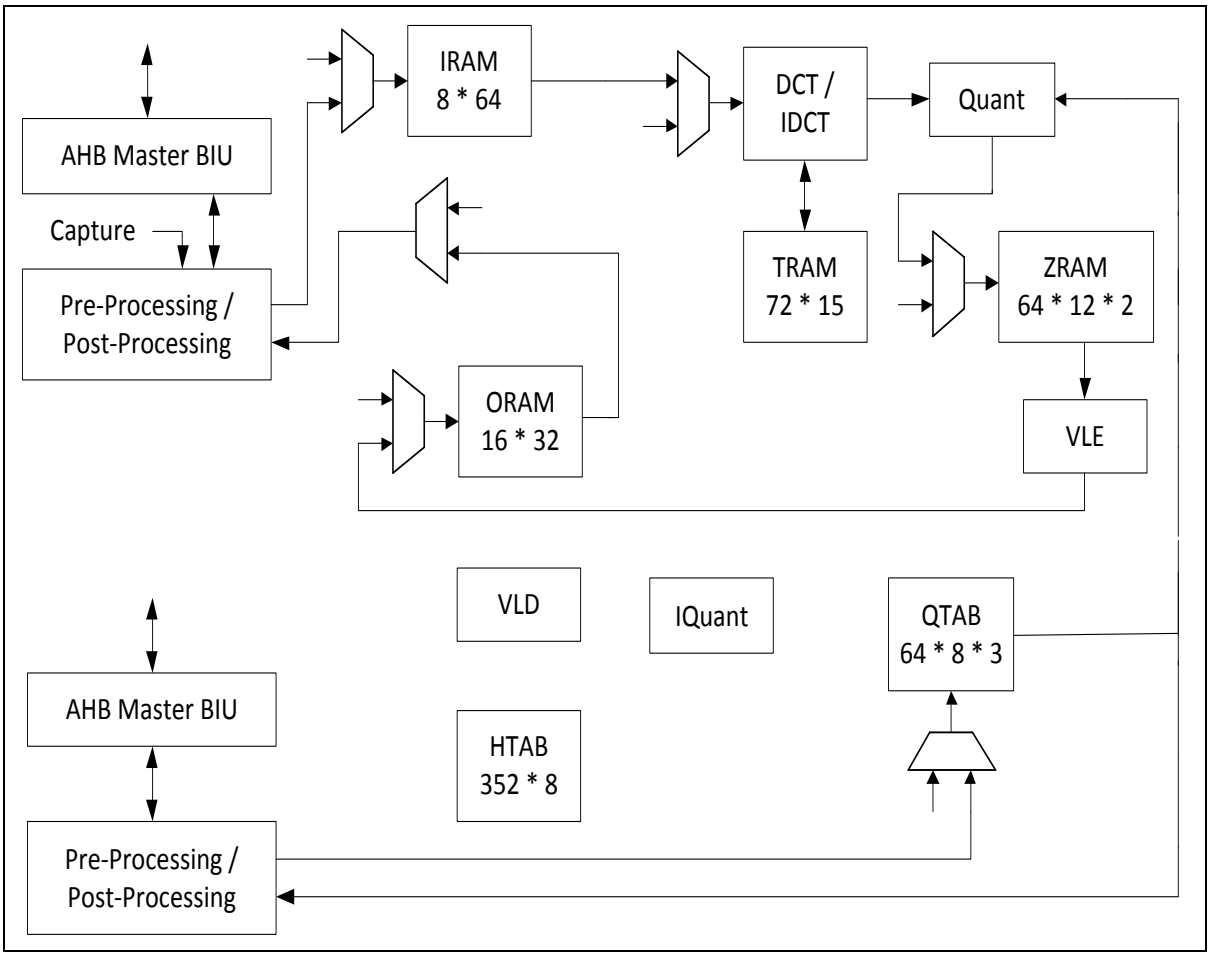



Figure 5.29-1 JPEG Codec block diagram (Encode)

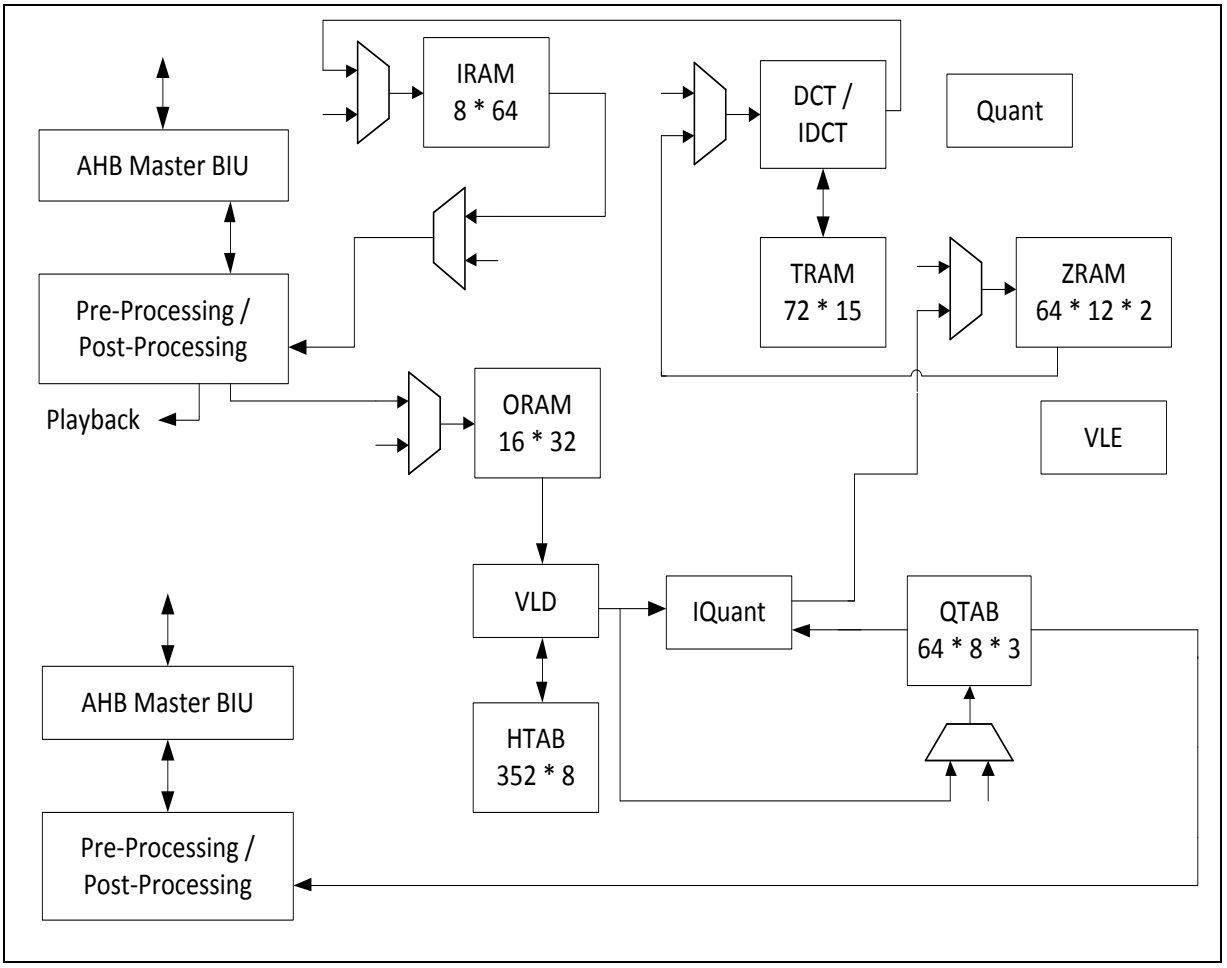



Figure 5.29-2 JPEG Codec block diagram (Decode)

### 5.29.4 Basic Configuration

Before using JPEG engine, it's necessary to enable clock of JPEG engine. Set JPEG (CLK_HCLKEN[29]) high to enable clock for JPEG engine operation.

#### 5.29.4.1 *JPEG Encode Programming*

The programming flow for JPEG encode operation is described as follows:

The programmer needs to program all the required control registers parameters, like image format, width, height, header format, quantization-table, scaling-factor, memory address, etc. before triggering the JPEG engine. Apply engine soft reset by setting reset ENG_RST(JMCR[1]) to 1 and then to 0. For planar format, PLANAR_ON (JITCR[15]) set to 1. For single mode, the programmer can trigger the JPEG engine once by setting the engine enable bit JPG_EN (JITCR[15]) to 1 and then to 0. For continue mode, the JPEG engine will continually operate if the engine enable bit JPG_EN (JITCR[15]) is kept in 1, and will stop operate when JPG_EN (JITCR[15]) is set to 0 and the current picture is encoded completely. When the encode operation for one picture (with both primary and thumbnail images if thumbnail encode is enabled) is complete, the JPEG codec will issue an interrupt to host.

For packet format, PLANAR_ON (JITCR[15]) set to 0.

### 5.29.5 Functional Description

#### 5.29.5.1 *JPEG Encode*

The JPEG Codec supports Baseline Sequential Mode JPEG still image compression and decompression that is fully compliant with ISO/IEC International Standard 10918-1 (T.81). The JPEG codec also supports the thumbnail image compression for EXIF (Exchangeable image file format for digital still camera, JEIDA). The following description describes the feature of the JPEG encoder. For the DCT-based sequential mode, 8x8 blocks are typically input block-by-block from left to right, and block-row by block-row from up to bottom. Each block is transformed by the forward DCT (FDCT) into a set of 64 values referred to as DCT coefficients. Each of these 64 coefficients is quantized by one of 64 corresponding values selected from the quantization-table. After quantization, the DC coefficient is coded by DPCM algorithm and the 63 AC coefficients are converted into one-dimension zig-zag sequence. Then the run-size symbols are passed to a Huffman encoder for entropy coding and the compressed JPEG bit-stream is generated by variable-length-encoder (VLE).

The JPEG encoder supports interleaved YUV422, YUV420 format and non-interleaved Y-component only format if planar format set and packet YUYV format if packet format set. Besides the standard compression, the JPEG encoder integrates a pre-processing unit that can scale-up or scale-down the source image in horizontal and vertical directions.

#### 5.29.5.2 *JPEG Encode Operation*

The JPEG encoder supports single compression mode and continuous compression mode. In single mode, the programmer can use dual-buffer or fix-buffer to store JPEG bit-stream. In continue mode, the programmer can store neighbor JPEG bit-stream in Frame Memory continuously. The JPEG encoder also supports thumbnail image encode.

The JPEG Codec can encode the image with three components (Y, Cb, Cr) or Y component only, where Y component represents the luminance information, and Cb & Cr represent the chrominance information in planar format. It also can encode packet YUYV in packet format. The three components are stored in frame memory separately. The JPEG Codec can compress YUV 420 or YUV 422 format by programming the control register bit EY422 (JMCR[3]). The control registers JYADDR0, JUADDR0, JVADDR0, JYADDR1, JUADDR1, JVADDR1 specify the memory starting address (buffer-0 & buffer-1) of Y, Cb and Cr components and packet data. The control registers JYSTRIDE, JUSTRIDE, JVSTRIDE specify the stride that is the address distance between adjacent lines for each component. The control registers IO_IADDR0, IO_IADDR1 specify the memory starting address (buffer-0 & buffer-1) for the JPEG bit-stream. The following figure depicts the source image starting address and stride.

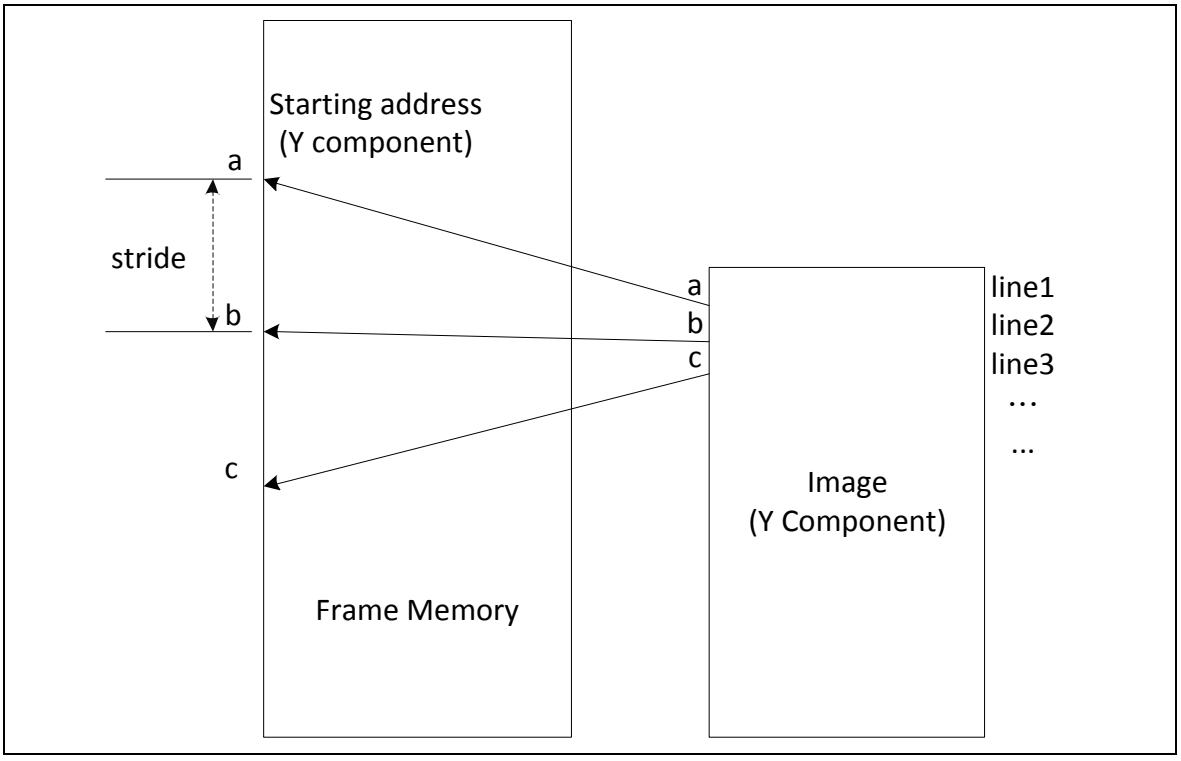



Figure 5.29-3 Image starting address and stride

The JPEG Codec supports thumbnail encode, the programmer can set the register bit THB (JMCR[4]) for thumbnail encode. It will automatically trigger JPEG engine twice, where the first encode operation is for thumbnail image, and the second one is for primary image. The following figure specifies the encode path. When the programmer turns on thumbnail encode, the JPEG engine supports an option for inserting one buffer region into primary JPEG bit-stream. This option can be turned on by setting the register bit A_JUMP (JMCR[4]). The buffer size can be programmed by specifying the registers JRESERVE. In general, the buffer is used to store the thumbnail JPEG bit-stream and some information about this encoded image. The starting address of the JPEG bit-stream for thumbnail image is equal to IO_IADDR plus an offset size specified by register JOFFSET. When the encode operation is completed, the programmer can get the size information of the encoded JPEG bit-stream by reading the registers JPRI_SIZE and JTHB_SIZE.

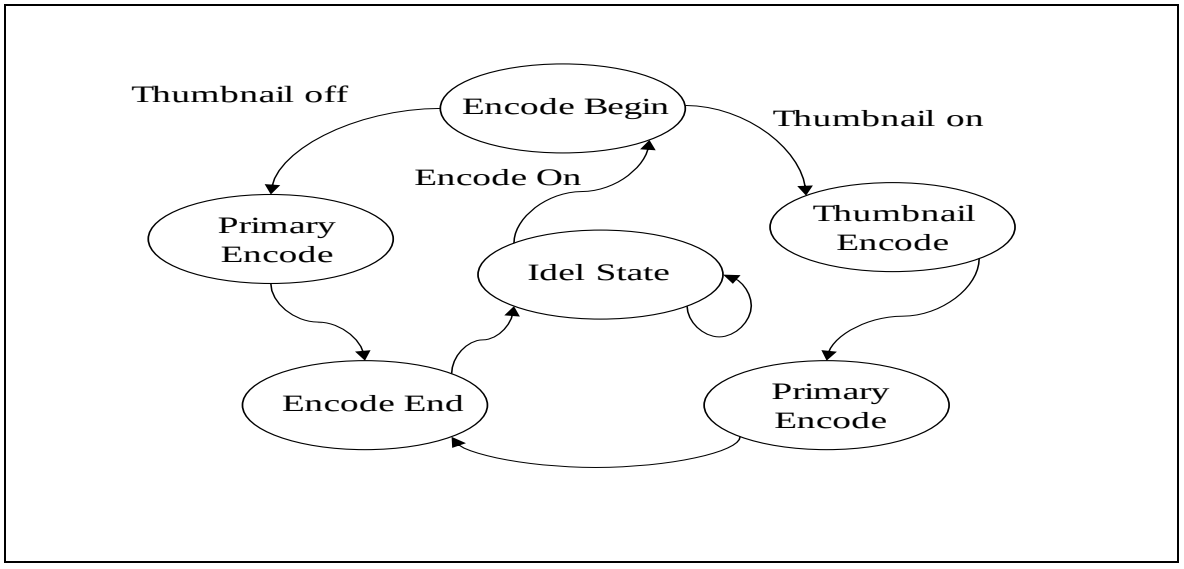



Figure 5.29-4 Primary and thumbnail encode

The JPEG Codec supports the planar format up-scaling and down-scaling function to adjust the encoded image size. The primary image encode supports 1~8X arbitrarily up-scaling in horizontal and vertical direction. But the thumbnail image encode doesn't support the up-scaling function. The JPEG Codec also supports the planar format down-scaling function with the 1/2, 1/3, 1/4, …, 1/64 ratio in vertical direction, and 1/2, 1/4, 1/6, 1/8, …, 1/62, 1/64 ratio in horizontal direction. The programmer can set the registers JPSCALU, JPSCALD, JTSCALD for image scaling up or down. The registers JPRIWH and JTHBWH specify the width and height of the encoded image after scaling. For planar format up-scaling mode, the programmer needs to specify the up-scale ratio by registers JUPRAT and the source image height register JSRCH. The rotation function is only supported for planar format. The source image can be encoded by rotate left or right 90°. It should be mentioned that the rotation function can only be applied only when encode YCbCr 4:2:0 source.

The standard JPEG bit-stream format includes some headers that specify some information, For example, Quantization-table (QTAB) and Huffman-table (HTAB) belong to the part of the header. The JPEG Codec supports some options whether you can insert these headers into the JPEG bit-stream or not. These options are defined in the control register JHEADER. The JPEG codec also supports quantization-table adjustment to control the size of JPEG bit-stream. When the bit-stream size is too large, it can set or adjust the value of quantization-table to reduce the bit-stream size. The adjustment control is defined in registers JPRIQC and JTHBQC. In addition, three programmable quantization-tables are provided. The programmer can specify using two or three tables for encoding by register bit E3QTAB (JITCR[3]). The Quantization-table registers are defined in JQTAB0~JQTAB2.

The JPEG Codec supports two coding modes: single mode and continue mode. For single mode, the programmer can use dual-buffer or fix-buffer to store JPEG bit-stream in frame memory. For continue mode, the programmer can store the neighbor JPEG bit-streams into frame memory continuously or store each JPEG bit-stream into frame memory by creating same buffer size that is specified in register JFSTRIDE. The following figure specifies these two encode modes.

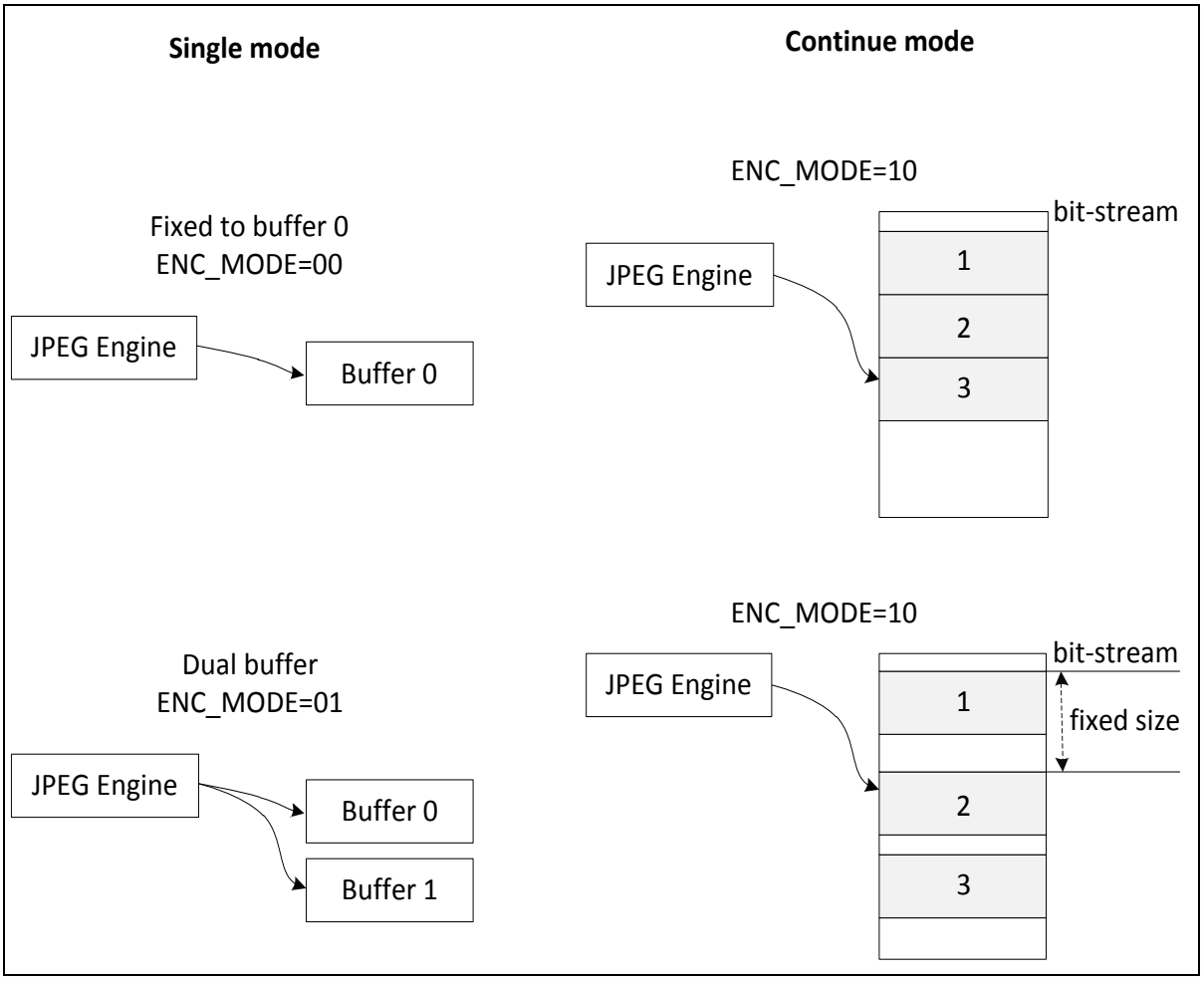



Figure 5.29-5 Single mode and continue mode

5.29.5.3 *JPEG Decode*

The JPEG decoding operation is very similar to the feedback loop of the JPEG encoding. After operation is triggered, the JPEG bit-stream is fetched from frame memory and processed by the variable-length decoder (VLD). The decoded data are parsed, inverse zig-zag scanned (IZZ), inverse quantization (IQ) and inverse DCT (IDCT). An offset value is added to the output data of IDCT to become the reconstruction picture.

The JPEG decoder also integrates a post-processing unit that can apply some post-processing to the decoded image. It can scale-down the image in horizontal and vertical directions.

5.29.5.4 *JPEG Decode Operation*

The JPEG decoder can decode the JPEG bit-stream that is baseline JPEG format. When the programmer want to decode JPEG bit-stream, he just needs to specify the starting address of the JPEG bit-stream in frame memory by registers JIOADDR0 or JIOADDR1 and set JPEG decode mode by register bit ENC_DEC (JMCR[7]), and then trigger the JPEG engine. When the decode operation is

complete, the JPEG engine will issue an interrupt to host. The status register DYUVMODE reports the image color format (4:4:4, 4:2:2, 4:2:0, or 4:1:1) that has been decoded. The register JDECWH reports the original width & height of the decoded JPEG image. The register JPRIWH should be written the real size image width & height after scaling or not for packet format. The programmer can also get the quantization-table of the decoded JPEG bit-stream by reading the registers JQTAB0~JQTAB2. For RGB555 packet output, set ORDER (JITCR[14]) to 1 in the register JITCR.

For 4:4:4 color format images, if the original width is not multiple-of-8, the decoded image will be padded to multiple-of-8. For 4:2:2 and 4:2:0 color format images, if the original width is not multiple-of-16, the decoded image will be padded to multiple-of-16. For 4:1:1 color format images, if the original width is not multiple-of-32, the decoded image will be padded to multiple-of-32.

The JPEG decoder supports the programmable Huffman-table function and can decode the bitstream that is coded by the user-defined Huffman-table. The programmer can choose to turn-on the programmable Huffman-table function or directly use the default Huffman-table to decode the JPEG bitstream by setting the register bit PDHTAB (JITCR[0]).

The JPEG decoder supports 1/2, 1/4 and 1/8 planar format down-scaling in horizontal and vertical direction to adjust the decoded image size. The programmer can specify the register JPSCALD for image planar format down-scaling function. The JPEG decoder also supports 1~16X arbitrarily packet format down-scaling in horizontal and vertical direction. For planar format down-scaling mode, the programmer needs to specify the down-scale ratio by registers JUPRAT.

The JPEG decoder supports specified window decode mode. This function allows user to specify a sub-window region within the whole image to be decoded as shown in the following figure. Only the specified window region image will be decoded and stored to frame memory. This function can be enabled by setting register bit WIN_DEC (JMCR[6]), and the window region can be specified in registers JWINDEC0~JWINDEC2.

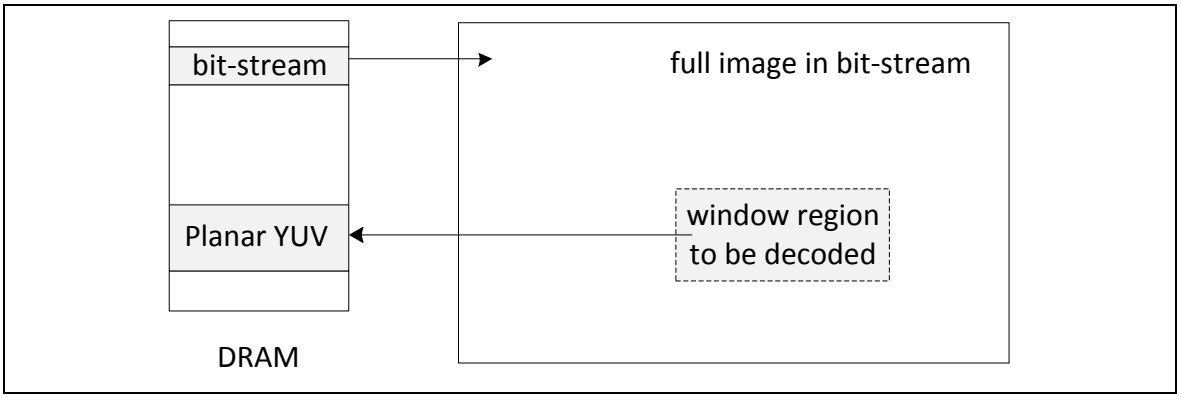



Figure 5.29-6 Specified window decode mode

When the “header decode complete wait” bit DHEND (JITCR[5]) is set, JPEG will enter the pending state after the header information has been decoded, and will resume the decode operation after the interrupt status is cleared.

5.29.5.5 *JPEG Codec Interrupt*

The JPEG codec supports encode complete, decode complete, encode error, decode error and header decode complete interrupts. When encode or decode operation is complete with no error, encode or decode complete interrupt will be issued to host. When decode bitstream and some error occurs, a decode error interrupt will be issued to host. When DHEND (JITCR[5]) is set and the bitstream header has been decoded, a header decode complete interrupt will be issued to host. The interrupt status is reflected on register JINTCR.

### 5.29.6 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 |
|---|---|---|---|---|
| **JPEG Base Address:**<br>**JPEG_BA = 0xB000_A000** | | | | |
| **JMCR** | JPEG_BA + 0x000 | R/W | JPEG Engine Mode Control Register | 0x0000_0000 |
| **JHEADER** | JPEG_BA + 0x004 | R/W | JPEG Encode Header Control Register | 0x0000_0000 |
| **JITCR** | JPEG_BA + 0x008 | R/W | JPEG Image Type Control Register | 0x0000_0000 |
| **RESERVED** | JPEG_BA + 0x00C | R/W | Reserved | 0x0000_0000 |
| **JPRIQC** | JPEG_BA + 0x010 | R/W | JPEG Encode Primary Q-Table Control Register | 0x0000_00F4 |
| **JTHBQC** | JPEG_BA + 0x014 | R/W | JPEG Encode Thumbnail Q-Table Control Register | 0x0000_00F4 |
| **JPRIWH** | JPEG_BA + 0x018 | R/W | JPEG Primary Width/Height Register | 0x0000_0000 |
| **JTHBWH** | JPEG_BA + 0x01C | R/W | JPEG Encode Thumbnail Width/Height Register (For Planar Format Only) | 0x0000_0000 |
| **JPRST** | JPEG_BA + 0x020 | R/W | JPEG Encode Primary Restart Interval Register | 0x0000_0004 |
| **JTRST** | JPEG_BA + 0x024 | R/W | JPEG Encode Thumbnail Restart Interval Register | 0x0000_0004 |
| **JDECWH** | JPEG_BA + 0x028 | R | JPEG Decode Image Width/Height Register | 0x0000_0000 |
| **JINTCR** | JPEG_BA + 0x02C | R/W | JPEG Interrupt Control and Status Register | 0x0020_0000 |
| **JDOWFBS** | JPEG_BA + 0x03C | R/W | JPEG Decoding Output Wait Frame Buffer Size | 0xFFFF_FFFF |
| **JWINDEC0** | JPEG_BA + 0x044 | R/W | JPEG Window Decode Mode Control Register 0 | 0x0000_0000 |
| **JWINDEC1** | JPEG_BA + 0x048 | R/W | JPEG Window Decode Mode Control Register 1 | 0x0000_0000 |
| **JWINDEC2** | JPEG_BA + 0x04C | R/W | JPEG Window Decode Mode Control Register 2 | 0x0000_0000 |
| **JMACR** | JPEG_BA + 0x050 | R/W | JPEG Memory Address Mode Control Register | 0x0000_0000 |
| **JPSCALU** | JPEG_BA + 0x054 | R/W | JPEG Primary Scaling-Up Control Register | 0x0000_0000 |
| **JPSCALD** | JPEG_BA + 0x058 | R/W | JPEG Primary Scaling-Down Control Register | 0x0000_0000 |
| **JTSCALD** | JPEG_BA + 0x05C | R/W | JPEG Thumbnail Scaling-Down Control Register | 0x0000_0000 |
| **JDBCR** | JPEG_BA + 0x060 | R/W | JPEG Dual-Buffer Control Register | 0x0000_0000 |
| **JRESERVE** | JPEG_BA + 0x070 | R/W | JPEG Encode Primary Bit-stream Reserved Size Register | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **JOFFSET** | JPEG_BA + 0x074 | R/W | JPEG Address Offset Between Primary & Thumbnail Register | 0x0000_0000 |
| **JFSTRIDE** | JPEG_BA + 0x078 | R/W | JPEG Encode Bit-stream Frame Stride Register | 0x0000_0000 |
| **JYADDR0** | JPEG_BA + 0x07C | R/W | JPEG Y Component or Packet Format Frame Buffer 0 Starting Address Register | 0x0000_0000 |
| **JUADDR0** | JPEG_BA + 0x080 | R/W | JPEG U Component Frame Buffer-0 Starting Address Register | 0x0000_0000 |
| **JVADDR0** | JPEG_BA + 0x084 | R/W | JPEG V Component Frame Buffer-0 Starting Address Register | 0x0000_0000 |
| **JYADDR1** | JPEG_BA + 0x088 | R/W | JPEG Y Component or Packet Format Frame Buffer-1 Starting Address Register | 0x0000_0000 |
| **JUADDR1** | JPEG_BA + 0x08C | R/W | JPEG U Component Frame Buffer-1 Starting Address Register | 0x0000_0000 |
| **JVADDR1** | JPEG_BA + 0x090 | R/W | JPEG V Component Frame Buffer-1 Starting Address Register | 0x0000_0000 |
| **JYSTRIDE** | JPEG_BA + 0x094 | R/W | JPEG Y Component Frame Buffer Stride Register | 0x0000_0000 |
| **JUSTRIDE** | JPEG_BA + 0x098 | R/W | JPEG U Component Frame Buffer Stride Register | 0x0000_0000 |
| **JVSTRIDE** | JPEG_BA + 0x09C | R/W | JPEG V Component Frame Buffer Stride Register | 0x0000_0000 |
| **JIOADDR0** | JPEG_BA + 0x0A0 | R/W | JPEG Bit-stream Frame Buffer 0 Starting Address Register | 0x0000_0000 |
| **JIOADDR1** | JPEG_BA + 0x0A4 | R/W | JPEG Bit-stream Frame Buffer 1 Starting Address Register | 0x0000_0000 |
| **JPRI_SIZE** | JPEG_BA + 0x0A8 | R | JPEG Encode Primary Image Bit-stream Size Register | 0x0000_0000 |
| **JTHB_SIZE** | JPEG_BA + 0x0AC | R | JPEG Encode Thumbnail Bit-stream Size Register | 0x0000_0000 |
| **JUPRAT** | JPEG_BA + 0x0B0 | R/W | JPEG Planar Format Encode Up-Scale and Packet Format Decode Down-Scale Ratio Register | 0x0000_0000 |
| **JBSFIFO** | JPEG_BA + 0x0B4 | R/W | JPEG Bit-stream FIFO Control Register | 0x0000_0032 |
| **JSRCH** | JPEG_BA + 0x0B8 | R/W | JPEG Encode Source Image Height Register | 0x0000_0FFF |
| **JQTAB0ER0** | JPEG_BA+0x100 | R/W | JPEG Quantization Table 0 Element Register 0 | 0x0000_0000 |
| **JQTAB0ER1** | JPEG_BA+0x104 | R/W | JPEG Quantization Table 0 Element Register 1 | 0x0000_0000 |
| **JQTAB0ER2** | JPEG_BA+0x108 | R/W | JPEG Quantization Table 0 Element Register 2 | 0x0000_0000 |
| **JQTAB0ER3** | JPEG_BA+0x10C | R/W | JPEG Quantization Table 0 Element Register 3 | 0x0000_0000 |
| **JQTAB0ER4** | JPEG_BA+0x110 | R/W | JPEG Quantization Table 0 Element Register 4 | 0x0000_0000 |
| **JQTAB0ER5** | JPEG_BA+0x114 | R/W | JPEG Quantization Table 0 Element Register 5 | 0x0000_0000 |
| **JQTAB0ER6** | JPEG_BA+0x118 | R/W | JPEG Quantization Table 0 Element Register 6 | 0x0000_0000 |
| **JQTAB0ER7** | JPEG_BA+0x11C | R/W | JPEG Quantization Table 0 Element Register 7 | 0x0000_0000 |
| **JQTAB0ER8** | JPEG_BA+0x120 | R/W | JPEG Quantization Table 0 Element Register 8 | 0x0000_0000 |
| **JQTAB0ER9** | JPEG_BA+0x124 | R/W | JPEG Quantization Table 0 Element Register 9 | 0x0000_0000 |
| **JQTAB0ER10** | JPEG_BA+0x128 | R/W | JPEG Quantization Table 0 Element Register 10 | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **JQTAB0ER11** | JPEG_BA+0x12C | R/W | JPEG Quantization Table 0 Element Register 11 | 0x0000_0000 |
| **JQTAB0ER12** | JPEG_BA+0x130 | R/W | JPEG Quantization Table 0 Element Register 12 | 0x0000_0000 |
| **JQTAB0ER13** | JPEG_BA+0x134 | R/W | JPEG Quantization Table 0 Element Register 13 | 0x0000_0000 |
| **JQTAB0ER14** | JPEG_BA+0x138 | R/W | JPEG Quantization Table 0 Element Register 14 | 0x0000_0000 |
| **JQTAB0ER15** | JPEG_BA+0x13C | R/W | JPEG Quantization Table 0 Element Register 15 | 0x0000_0000 |
| **JQTAB1ER0** | JPEG_BA+0x140 | R/W | JPEG Quantization Table 1 Element Register 0 | 0x0000_0000 |
| **JQTAB1ER1** | JPEG_BA+0x144 | R/W | JPEG Quantization Table 1 Element Register 1 | 0x0000_0000 |
| **JQTAB1ER2** | JPEG_BA+0x148 | R/W | JPEG Quantization Table 1 Element Register 2 | 0x0000_0000 |
| **JQTAB1ER3** | JPEG_BA+0x14C | R/W | JPEG Quantization Table 1 Element Register 3 | 0x0000_0000 |
| **JQTAB1ER4** | JPEG_BA+0x150 | R/W | JPEG Quantization Table 1 Element Register 4 | 0x0000_0000 |
| **JQTAB1ER5** | JPEG_BA+0x154 | R/W | JPEG Quantization Table 1 Element Register 5 | 0x0000_0000 |
| **JQTAB1ER6** | JPEG_BA+0x158 | R/W | JPEG Quantization Table 1 Element Register 6 | 0x0000_0000 |
| **JQTAB1ER7** | JPEG_BA+0x15C | R/W | JPEG Quantization Table 1 Element Register 7 | 0x0000_0000 |
| **JQTAB1ER8** | JPEG_BA+0x160 | R/W | JPEG Quantization Table 1 Element Register 8 | 0x0000_0000 |
| **JQTAB1ER9** | JPEG_BA+0x164 | R/W | JPEG Quantization Table 1 Element Register 9 | 0x0000_0000 |
| **JQTAB1ER10** | JPEG_BA+0x168 | R/W | JPEG Quantization Table 1 Element Register 10 | 0x0000_0000 |
| **JQTAB1ER11** | JPEG_BA+0x16C | R/W | JPEG Quantization Table 1 Element Register 11 | 0x0000_0000 |
| **JQTAB1ER12** | JPEG_BA+0x170 | R/W | JPEG Quantization Table 1 Element Register 12 | 0x0000_0000 |
| **JQTAB1ER13** | JPEG_BA+0x174 | R/W | JPEG Quantization Table 1 Element Register 13 | 0x0000_0000 |
| **JQTAB1ER14** | JPEG_BA+0x178 | R/W | JPEG Quantization Table 1 Element Register 14 | 0x0000_0000 |
| **JQTAB1ER15** | JPEG_BA+0x17C | R/W | JPEG Quantization Table 1 Element Register 15 | 0x0000_0000 |
| **JQTAB2ER0** | JPEG_BA+0x180 | R/W | JPEG Quantization Table 2 Element Register 0 | 0x0000_0000 |
| **JQTAB2ER1** | JPEG_BA+0x184 | R/W | JPEG Quantization Table 2 Element Register 1 | 0x0000_0000 |
| **JQTAB2ER2** | JPEG_BA+0x188 | R/W | JPEG Quantization Table 2 Element Register 2 | 0x0000_0000 |
| **JQTAB2ER3** | JPEG_BA+0x18C | R/W | JPEG Quantization Table 2 Element Register 3 | 0x0000_0000 |
| **JQTAB2ER4** | JPEG_BA+0x190 | R/W | JPEG Quantization Table 2 Element Register 4 | 0x0000_0000 |
| **JQTAB2ER5** | JPEG_BA+0x194 | R/W | JPEG Quantization Table 2 Element Register 5 | 0x0000_0000 |
| **JQTAB2ER6** | JPEG_BA+0x198 | R/W | JPEG Quantization Table 2 Element Register 6 | 0x0000_0000 |
| **JQTAB2ER7** | JPEG_BA+0x19C | R/W | JPEG Quantization Table 2 Element Register 7 | 0x0000_0000 |
| **JQTAB2ER8** | JPEG_BA+0x1A0 | R/W | JPEG Quantization Table 2 Element Register 8 | 0x0000_0000 |
| **JQTAB2ER9** | JPEG_BA+0x1A4 | R/W | JPEG Quantization Table 2 Element Register 9 | 0x0000_0000 |
| **JQTAB2ER10** | JPEG_BA+0x1A8 | R/W | JPEG Quantization Table 2 Element Register 10 | 0x0000_0000 |
| **JQTAB2ER11** | JPEG_BA+0x1AC | R/W | JPEG Quantization Table 2 Element Register 11 | 0x0000_0000 |
| **JQTAB2ER12** | JPEG_BA+0x1B0 | R/W | JPEG Quantization Table 2 Element Register 12 | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **JQTAB2ER13** | JPEG_BA+0x1B4 | R/W | JPEG Quantization Table 2 Element Register 13 | 0x0000_0000 |
| **JQTAB2ER14** | JPEG_BA+0x1B8 | R/W | JPEG Quantization Table 2 Element Register 14 | 0x0000_0000 |
| **JQTAB2ER15** | JPEG_BA+0x1BC | R/W | JPEG Quantization Table 2 Element Register 15 | 0x0000_0000 |

### 5.29.7 Register Description

### JPEG Engine Mode Control Register (JMCR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JMCR** | JPEG_BA + 0x000 | R/W | JPEG Engine 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** | | | | | | **RESUMEI** | **RESUMEO** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ENC_DEC** | **WIN_DEC** | **PRI** | **THB** | **EY422** | **QT_BUSY** | **ENG_RST** | **JPG_EN** |

| Bits | Description | |
|---|---|---|
| [31:10] | **Reserved** | Reserved. |
| [9] | **RESUMEI** | **Resume JPEG Operation for Input On-the-fly Mode**<br>Write a “1” to this bit to restart JPEG from a pending state after an input wait interrupt event occurs. This bit will be automatically set to “0” after JPEG receives it.<br>**Note:** This bit can be set when the next source data is filled into frame buffer by host no matter JPEG is pending or not. |
| [8] | **RESUMEO** | **Resume JPEG Operation for Output On-the-fly Mode**<br>Write a “1” to this bit to restart JPEG from a pending state after an output wait interrupt event occurs. This bit will be automatically set to “0” after JPEG receives it.<br>**Note:** This bit can be set when the JPEG generated data is fetched from frame buffer by host no matter JPEG is pending or not. |
| [7] | **ENC_DEC** | **JPEG Encode/Decode Mode**<br>0 = Decode.<br>1 = Encode. |
| [6] | **WIN_DEC** | **JPEG Window Decode Mode**<br>0 = Disable, decode full image.<br>1 = Enable, only the window region defined by registers **JWINDEC** of the whole image will be decoded.<br>**Note:** This bit is only valid when JPEG engine is operates in decode mode. If window decode mode is enabled, the up-scaling and down-scaling functions are not allowed. |
| [5] | **PRI** | **Encode Primary Image**<br>0 = Disable encoding primary image.<br>1 = Enable encoding primary image. |
| [4] | **THB** | **Encode Thumbnail Image**<br>0 = Disable encoding thumbnail image.<br>1 = Enable encoding thumbnail image. |

| | | |
|---|---|---|
| [3] | **EY422** | **Encode Image Format**<br>0 = YUV 4:2:0.<br>1 = YUV 4:2:2. |
| [2] | **QT_BUSY** | **Quantization-table Busy Status (Read-only)**<br>0 = Quantization-Table is ready for host access.<br>1 = Quantization-Table is busy and can't be accessed. |
| [1] | **ENG_RST** | **Soft Reset JPEG Engine (Except JPEG Control Registers)**<br>0 = Disable. Normal operation<br>1 = Reset JPEG engine, but the value of control registers keep no change. |
| [0] | **JPG_EN** | **JPEG Engine Operation Control**<br>0 = Disable.<br>1 = Enable. |

## JPEG Encode Header Control Register (JHEADER)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| JHEADER | JPEG_BA + 0x004 | R/W | JPEG Encode Header 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 |
| P_JFIF | P_HTAB | P_QTAB | P_DRI | T_JFIF | T_HTAB | T_QTAB | T_DRI |

| Bits | Description | |
|---|---|---|
| [31:8] | Reserved | Reserved. |
| [7] | P_JFIF | Primary JPEG Bit-stream Include JFIF Header<br>0 = Not Include.<br>1 = Include. |
| [6] | P_HTAB | Primary JPEG Bit-stream Include Huffman-table<br>0 = Not Include.<br>1 = Include. |
| [5] | P_QTAB | Primary JPEG Bit-stream Include Quantization-table<br>0 = Not Include.<br>1 = Include. |
| [4] | P_DRI | Primary JPEG Bit-stream Include Restart Interval<br>0 = Not Include.<br>1 = Include. |
| [3] | T_JFIF | Thumbnail JPEG Bit-stream Include JFIF Header<br>0 = Not Include.<br>1 = Include. |
| [2] | T_HTAB | Thumbnail JPEG Bit-stream Include Huffman-table<br>0 = Not Include.<br>1 = Include. |
| [1] | T_QTAB | Thumbnail JPEG Bit-stream Include Quantization-table<br>0 = Not Include.<br>1 = Include. |

| | | |
|---|---|---|
| [0] | **T_DRI** | **Thumbnail JPEG Bit-stream Include Restart Interval**<br>0 = Not Include.<br>1 = Include. |

## JPEG Image Type Control Register (JITCR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| JITCR | JPEG_BA + 0x008 | R/W | JPEG Image Type Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | Dec_Output_Wait_Go | Reserved | ARGB8888 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PLANAR_ON | ORDER | RGB_555_565 | ROTATE | | DYUV_MODE | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EXIF | EY_ONLY | DHEND | DTHB | E3QTAB | D3QTAB | ERR_DIS | PDHTAB |

| Bits | Description | |
|---|---|---|
| [31:19] | **Reserved** | Reserved. |
| [18] | **Dec_Output_Wait_Go** | **Decode Output Wait and Go Mode Enable**<br>While the frame size and start address are setting, writing “1” will trigger the decoding output wait feature The decoding processing will be halted and an interrupt will issued while the current frame buffer is full The decoding process will be resumed and new frame buffer size and start address will be loaded while another “1” is written<br>Check 0x3C about the packet Frame Buffer Size. |
| [17] | **Reserved** | Reserved. |
| [16] | **ARGB8888** | **ARGB888 Format Enable**<br>This bit defines if the RGB data format is ARGB888 or not. This bit only takes effect while ORDER (JITCR[14]) is high.<br>0 = ARGB888 data format Disabled.<br>1 = ARGB888 data format Enabled. |
| [15] | **PLANAR_ON** | **Planar Format Enable**<br>0 = Packet format.<br>1 = Planar format. |
| [14] | **ORDER** | **Decode Packet Format Output Data Order (Low Byte First)**<br>0 = Y0, U0, Y1, V0.<br>1 = RGB. |
| [13] | **RGB_555_565** | **RGB Data Format Selection**<br>This bit defines the RGB data format is RGB555 or RGB565. This bit only takes effect while ORDER (JITCR[14]) is high and ARGB888 (JITCR[16]) is low.<br>0 = RGB data format is RGB555.<br>1 = RGB data format is RGB565. |

| | | |
|---|---|---|
| [12:11] | **ROTATE** | **Encode Image Rotate(for Planar Format Only)**<br>00 = Normal encode.<br>01 = Reserved.<br>10 = The encoded image is rotated left from source image.<br>11 = The encoded image is rotated right from source image.<br>**Note:** The **JPRIWH** and **JTHBWH** specify the image width and height after rotation. However the **JPSCALD**, **JTSCALD** and **JUPRAT** specify the scale ratio before rotation. |
| [10:8] | **DYUV_MODE** | **Decoded Image YUV Color Format (Read-only)**<br>000 = The format of decoded image is YUV 4:2:0.<br>001 = The format of decoded image is YUV 4:2:2.<br>010 = The format of decoded image is YUV 4:4:4.<br>011 = The format of decoded image is YUV 4:1:1.<br>100 = The format of decoded image is gray-level (Y only).<br>101 = The format of decoded image is YUV 4:2:2 transpose. |
| [7] | **EXIF** | **Encode Quantization-table & Huffman-table Header Format Selection**<br>0 = General format. The header QTAB/HTAB for each component is defined in separated DQT/DHT marker.<br>1 = EXIF compatible format. Three QTAB are defined for Y, Cb, and Cr, header QTAB/HTAB is defined in only one DQT/DHT marker. |
| [6] | **EY_ONLY** | **Encode Gray-level (Y-component Only) Image**<br>0 = Encode normal Y/Cb/Cr color image.<br>1 = Encode gray-level image. |
| [5] | **DHEND** | **Header Decode Complete Stop Enable**<br>0 = JPEG engine will not stop after the header information of JPEG bitstream has been decoded.<br>1 = JPEG engine will enter the pending state after the header information of JPEG bitstream has been decoded. Clear header-decode-end interrupt status can resume JPEG decoding operation |
| [4] | **DTHB** | **Decode Thumbnail Image Only**<br>0 = Decode primary image.<br>1 = Decode thumbnail image only.<br>**Note:** If the JPEG bit-stream contains thumbnail, the programmer can select to decode the primary image or decode thumbnail image only by this bit. |
| [3] | **E3QTAB** | **Numbers of Quantization-table Are Used for Encode**<br>0 = Two QTAB, one for Y and another for Cb & Cr.<br>1 = Three QTAB, one for Y, one for Cb, and one for Cr.<br>**Note:** If **EXIF** is enable, three QTAB are always used for Y, Cb, and Cr. |
| [2] | **D3QTAB** | **Numbers of Quantization-table Are Used for Decode (Read-only)**<br>0 = Two QTAB.<br>1 = Three QTAB. |
| [1] | **ERR_DIS** | **Decode Error Engine Abort**<br>0 = JPEG decode operation will abort if decode error occurs.<br>1 = JPEG decode operation will continue if decode error occurs. |

| | | |
|---|---|---|
| [0] | **PDHTAB** | **Programmable Huffman-table Function for Decode**<br>0 = Disable. Use default huffman-table for JPEG decode<br>1 = Enable. Allow user-defined Huffman-table in JPEG bit-stream |

## JPEG Encode Primary Quantization-Table Control Register (JPRIQC)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JPRIQC** | JPEG_BA + 0x010 | R/W | JPEG Encode Primary Q-Table Control Register | 0x0000_00F4 |

| 31 | 30 | 29 | 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** |
| **P_QADJUST** | | | | **P_QVS** | | | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:4] | **P_QADJUST** | **Primary Quantization-table Adjustment**<br>If the sum of the position (x, y) of quantization-table is greater than **P_QADJUST**, the quantization value will be set to 127. Otherwise the value will keep as the original.<br>8x8 DCT block: x = 0~7, y = 0~7.<br>if ((x+y) > P_QADJUST) => Q' = 127.<br>else => Q' = Q. |
| [3:0] | **P_QVS** | **Primary Quantization-table Scaling Control**<br>Q' = (P_QVS[3]*2*Q)+(P_QVS[2]*Q)+(P_QVS[1]*Q/2)+(P_QVS[0]*Q/4). |

## JPEG Encode Thumbnail Quantization-Table Control Register (JTHBQC)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JTHBQC** | JPEG_BA + 0x014 | R/W | JPEG Encode Thumbnail Q-Table Control Register | 0x0000_00F4 |

| 31 | 30 | 29 | 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** |
| **T_QADJUST** | | | | **T_QVS** | | | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:4] | **T_QADJUST** | **Thumbnail Quantization-table Adjustment**<br>If the sum of the position (x, y) of quantization-table is greater than **T_QADJUST**, the quantization value will be set to 127. Otherwise the value will keep as the original.<br>8x8 DCT block: x = 0~7, y = 0~7.<br>if ((x+y) > T_QADJUST) => Q' = 127.<br>else => Q' = Q. |
| [3:0] | **T_QVS** | **Thumbnail Quantization-table Scaling Control**<br>Q' = (T_QVS[3]*2*Q)+(T_QVS[2]*Q)+(T_QVS[1]*Q/2)+(T_QVS[0]*Q/4). |

## JPEG Primary Image Width/Height Register (JPRIWH)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| JPRIWH | JPEG_BA + 0x018 | R/W | JPEG Primary Width/Height Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:29] | **Reserved** | Reserved. |
| [28:16] | **P_HEIGHT** | **Primary Encode and Packet Format Decode Image Height**<br>A 13-bit value specifies the height of encoded and decoded JPEG image. The value is equal to the size after scaling-up or scaling-down.<br>**Note:** The JPEG engine supports horizontal and vertical arbitrarily up-scaling 1X~8X in planar format encode mode. When the vertical up-scaling mode (**Y2**) is enabled, the height of source image needs to be specified by **JCRCH**.<br>When the vertical down-scaling mode is enable in packet format decode, the size is ceil((YSF/1024)*(height of image)) |
| [15:13] | **Reserved** | Reserved. |
| [12:0] | **P_WIDTH** | **Primary Encode and Packet Format Decode Image Width**<br>A 13-bit value specifies the width of encoded and decoded JPEG image. The value is equal to the size after scaling-up or scaling-down.<br>When the down-scaling mode is enable in packet format decode, the size is<br>ceil((XSF/1024)*(width of image)/16) * (16 + Block1),<br>while Block1 = 1 when mod((XSF/1024)*(width of image)/16) is 0. |

## JPEG Encode Thumbnail Image Width/Height Register (JTHBWH) (For Planar Format Only)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JTHBWH** | JPEG_BA + 0x01C | R/W | JPEG Encode Thumbnail Width/Height Register (For Planar Format Only) | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:29] | **Reserved** | Reserved. |
| [28:16] | **T_HEIGHT** | **Thumbnail Encode Image Height (for Planar Format Only)**<br>A 13-bit value specifies the height of encoded JPEG thumbnail image. The value is equal to the size after scaling-down. |
| [15:13] | **Reserved** | Reserved. |
| [12:0] | **T_WIDTH** | **Thumbnail Encode Image Width (for Planar Format Only)**<br>A 13-bit value specifies the width of encoded JPEG thumbnail image. The value is equal to the size after scaling-down. |

## JPEG Encode Primary Restart Interval Register (JPRST)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JPRST** | JPEG_BA + 0x020 | R/W | JPEG Encode Primary Restart Interval Register | 0x0000_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** |
| **P_RST** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **P_RST** | **Primary Encode Restart Interval Value**<br>An 8-bit value specifies the restart interval for encoding primary JPEG image. |

## JPEG Encode Thumbnail Restart Interval Register (JTRST)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JTRST** | JPEG_BA + 0x024 | R/W | JPEG Encode Thumbnail Restart Interval Register | 0x0000_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** |
| **T_RST** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **T_RST** | **Thumbnail Encode Restart Interval Value**<br>An 8-bit value specifies the restart interval for encoding thumbnail JPEG image. |

## JPEG Decode Image Width/Height Register (JDECWH)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JDECWH** | JPEG_BA + 0x028 | R | JPEG Decode Image Width/Height Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **DEC_HEIGHT** | **Decode Image Height**<br>A 16-bit value reports the height of decoded JPEG image. |
| [15:0] | **DEC_WIDTH** | **Decode Image Width**<br>A 16-bit value reports the width of decoded JPEG image.<br>**Note:** 1. The value of width and height are extracted from bitstream header and are not the width and height after up-scaling or down-scaling.<br>2. The real decoded image width (stored to DRAM) will be aligned to multiple of 16 for 4:2:2/4:2:0, and multiple of 8 for 4:4:4/y-only. The real decoded image height (stored to DRAM) will be aligned to multiple of 16 for 4:2:0, and multiple of 8 for 4:4:4/4:2:2/y-only.<br>3. If up-scaling or down-scaling function is enabled, the real image width/height (stored to DRAM) is equal to aligned width/height (described above) * scaling-factor. |

## JPEG Interrupt Control and Status Register (JINTCR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| JINTCR | JPEG_BA + 0x02C | R/W | JPEG Interrupt Control and Status Register | 0x0020_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | DOW_INTE | Reserved | | | DOW_INTS |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| JPG_WAITI | JPG_WAITO | Reserved | | | | | BAbort |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | DHE_INTE | IPW_INTE | OPW_INTE | ENC_INTE | DEC_INTE | DER_INTE | EER_INTE |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | DHE_INTS | IPW_INTS | OPW_INTS | ENC_INTS | DEC_INTS | DER_INTS | EER_INTS |

| Bits | Description | |
|---|---|---|
| [31:29] | **Reserved** | Reserved. |
| [28] | **DOW_INTE** | **Decoding Output Wait Interrupt Enable**<br>0 = DOW_INTS (JINTCR[24]) high to trigger JPEG interrupt Disabled.<br>1 = DOW_INTS (JINTCR[24]) high to trigger JPEG interrupt Enabled. |
| [27:25] | **Reserved** | Reserved. |
| [24] | **DOW_INTS** | **Status of Decoding Output Wait**<br>0 = No Interrupt.<br>1 = Decoding packet buffer is full.<br>Writing “1” will clear the status |
| [23] | **JPG_WAITI** | **JPEG Input Wait Status (Read-only)**<br>0 = JPEG is operating or idle.<br>1 = JPEG is pending and is waiting for a input resume. |
| [22] | **JPG_WAITO** | **JPEG Output Wait Status (Read-only)**<br>0 = JPEG is operating or idle.<br>1 = JPEG is pending and is waiting for a output resume. |
| [21:17] | **Reserved** | Reserved. |
| [16] | **BAbort** | **JPEG Memory Access Error Status (Read-only)**<br>0 = Normal operation.<br>1 = Wrong frame memory space is accessed. |
| [15] | **Reserved** | Reserved. |
| [14] | **DHE_INTE** | **JPEG Header Decode End Wait Interrupt Enable**<br>0 = Disable.<br>1 = Enable. |

| | | |
|---|---|---|
| [13] | **IPW_INTE** | **Input Wait Interrupt Enable**<br>0 = Disable.<br>1 = Enable. |
| [12] | **OPW_INTE** | **Output Wait Interrupt Enable**<br>0 = Disable.<br>1 = Enable. |
| [11] | **ENC_INTE** | **Encode Complete Interrupt Enable**<br>0 = Disable.<br>1 = Enable. |
| [10] | **DEC_INTE** | **Decode Complete Interrupt Enable**<br>0 = Disable.<br>1 = Enable. |
| [9] | **DER_INTE** | **Decode Error Interrupt Enable**<br>0 = Disable.<br>1 = Enable. |
| [8] | **EER_INTE** | **Encode (On-the-fly) Error Interrupt Enable**<br>0 = Disable.<br>1 = Enable. |
| [7] | **Reserved** | Reserved. |
| [6] | **DHE_INTS** | **JPEG Header Decode End Wait Interrupt Status**<br>0 = No Interrupt.<br>1 = Interrupt Generated.<br>**Note:** When write value “1” to this bit, the interrupt will be clear and JPEG will resume operating from a pending state. |
| [5] | **IPW_INTS** | **Input Wait Interrupt Status**<br>0 = No Interrupt.<br>1 = Interrupt Generated.<br>**Note:** When write value “1” to this bit, the interrupt will be clear. |
| [4] | **OPW_INTS** | **Output Wait Interrupt Status**<br>0 = No Interrupt.<br>1 = Interrupt Generated.<br>**Note:** When write value “1” to this bit, the interrupt will be clear. |
| [3] | **ENC_INTS** | **Encode Complete Interrupt Status**<br>0 = No Interrupt.<br>1 = Interrupt Generated.<br>**Note:** When write value “1” to this bit, the interrupt will be clear. |
| [2] | **DEC_INTS** | **Decode Complete Interrupt Status**<br>0 = No Interrupt.<br>1 = Interrupt Generated.<br>**Note:** When write value “1” to this bit, the interrupt will be clear. |

| | | |
|---|---|---|
| [1] | **DER_INTS** | **Decode Error Interrupt Status**<br>0 = No Interrupt.<br>1 = Interrupt Generated.<br>**Note:** When write value “1” to this bit, the interrupt will be clear. |
| [0] | **EER_INTS** | **Encode (On-the-fly) Error Interrupt Status**<br>0 = No Interrupt.<br>1 = Interrupt Generated.<br>**Note:** When write value “1” to this bit, the interrupt will be clear. |

## JPEG Decoding Output Wait Frame Buffer Size (JDOWFBS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JDOWFBS** | JPEG_BA + 0x03C | R/W | JPEG Decoding Output Wait Frame Buffer Size | 0xFFFF_FFFF |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **JDOWFBS** | **JPEG Decoding Output Wait Frame Buffer Size**<br>The JPEG output decoding process will be paused while this buffer is full and decoding process will be resumed while new buffer size/address is set and “1” is written to bit[18] of register 0x08.<br>**Note:** The buffer size must be multiples of MCU-line. |

## JPEG Window Decode Mode Control Register 0 (JWINDEC0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JWINDEC0** | JPEG_BA + 0x044 | R/W | JPEG Window Decode Mode Control Register 0 | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:26] | **Reserved** | Reserved. |
| [25:16] | **MCU_S_Y** | **MCU(Minimum Coded Unit) Start Position Y for Window Decode Mode**<br>A 10-bit value specifies the MCU start position y of the window region within the whole image to be decoded when the window decode mode (**WIN_DEC**) is enabled. The position y is started from 0.<br>**Note:** The MCU size is fixed to 16x16. |
| [15:10] | **Reserved** | Reserved. |
| [9:0] | **MCU_S_X** | **MCU Start Position X for Window Decode Mode**<br>A 10-bit value specifies the MCU start position x of the window region within the whole image to be decoded when the window decode mode (**WIN_DEC**) is enabled. The position x is started from 0. |

## JPEG Window Decode Mode Control Register 1 (JWINDEC1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JWINDEC1** | JPEG_BA + 0x048 | R/W | JPEG Window Decode Mode Control Register 1 | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:26] | **Reserved** | Reserved. |
| [25:16] | **MCU_E_Y** | **MCU End Position Y for Window Decode Mode**<br>A 10-bit value specifies the MCU end position y of the window region within the whole image to be decoded when the window decode mode (**WIN_DEC**) is enabled. |
| [15:10] | **Reserved** | Reserved. |
| [9:0] | **MCU_E_X** | **MCU End Position X for Window Decode Mode**<br>A 10-bit value specifies the MCU end position x of the window region within the whole image to be decoded when the window decode mode (**WIN_DEC**) is enabled. |

## JPEG Window Decode Mode Control Register 2 (JWINDEC2)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JWINDEC2** | JPEG_BA + 0x04C | R/W | JPEG Window Decode Mode Control 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** |
| **Reserved** | | | **WD_WIDTH** | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **WD_WIDTH** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:13] | **Reserved** | Reserved. |
| [12:0] | **WD_WIDTH** | **Image Width (Y-stride) for Window Decode Mode**<br>A 13-bit value specifies the memory line space (Y-Stride) for the window image within the whole image to be decoded when the window decode mode (**WIN_DEC**) is enabled: byte address of word aligned. |

## JPEG Memory Address Mode Control Register (JMACR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| JMACR | JPEG_BA + 0x050 | R/W | JPEG Memory Address Mode Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | FLY_SEL | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FLY_TYPE | | Reserved | | | | BSF_SEL | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| BSF_SEL | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FLY_ON | Reserved | | | IP_SF_ON | OP_SF_ON | ENC_MODE | |

| Bits | Description | |
|---|---|---|
| [31:30] | **Reserved** | Reserved. |
| [29:24] | **FLY_SEL** | **Hardware Memory On-the-fly Access Image Buffer-size Selection for Encode**<br>The numbers of lines used as frame-buffer = (FLY_SEL+1) * 16.<br>**Note:**<br>This setting is only valid when **FLY_TYPE =** 2'b01. Otherwise the buffer-size is always fixed.<br>The minimum buffer-size is 32-line for 4:2:0 format image and is 16-line for 4:2:2 format image. If down-scaling in vertical direction is applied, the buffer-size must larger than (16 x down-scaling-factor) in 4:2:2 and (32 x down-scaling-factor) in 4:2:0 image. Ex: For scaling-down 1/3 in vertical direction, 4:2:2 image, the minimum buffer-size must be 48-line (= 16 x 3).<br>If down-scaling in vertical direction is applied, the source image height (**S_HEIGHT**) needs to be specified. |
| [23:22] | **FLY_TYPE** | 01 = Dual buffer on-the fly.<br>10 = Single buffer on-the-fly. |
| [21:18] | **Reserved** | Reserved. |
| [17:8] | **BSF_SEL** | **Memory On-the-fly Access Bitstream Buffer-size Selection**<br>A 10-bit value specifies the memory space for JPEG bitstream. The unit of this register is 2K Bytes.<br>**Note:** The buffer region is used in a dual-buffer manner. For example, if the buffer-size is 2KB (BSF_SEL = 1), host needs to fill/remove 1KB bitstream data into/from one of the half buffer region before triggering or resuming JPEG operation, and JPEG will issue an input-wait interrupt while 1KB bitstream data stored in one of the half buffer region is processed or an output-wait interrupt while 1KB encoded bitstream data is stored into one of the half buffer region in decode and encode modes individually. |
| [7] | **FLY_ON** | **Hardware Memory On-the-fly Access Mode**<br>0 = Disable.<br>1 = Enable, the buffer size for source image data in encode mode is defined by **FLY_SEL.** |
| [6:4] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [3] | **IP_SF_ON** | **Software Memory On-the-fly Access Mode for Data Input**<br>0 = Disable, JPEG can only be triggered after the whole image or bitstream data is stored in frame-buffer in encode or decode mode individually.<br>1 = Enable, JPEG can encode partial image or decode partial bitstream data by re-using a small size frame-buffer; the buffer size for the image data to be encoded is fixed, and the buffer size for the bitstream to be decoded is defined by **BSF_SEL.** |
| [2] | **OP_SF_ON** | **Software Memory On-the-fly Access Mode for Data Output**<br>0 = Disable, JPEG will continue to write the whole encoded bitstream or decoded image data into frame-buffer.<br>1 = Enable, JPEG can write partial encoded bitstream or decoded image data by re-using a small size frame-buffer; the buffer size for the encoded bitstream is defined by **BSF_SEL**, and the buffer size for the decoded image data is fixed. |
| [1:0] | **ENC_MODE** | **JPEG Memory Address Mode Control**<br>00 = Still image encode mode, the encoded bit-stream is always placed into output buffer-0.<br>01 = Still image encode mode, the output dual-buffer is controlled by Register **JDBCR[0].**<br>10 = Continue image encode mode, the encoded bit-stream is placed into the continuous memory address.<br>11 = Continue image encode mode, the distance of memory address for adjacent image is fixed and is specified by JPEG Encode Bit-stream Frame Stride Register (**F_STRIDE**). |

## JPEG Primary Scaling-Up Control Register (JPSCALU)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JPSCALU** | JPEG_BA + 0x054 | R/W | JPEG Primary Scaling-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** | **8X** | **Reserved** | | | **A_JUMP** | **Reserved** | |

| Bits | Description | |
|---|---|---|
| [31:7] | **Reserved** | Reserved. |
| [6] | **8X** | **Primary Image Up-scaling for Encode**<br>0 = No up-scaling, original size.<br>1 = In encode mode, the image is arbitrarily up-scaled 1X~8X;.<br>**Note:** When **FLY_TYPE =** 2'b1x, the up-scaling function is not supported. |
| [5:3] | **Reserved** | Reserved. |
| [2] | **Reserved** | Reserved. |
| [1:0] | **Reserved** | Reserved. |

## JPEG Primary Scaling-Down Control Register (JPSCALD)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JPSCALD** | JPEG_BA + 0x058 | R/W | JPEG Primary Scaling-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** |
| **PSX_ON** | **PS_LPF_ON** | **Reserved** | **PSCALX_F** | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | **PSCALY_F** | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15] | **PSX_ON** | **Primary Image Horizontal Down-scaling for Encode/Decode**<br>0 = Disable, no horizontal down-scale.<br>1 = Enable.<br>**Note:** When **FLY_TYPE** = 2'b1x, the down-scaling function is not supported.<br>In packet format decode mode, the image is arbitrarily down-scaled 1X~16X for Y422 and Y420, 1X~8X for Y444. |
| [14] | **PS_LPF_ON** | **Primary Image Down-scaling Low Pass Filter for Decode**<br>**0 = Disable, no down-scale low pass filter.**<br>**1 = Enable.** |
| [13] | **Reserved** | Reserved. |
| [12:8] | **PSCALX_F** | **Primary Image Horizontal Down-scaling Factor**<br>A 5-bit value specifies the horizontal down-scaling factor. The scaling factor is equal to 2*(1+SCALX_F). For example, if SCALX_F = 1, the image will shrink 4 times in horizontal direction.<br>**Note:** 1. For planar format encode mode, SCALX_F can be any value from 0 to 31. For planar format decode mode, the value of SCALX_F can only be 0, 1, 3 i.e. scaling-down 1/2, 1/4 and 1/8.<br>2. For planar format decode mode, the image width after down-scaling needs to be multiple of 4.<br>3. For packet format, SCALX_F is reserved. |
| [7:6] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [5:0] | **PSCALY_F** | **Primary Image Vertical Down-scaling Factor**<br>A 6-bit value specifies the vertical down-scaling factor. The scaling factor is equal to (1+SCALY_F). For example, if SCALY_F = 3, the image will shrink 4 times in vertical direction.<br>**Note:** For planar format encode mode, SCALY_F can be any value from 0 to 63. For planar format decode mode, the value of SCALY_F can only be 0, 1, 3, 7, i.e. scaling-down 1/1, 1/2, 1/4 and 1/8.<br>For packet format, SCALY_F is reserved. |

## JPEG Thumbnail Scaling-Down Control Register (JTSCALD)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| JTSCALD | JPEG_BA + 0x05C | R/W | JPEG Thumbnail Scaling-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 |
| TSX_ON | Reserved | | TSCALX_F | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | TSCALY_F | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15] | **TSX_ON** | **Thumbnail Image Horizontal Down-scaling for Encode**<br>0 = Disable, no horizontal down-scale.<br>1 = Enable. |
| [14:13] | **Reserved** | Reserved. |
| [12:8] | **TSCALX_F** | **Thumbnail Image Horizontal Down-scaling Factor**<br>A 5-bit value specifies the horizontal down-scaling factor. The scaling factor is equal to 2*(1+SCALX_F).<br>EX: If SCALX_F = 1, the image will shrink 4 times in horizontal direction. |
| [7:6] | **Reserved** | Reserved. |
| [5:0] | **TSCALY_F** | **Thumbnail Image Vertical Down-scaling Factor**<br>A 6-bit value specifies the vertical down-scaling factor. The scaling factor is equal to (1+SCALY_F).<br>EX: If SCALY_F = 3, the image will shrink 4 times in vertical direction. |

## JPEG Dual-Buffer Control Register (JDBCR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| JDBCR | JPEG_BA + 0x060 | R/W | JPEG Dual-Buffer 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 |
| DBF_EN | Reserved | | IP_BUF | Reserved | | | OP_BUF |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7] | **DBF_EN** | **Dual Buffering Control**<br>0 = Disable dual buffering.<br>1 = Enable dual buffering. |
| [6:5] | **Reserved** | Reserved. |
| [4] | **IP_BUF** | **Input Dual Buffer Control**<br>0 = Input data from buffer-0.<br>1 = Input data from buffer-1.<br>**Note:** If **DBF_EN** is disabled, this bit is unused. |
| [3:1] | **Reserved** | Reserved. |
| [0] | **OP_BUF** | **Output Dual Buffer Control**<br>0 = Output data to buffer-0.<br>1 = Output data to buffer-1.<br>**Note:** If **DBF_EN** is disabled, this bit is unused. |

## JPEG Encode Primary Bit-stream Reserved Size Register (JRESERVE)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JRESERVE** | JPEG_BA + 0x070 | R/W | JPEG Encode Primary Bit-stream Reserved Size 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** |
| **RES_SIZE** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **RES_SIZE** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **RES_SIZE** | **Primary Encode Bit-stream Reserved Size**<br>A 16-bit value specifies the reserved size (byte address) in encoded primary JPEG bit-stream.<br>**Note:** When the function of reserved size (**A_JUMP**) is enabled, the value of reserved size must greater than zero, be multiple of 2 but can't be multiple of 4. The actual byte counts reserved in bit-stream is equal to (RES_SIZE - 2). |

## JPEG Address Offset Between Primary/Thumbnail Start Address Register (JOFFSET)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JOFFSET** | JPEG_BA + 0x074 | R/W | JPEG Address Offset Between Primary & Thumbnail Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **OFFSET_SIZE** | **Primary/Thumbnail Starting Address Offset Size**<br>A 24-bit value specifies the offset size (byte address) between the starting address of primary and thumbnail bit-stream.<br>**Note:** When thumbnail encode is enabled, the value of offset size must greater than zero and be multiple of 4. |

## JPEG Encode Bit-Stream Frame Stride Register (JFSTRIDE)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JFSTRIDE** | JPEG_BA + 0x078 | R/W | JPEG Encode Bit-stream Frame Stride Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **F_STRIDE** | **JPEG Encode Bit-stream Frame Stride**<br>A 24-bit value specifies the memory distance between neighbor JPEG bit-stream (byte address of word aligned). |

## JPEG Y Component or Packet Format Frame Buffer 0 Starting Address Register (JYADDR0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JYADDR0** | JPEG_BA + 0x07C | R/W | JPEG Y Component or Packet Format Frame Buffer 0 Starting Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Y_IADDR0** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Y_IADDR0** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Y_IADDR0** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Y_IADDR0** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:0] | **Y_IADDR0** | **JPEG Y Component Frame Buffer-0 Starting Address**<br>A 32-bit value specifies the starting address bits 31 to 0 of frame buffer-0 for Y component or packet format (byte address of word aligned).<br>If a “1” is written to Decoding_Output_Wait_Go (Bit[18] of 0x08), the content of this register will be reloaded as start address of next packet frame buffer and the content of 0x1C0 will be loaded as buffer size in next packet ytransferring. |

### JPEG U Component Frame Buffer-0 Starting Address Register (JUADDR0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JUADDR0** | JPEG_BA + 0x080 | R/W | JPEG U Component Frame Buffer-0 Starting Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **U_IADDR0** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **U_IADDR0** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **U_IADDR0** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **U_IADDR0** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:0] | **U_IADDR0** | **JPEG U Component Frame Buffer-0 Starting Address**<br>A 32-bit value specifies the starting address bits 31 to 0 of frame buffer-0 for U component (byte address of word aligned). |

## JPEG V Component Frame Buffer-0 Starting Address Register (JVADDR0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JVADDR0** | JPEG_BA + 0x084 | R/W | JPEG V Component Frame Buffer-0 Starting Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **V_IADDR0** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **V_IADDR0** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **V_IADDR0** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **V_IADDR0** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:0] | **V_IADDR0** | **JPEG V Component Frame Buffer-0 Starting Address**<br>A 32-bit value specifies the starting address bits 31 to 0 of frame buffer-0 for V component (byte address of word aligned). |

## JPEG Y Component or Packet Format Frame Buffer-1 Starting Address Register (JYADDR1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JYADDR1** | JPEG_BA + 0x088 | R/W | JPEG Y Component or Packet Format Frame Buffer-1 Starting Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Y_IADDR1** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Y_IADDR1** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Y_IADDR1** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Y_IADDR1** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:0] | **Y_IADDR1** | **JPEG Y Component or Packet Format Frame Buffer-1 Starting Address**<br>A 32-bit value specifies the starting address bits 31 to 0 of frame buffer-1 for Y component or packet format (byte address of word aligned) |

## JPEG U Component Frame Buffer-1 Starting Address Register (JUADDR1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JUADDR1** | JPEG_BA + 0x08C | R/W | JPEG U Component Frame Buffer-1 Starting Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **U_IADDR1** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **U_IADDR1** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **U_IADDR1** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **U_IADDR1** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:0] | **U_IADDR1** | **JPEG U Component Frame Buffer-1 Starting Address**<br>A32-bit value specifies the starting address bits 31 to 0 of frame buffer-1 for U component (byte address of word aligned). |

### JPEG V Component Frame Buffer-1 Starting Address Register (JVADDR1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JVADDR1** | JPEG_BA + 0x090 | R/W | JPEG V Component Frame Buffer-1 Starting Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **V_IADDR1** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **V_IADDR1** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **V_IADDR1** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **V_IADDR1** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:0] | **V_IADDR1** | **JPEG V Component Frame Buffer-1 Starting Address**<br>A 32-bit value specifies the starting address bits 31 to 0 of frame buffer-1 for V component (byte address of word aligned). |

## JPEG Y Component Frame Buffer Stride Register (JYSTRIDE)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| JYSTRIDE | JPEG_BA + 0x094 | R/W | JPEG Y Component Frame Buffer Stride 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 | | | | Y_STRIDE | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Y_STRIDE | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:12] | **Reserved** | Reserved. |
| [11:0] | **Y_STRIDE** | **JPEG Y Component Frame Buffer Stride**<br>A 12-bit value specifies the byte offset of memory address of vertical adjacent line for Y component (byte address of word aligned).<br>For packet format, the stride is the difference between the final output width and the input image width after scaling. |

## JPEG U Component Frame Buffer Stride Register (JUSTRIDE)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JUSTRIDE** | JPEG_BA + 0x098 | R/W | JPEG U Component Frame Buffer Stride 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** | | | | **U_STRIDE** | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **U_STRIDE** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:12] | **Reserved** | Reserved. |
| [11:0] | **U_STRIDE** | **JPEG U Component Frame Buffer Stride**<br>A 12-bit value specifies the byte offset of memory address of vertical adjacent line for U component (byte address of word aligned). |

## JPEG V Component Frame Buffer Stride Register (JVSTRIDE)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JVSTRIDE** | JPEG_BA + 0x09C | R/W | JPEG V Component Frame Buffer Stride 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** | | | | **V_STRIDE** | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **V_STRIDE** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:12] | **Reserved** | Reserved. |
| [11:0] | **V_STRIDE** | **JPEG V Component Frame Buffer Stride**<br>A 12-bit value specifies the byte offset of memory address of vertical adjacent line for V component (byte address of word aligned). |

## JPEG Bit-Stream Frame Buffer 0 Starting Address Register (JIOADDR0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JIOADDR0** | JPEG_BA + 0x0A0 | R/W | JPEG Bit-stream Frame Buffer 0 Starting Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **IO_IADDR0** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **IO_IADDR0** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **IO_IADDR0** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **IO_IADDR0** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:0] | **IO_IADDR0** | **JPEG Bit-stream Frame Buffer-0 Starting Address**<br>A 32-bit value specifies the starting address bits 31 to 0 of frame buffer-0 for JPEG bit-stream (byte address of word aligned). |

## JPEG Bit-Stream Frame Buffer 1 Starting Address Register (JIOADDR1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JIOADDR1** | JPEG_BA + 0x0A4 | R/W | JPEG Bit-stream Frame Buffer 1 Starting Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **IO_IADDR1** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **IO_IADDR1** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **IO_IADDR1** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **IO_IADDR1** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:0] | **IO_IADDR1** | **JPEG Bit-stream Frame Buffer-1 Starting Address**<br>A 32-bit value specifies the starting address bits 31 to 0 of frame buffer-1 for JPEG bit-stream (byte address of word aligned). |

### JPEG Encode Primary Image Bit-Stream Size Register (JPRI_SIZE)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JPRI_SIZE** | JPEG_BA + 0x0A8 | R | JPEG Encode Primary Image Bit-stream Size Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **PRI_SIZE** | **JPEG Primary Image Encode Bit-stream Size**<br>A 24-bit value reports the bit-stream byte size of encoded primary image. |

### JPEG Encode Thumbnail Bit-Stream Size Register (JTHB_SIZE)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JTHB_SIZE** | JPEG_BA + 0x0AC | R | JPEG Encode Thumbnail Bit-stream Size 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** |
| **THB_SIZE** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **THB_SIZE** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **THB_SIZE** | **JPEG Thumbnail Image Encode Bit-stream Size**<br>A 16-bit value reports the bit-stream byte size of encoded thumbnail image. |

## JPEG Planar Format Encode Up-Scale and Packet Format Decode Down-Scale Ratio (JUPRAT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JUPRAT** | JPEG_BA + 0x0B0 | R/W | JPEG Planar Format Encode Up-Scale and Packet Format Decode Down-Scale Ratio Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:30] | **Reserved** | Reserved. |
| [29:16] | **S_HEIGHT** | **JPEG Image Height Planar Format Encode Up-scale or Packet Format Decode Down-scale Ratio**<br>A 14-bit value specifies image height planar format encode up-scale or packet format decode down-scale ratio. For planar format, the first 4 bits are integer part and the others are decimal part. For packet format, 13 bits are decimal part. The JPEG engine supports vertical arbitrarily up-scaling in planar format encode mode and arbitrarily down-scaling in packet format decode mode. This value needs to be specified only when vertical up-scaling (**Y2**) or down-scaling (**PSX_ON**) is enabled.<br>Note : if up-scale from 128 to 256, the up-scale ratio is (256-1)/(128-1) instead of 2. if down-scale from 256 to 128, the down-scale ratio is ceil(4096/8192). The height is floor[(256 – 1)*(4096/8192)] + 1 . |
| [15:14] | **Reserved** | Reserved. |
| [13:0] | **S_WIDTH** | **JPEG Image Width Planar Format Encode Up-scale or Packet Format Decode Down-scale Ratio**<br>A 14-bit value specifies source image width planar format encode up-scale or packet format decode down-scale ratio. For planar format, the first 4 bits are integer part and the others are decimal part. For packet format, 13 bits are decimal part. The JPEG engine supports horizontal arbitrarily up-scaling in planar format encode mode and arbitrarily down-scaling in packet format decode mode. This value needs to be specified only when horizontal up-scaling (**X2**) or down-scaling (**PSX_ON**) is enabled.<br>Note : if up-scale from 128 to 256, the up-scale ratio is (256-1)/(128-1) instead of 2. If down-scale from 256 to 128, the down-scale ratio is ceil(4096/8192). The width is floor[(256 – 1)*(4096/8192)] + 1; |

## JPEG Bit-stream FIFO Control Register (JBSFIFO)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| JBSFIFO | JPEG_BA + 0x0B4 | R/W | JPEG Bit-stream FIFO Control Register | 0x0000_0032 |

| 31 | 30 | 29 | 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 | BSFIFO_HT | | | Reserved | BSFIFO_LT | | |

| Bits | Description | |
|---|---|---|
| [31:7] | **Reserved** | Reserved. |
| [6:4] | **BSFIFO_HT** | **Bit-stream FIFO High-threshold Control**<br>While the fullness of bit-stream output FIFO is higher than the high-threshold in encode mode, the priority for output will become higher than input.<br>000 = 2 words.<br>001 = 4 words.<br>010 = 6 words.<br>011 = 8 words.<br>100 = 10 words.<br>101 = 12 words.<br>110 = 14 words.<br>111 = 16 words. |
| [3] | **Reserved** | Reserved. |
| [2:0] | **BSFIFO_LT** | **Bit-stream FIFO Low-threshold Control**<br>The JPEG engine may start to request memory access while the fullness of bit-stream output FIFO in encode mode or emptiness of bit-stream input FIFO in decode mode is higher than the low-threshold.<br>000 = 1 word.<br>001 = 2 words.<br>010 = 4 words.<br>011 = 6 words.<br>100 = 8 words.<br>101 = 10 words.<br>110 = 12 words.<br>111 = 14 words. |

## JPEG Encode Source Image Height Register (JSRCH)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JSRCH** | JPEG_BA + 0x0B8 | R/W | JPEG Encode Source Image Height Register | 0x0000_0FFF |

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

| Bits | Description | |
|---|---|---|
| [31:12] | **Reserved** | Reserved. |
| [11:0] | **JSRCH** | **JPEG Encode Source Image Height**<br>A 12-bit value specifies source image height. The JPEG engine supports vertical arbitrarily up-scaling in encode mode. This value needs to be specified only when vertical up-scaling (**Y2**) is enabled. |

## JPEG Quantization Table 0 Element Register n (JQTAB0ERn, n = 0 ~ 15)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JQTAB0ER0** | JPEG_BA+0x100 | R/W | JPEG Quantization Table 0 Element Register 0 | 0x0000_0000 |
| **JQTAB0ER1** | JPEG_BA+0x104 | R/W | JPEG Quantization Table 0 Element Register 1 | 0x0000_0000 |
| **JQTAB0ER2** | JPEG_BA+0x108 | R/W | JPEG Quantization Table 0 Element Register 2 | 0x0000_0000 |
| **JQTAB0ER3** | JPEG_BA+0x10C | R/W | JPEG Quantization Table 0 Element Register 3 | 0x0000_0000 |
| **JQTAB0ER4** | JPEG_BA+0x110 | R/W | JPEG Quantization Table 0 Element Register 4 | 0x0000_0000 |
| **JQTAB0ER5** | JPEG_BA+0x114 | R/W | JPEG Quantization Table 0 Element Register 5 | 0x0000_0000 |
| **JQTAB0ER6** | JPEG_BA+0x118 | R/W | JPEG Quantization Table 0 Element Register 6 | 0x0000_0000 |
| **JQTAB0ER7** | JPEG_BA+0x11C | R/W | JPEG Quantization Table 0 Element Register 7 | 0x0000_0000 |
| **JQTAB0ER8** | JPEG_BA+0x120 | R/W | JPEG Quantization Table 0 Element Register 8 | 0x0000_0000 |
| **JQTAB0ER9** | JPEG_BA+0x124 | R/W | JPEG Quantization Table 0 Element Register 9 | 0x0000_0000 |
| **JQTAB0ER10** | JPEG_BA+0x128 | R/W | JPEG Quantization Table 0 Element Register 10 | 0x0000_0000 |
| **JQTAB0ER11** | JPEG_BA+0x12C | R/W | JPEG Quantization Table 0 Element Register 11 | 0x0000_0000 |
| **JQTAB0ER12** | JPEG_BA+0x130 | R/W | JPEG Quantization Table 0 Element Register 12 | 0x0000_0000 |
| **JQTAB0ER13** | JPEG_BA+0x134 | R/W | JPEG Quantization Table 0 Element Register 13 | 0x0000_0000 |
| **JQTAB0ER14** | JPEG_BA+0x138 | R/W | JPEG Quantization Table 0 Element Register 14 | 0x0000_0000 |
| **JQTAB0ER15** | JPEG_BA+0x13C | R/W | JPEG Quantization Table 0 Element Register 15 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **QTAB0_3** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **QTAB0_2** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **QTAB0_1** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **QTAB0_0** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **QTAB0_3** | **JPEG Quantization-table 0 – 3**<br>An 8-bit value specifies one element (3, 7, 11, …, 59, 63) of the Quantization-Table 0.<br>**Note:**<br>1. The sequence order of QTAB is from the left to right and from the top to bottom.<br>2. You need to read the same address twice to get the correct data |

| | | |
|---|---|---|
| [23:16] | **QTAB0_2** | **JPEG Quantization-table 0 – 2**<br>An 8-bit value specifies one element (2, 6, 10, …, 58, 62) of the Quantization-Table 0. |
| [15:8] | **QTAB0_1** | **JPEG Quantization-table 0 – 1**<br>An 8-bit value specifies one element (1, 5, 9, …, 57, 61) of the Quantization-Table 0. |
| [7:0] | **QTAB0_0** | **JPEG Quantization-table 0 – 0**<br>An 8-bit value specifies one element (0, 4, 8, …, 56, 60) of the Quantization-Table 0. |

## JPEG Quantization Table 1 Element Register n (JQTAB1ERn, n = 0 ~ 15)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JQTAB1ER0** | JPEG_BA+0x140 | R/W | JPEG Quantization Table 1 Element Register 0 | 0x0000_0000 |
| **JQTAB1ER1** | JPEG_BA+0x144 | R/W | JPEG Quantization Table 1 Element Register 1 | 0x0000_0000 |
| **JQTAB1ER2** | JPEG_BA+0x148 | R/W | JPEG Quantization Table 1 Element Register 2 | 0x0000_0000 |
| **JQTAB1ER3** | JPEG_BA+0x14C | R/W | JPEG Quantization Table 1 Element Register 3 | 0x0000_0000 |
| **JQTAB1ER4** | JPEG_BA+0x150 | R/W | JPEG Quantization Table 1 Element Register 4 | 0x0000_0000 |
| **JQTAB1ER5** | JPEG_BA+0x154 | R/W | JPEG Quantization Table 1 Element Register 5 | 0x0000_0000 |
| **JQTAB1ER6** | JPEG_BA+0x158 | R/W | JPEG Quantization Table 1 Element Register 6 | 0x0000_0000 |
| **JQTAB1ER7** | JPEG_BA+0x15C | R/W | JPEG Quantization Table 1 Element Register 7 | 0x0000_0000 |
| **JQTAB1ER8** | JPEG_BA+0x160 | R/W | JPEG Quantization Table 1 Element Register 8 | 0x0000_0000 |
| **JQTAB1ER9** | JPEG_BA+0x164 | R/W | JPEG Quantization Table 1 Element Register 9 | 0x0000_0000 |
| **JQTAB1ER10** | JPEG_BA+0x168 | R/W | JPEG Quantization Table 1 Element Register 10 | 0x0000_0000 |
| **JQTAB1ER11** | JPEG_BA+0x16C | R/W | JPEG Quantization Table 1 Element Register 11 | 0x0000_0000 |
| **JQTAB1ER12** | JPEG_BA+0x170 | R/W | JPEG Quantization Table 1 Element Register 12 | 0x0000_0000 |
| **JQTAB1ER13** | JPEG_BA+0x174 | R/W | JPEG Quantization Table 1 Element Register 13 | 0x0000_0000 |
| **JQTAB1ER14** | JPEG_BA+0x178 | R/W | JPEG Quantization Table 1 Element Register 14 | 0x0000_0000 |
| **JQTAB1ER15** | JPEG_BA+0x17C | R/W | JPEG Quantization Table 1 Element Register 15 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **QTAB1_3** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **QTAB1_2** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **QTAB1_1** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **QTAB1_0** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **QTAB1_3** | **JPEG Quantization-table 1 – 3**<br>An 8-bit value specifies one element of the Quantization-Table 1.<br>**Note:** When three-QTAB mode (**E3QTAB**) is enabled, JPEG encoder uses this table for coding Cb component. Otherwise JPEG encoder uses this table for coding both Cb and Cr component. The other requirements are the same as QTAB0 described above. |
| [23:16] | **QTAB1_2** | **JPEG Quantization-table 1 – 2**<br>An 8-bit value specifies one element of the Quantization-Table 1. |

| | | |
|---|---|---|
| [15:8] | **QTAB1_1** | **JPEG Quantization-table 1 – 1**<br>An 8-bit value specifies one element of the Quantization-Table 1. |
| [7:0] | **QTAB1_0** | **JPEG Quantization-table 1 – 0**<br>An 8-bit value specifies one element of the Quantization-Table 1. |

## JPEG Quantization Table 2 Element Register n (JQTAB2ERn, n = 0 ~ 15)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **JQTAB2ER0** | JPEG_BA+0x180 | R/W | JPEG Quantization Table 2 Element Register 0 | 0x0000_0000 |
| **JQTAB2ER1** | JPEG_BA+0x184 | R/W | JPEG Quantization Table 2 Element Register 1 | 0x0000_0000 |
| **JQTAB2ER2** | JPEG_BA+0x188 | R/W | JPEG Quantization Table 2 Element Register 2 | 0x0000_0000 |
| **JQTAB2ER3** | JPEG_BA+0x18C | R/W | JPEG Quantization Table 2 Element Register 3 | 0x0000_0000 |
| **JQTAB2ER4** | JPEG_BA+0x190 | R/W | JPEG Quantization Table 2 Element Register 4 | 0x0000_0000 |
| **JQTAB2ER5** | JPEG_BA+0x194 | R/W | JPEG Quantization Table 2 Element Register 5 | 0x0000_0000 |
| **JQTAB2ER6** | JPEG_BA+0x198 | R/W | JPEG Quantization Table 2 Element Register 6 | 0x0000_0000 |
| **JQTAB2ER7** | JPEG_BA+0x19C | R/W | JPEG Quantization Table 2 Element Register 7 | 0x0000_0000 |
| **JQTAB2ER8** | JPEG_BA+0x1A0 | R/W | JPEG Quantization Table 2 Element Register 8 | 0x0000_0000 |
| **JQTAB2ER9** | JPEG_BA+0x1A4 | R/W | JPEG Quantization Table 2 Element Register 9 | 0x0000_0000 |
| **JQTAB2ER10** | JPEG_BA+0x1A8 | R/W | JPEG Quantization Table 2 Element Register 10 | 0x0000_0000 |
| **JQTAB2ER11** | JPEG_BA+0x1AC | R/W | JPEG Quantization Table 2 Element Register 11 | 0x0000_0000 |
| **JQTAB2ER12** | JPEG_BA+0x1B0 | R/W | JPEG Quantization Table 2 Element Register 12 | 0x0000_0000 |
| **JQTAB2ER13** | JPEG_BA+0x1B4 | R/W | JPEG Quantization Table 2 Element Register 13 | 0x0000_0000 |
| **JQTAB2ER14** | JPEG_BA+0x1B8 | R/W | JPEG Quantization Table 2 Element Register 14 | 0x0000_0000 |
| **JQTAB2ER15** | JPEG_BA+0x1BC | R/W | JPEG Quantization Table 2 Element Register 15 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **QTAB2_3** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **QTAB2_2** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **QTAB2_1** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **QTAB2_0** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **QTAB2_3** | **JPEG Quantization-table 2 – 3**<br>An 8-bit value specifies one element of the Quantization-Table 2.<br>**Note:** When three-QTAB mode (**E3QTAB**) is enabled, JPEG encoder uses this table for coding Cr component. Otherwise this table is unused. The other requirements are the same as QTAB0 described above. |
| [23:16] | **QTAB2_2** | **JPEG Quantization-table 2 – 2**<br>An 8-bit value specifies one element of the Quantization-Table 2. |

| | | |
|---|---|---|
| [15:8] | **QTAB2_1** | **JPEG Quantization-table 2 – 1**<br>An 8-bit value specifies one element of the Quantization-Table 2. |
| [7:0] | **QTAB2_0** | **JPEG Quantization-table 2 – 0**<br>An 8-bit value specifies one element of the Quantization-Table 2. |

## 5.30 LCD Display Interface Controller (LCM)

### 5.30.1 Overview

The main purpose of Display Controller is used to display the video/image data to LCD device or connect with external TV-encoder. The video/image data source may come from the image sensor, JPEG decoder and the OSD pattern which have been stored in system memory (SDRAM). The input data format of the display controller can be packet YUV422, packet YUV444, packet RGB444, packet RGB565, packet RGB666, and packet RGB888. The OSD (On Screen Display) function supports packet YUV422 and 8/16/24-bit direct-color mode. The LCD controller supports both sync-type and MPU-type LCDM. This LCD Controller is a bus master and can transfer display data from system memory (SDRAM) without CPU intervention.

### 5.30.2 Features

- Input data format
  - YUV422, YUV444
  - RGB444, RGB565, RGB666, RGB888
- Output format
  - YUV422, YUV444
  - RGB444, RGB565, RGB666, RGB888
- Input size: Maximum size 1024 * 768
- Image resize
  - Horizontal up-scaling 1~8X in fractional steps
  - Vertical up-scaling 1~8X in fractional steps
- Convert full range YUV to CCIR601
- Windowing support for three OSD graphic or text overlay
- Support CCIR-656 (with header), CCIR-601(with hsync and vsync) 8/16-bit YUV data output format to connect with external TV encoder
- Support both sync-type and MPU-type LCM
- Support the 8/9/16/18/24-bit data output to connect with 80/68 series MPU type LCM module

The LCD Controller includes the following main functions：

- Video post-processing
- Display & overlay control
- Video output control
- Hardware cursor control

### 5.30.3 Block Diagram

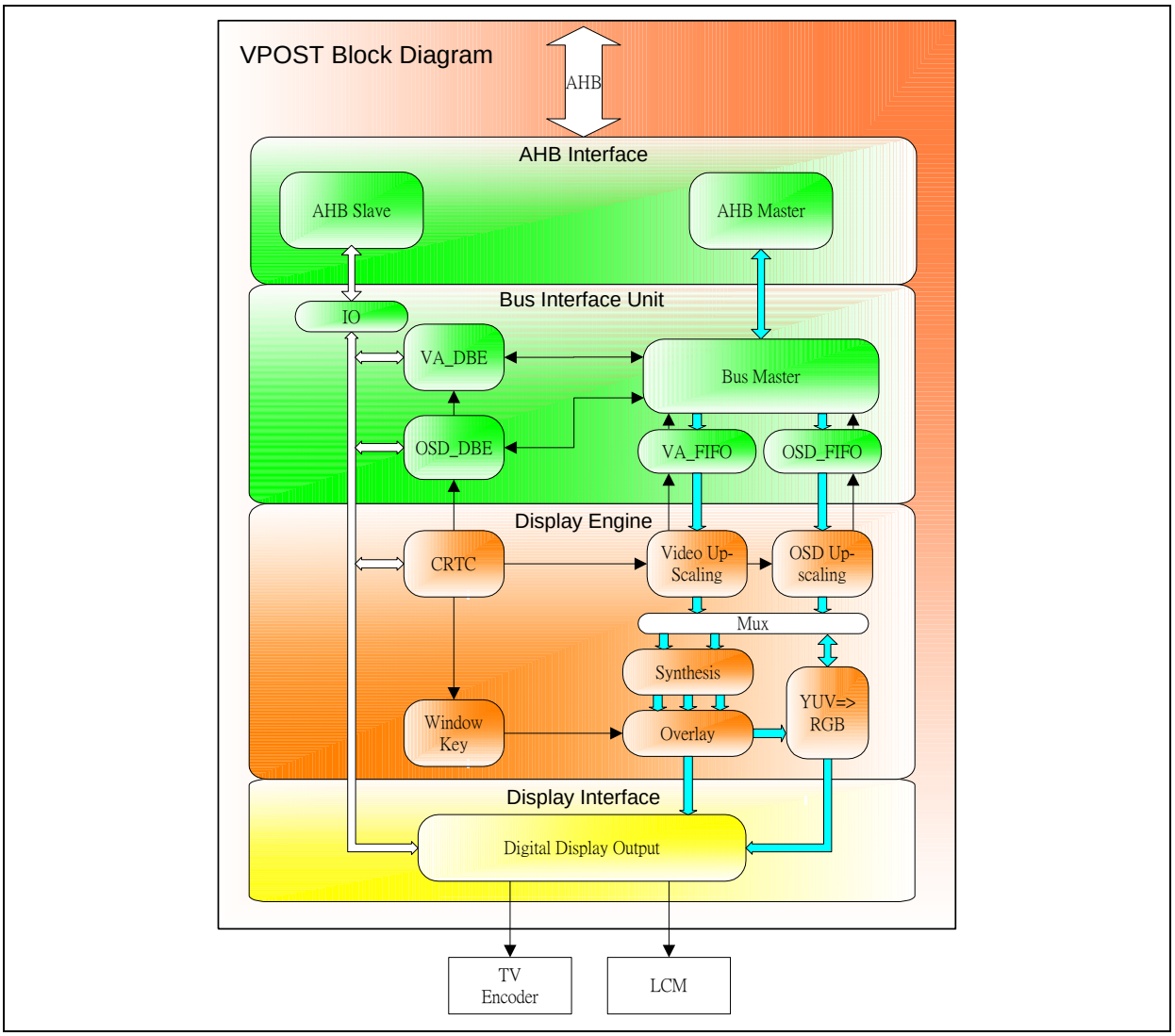



Figure 5.30-1 LCD Display Interface Controller Functional Block Diagram

### 5.30.4 Basic Configuration

Before using LCD display interface, it's necessary to configure related pins as the LCD function and enable LCD's clock.

For LCD display interface related pin configuration, please refer to the register SYS_MFP_GPAL, SYS_MFP_GPAH, SYS_MFP_GPDH, SYS_MFP_GPGL and SYS_MFP_GPGH to know how to configure related pins as the LCD display interface function.

Set LCD (CLK_HCLKEN[26]) high to turn on clock for LCD display controller. In addition, it's necessary to configure LCD_S (CLK_DIVCTL1[4:0]) and LCD_N (CLK_DIVCTL1[15:8]) to generate LCD_CLK to LCD panel properly.

### 5.30.5 Functional Description

5.30.5.1 *VPOST Processor*

The Video Engine is used to scale-up the video for display. The video can be arbitrarily up-scaled to full-screen size in horizontal and vertical direction by programming the scaling-factor registers VA_SCALE. Similarly, the OSD function also supports up-scaling in horizontal and vertical direction. But it only supports 2X and 4X up-scaling.

5.30.5.2 *Display & Overlay Control*

The Display Control unit includes timing controller and overlay controller. The timing controller generates the required horizontal and vertical timing for display device. The display timing is defined in the control registers, CRTC_SIZE~CRTC_VR and OSD_WINS~OSD_WINE. The following figure specifies the registers definition.

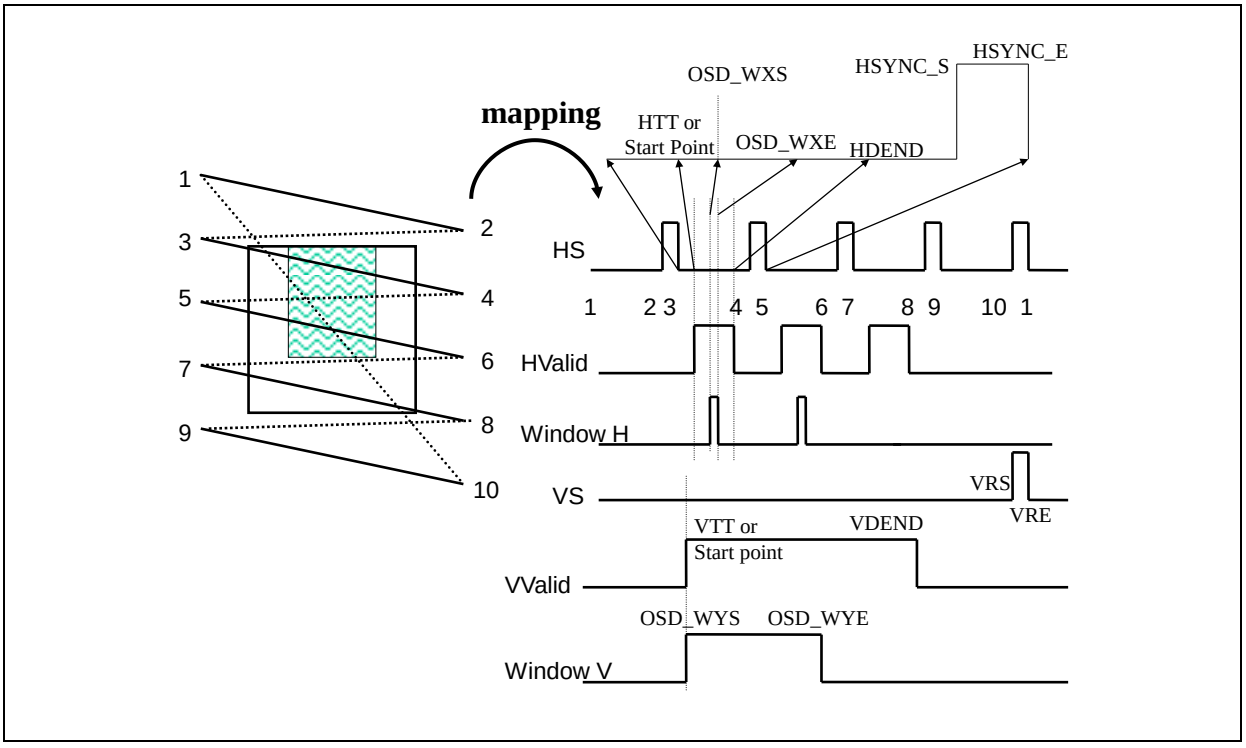



Figure 5.30-2 Display and Overlay Control Timing

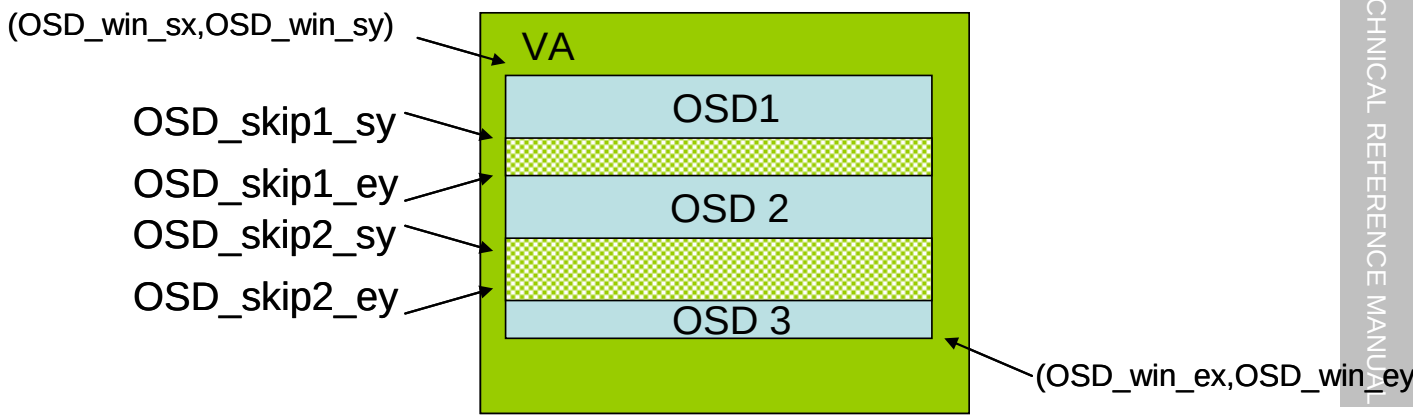
The Display Controller supports various video/OSD overlay control function. The OSD can be turned on and overlaid with video data by setting register OSD_EN (DCCS[2]) and OSD window coordinates registers, OSD_WINS and OSD_WINE. The OSD data format supports 8/16/24-bit direct-color mode. The OSD format is selected by register OSD_SRC (DCCS[14:12]). The OSD data can be transparent, blinking or mixed with video data by setting overlay control registers OSD_OVERLAY. The display condition is depicted in the following table:

| Window-Key | Color-Key | OCR1 | OCR0 | Display |
|---|---|---|---|---|
| 0 | X | X | X | Video |
| 1 | 0 | X | 0 | Video |
| 1 | 0 | X | 1 | OSD |
| 1 | 0 | X | 2 | Video+OSD |
| 1 | 1 | 0 | X | Video |
| 1 | 1 | 1 | X | OSD |
| 1 | 1 | 2 | X | Video+OSD |

The Color-Key value indicates the color-key condition is matched or un-matched. The OSD color-key pattern is defined in register OSD_CKEY for V/U/Y or B/G/R components according to the source color format OSD_SRC (DCCS[14:12]). The color-key mask is also provided in registers OSD_CMASK. Only the color-key pattern, which with the mask bits has been set to “1”, will be compared with the OSD graphic data.

Setting the value of register VA_SYNW (OSD_OVERLAY[6:4]) can change the weighting of mixing the data of video and OSD streams. In addition, the OSD can be periodically blinking by setting the blinking cycle time in register BLINK_VCNT (OSD_OVERLAY[23:16]).

Setting the value of registers OSK_SKIP1 and OSD_SKIP2 can separate the OSD window into 3 sub-windows. The OSD data fetching of lines enclosed by SKIP regions is skipped. So, the OSD frame buffer size can be reduced. The display condition is depicted in the following diagram:



5.30.5.3 *Digital Display Output Control*

Various digital video output modes are supported:

(1) 8-bit/16-bit YUV output for external TV-encoder;

(2) 8-bit RGB output for sync-based TFT-LCD device;

(3) 8-bit/16-bit/18-bit RGB output for high-color sync-based TFT-LCD device;

(4) 8-bit/9-bit/16-bit/18-bit RGB output for MPU-interfaced LCD device.

The display device is defined in register DEVICE (DEVICE_CTRL[7:5]). The data bus 8-bit/16-bit or 9-bit/18-bit is selected by DBWORD (DEVICE_CTRL[26]). For the MPU-interfaced LCD, 68-series and 80-series MPU interface are supported. The display color formats can be 4096 (RGB444), 65536 (RGB565), and 262144 (RGB666) colors both in 8-bit and 16-bit or 9-bit and 18-bit data bus modes. The related control signals for MPU-interfaced LCD are defined in register DEVICE_CTRL. In addition, the video source color format can be YUV or RGB by setting register VA_SRC (DCCS[10:8]).

5.30.5.4 *Display Pin Assignment*

| **Pad Name** | **VD [23:0]** | **HSYNC** | **VSYNC** | **VDEN** | **VICLK** | **VOCLK** |
|---|---|---|---|---|---|---|
| **Sync mode** | Video data bus(O) | HSYNC(O) | VSYNC(O) | Data enable(O) | Clock in (I) | Clock out (O) |
| **MPU80** | Video data bus(I/O) | Write (WR) (O) | Read (RD) (O) | MPU-LCD (RS) (O) | Non used | Chip select(CS) (O) |
| **MPU80+VSync** | Video data bus(I/O) | Write (WR) (O) | Read (RD) (O) | MPU-LCD (RS) (O) | Vsync (O) | Chip select(CS) (O) |
| **MPU80+FMARK** | Video data bus(I/O) | Write (WR) (O) | Read (RD) (O) | MPU-LCD (RS) (O) | FMARK (I) | Chip select(CS) (O) |
| **MPU68** | Video data bus(I/O) | Enable (EN) (O) | Read/Write (RW) (O) | MPU-LCD (RS) (O) | Non used | Chip select(CS) (O) |
| **MPU68+VSync** | Video data bus(I/O) | Enable (EN) (O) | Read/Write (RW) (O) | MPU-LCD (RS) (O) | Vsync (O) | Chip select(CS) (O) |
| **MPU68+FMARK** | Video data bus(I/O) | Enable (EN) (O) | Read/Write (RW) (O) | MPU-LCD (RS) (O) | FMARK (I) | Chip select(CS) (O) |

### 5.30.6 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **LCM Base Address:**<br>**LCM_BA = 0xB000_8000** | | | | |
| **DCCS** | LCM_BA + 0x00 | R/W | Display Controller Control and Status Register | 0x0000_0000 |
| **DEVICE_CTRL** | LCM_BA + 0x04 | R/W | Display Output Device Control Register | 0x0000_00E0 |
| **MPULCD_CMD** | LCM_BA + 0x08 | R/W | MPU-Interface LCD Write Command Register | 0x0000_0000 |
| **INT_CS** | LCM_BA + 0x0C | R/W | Interrupt Control/Status Register | 0x0000_0000 |
| **CRTC_SIZE** | LCM_BA + 0x10 | R/W | CRTC Display Size Register | 0x0000_0000 |
| **CRTC_DEND** | LCM_BA + 0x14 | R/W | CRTC Display Enable End Register | 0x0000_0000 |
| **CRTC_HR** | LCM_BA + 0x18 | R/W | CRTC Internal Horizontal Retrace Timing Register | 0x0000_0000 |
| **CRTC_HSYNC** | LCM_BA + 0x1C | R/W | CRTC Horizontal Sync Timing Register | 0x0000_0000 |
| **CRTC_VR** | LCM_BA + 0x20 | R/W | CRTC Internal Vertical Retrace Timing Register | 0x0000_0000 |
| **VA_BADDR0** | LCM_BA + 0x24 | R/W | Video Stream Frame Buffer-0 Starting Address Register | 0x0000_0000 |
| **VA_BADDR1** | LCM_BA + 0x28 | R/W | Video Stream Frame Buffer-1 Starting Address Register | 0x0000_0000 |
| **VA_FBCTRL** | LCM_BA + 0x2C | R/W | Video Stream Frame Buffer Control Register | 0x0000_0000 |
| **VA_SCALE** | LCM_BA + 0x30 | R/W | Video Stream Scaling Control Register | 0x0000_0000 |
| **VA_TEST** | LCM_BA + 0x34 | R/W | Test Mode Control Register | 0x0000_0000 |
| **VA_WIN** | LCM_BA + 0x38 | R/W | Video Stream Active Window Coordinates Register | 0x0001_07FF |
| **VA_STUFF** | LCM_BA + 0x3C | R/W | Video Stream Stuff Register | 0x0000_0000 |
| **OSD_WINS** | LCM_BA + 0x40 | R/W | OSD Window Starting Coordinates Register | 0x0000_0000 |
| **OSD_WINE** | LCM_BA + 0x44 | R/W | OSD Window Ending Coordinates Register | 0x0000_0000 |
| **OSD_BADDR** | LCM_BA + 0x48 | R/W | OSD Stream Frame Buffer Starting Address Register | 0x0000_0000 |
| **OSD_FBCTRL** | LCM_BA + 0x4C | R/W | OSD Stream Frame Buffer Control Register | 0x0000_0000 |
| **OSD_OVERLAY** | LCM_BA + 0x50 | R/W | OSD Overlay Control Register | 0x0000_0000 |
| **OSD_CKEY** | LCM_BA + 0x54 | R/W | OSD Overlay Color-Key Pattern Register | 0x0000_0000 |
| **OSD_CMASK** | LCM_BA + 0x58 | R/W | OSD Overlay Color-Key Mask Register | 0x0000_0000 |
| **OSD_SKIP1** | LCM_BA + 0x5C | R/W | OSD Window Skip1 Register | 0x0000_0000 |
| **OSD_SKIP2** | LCM_BA + 0x60 | R/W | OSD Window Skip2 Register | 0x0000_0000 |
| **OSD_SCALE** | LCM_BA + 0x64 | R/W | OSD Scaling Control Register | 0x0000_0000 |
| **MPU_VSYNC** | LCM_BA + 0x68 | R/W | MPU Vsync Control Register | 0x0000_0000 |
| **HC_CTRL** | LCM_BA + 0x6C | R/W | Hardware Cursor Control Register | 0x0000_0000 |
| **HC_POS** | LCM_BA + 0x70 | R/W | Hardware Cursor Position Register | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **HC_WBCTRL** | LCM_BA + 0x74 | R/W | Hardware Cursor Window Buffer Control Register | 0x0000_0000 |
| **HC_BADDR** | LCM_BA + 0x78 | R/W | Hardware Cursor Memory Base Address Register | 0x0000_0000 |
| **HC_COLOR0** | LCM_BA + 0x7C | R/W | Hardware Cursor Color RAM 0 Register | 0x0000_0000 |
| **HC_COLOR1** | LCM_BA + 0x80 | R/W | Hardware Cursor Color RAM 1 Register | 0x0000_0000 |
| **HC_COLOR2** | LCM_BA + 0x84 | R/W | Hardware Cursor Color RAM 2 Register | 0x0000_0000 |
| **HC_COLOR3** | LCM_BA + 0x88 | R/W | Hardware Cursor Color RAM 3 Register | 0x0000_0000 |

### 5.30.7 Register Description

**Display Controller Control and Status Register (DCCS)**

The register includes Display-Output-Control, Stream-Control and Display-Video-Source-Format-Control registers.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| DCCS | LCM_BA + 0x00 | R/W | Display Controller Control and Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | LACE_F | VSYNC | HACT | VACT | DISP_ON | Reserved |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | OSD_HUP | | OSD_VUP | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| ITU_EN | OSD_SRC | | | Reserved | VA_SRC | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SINGLE | FIELD_INTR | CMD_ON | DISP_INT_EN | DISP_OUT_EN | OSD_EN | VA_EN | ENG_RST |

| Bits | Description | |
|---|---|---|
| [31:30] | **Reserved** | Reserved. |
| [29] | **LACE_F** | **Interlace Mode Display Field Status (Read Only)**<br>0 = Current displayed field is even field.<br>1 = Current displayed field is odd field. |
| [28] | **VSYNC** | **Internal Vertical Sync Status (Read Only)**<br>When V_POL (DEVICE_CTRL[21]) = 1 (high active).<br>0 = Display operation is not within vertical sync period.<br>1 = Display operation is within vertical sync period.<br>When V_POL (DEVICE_CTRL[21]) = 0 (low active).<br>0 = Display operation is within vertical sync period.<br>1 = Display operation is not within vertical sync period. |
| [27] | **HACT** | **Display Horizontal Line (Read Only)**<br>0 = Display Controller is not operating for horizontal line display.<br>1 = Display Controller is operating for horizontal line display. |
| [26] | **VACT** | **Display Video Frame (Read Only)**<br>0 = Display Controller is not operating for video frame display.<br>1 = Display Controller is operating for video frame display. |
| [25] | **DISP_ON** | **Display Controller Active (Read Only)**<br>0 = Display Controller is active.<br>1 = Display Controller is off. |

| | | |
|---|---|---|
| [24:20] | **Reserved** | Reserved. |
| [19:18] | **OSD_HUP** | **OSD Stream Horizontal Up-scaling**<br>00 = Original.<br>01 = 2X.<br>10 = Reserved.<br>11 = Reserved. |
| [17:16] | **OSD_VUP** | **OSD Stream Vertical Up-scaling**<br>00 = 1X.<br>01 = 2X.<br>10 = 4X.<br>11 = Reserved. |
| [15] | **ITU_EN** | **ITU656 Format Header Encode**<br>When DEVICE (DEVICE_CTRL[7:5]) = 000 and DBWORD (DEVICE_CTRL[26]) = 0,.<br>0 = Disable.<br>1 = Enable. |
| [14:12] | **OSD_SRC** | **OSD Stream Source Color Format**<br>000 = YUV422.<br>001 = YCBCR422.<br>010 = RGB888.<br>011 = RGB666.<br>100 = RGB565.<br>101 = RGB444 Low: {4'h0,R,G,B}.<br>111 = RGB444 High: {R,G,B,4'h0}.<br>110 = RGB332. |
| [11] | **Reserved** | Reserved. |
| [10:8] | **VA_SRC** | **Video Stream Source Color Format**<br>000 = YUV422.<br>001 = YCBCR422.<br>010 = RGB888.<br>011 = RGB666.<br>100 = RGB565.<br>101 = RGB444 Low: {4'h0,R,G,B}.<br>111 = RGB444 High: {R,G,B,4'h0}.<br>110 = Reserved. |
| [7] | **SINGLE** | **Display Single Frame Mode**<br>0 = Display continuous video frame.<br>1 = Display single picture frame, the Display Controller will stop operating after finishing display one frame. |
| [6] | **FIELD_INTR** | **Interrupt Mode Control**<br>0 = Interrupt signal responses at each frame display complete.<br>1 = Interrupt signal responses at each field display complete.<br>**Note:** The setting for this is only meaningful when the display mode is operated at interlaced mode. |

| | | |
|---|---|---|
| [5] | **CMD_ON** | **Command Mode**<br>0 = Normal video display mode.<br>1 = Turn-on command mode for sending LCD command or parameter data. |
| [4] | **DISP_INT_EN** | **Display Controller Interrupt Output Enable**<br>**0 = Disable.**<br>**1 = Enable.** |
| [3] | **DISP_OUT_EN** | **Display-relative Output Pins Tri-state Mode**<br>0 = Output disabled, output pins in tri-state mode.<br>1 = Display output enable, normal mode. |
| [2] | **OSD_EN** | **OSD Data Fetch Control**<br>0 = Disable.<br>1 = Enable. |
| [1] | **VA_EN** | **Video**<br>0 = Disable.<br>1 = Enable. |
| [0] | **ENG_RST** | **Display Engine Reset (Except Display Control Registers)**<br>0 = Disable, normal operation.<br>1 = Reset the Display Engine, but the value of the display control registers keep no change. |

## Display Output Device Control Register (DEVICE_CTRL)

The type of external output device is controlled by this register.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| DEVICE_CTRL | LCM_BA + 0x04 | R/W | Display Output Device Control Register | 0x0000_00E0 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CMD_LOW | CM16t18 | CMD16 | DE_POL | MCU68 | DBWORD | RGB_SCALE | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| LACE | VR_LACE | V_POL | H_POL | FAL_D | LCD_ODD | SEL_ODD | YUV2CCIR |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LCD_DDA | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DEVICE | | | RGB_SHIFT | | SWAP_YCbCr | | Reserved |

| Bits | Description | |
|---|---|---|
| [31] | **CMD_LOW** | **Command Low**<br>0 = Output pin RS = 1: command data, 0: display/parameter data.<br>1 = Output pin RS = 0: command data, 1: display/parameter data. |
| [30] | **CM16t18** | **Command Mapping From 16-bit to 18-bit or 8-bit to 9-bit Data Bus**<br>Used for 18-bit/9-bit RGB666 MPU-Interfaced LCD device mode.<br>0 = For 18-bit data bus mode,.<br>Data[17:0] = {Command[15:8], 1'b0, Command[7:0],. 1'b0};<br>For 9-bit data bus mode,<br>Data[8:0] = {Command[7:0],. 1'b0};<br>1 = For 18-bit data bus mode,.<br>Data[17:0] = {2'b00, Command[...15:0]};<br>For 9-bit data bus mode,<br>Data[8:0] = {1'b0, Command.[7:0]}; |
| [29] | **CMD16** | **Command Data 16-bit Mode**<br>0 = The command data is 8-bit.<br>1 = The command data is 16-bit.<br>**Note:** The 16-bit command mode is only valid when DBWORD (DEVICE_CTRL[26]) is active. |
| [28] | **DE_POL/ IM_262K (Share bit)** | **Active Polarity of Display Output Enable for Sync-type LCM**<br>0 = Active high.<br>1 = Active low.<br>Interface Mode Selection for 262K MPU-Interface LCM<br>0 = 9/18 bit data bus.<br>1 = 8/16 bit data bus. |

| | | |
|---|---|---|
| [27] | **MPU68** | **MPU Interface Selection**<br>0 = 80-series MPU interface.<br>1 = 68-series MPU interface. |
| [26] | **DBWORD** | **Digital LCD Data Bus Width Selection**<br>(Data bus width is equal to WORD length):<br>0 = Bus width is equal to half-word, two bus transactions are required for single pixel.<br>1 = Bus width is equal to word, one bus transaction is required for single pixel.<br>For YUV422 output mode:<br>0 = Data bus is 8-bit.<br>1 = Data bus is 16-bit.<br>For 256/4096/65536 colors mode:<br>0 = Data bus is 8-bit.<br>1 = Data bus is 16-bit.<br>For 262144 colors mode:<br>0 = Data bus is 8/9-bit.<br>1 = Data bus is 16/18-bit.<br>For 1677721 colors mode:<br>0 = Data bus is 8-bit – 3 cycles per pixel at SWAP_YCbCr[1] (DEVICE[2]) = 0.<br>4 cycles per pixel at SWAP_YCbCr[1] (DEVICE[2]) = 1.<br>1 = Data bus is 24-bit – 1 cycles per pixel. |
| [25:24] | **RGB_SCALE** | **RGB Color Type**<br>00 = 4096 colors mode.<br>01 = 65536 colors mode.<br>10 = 262144 colors mode.<br>11 = 16777216 colors mode. |
| [23] | **LACE** | **Display Data Output Mode**<br>0 = Non-interlace.<br>1 = Interlace. |
| [22] | **VR_LACE** | **Sync (Horizontal and Vertical Sync) Interlace**<br>0 = Non-interlace.<br>1 = Interlace. |
| [21] | **V_POL** | **V_POL (Vertical Polarity)**<br>0 = Low Active.<br>1 = High Active. |
| [20] | **H_POL** | **H_POL (Horizontal Polarity)**<br>0 = Low Active.<br>1 = High Active. |
| [19] | **FAL_D** | **FAL_D**<br>0 = Falling Latch Out.<br>1 = Rising Latch Out. |

| | | |
|---|---|---|
| [18:17] | **[LCD_ODD : SEL_ODD]** | **Control LCD Line Data Out**<br>00 = First line data is RGB, second line data is GBR.<br>01 = First line data is BGR, second line data is RBG.<br>10 = First line data is GBR, second line data is RGB.<br>11 = First line data is RBG, second line data is BGR. |
| [16] | **YUV2CCIR** | 0 = No operation.<br>1 = Convert full range YUV to CCIR601. |
| [15:8] | **LCD_DDA** | **Generate LCD Clock Frequency for TFT-ICD Panel**<br>**Note:** Only for DEVICE (DEVICE_CTRL[7:5]) “100” and “110”<br>Set LCD_DDA = 0 will disable DDA operation. |
| [7:5] | **DEVICE** | **DEVICE Setting**<br>000 = Packed YUV422.<br>001 = Packed YUV444.<br>100 = Sync-based TFT-LCD (UNIPAC).<br>101 = Sync-based TFT-LCD (SEIKO EPSON).<br>110 = Sync-based High-color TFT-LCD (RGB565/RGB666/RGB888).<br>111 = MPU-Interfaced LCD (RGB332/RGB444/RGB565/RGB666).<br>**Note1:** Device “000” is supported both in 8-bit and 16-bit data bus.<br>**Note2:** Device “110” is supported both in 16-bit and 18-bit. The 16-bit and 18-bit data bus is selected by RGB_SCALE (DEVICE_CTRL[25:24]).<br>**Note3:** Device “111” is supported in 8-bit, 9-bit, 16-bit, and 18-bit data bus.<br>**Note4:** The 8-bit/16-bit or 9-bit/18-bit data bus is selected by the combination of DBWORD (DEVICE_CTRL[26]), RGB_SCALE (DEVICE_CTRL[25:24]) and DE_POL (DEVICE_CTRL[28]). |

| | | |
|---|---|---|
| [4:3] | **RGB_SHIFT/ DM_262K (Share bit)** | **RGB Data Output Shift for Sync-type LCD Panel**<br>When DEVICE (DEVICE_CTRL[7:5]) = 100, 101.<br>00 = Not Shift.<br>01 = Shift One Cycle.<br>10 = Shift 2 Cycle.<br>11 = Not Defined.<br><br>RGB Data Output Arrangement for 262K MPU-Interface LCM.<br><br>When DEVICE (DEVICE_CTRL[7:5]) = 111, and 16-bit data bus mode (*denote don't care bit).<br>00 = RRRRRRGGGGGGBBBB, **********BB.<br>01 = **********RR, RRRRGGGGGGBBBBBB.<br>10 = RRRRRR**GGGGGG**, ********BBBBBB**.<br>11 = RRRRRR**GGGGGG**, BBBBBB**********.<br><br>When DEVICE (DEVICE_CTRL[7:5]) = 111, and 8-bit data bus mode.<br>*0 = RRRRRR**, GGGGGG**, BBBBBB**.<br>*1 = RRRRRRGG, GGGGBBBB, ******BB.<br><br>**ITU656 format select**<br>When DEVICE (DEVICE_CTRL[7:5]) = 000, ITU_EN (DCCS[15]) =1 and 8-bit data bus mode.<br>01 = NTSC.<br>10 = PAL. |
| [2-1] | **SWAP_YcbCr (share_bit DEVICE (DEVICE_CTRL[7:5]) = 000)** | **YUV Data Output Swap (for Packed YUV Mode)**<br>When DEVICE (DEVICE_CTRL[7:5]) = 000:.<br>00 = UYVY.<br>01 = YUYV.<br>10 = VYUY.<br>11 = YVYU. |
| [2] | **SWAP_YcbCr[1] (share_bit DEVICE (DEVICE_CTRL[7:5]) = 100)** | **Delay Control**<br>Make the cycle of reading data from FIFO to be delay one cycles per two pixel so the output rate is 1.5 cycles per pixel .When DEVICE (DEVICE_CTRL[7:5]) = 100, 8-bit data bus , SWAP_YcbCr[0] (DEVICE_CTRL[1]) = 0, and DBWORD (DEVICE[26]) = 1 (Pixel data read from FIFO is 1 cycle per pixel),.<br>=> Unipac sub-sampling one component from each RGB pixel.<br>0 = 1/3 subsampling, per pixel is subsampling a component.<br>Seq: R0G1B2,R3G4B5,….. output 1 cycles per pixel<br>For 960*240 panel, (source are expand from 320*240 to 960*240)<br>1= 1/2 subsampling, even pixel is subsampled two components and odd pixel is subsampled one components.<br>Seq: R0G0B1,R2G3B3,R4G4B5….. output 1.5 cycles per pixel<br>For 480*240 (source is 320*240) |

| | | |
|---|---|---|
| [1] | **SWAP_YcbCr[0] (share_bit DEVICE (DEVICE_CTRL[7:5]) = 100)** | **Read Cycles Per Pixel**<br>When DEVICE (DEVICE_CTRL[7:5]) = 100, 8-bit data bus, DBWORD (DEVICE[26]) = 0, and DEVICE_CTRL [SWAP_YcbCr[1]]=0,.<br>0 = Pixel data read from FIFO is 2 cycles per pixel,.<br>Seq: R0G0B1,R1G2B2,R3G3B,4R4….  2 cycles per pixel<br>For 640 *240 panel (source 320*240)<br>1= Pixel data read from FIFO is 3 cycles per pixel,.<br>Seq: R0G0B0,R1G1B1,R2G2B2,R3G3B3... 3 cycles per pixel<br>For 960*240 panel, (source is 320*240) |
| [2] | **SWAP_YCbCr[1] (share_bit DEVICE (DEVICE_CTRL[7:5]) = 110)** | **RGB_Dumy Format (for Sync-based High-color TFT-ICD -interfaced LCD)**<br>When DEVICE (DEVICE_CTRL[7:5]) = 110, 8-bit data bus & 16M-color mode, DBWORD (DEVICE[26]) = 0, SWAP_YCbCr[0] = x.<br>0 = RGB – output 8-bit data in the sequence of " R0G0B0R1G1B1….".,. 3 cycles per pixel.<br>1 = RGBX– output 8-bit data in the sequence of " R0G0B0XR1G1B1X….".,. 4 cycles per pixel. |
| [1] | **SWAP_YCbCr[0] (share_bit DEVICE (DEVICE_CTRL[7:5]) = 110)** | **RGB Data Output Swap (for 65536-color LCD & 262144-color LCD)**<br>When DEVICE (DEVICE_CTRL[7:5]) = 110, 8-bit data bus & 65536-color, 9-bit data bus & 256K-color mode and SWAP_YCbCr[1] = 0.<br>0 = the high-byte of 16-bit RGB565 pixel data is output first, the low-byte data is output secondly; the msb 9-bit of 18-bit RGB666 pixel data is output first, the LSB 9-bit data is output secondly.<br>1 = the low-byte of 16-bit RGB565 pixel data is output first, the low-byte data is output secondly;.<br>the LSB 9-bit of 18-bit RGB666 pixel data is output first, the msb 9-bit data is output secondly. |
| [2] | **SWAP_YCbCr[1] (share_bit DEVICE (DEVICE_CTRL[7:5]) = 111)** | **RGB Data Output Swap (for MPU-interfaced LCD)**<br>When DEVICE (DEVICE_CTRL[7:5]) = 111, 16-bit data bus & 4096-color mode.<br>0 = Data format of DDATA[15:0] is {R,G,B,4'h0} in 4096-color mode.<br>1 = Data format of DDATA[15:0] is {4'h0,R,G,B} in 4096-color mode. |
| [1] | **SWAP_YCbCr[0] (share_bit DEVICE (DEVICE_CTRL[7:5]) = 111)** | **RGB Data Output Swap (for 65536-color LCD & 262144-color LCD)**<br>When DEVICE (DEVICE_CTRL[7:5]) = 111, 8-bit data bus & 65536-color, 9-bit data bus & 256K-color mode.<br>0 = the high-byte of 16-bit RGB565 pixel data is output first, the low-byte data is output secondly; the msb 9-bit of 18-bit RGB666 pixel data is output first, the LSB 9-bit data is output secondly.<br>1 = the low-byte of 16-bit RGB565 pixel data is output first, the low-byte data is output secondly;.<br>the LSB 9-bit of 18-bit RGB666 pixel data is output first, the msb 9-bit data is output secondly. |
| [0] | **Reserved** | Reserved. |

### Data bus arrangement for different pixel and bus for MPU-Interface LCM

| Gray Scale Selection (RGB_SCALE) | Bus Interface Mode (DBWORD, IM_262K) | SWAP_YCbCr | Data Mode For 262K Panels (DM_262K) | Data Bus Arrangement ("*" Denote Don't Care Bit ) | Note |
|---|---|---|---|---|---|
| 12 bits/pixel (00) | 8 bits | ** | ** | RRRRGGGG<br>BBBBRRRR<br>GGGGBBBB | 3*fer/2pixels |
| | 16 bits | 0* | ** | RRRRGGGGBBBB**** | 1*fer/1pixel |

| | | | | | |
|---|---|---|---|---|---|
| | 16 bits | 1* | ** | ****RRRRGGGGBBBB | 1*fer/1pixel |
| 16 bits/pixel (01) | 8 bits | *0 | ** | RRRRRGGG<br>GGGBBBBB | 2*fer/1pixel |
| | 8 bits | *1 | ** | GGGBBBBB<br>RRRRRGGG | 2*fer/1pixel |
| | 16 bits | ** | ** | RRRRRGGGGGGBBBBB | 1*fer/1pixel |
| 18 bits/pixel (10) | 8 bits | ** | *0 | RRRRRR**<br>GGGGGG**<br>BBBBBB** | 3*fer/1pixel |
| | | ** | *1 | RRRRRRGG<br>GGGGBBBB<br>******BB | 3*fer/1pixel |
| | 9 bits | *0 | ** | RRRRRRGGG<br>GGGBBBBBB | 2*fer/1pixel |
| | 9 bits | *1 | ** | GGGBBBBBB<br>RRRRRRGGG | 2*fer/1pixel |
| | 16 bits | ** | 00 | RRRRRRGGGGGGBBBB<br>**********BB | 2*fer/1pixel |
| | | ** | 01 | **********RR<br>RRRRGGGGGGBBBBBB | 2*fer/1pixel |
| | | ** | 10 | RRRRRR**GGGGGG**<br>********BBBBBB** | 2*fer/1pixel |
| | | ** | 11 | RRRRRR**GGGGGG**<br>BBBBBB********** | 2*fer/1pixel |
| | 18 bits | | ** | RRRRRRGGGGGGBBBBBB | 1*fer/1pixel |

**Data bus arrangement for different pixel for Unipac - Interface LCM at 16 M colors and 8-bit data bus**

| **Gray Scale Selection** | **Bus Interface Mode (DBWORD)** | **SWAP_YCbCr** | **Control LCD Line Data Out [LCD_ODD : SEL_ODD]** | **Data Bus Arrangement (“*” Denote Don’t Care Bit ) First Line(Odd Line)** | **Data Bus Arrangement (“*” Denote Don’t Care Bit ) Secondt Line(Even Line)** | **Note** |
|---|---|---|---|---|---|---|
| 24 bits/pixel | 1 | 00 | 00 | RRRRRRRR0 (pixel 0)<br>GGGGGGGG1 (pixel 1)<br>BBBBBBBB2 (pixel 2) | GGGGGGGG0 (pixel 0)<br>BBBBBBBB1 (pixel 1)<br>RRRRRRRR2 (pixel 2) | 1*fer/1pixel |
| | 1 | 00 | 01 | BBBBBBBB0 (pixel 0)<br>GGGGGGGG1 (pixel 1)<br>RRRRRRRR2 (pixel 2) | RRRRRRRR0 (pixel 0)<br>BBBBBBBB1 (pixel 1)<br>GGGGGGGG2 (pixel 2) | 1*fer/1pixel |
| | 1 | 00 | 10 | GGGGGGGG0 (pixel 0) | RRRRRRRR0 (pixel 0) | 1*fer/1pixel |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | BBBBBBBB1 (pixel 1)<br>RRRRRRRR2 (pixel 2) | GGGGGGGG1 (pixel 1)<br>BBBBBBBB2 (pixel 2) | |
| | 1 | 00 | 11 | RRRRRRRR0 (pixel 0)<br>BBBBBBBB1 (pixel 1)<br>GGGGGGGG2 (pixel 2) | BBBBBBBB0 (pixel 0)<br>GGGGGGGG1 (pixel 1)<br>RRRRRRRR2 (pixel 2) | 1*fer/1pixel |
| | 1 | 10 | 00 | RRRRRRRR0 (pixel 0)<br>GGGGGGGG0 (pixel 0)<br>BBBBBBBB1 (pixel 2) | GGGGGGGG0 (pixel 0)<br>BBBBBBBB0 (pixel 0)<br>RRRRRRRR1 (pixel 1) | 1.5*fer / 1pixel |
| | 1 | 10 | 01 | BBBBBBBB0 (pixel 0)<br>GGGGGGGG0 (pixel 0)<br>RRRRRRRR1 (pixel 1)<br>BBBBBBBB2 (pixel 2)<br>GGGGGGGG2 (pixel 2)<br>RRRRRRRR3 (pixel 3) | RRRRRRRR0 (pixel 0)<br>BBBBBBBB0 (pixel 0)<br>GGGGGGGG1 (pixel 1)<br>RRRRRRRR2 (pixel 2)<br>BBBBBBBB2 (pixel 2)<br>GGGGGGGG3 (pixel 3) | 1.5*fer / 1pixel |
| | 1 | 10 | 10 | GGGGGGGG0 (pixel 0)<br>BBBBBBBB0 (pixel 0)<br>RRRRRRRR1 (pixel 1) | RRRRRRRR0 (pixel 0)<br>GGGGGGGG0 (pixel 0)<br>BBBBBBBB1 (pixel 1) | 1.5*fer / 1pixel |
| | 1 | 10 | 11 | RRRRRRRR0 (pixel 0)<br>BBBBBBBB0 (pixel 0)<br>GGGGGGGG1 (pixel 1) | BBBBBBBB0 (pixel 0)<br>GGGGGGGG0 (pixel 0)<br>RRRRRRRR1 (pixel 1) | 1.5*fer / 1pixel |
| | 0 | 00 | 00 | RRRRRRRR0 (pixel 0)<br>GGGGGGGG0 (pixel 0)<br>BBBBBBBB1 (pixel 1) | GGGGGGGG0 (pixel 0)<br>BBBBBBBB0 (pixel 0)<br>RRRRRRRR1 (pixel 1) | 2*fer/1pixel |
| | 0 | 00 | 01 | BBBBBBBB0 (pixel 0)<br>GGGGGGGG0 (pixel 0)<br>RRRRRRRR1 (pixel 1) | RRRRRRRR0 (pixel 0)<br>BBBBBBBB0 (pixel 0)<br>GGGGGGGG1 (pixel 1) | 2*fer/1pixel |
| | 0 | 00 | 10 | GGGGGGGG0 (pixel 0)<br>BBBBBBBB0 (pixel 0)<br>RRRRRRRR1 (pixel 1) | RRRRRRRR0 (pixel 0)<br>GGGGGGGG0 (pixel 0)<br>BBBBBBBB1 (pixel 1) | 2*fer/1pixel |
| | 0 | 00 | 11 | RRRRRRRR0 (pixel 0)<br>BBBBBBBB0 (pixel 0)<br>GGGGGGGG1 (pixel 1) | BBBBBBBB0 (pixel 0)<br>GGGGGGGG0 (pixel 0)<br>RRRRRRRR1 (pixel 1) | 2*fer/1pixel |
| | 0 | 01 | 00 | RRRRRRRR0 (pixel 0)<br>GGGGGGGG0 (pixel 0)<br>BBBBBBBB0 (pixel 0) | GGGGGGGG0 (pixel 0)<br>BBBBBBBB0 (pixel 0)<br>RRRRRRRR0 (pixel 0) | 3*fer/1pixel |
| | 0 | 01 | 01 | BBBBBBBB0 (pixel 0)<br>GGGGGGGG0 (pixel 0)<br>RRRRRRRR0 (pixel 0) | RRRRRRRR0 (pixel 0)<br>BBBBBBBB0 (pixel 0)<br>GGGGGGGG0 (pixel 0) | 3*fer/1pixel |
| | 0 | 01 | 10 | GGGGGGGG0 (pixel 0)<br>BBBBBBBB0 (pixel 0)<br>RRRRRRRR0 (pixel 0) | RRRRRRRR0 (pixel 0)<br>GGGGGGGG0 (pixel 0)<br>BBBBBBBB0 (pixel 0) | 3*fer/1pixel |
| | 0 | 01 | 11 | RRRRRRRR0 (pixel 0) | BBBBBBBB0 (pixel 0) | 3*fer/1pixel |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | BBBBBBBB0 (pixel 0)<br>GGGGGGGG0 (pixel 0) | GGGGGGGG0 (pixel 0)<br>RRRRRRRR0 (pixel 0) | |
| | | | | | | |

**Data bus arrangement for different pixel for TFT High colors device - Interface LCM at 16 M colors and 8-bit data bus**

| Gray Scale Selection | Bus Interface Mode (DBWORD) | SWAP_YCbCr | Control LCD Line Data Out [LCD_ODD : SEL_ODD] | Data Bus Arrangement (“*” Denote Don’t Care Bit ) | | Note |
|---|---|---|---|---|---|---|
| 24 bits data bus | 1 | 00 | ** | RRRRRRRRGGGGGGGGBBBBBBBB (pixel 0) | | 1*fer/1pixel |
| 8 bits data bus | 0 | 00 | ** | BBBBBBBB0 (pixel 0)<br>GGGGGGGG0 (pixel 0)<br>RRRRRRRR0 (pixel 0)<br>BBBBBBBB1 (pixel 1)<br>GGGGGGGG1 (pixel 1)<br>RRRRRRRR1 (pixel 1) | | 3*fer/1pixel |
| | 0 | 10 | ** | BBBBBBBB0 (pixel 0)<br>GGGGGGGG0 (pixel 0)<br>RRRRRRRR0 (pixel 0)<br>******** 0 (pixel 0)<br>BBBBBBBB1 (pixel 1)<br>GGGGGGGG1 (pixel 1)<br>RRRRRRRR1 (pixel 1)<br>******** 1 (pixel 1) | | 4*fer/1pixel |

## MPU-Interfaced LCD Write Command Register (MPULCD_CMD)

When DEVICE (DEVICE_CTRL[7:5]) = 111, a 16-bit value represents MPU-interfaced LCD command/parameter data. For 8-bit data bus or 16-bit data bus with 8-bit command mode, the MPULCD_CMD[15:8] is discarded. When writing data to this register (MPULCD_CMD[7:0]), Display Controller will switch to command mode and write this data to LCM if MCD_ON (DCCS[5]) is enabled and Display Controller is not outputting display data. You can read HACT (DCCS[27]) and VACT (DCCS[26]) to get display status.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **MPULCD_CMD** | LCM_BA + 0x08 | R/W | MPU-Interface LCD Write Command Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31] | **CMD_BUSY** | **Command Interface Is Busy**<br>0 = Command interface is ready for next command.<br>1 = Command interface is busy for writing/reading pending command. |
| [30] | **WR_RS** | **Write/Read RS Setting**<br>0 = Output pin RS = 0 when sending command/parameter via MPULCD_CMD.<br>1 = Output pin RS = 1 when sending command/parameter via MPULCD_CMD. |
| [30] | **READ** | **Read Status or Data**<br>0 = Write command/parameter LCM.<br>1 = Read status/data from LCM.<br>**Note:** Data will be stored in MPULCD_CMD[17:0], when CMD_BUSY is inactive after read operation. |
| [29:18] | **Reserved** | Reserved. |
| [17:16] | **MPULCD_CMD** | **MPU-interfaced LCD Read Data (Read Only)** |
| [15:0] | **MPULCD_CMD** | **MPU-interfaced LCD Command/Parameter Data, Read Data** |

## Interrupt Control/Status Register (INT_CS)

Interrupts are the communication method for Display Controller-initiated communication with the Display Controller Driver. There are several events that may trigger an interrupt from the Display Controller. Each specific event sets a specific bit in the INT_CS register. The Display Controller requests an interrupt when all three of the following conditions are met:

- The **DISP_INT_EN** bit in *DCCS* is set to '1'.
- A status bit in *INT_CS* is set to '1'.
- The corresponding enable bit in *INT_CS* for the *Status* bit is set to '1'.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| INT_CS | LCM_BA + 0x0C | R/W | Interrupt Control/Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| DISP_F_INT | DISP_F_STATUS | UNDERRUN_INT | BUS_ERROR_INT | 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 | | | | | | UNDERRUN_EN | DISP_F_EN |

| Bits | Description | |
|---|---|---|
| [31] | **DISP_F_INT** | **Frame Display Complete Interrupt Status (Write-clear)**<br>When write "1" value on the bit, the interrupt will be cleared.<br>(This status bit can be internally written no matter DISP_F_EN is enable or not) |
| [30] | **DISP_F_STATUS** | **Frame Display Complete Internal Status (2), (Write-clear)**<br>When write "1" value on the bit, it will be cleared.<br>(This status bit can be internally written only if DISP_F_EN is enable)<br>**Note:** The interrupt status can be programmed for indicating frame displayed complete or field displayed complete by FIELD_INTR (DCCS[6]). |
| [29] | **UNDERRUN_INT** | **FIFO Under-run Interrupt Status (Write-clear)**<br>When write "1" value on the bit, the interrupt will be cleared. |
| [28] | **BUS_ERROR_INT** | **Bus Error Interrupt (Write-clear)**<br>When DMA bus master receive an error response from slaves, this bit will be set.<br>When write "1" value on the bit, the interrupt will be cleared.<br>**Note:** This interrupt is always enabled. |
| [27:2] | **Reserved** | Reserved. |
| [1] | **UNDERRUN_EN** | **FIFO Under-run Interrupt Enable** |

| | | |
|---|---|---|
| | | 0 = Disable.<br>1 = Enable. |
| [0] | **DISP_F_EN** | **Frame Display Complete Interrupt Enable**<br>0 = Disable.<br>1 = Enable. |

## CRTC Display Size Register (CRTC_SIZE)

This register controls the display size. It includes Horizontal-Total (HTT) and Vertical-Total (VTT) registers. The value of HTT specifies the total number of pixels in the CRTC horizontal scan line interval including retrace time. And the value of VTT specifies the total number of scan line for each field (frame), including the retrace time.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CRTC_SIZE | LCM_BA + 0x10 | R/W | CRTC Display Size Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **VTT** | **CRTC Vertical Total Scan Lines**<br>An 11-bit value specifies the total number of scan line for each field, including the retrace time. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **HTT** | **CRTC Horizontal Total Pixels**<br>An 11-bit value specifies the total number of pixels in the CRTC horizontal scan line interval including the retrace time. |

## CRTC Display Enable End Register (CRTC_DEND)

This register controls the actual display size of the output device. It includes HDEND and VDEND registers. The value of HDEND specifies the total number of displayed pixels for a scan line. And the value of VDEND specifies the total number of displayed scan line for each field (frame).

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRTC_DEND** | LCM_BA + 0x14 | R/W | CRTC Display Enable End Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **VDEND** | **CRTC Vertical Display Enable End**<br>An 11-bit value specifies the total number of displayed scan line for each field. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **HDEND** | **CRTC Horizontal Display Enable End**<br>An 11-bit value specifies the total number of displayed pixels for scan line. |

## CRTC Internal Horizontal Retrace Timing Register (CRTC_HR)

The internal horizontal retrace timing can be controlled by properly setting the values of retrace starting (HRS) and ending (HRE) registers included in this register. The values are programmed in number of pixels.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRTC_HR** | LCM_BA + 0x18 | R/W | CRTC Internal Horizontal Retrace Timing Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **HRE** | **CRTC Internal Horizontal Retrace End Low**<br>An 11-bit value programmed in pixels, at which the Internal Horizontal Retrace becomes inactive. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **HRS** | **CRTC Internal Horizontal Retrace Start Timing**<br>An 11-bit value programmed in pixels, at which the Internal Horizontal Retrace becomes active. |

## CRTC Horizontal Sync Timing Register (CRTC_HSYNC)

The horizontal sync timing can be controlled by properly setting the values of starting (HSYNC_S) and ending (HSYNC_E) registers included in this register. The values are programmed in numbers of pixel.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CRTC_HSYNC | LCM_BA + 0x1C | R/W | CRTC Horizontal Sync Timing Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| HSYNC_SHIFT [1:0] | | Reserved | | | HSYNC_E[10:8] | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HSYNC_E[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | HSYNC_S[10:8] | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HSYNC_S[7:0] | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:30] | **HSYNC_SHIFT** | **Hsync Signal Adjustment for Multi-cycles Per Pixel Mode of Sync-based Unipac-ICD**<br>When DEVICE (DEVICE_CTRL[7:5]) = 100,8-bit data bus, DBWORD (DEVICE[26]) = 0, and SWAP_YcbCr[1] (DEVICE_CTRL[2]) =0,.<br>If SWAP_YcbCr[0] (DEVICE_CTRL[1]) = 0, it means that 2 cycles per pel, so hsync's hrs would move three cycles if hrs added or substracted 1.In order to adjust hsync signal in pclk unit, RGB_SHIFT shared register aids this function in -LCD with 8 bits data bus mode.<br>00 = Hsync will not move.<br>01 = Hsync will left move 1 pclk cycle.<br>If SWAP_YcbCr[0] (DEVICE_CTRL[1]) = 1, it means that 3 cycles per pel, so hsync's hrs would move three cycles if hrs added or substracted 1.In order to adjust hsync signal in pclk unit, RGB_SHIFT shared register aids this function in High-color TFT-LCD with 8 bits data bus mode.<br>00 = Hsync will not move.<br>01 = Hsync will left move 1 pclk cycle.<br>10 = Hsync will left move 2 pclk cycle.<br>Hsync singal adjustment for multi-cycles per pixel mode of Sync-based High-color TFT-LCD.<br>When DEVICE (DEVICE_CTRL[7:5]) = 110 , RGB_SCALE (DEVICE_CTRL[25:24]) =3(16M-color mode) and DBWORD (DEVICE[26]) = 0,.<br>If SWAP_YcbCr[0] (DEVICE_CTRL[1]) = 0, it means that 3 cycles per pel, so hsync's hrs would move three cycles if hrs added or substracted 1.In order to adjust hsync signal in pclk unit, RGB_SHIFT shared register aids this function in High-color TFT-LCD with 8 bits data bus mode.<br>00 = Hsync will not move.<br>01 = Hsync will left move 1 pclk cycle. |

| | | |
|---|---|---|
| | | 10 = Hsync will left move 2 pclk cycle.<br>If SWAP_YcbCr[0] (DEVICE_CTRL[1]) = 1, it means that 4 cycles per pixel, so hsync's hrs would move three cycles if hrs added or substracted 1. In order to adjust hsync signal in pclk unit, RGB_SHIFT shared register aids this function in High-color. TFT-LCD with 8-bit data bus mode.<br>00 = Hsync will not move.<br>01 = Hsync will left move 1 pclk cycle.<br>10 = Hsync will left move 2 pclk cycle.<br>11 = Hsync will left move 3 pclk cycle. |
| [29:27] | **Reserved** | Reserved. |
| [26:16] | **HSYNC_E** | **CRTC Horizontal Sync End Timing**<br>An 11-bit value programmed in pixels, at which the Horizontal Sync Signal becomes inactive. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **HSYNC_S** | **CRTC Horizontal Sync Start Timing**<br>An 11-bit value programmed in pixels, at which the Horizontal Sync signal becomes active. |

## CRTC Internal Vertical Retrace Timing Register (CRTC_VR)

The vertical retrace timing can be controlled by properly setting the values of starting (VRS) and ending (VRE) registers included in this register. The values are programmed in numbers of scan-line.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CRTC_VR** | LCM_BA + 0x20 | R/W | CRTC Internal Vertical Retrace Timing Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **VRE** | **CRTC Vertical Internal Retrace End Low**<br>An 11-bit value is programmed in number of scan line, at which the internal vertical retrace becomes inactive. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **VRS** | **CRTC Vertical Internal Retrace Start Timing**<br>An 11-bit value is programmed in number of scan line, at which the internal vertical retrace becomes active. |

### Video Stream Frame Buffer-0 Starting Address Register (VA_BADDR0)

The value of this register represents the starting memory address of the frame buffer-0 for video data stream.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **VA_BADDR0** | LCM_BA + 0x24 | R/W | Video Stream Frame Buffer-0 Starting Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **VA_BADDR0[31:24]** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **VA_BADDR0[23:16]** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **VA_BADDR0[15:8]** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **VA_BADDR0[7:0]** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:0] | **VA_BADDR0** | **Starting Memory Address of the Frame Buffer-0 for Video Data Stream**<br>The value of this register represents the starting memory address of the frame buffer-0 for video data stream. |

## Video Stream Frame Buffer-1 Starting Address Register (VA_BADDR1)

The value of this register represents the starting memory address of the frame buffer-1 for video data stream.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **VA_BADDR1** | LCM_BA + 0x28 | R/W | Video Stream Frame Buffer-1 Starting Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **VA_BADDR1[31:24]** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **VA_BADDR1[23:16]** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **VA_BADDR1[15:8]** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **VA_BADDR1[7:0]** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:0] | **VA_BADDR1** | **Starting Memory Address of the Frame Buffer-1 for Video Data Stream**<br>The value of this register represents the starting memory address of the frame buffer-1 for video data stream. |

## Video Stream Frame Buffer Control Register (VA_FBCTRL)

The information contained in this register is used to efficiently control the frame buffer operation. The VA_STRIDE shows the word offset of memory address between two vertically adjacent lines. The VA_FF specifies the number of WORD SDRAM access cycle for a horizontal scan line fetching. The data buffer FIFO is divided into two or four regions depending on the value of the IO_REGION_HALF. If IO_REGION_HALF is not asserted, there are four regions of 8 words each. If IO_REGION_HALF is asserted, there are two regions of 16 words each. The size of the region affects the AHB burst transfer size. There are two pointers into the FIFO: one for the display engine data and one for the AHB data. These pointers are maintained in the Data Buffer Control module. When the pointers are not in the same region, an AHB burst cycle is issued to read or write the data in the region pointed to by the AHB pointer.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| VA_FBCTRL | LCM_BA + 0x2C | R/W | Video Stream Frame Buffer Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| DB_EN | START_BUF | FIELD_DUAL | O_REGION_HAL | Reserved | VA_FF[10:8] | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| VA_FF[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | Reserved | | | | VA_STRIDE[10:8] | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| VA_STRIDE[7:0] | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31] | **DB_EN** | **Dual Buffer Switch Enable**<br>0 = Dual buffer switch disable, Always fetch data from address which START_BUF indicated.<br>1 = Dual buffer switch enable. Switch starting address between VA_BADDR0 and VA_BADDR1 at each frame/field starts (controlled by FIELD_DUAL). The first display frame/field address is controlled by START_BUF. |
| [30] | **START_BUF** | **Starting Buffer of Dual-buffer**<br>0 = Starting fetch data from VA_BADDR0.<br>1 = Starting fetch data from VA_BADDR1. |
| [29] | **FIELD_DUAL** | **Dual-buffer Switch Control**<br>0 = Switch dual-buffer before each frame starts.<br>1 = Switch dual-buffer before each field starts. |
| [28] | **IO_REGION_HALF** | **Data Buffer Region Size**<br>0 = 8 words/region.<br>1 = 16 words/region.<br>**Note:** Both VA and OSD FIFO are controlled by this bit. |
| [27] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [26:16] | **VA_FF** | **Video Stream Fetch Finish**<br>An 11-bit value specifies the number of WORD SDRAM access cycle for a horizontal scan line fetching of video data stream. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **VA_STRIDE** | **Video Stream Frame Buffer Stride**<br>An 11-bit value specifies the word offset of memory address of vertically adjacent line for video stream. |

### Video Stream Scaling Control Register (VA_SCALE)

This register control the video up-scaling factors, both horizontal and vertical up-scaling ratios are ranging from 1.0 to 7.99999 in fractional steps. There are two modes of horizontal up-scaling, interpolation and duplication, which can be controlled by setting XCOPY.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| VA_SCALE | LCM_BA + 0x30 | R/W | Video Stream Scaling Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | VA_SCALE_V[12:8] | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| VA_SCALE_V[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| XCOPY | Reserved | | VA_SCALE_H[12:8] | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| VA_SCALE_H[7:0] | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:29] | **Reserved** | Reserved. |
| [28:16] | **VA_SCALE_V** | **Video Vertical Scaling Control**<br>A 13-bit value specifies the vertical scaling factor of 1.0~7.99999. Bit 12-10 specify the integral part and bits 9-0 specifies the decimal part of the scaling factor. |
| [15] | **XCOPY** | **Video Stream Horizontal Up-scaling Mode**<br>0 = Interpolation.<br>1 = Duplication. |
| [14:13] | **Reserved** | Reserved. |
| [12:0] | **VA_SCALE_H** | **Video Horizontal Scaling Control**<br>A 13-bit value specifies the horizontal scaling factor of 1.0~7.99999. Bit 12-10 specifies the integral part and bits 9-0 specifies the decimal part of the scaling factor. |

## Test Mode Control Register (VA_TEST)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| VA_TEST | LCM_BA + 0x34 | R/W | Test Mode Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| SIGN_DATA [15:8] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| SIGN_DATA [7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | BistFail | | BistFinish | BistEn |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SIGN_BUSY | Reserved | TEST_GARY | SIGN_SELF | SIGN_PRESET | SIGN_SRC | | SIGN_ON |

| Bits | Description | |
|---|---|---|
| [31:16] | **SIGN_DATA** | **A 16-bit Value Specifies Signature Data (Read Only)** |
| [15:12] | **Reserved** | Reserved. |
| [11:10] | **BistFail** | The BistFail indicates if the BIST fails or succeeds If the value of BistFail is 2'b00 at the end, the embedded SRAM pass the BIST test, otherwise, it is faulty. |
| [9] | **BistFinish** | The Finish indicates the end of the BIST operation When BIST controller finishes all operations, this bit will be set to one Clearing BistEn also cause this bit to be reset. |
| [8] | **BistEn** | The BistEn is used to enable the BIST operations. |
| [7] | **SIGN_BUSY** | **Signature Analyzer Status (Read Only)**<br>0 = Signature analyzer Idle.<br>1 = Signature analyzer Busy. |
| [6] | **Reserved** | Reserved. |
| [5] | **TEST_GARY** | It is necessary to fix at "0" for normal operation. |
| [4] | **SIGN_SELF** | **Signature Analyzer Self Test**<br>0 = Disable.<br>1 = Enable. |
| [3] | **SIGN_PRESET** | **Preset Signature Analyzer Operation**<br>0 = Preset signature Value when analysis begins.<br>1 = Do not preset. |
| [2:1] | **SIGN_SRC** | **Signature Analyzer Data Source Select**<br>00 = Null (Zero Input).<br>01 = R data.<br>10 = G data.<br>11 = B data. |

| | | |
|---|---|---|
| [0] | **SIGN_ON** | **Signature Analyzer**<br>0 = Disable.<br>1 = Start signature analyzer operation. |

### Video Stream Active Window Coordinates Register (VA_WIN)

This pair of registers (VA_WYS, VA_WYE) specifies the area which video stream will occupy in the screen. It is called Active window for Vide Stream. The pixels outside the active window will be filled with the color specified by VA_STUFF. When the value of VA_WYE is greater than VDEND (CRTC_DEND[26:16]), the Active window will be actually ended at VDEND (CRTC_DEND[26:16]).

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **VA_WIN** | LCM_BA + 0x38 | R/W | Video Stream Active Window Coordinates Register | 0x0001_07FF |

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

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **VA_WYS** | **VA Active Window Y-start**<br>An 11-bit value specifies the vertical starting scan line of the Active VA window. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **VA_WYE** | **VA Active Window Y-end**<br>An 11-bit value specifies the last vertical scan line of the Active VA window. |

### Video Stream Stuff Register (VA_STUFF)

A 24-bit value specifies stuff pattern for non-active window area in Video Stream.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| VA_STUFF | LCM_BA + 0x3C | R/W | Video Stream Stuff Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:16] | **VA_STUFF [23:16]** | The 8 higher-order bits are used for Y or R component according to the source color format. |
| [15:8] | **VA_STUFF [15:8]** | The 8 middle-order bits are used for U or G component according to the source color format. |
| [7:0] | **VA_STUFF [7:0]** | The 8 lower-order bits are used for V or B component according to the source color format. |

## OSD Window Starting Coordinates Register (OSD_WINS)

The starting coordinates of the OSD window is specified in this register. Two values form the coordinates; they are the horizontal starting pixel (OSD_WXS) and the vertical starting scan line (OSD_WYS).

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **OSD_WINS** | LCM_BA + 0x40 | R/W | OSD Window Starting Coordinates Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **OSD_WYS** | **OSD Window Y-start**<br>An 11-bit value specifies the vertical starting scan line of the OSD window. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **OSD_WXS** | **OSD Window X-start**<br>An 11-bit value specifies the horizontal starting pixel position of the OSD window. |

## OSD Window Ending Coordinates Register (OSD_WINE)

The ending coordinates of the OSD window is specified in this register. Two values form the coordinates; they are the last horizontal pixel (OSD_WXE) and the last vertical scan-line (OSD_WYS).

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **OSD_WINE** | LCM_BA + 0x44 | R/W | OSD Window Ending Coordinates Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **OSD_WYE** | **OSD Window Y-end**<br>An 11-bit value specifies the last vertical scan line of the OSD window. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **OSD_WXE** | **OSD Window X-end**<br>An 11-bit value specifies the last horizontal pixel position of the OSD window. |

### OSD Stream Frame Buffer Starting Address Register (OSD_BADDR)

The value of this register represents the starting memory address of the frame buffer for OSD data stream.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **OSD_BADDR** | LCM_BA + 0x48 | R/W | OSD Stream Frame Buffer Starting Address Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **OSD_BADDR [31:24]** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **OSD _BADDR [23:16]** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **OSD _BADDR [15:8]** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **OSD _BADDR [7:0]** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:0] | **OSD_BADDR** | **Starting Memory Address of the Frame Buffer for OSD Data Stream**<br>The value of this register represents the starting memory address of the frame buffer for OSD data stream. |

## OSD Stream Frame Buffer Control Register (OSD_FBCTRL)

The information contained in this register is used to efficiently control the frame buffer operation. The OSD_STRIDE shows the word offset of memory address between two vertically adjacent lines. The OSD_FF specifies the number of WORD SDRAM access cycle for a horizontal scan line fetching.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **OSD_FBCTRL** | LCM_BA + 0x4C | R/W | OSD Stream Frame Buffer Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | **OSD_FF[10:8]** | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **OSD_FF[7:0]** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | **OSD _STRIDE[10:8]** | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **OSD _STRIDE[7:0]** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **OSD_FF** | **OSD Stream Fetch Finish**<br>An 11-bit value specifies the number of WORD SDRAM access cycle for a horizontal scan line fetching of OSD data stream. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **OSD_STRIDE** | **OSD Stream Frame Buffer Stride**<br>An 11-bit value specifies the word offset of memory address of vertically adjacent line for OSD stream. |

## OSD Overlay Control Register (OSD_OVERLAY)

Setting this register can control the display effect of the overlay area. It can be periodic blanking, VA and OSD data mixing, and VA or OSD data alone.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| OSD_OVERLAY | LCM_BA + 0x50 | R/W | OSD Overlay Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BLINK_VCNT[7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | BLI_ON | CKEY_ON |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | VA_SYNW[2:0] | | | OCR1[1:0] | | OCR0[1:0] | |

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:16] | **BLINK_VCNT** | **OSD Blinking Cycle Time**<br>An 8-bit value specifies the OSD blinking cycle time (unit : Vsync). |
| [15:10] | **Reserved** | Reserved. |
| [9] | **BLI_ON** | **OSD Blinking Control**<br>0 = Blinking Disable.<br>1 = Blinking Enable.<br>**Note:** Blinking control mode share the same color-key pattern registers with color-key control mode |
| [8] | **CKEY_ON** | **OSD Color-key Control**<br>0 = Color-Key Disable.<br>1 = Color-Key Enable. |
| [7] | **Reserved** | Reserved. |
| [6:4] | **VA_SYNW** | **Synthesis Video Weighting**<br>000 = Synthesized Video = Video; otherwise.<br>Synthesized Video = ((Video * VA_SYNW) + (OSD * (8-VA_SYNW))) / 8. |
| [3:2] | **OCR1** | **Video/OSD Overlay Control 1**<br>When VA_EN (DCCS[1]) = 1, OSD_EN (DCCS[2]) = 1, CKEY_ON (OSD_OVERLAY[8]) = 1, Display region within OSD window, color-key condition match,.<br>00 = Display Video data.<br>01 = Display OSD data.<br>10 = Display synthesized (Video + OSD) data. |

| | | |
|---|---|---|
| | | 11 = Reserved. |
| [1:0] | **OCR0** | **Video/OSD Overlay Control 0**<br>When VA_EN (DCCS[1]) = 1, OSD_EN (DCCS[2]) = 1, CKEY_ON (OSD_OVERLAY[8]) = 1, Display region within OSD window, color-key condition un-match,.<br>00 = Display Video data.<br>01 = Display OSD data.<br>10 = Display synthesized (Video + OSD) data.<br>11 = Reserved. |

## OSD Overlay Color-Key Pattern Register (OSD_CKEY)

A 24-bit value specifies OSD color-key pattern. When 24-bit OSD data is equal to the specified pattern data, the color-key condition is matched.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **OSD_CKEY** | LCM_BA + 0x54 | R/W | OSD Overlay Color-Key Pattern Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:16] | **OSD_CKEY[23:16]** | The 8 higher-order bits are used for OSD data comparing of Y or R component according to the source color format. |
| [15:8] | **OSD_CKEY[15:8]** | The 8 middle-order bits are used for OSD data comparing of U or G component according to the source color format. |
| [7:0] | **OSD_CKEY[7:0]** | The 8 lower-order bits are used for OSD data comparing of V or B component according to the source color format. |

### OSD Overlay Color-Key Mask Register (OSD_CMASK)

A 24-bit value serves as the mask of OSD color-key pattern comparing. The OSD data only compare with the color-key pattern where the mask bits are set as 1.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **OSD_CMASK** | LCM_BA + 0x58 | R/W | OSD Overlay Color-Key Mask Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:16] | **OSD_CMASK[23:16]** | The 8 higher-order bits are used for pattern mask of Y or R component according to the source color format. |
| [15:8] | **OSD_CMASK[15:8]** | The 8 middle-order bits are used for pattern mask of U or G component according to the source color format. |
| [7:0] | **OSD_CMASK[7:0]** | The 8 lower-order bits are used for pattern mask of V or B component according to the source color format. |

## OSD Window Skip1 Register (OSD_SKIP1)

This register is used to separate OSD into two sub-windows. The OSD-data fetching of lines enclosed by this skip region in OSD window is skipped. The value of starting-Y address (OSK_SKIP1_YS) can't be the same with OSD_WYS. The minimum value of it is OSD_WYS+1. In interlace mode, the minimum value of ending address (OSK_SKIP1_YE) is OSD_SKIP1_YS +1.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **OSD_SKIP1** | LCM_BA + 0x5C | R/W | OSD Window Skip1 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | **OSD_SKIP1_YS[10:8]** | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **OSD_SKIP1_YS[7:0]** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | **OSD_SKIP1_YE[10:8]** | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **OSD_SKIP1_YE[7:0]** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **OSD_SKIP1_YS** | **OSD Window Skip1 Y-start**<br>An 11-bit value specifies the first vertical scan line of the OSD skip1 window. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **OSD_SKIP1_YE** | **OSD Window X-end**<br>An 11-bit value specifies the last vertical scan line of the OSD skip1 window. |

## OSD Window Skip2 Register (OSD_SKIP2)

This register is used to separate OSD into two sub-windows. The OSD-data fetching of lines enclosed by this skip region in OSD window is skipped. The value of starting-Y address (OSK_SKIP2_YS) can't be the same with OSD_WYS. The minimum value of it is OSD_WYS+1. In interlace mode, the minimum value of ending address (OSK_SKIP2_YE) is OSD_SKIP2_YS +1.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **OSD_SKIP2** | LCM_BA + 0x60 | R/W | OSD Window Skip2 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | **OSD_SKIP2_YS[10:8]** | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **OSD_SKIP2_YS[7:0]** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | **OSD_SKIP2_YE[10:8]** | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **OSD_SKIP2_YE[7:0]** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **OSD_SKIP2_YS** | **OSD Window SKIP2 Y-start**<br>An 11-bit value specifies the first vertical scan line of the OSD SKIP2 window. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **OSD_SKIP2_YE** | **OSD Window X-end**<br>An 11-bit value specifies the last vertical scan line of the OSD SKIP2 window. |

### OSD Scaling Control Register (OSD_SCALE)

This register control the OSD up-scaling factors, the horizontal up-scaling ratios are ranging from 1.0 to 7.99999 in fractional steps. There is only one mode of horizontal up-scaling by duplication.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **OSD_SCALE** | LCM_BA + 0x64 | R/W | OSD Scaling 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** | | **OSD_SCALE_H[12:8]** | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **OSD_SCALE_H[7:0]** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:13] | **Reserved** | Reserved. |
| [12:0] | **OSD_SCALE_H** | **OSD Scaling Setting** |

## Hardware Cursor Control Register (HC_CTRL)

This register is used to control the modes of hardware cursor. (HC_TIP_X, HC_TIP_Y) specifies which the cursor's tip is located at on hardware cursor block.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| HC_CTRL | LCM_BA + 0x6C | R/W | Hardware Cursor Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | HC_TIP_Y [5:0] | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | HC_TIP_X [5:0] | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | HC_MODE[2:0] | | |

| Bits | Description | |
|---|---|---|
| [31:22] | **Reserved** | Reserved. |
| [21:16] | **HC_TIP_Y** | Y position of Hardware cursor picture's tip on hardware cursor bit map. |
| [15:14] | **Reserved** | Reserved. |
| [13:8] | **HC_TIP_X** | X position of Hardware cursor picture's tip on hardware cursor bit map. |
| [7:3] | **Reserved** | Reserved. |
| [2:0] | **HC_MODE** | **Hardware Cursor Mode Setting**<br>0 = 32*32*2bpp – 4 color mode.<br>1 = 32*32*2bpp – 3 color mode and transparency mode.<br>2 = 64*64*2bpp – 4 color mode.<br>3 = 64*64*2bpp – 3 color mode and transparency mode.<br>4 = 128*128*1bpp – 2 color mode.<br>5 = 128*128*1bpp – 1 color mode and transparency mode. |

## Hardware Cursor Position Register (HC_POS)

This register is used to control the position of hardware cursor coordinate on va picture. (HC_X, HC_Y) can be changed dynamically at by software setting.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| HC_POS | LCM_BA + 0x70 | R/W | Hardware Cursor Position Register | 0x0000_0000 |

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

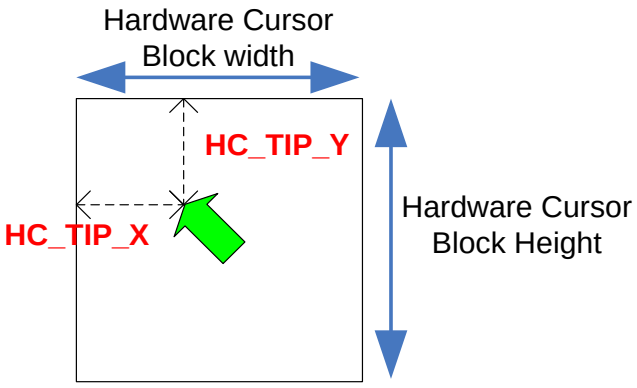
| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **HC_Y** | Y position of hardware cursor's tip on va picture. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **HC_X** | X position of hardware cursor's tip on va picture. |

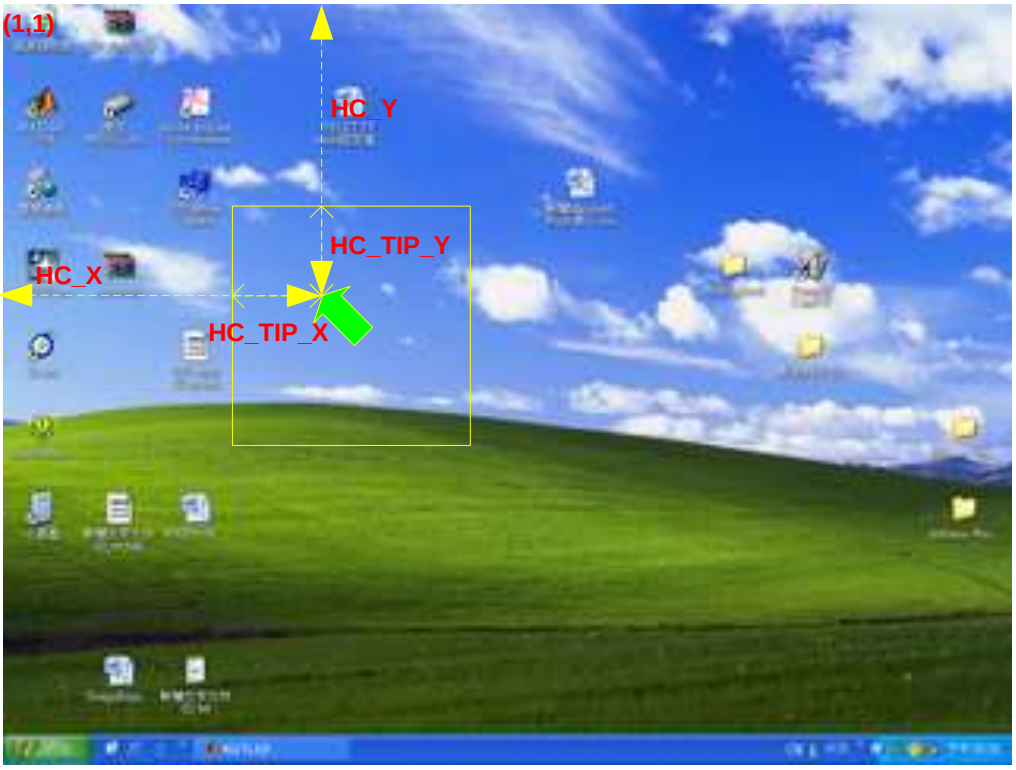
Hardware Cursor block width and height depend on HC_MODE (HC_CTRL[2:0]) setting.

HC_MODE (HC_CTRL[2:0]) = 0 , 1 => hardware cursor block width = hardware cursor block height = 32

HC_MODE (HC_CTRL[2:0]) = 2 , 3 => hardware cursor block width = hardware cursor block height = 64

HC_MODE (HC_CTRL[2:0]) = 4 , 5 => hardware cursor block width = hardware cursor block height = 128


Hardware Cursor
Block width
HC_TIP_Y
HC_TIP_X
Hardware Cursor
Block Height


(1,1)
HC_Y
HC_TIP_Y
HC_X
HC_TIP_X

## Hardware Cursor Window Buffer Control Register (HC_WBCTRL)

The information contained in this register is used to efficiently control the hardware cursor window buffer operation. The HC_STRIDE shows the word offset of memory address between two vertically adjacent lines. The VA_FF specifies the number of WORD SDRAM access cycle for a horizontal scan line fetching.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| HC_WBCTRL | LCM_BA + 0x74 | R/W | Hardware Cursor Window Buffer Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **HC_FF** | **Hardware Cursor Fetch Finish**<br>An 11-bit value specifies the number of WORD SDRAM access cycle for a horizontal scan line fetching of Hardware cursor window. |
| [15:11] | **Reserved** | Reserved. |
| [10:0] | **HC_STRIDE** | **Hardware Cursor Window Buffer Stride**<br>An 11-bit value specifies the word offset of memory address of vertically adjacent line for Hardware cursor window. |

### Hareware Cursor Memory Base Address Register (HC_BADDR)

This register is used to control the starting memory address of the frame buffer for Hardware cursor data stream.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HC_BADDR** | LCM_BA + 0x78 | R/W | Hardware Cursor Memory Base Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **HC_BADDR** | **Starting Memory Address of the Frame Buffer for Hardware Cursor Data Stream**<br>The value of this register represents the starting memory address of the frame buffer. |

### Hardware Cursor Color RAM 0 Register (HC_COLOR0)

This register is used to control the color of hardware cursor according to bpp value 0.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| HC_COLOR0 | LCM_BA + 0x7C | R/W | Hardware Cursor Color RAM 0 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HC_COLOR0_R | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| HC_COLOR0_G | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HC_COLOR0_B | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:16] | **HC_COLOR0_R** | **Hardware Cursor Color 0 R** |
| [15:8] | **HC_COLOR0_G** | **Hardware Cursor Color 0 G** |
| [7:0] | **HC_COLOR0_B** | **Hardware Cursor Color 0 B** |

### Hardware Cursor Color RAM 1 Register (HC_COLOR1)

This register is used to control the color of hardware cursor according to bpp value 1.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| HC_COLOR1 | LCM_BA + 0x80 | R/W | Hardware Cursor Color RAM 1 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HC_COLOR1_R | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| HC_COLOR1_G | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HC_COLOR1_B | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:16] | **HC_COLOR1_R** | **Hardware Cursor Color 1 R** |
| [15:8] | **HC_COLOR1_G** | **Hardware Cursor Color 1 G** |
| [7:0] | **HC_COLOR1_B** | **Hardware Cursor Color 1 B** |

### Hardware Cursor Color RAM 2 Register (HC_COLOR2)

This register is used to control the color of hardware cursor according to bpp value 2.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| HC_COLOR2 | LCM_BA + 0x84 | R/W | Hardware Cursor Color RAM 2 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HC_COLOR2_R | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| HC_COLOR2_G | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HC_COLOR2_B | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:16] | **HC_COLOR2_B** | **Hardware Cursor Color 2 R** |
| [15:8] | **HC_COLOR2_G** | **Hardware Cursor Color 2 G** |
| [7:0] | **HC_COLOR2_R** | **Hardware Cursor Color 2 B** |

## Hardware Cursor Color RAM 3 Register (HC_COLOR3)

This register is used to control the color of hardware cursor according to bpp value 3. When transparency is enabled, this color ram will be ignored.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| HC_COLOR3 | LCM_BA + 0x88 | R/W | Hardware Cursor Color RAM 3 Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HC_COLOR3_R | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| HC_COLOR3_G | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HC_COLOR3_B | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:16] | **HC_COLOR3_R** | **Hardware Cursor Color 3 R**<br>When transparency is enabled, this color ram will be ignored. |
| [15:8] | **HC_COLOR3_G** | **Hardware Cursor Color 3 G**<br>When transparency is enabled, this color ram will be ignored. |
| [7:0] | **HC_COLOR3_B** | **Hardware Cursor Color 3 B**<br>When transparency is enabled, this color ram will be ignored. |

## 5.31 Capture Sensor Interface Controller (CAP)

### 5.31.1 Overview

The Image Capture Interface is designed to capture image data from a sensor. After capturing or fetching image data, it will process the image data, and then FIFO output them into frame buffer.

### 5.31.2 Feature

- 8-bit RGB565 sensor
- 8-bit YUV422 sensor
- Supports CCIR601 YCbCr color range scale to full YUV color range
- Supports 4 packaging format for packet data output: YUYV, Y only, RGB565, RGB555
- Supports YUV422 planar data output
- Supports the CROP function to crop input image to the required size for digital application.
- Supports the down scaling function to scale input image to the required size for digital application.
- Supports frame rate control
- Supports field detection and even/odd field skip mechanism
- Supports packet output dual buffer control through hardware buffer controller
- Supports negative/sepia/posterization color effect
- Supports two independent capture interfaces

### 5.31.3 Block Diagram

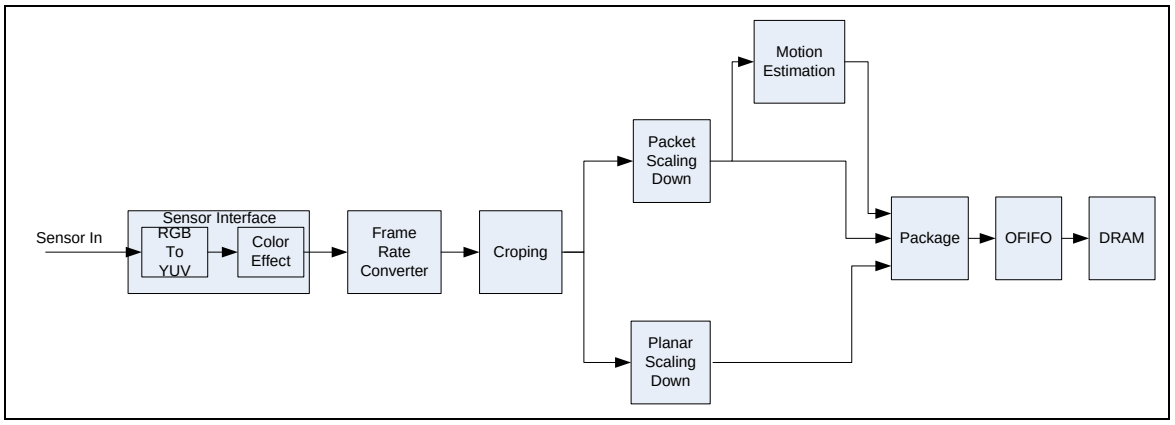



Figure 5.31-1 Image Capture Functional Block Diagram

### 5.31.4 Basic Configuration

Before using Capture Sensor interface, it's necessary to configure related pins as the capture function and enable CAP's clock.

For Capture Sensor interface related pin configuration, please refer to the register SYS_MFP_GPIL and SYS_MFP_GPIH to know how to configure related pins as the Capture Sensor interface function.

The CAP peripheral clock can be enabled by setting CAP (CLK_HCLKEN[26]) high. The CAP engine clock source is selected by SENSOR_S (CLK_DIVCTL3[2016]) and CAP engine clock divider is determined by SENSOR_N (CLK_DIVCTL3[27:24]).

### 5.31.5 Functional Description

#### 5.31.5.1 *Image Capture Flow Chart*

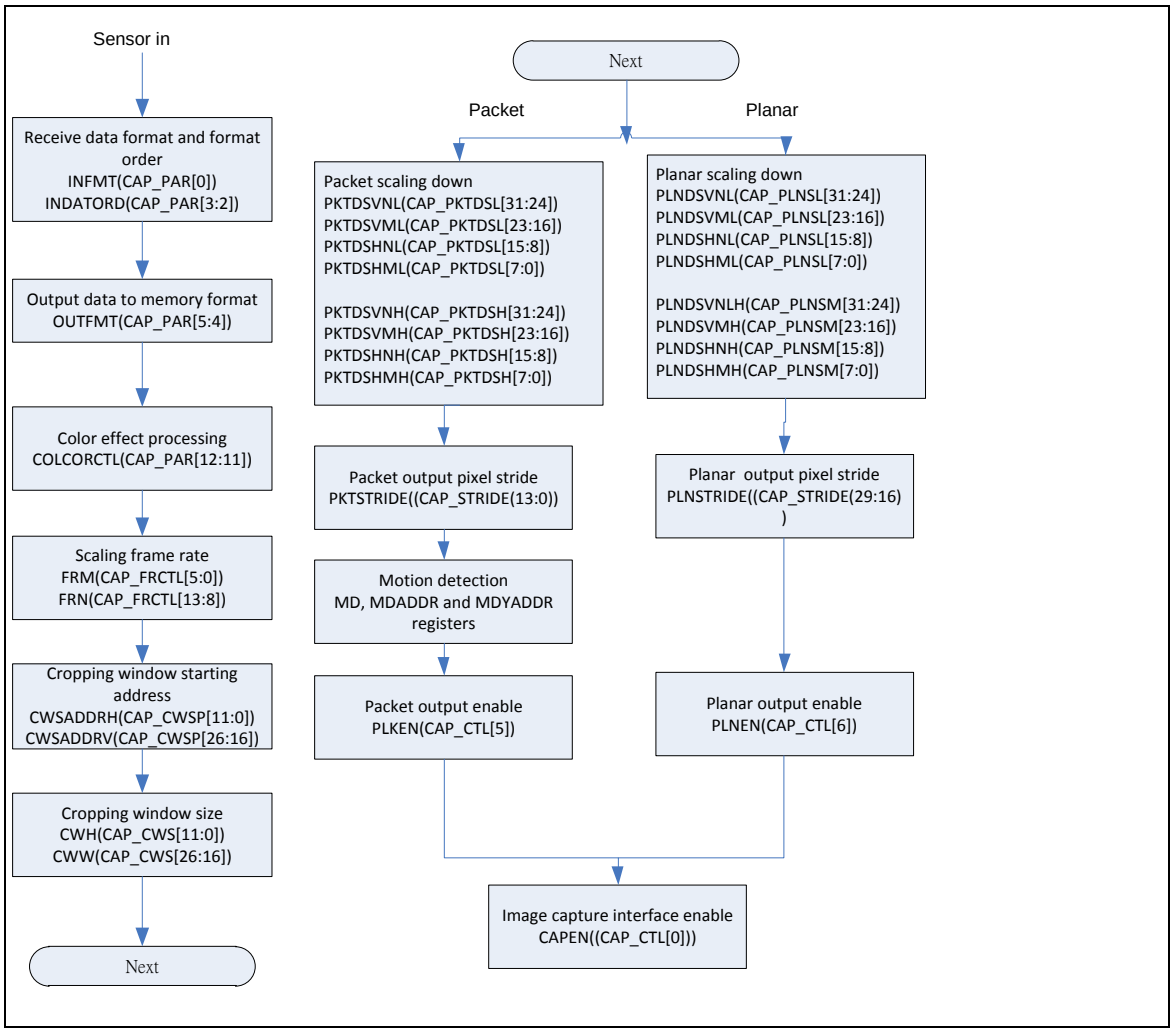



Figure 1-2 Image Capture Flow Chart

#### 5.31.5.2 *Cropping Feature*

The capture interface can select a window from the received image. The size of the window is specified by the number of pixel clocks (horizontal dimension) and the number of lines (vertical dimension). The start (left upper corner) coordinates can be specified by the CAP_CWSP register. The size (vertical dimension in number of lines and horizontal dimension in number of pixel clocks) can be specified by the CAP_CWS register.

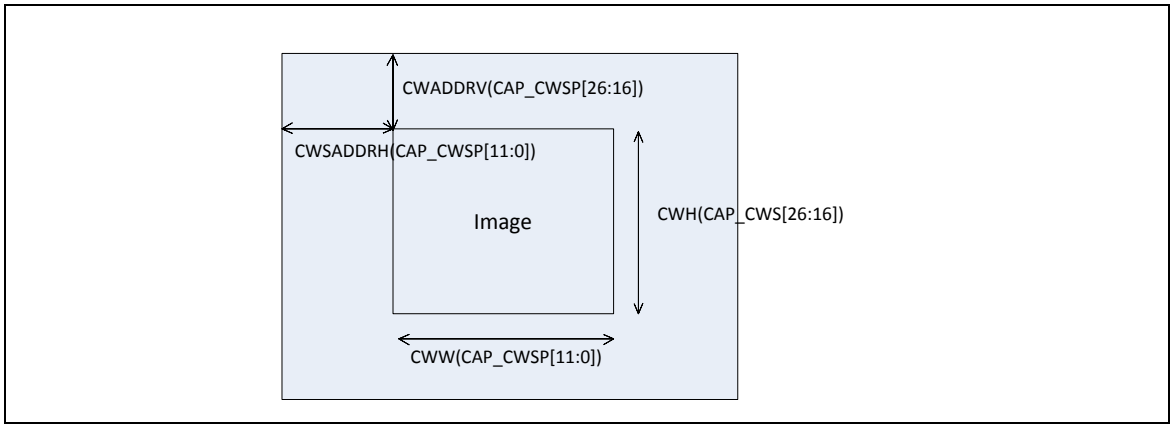



Figure 1-3 Image Start and Size of the Window after Cropping Block

5.31.5.3 *One Shutter Mode (Single Frame)*

In this mode, a single frame is captured. After the SHUTTER (CAP_CTL[16]) bit is set, the Image Capture interface automatically disables the capture interface after a frame is captured.

5.31.5.4 *Motion Detection*

The feature is used to detect object movement. MDSM (CAP_MD[9]) and MDBS (CAP_MD[8]) are determined using the method of storage. If the difference between the center of BASEADDR(CAP_PKTBA0[31:0]) block and the center of MDYADDR(CAP_MDYADDR[31:0]) block is greater than MDTHR (CAP_MD[20:16]) the first bit will change to 1, and other bits will be different, as shown in the following figure.

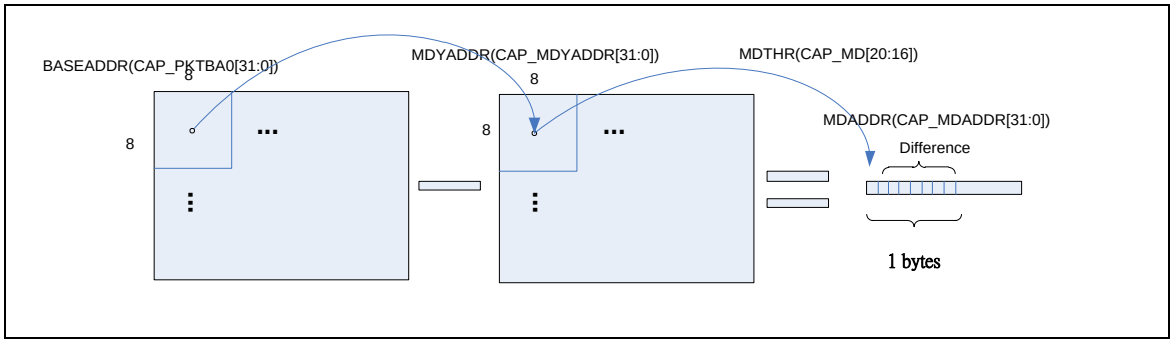



Figure 1-4 MDSM (MD[9]) is set to 0 and MDBS (MD[8]) is set to 1

If the difference between the center of BASEADDR(CAP_PKTBA0[31:0]) block and the center of CAP_MDYADDR block is greater than MDTHR (CAP_MD[20:16]) the first bit is set to 1, and if the next difference between the center of PACBA0 block and the center of MDYADDR(CAP_MDYADDR[31:0]) block is greater than MDTHR (CAP_MD[20:16]) the second bit will change to 1, as shown in the following figure.

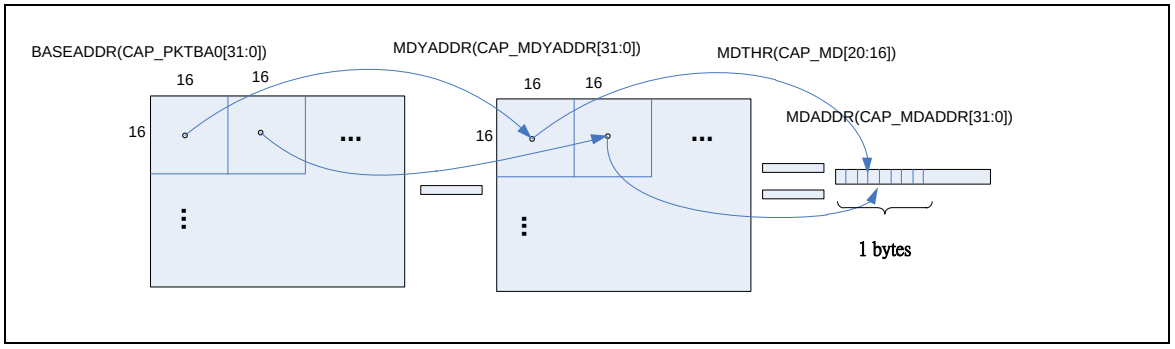


Figure 1-5 MDSM (MD[9]) is set to 1 and MDBS (MD[8]) is set to 0

### 5.31.6 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAP Base Address:**<br>**CAP_BA = 0xB000_E000** | | | | |
| **CAP_CTL** | CAP_BA+0x00 | R/W | Image Capture Interface Control Register | 0x0000_0040 |
| **CAP_PAR** | CAP_BA+0x04 | R/W | Image Capture Interface Parameter Register | 0x0000_0000 |
| **CAP_INT** | CAP_BA+0x08 | R/W | Image Capture Interface Interrupt Register | 0x0000_0000 |
| **CAP_POSTERIZE** | CAP_BA+0x0C | R/W | YUV Component Posterizing Factor Register | 0x0000_0000 |
| **CAP_MD** | CAP_BA+0x10 | R/W | Motion Detection Register | 0x0000_0000 |
| **CAP_MDADDR** | CAP_BA+0x14 | R/W | Motion Detection Output Address Register | 0x0000_0000 |
| **CAP_MDYADDR** | CAP_BA+0x18 | R/W | Motion Detection Temp Y Output Address Register | 0x0000_0000 |
| **CAP_SEPIA** | CAP_BA+0x1C | R/W | Sepia Effect Control Register | 0x0000_0000 |
| **CAP_CWSP** | CAP_BA+0x20 | R/W | Cropping Window Starting Address Register | 0x0000_0000 |
| **CAP_CWS** | CAP_BA+0x24 | R/W | Cropping Window Size Register | 0x0000_0000 |
| **CAP_PKTSL** | CAP_BA+0x28 | R/W | Packet Scaling Vertical/Horizontal Factor Register (LSB) | 0x0000_0000 |
| **CAP_PLNSL** | CAP_BA+0x2C | R/W | Planar Scaling Vertical/Horizontal Factor Register (LSB) | 0x0000_0000 |
| **CAP_FRCTL** | CAP_BA+0x30 | R/W | Scaling Frame Rate Factor Register | 0x0000_0000 |
| **CAP_STRIDE** | CAP_BA+0x34 | R/W | Frame Output Pixel Stride Width Register | 0x0000_0000 |
| **CAP_FIFOTH** | CAP_BA+0x3C | R/W | FIFO Threshold Register | 0x070D_0507 |
| **CAP_CMPADDR** | CAP_BA+0x40 | R/W | Compare Memory Base Address Register | 0xFFFF_FFFC |
| **CAP_PKTSM** | CAP_BA+0x48 | R/W | Packet Scaling Vertical/Horizontal Factor Register (MSB) | 0x0000_0000 |
| **CAP_PLNSM** | CAP_BA+0x4C | R/W | Planar Scaling Vertical/Horizontal Factor Register (MSB) | 0x0000_0000 |
| **CAP_CURADDRP** | CAP_BA+0x50 | R | Current Packet System Memory Address Register | 0x0000_0000 |
| **CAP_CURADDRY** | CAP_BA+0x54 | R | Current Planar Y System Memory Address Register | 0x0000_0000 |
| **CAP_CURADDRU** | CAP_BA+0x58 | R | Current Planar U System Memory Address Register | 0x0000_0000 |
| **CAP_CURADDRV** | CAP_BA+0x5C | R | Current Planar V System Memory Address Register | 0x0000_0000 |
| **CAP_PKTBA0** | CAP_BA+0x60 | R/W | System Memory Packet Base Address 0 Register | 0x0000_0000 |
| **CAP_PKTBA1** | CAP_BA+0x64 | R/W | System Memory Packet Base Address 1 Register | 0x0000_0000 |
| **CAP_YBA** | CAP_BA+0x80 | R/W | System Memory Planar Y Base Address Register | 0x0000_0000 |
| **CAP_UBA** | CAP_BA+0x84 | R/W | System Memory Planar U Base Address Register | 0x0000_0000 |
| **CAP_VBA** | CAP_BA+0x88 | R/W | System Memory Planar V Base Address Register | 0x0000_0000 |

### 5.31.7 Register Description

### Image Capture Interface Control Register (CAP_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CAP_CTL | CAP_BA+0x00 | R/W | Image Capture Interface Control Register | 0x0000_0040 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | VPRST |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | UPDATE | Reserved | | | SHUTTER |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | PKTEN | PLNEN | Reserved | ADDRSW | Reserved | Reserved | CAPEN |

| Bits | Description | |
|---|---|---|
| [31:25] | **Reserved** | Reserved. |
| [24] | **VPRST** | **Capture Interface Reset**<br>0 = Capture interface reset Disabled.<br>1 = Capture interface reset Enabled. |
| [23:21] | **Reserved** | Reserved. |
| [20] | **UPDATE** | **Update Register at New Frame**<br>0 = Update register at new frame Disabled.<br>1 = Update register at new frame Enabled (Auto clear to 0 when register updated). |
| [19:17] | **Reserved** | Reserved. |
| [16] | **SHUTTER** | **Image Capture Interface Automatically Disable the Capture Inteface After a Frame Had Been Captured**<br>0 = Shutter Disabled.<br>1 = Shutter Enabled. |
| [15:7] | **Reserved** | Reserved. |
| [6] | **PKTEN** | **Packet Output Enable**<br>0 = Packet output Disabled.<br>1 = Packet output Enabled. |
| [5] | **PLNEN** | **Planar Output Enable**<br>0 = Planar output Disabled.<br>1 = Planar output Enabled. |
| [4] | **Reserved** | Reserved. |
| [3] | **ADDRSW** | **Packet Buffer Address Switch**<br>0 = Packet buffer address switch Disabled.<br>1 = Packet buffer address switch Enabled. |

| | | |
|---|---|---|
| [2:1] | **Reserved** | Reserved. |
| [0] | **CAPEN** | **Image Capture Interface Enable**<br>0 = Image Capture Interface Disabled.<br>1 = Image Capture Interface Enabled. |

## Image Capture Interface Parameter Register (CAP_PAR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CAP_PAR | CAP_BA+0x04 | R/W | Image Capture Interface Parameter Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | FBB | Reserved | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | COLORCTL | | VSP | HSP | PCLKP |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PLNFMT | RANGE | OUTFMT | | INDATORD | | SENTYPE | INFMT |

| Bits | Description | |
|---|---|---|
| [31:19] | **Reserved** | Reserved. |
| [18] | **FBB** | **Field by Blank**<br>Hardware will tag field0 or field1 by vertical blanking instead of FIELD flag in ccir-656 mode.<br>0 = Field by blank Disabled.<br>1 = Field by blank Enabled. |
| [17:13] | **Reserved** | Reserved. |
| [12:11] | **COLORCTL** | **Special COLORCTL Processing**<br>00 = Normal Color.<br>01 = Sepia effect, corresponding U,V component value is set at register CAP_SEPIA.<br>10 = Negative picture.<br>11 = Posterize image, the Y, U, V components posterizing factor are set at register CAP_POSTERIZE. |
| [10] | **VSP** | **Sensor Vsync Polarity**<br>0 = Sync Low.<br>1 = Sync High. |
| [9] | **HSP** | **Sensor Hsync Polarity**<br>0 = Sync Low.<br>1 = Sync High. |
| [8] | **PCLKP** | **Sensor Pixel Clock Polarity**<br>0 = Input video data and signals are latched by falling edge of Pixel Clock.<br>1 = Input video data and signals are latched by rising edge of Pixel Clock. |
| [7] | **PLNFMT** | **Planar Output YUV Format**<br>0 = YUV422. |

| | | |
|---|---|---|
| | | 1 = YUV420. |
| [6] | **RANGE** | **Scale Input YUV CCIR601 Color Range to Full Range**<br>0 = Default.<br>1 = Scale to full range. |
| [5:4] | **OUTFMT** | **Image Data Format Output to System Memory**<br>00 = YCbCr422.<br>01 = Only output Y.<br>10 = RGB555.<br>11 = RGB565. |
| [3:2] | **INDATORD** | **Sensor Input Data Order**<br>If INFMT (CAP_PAR[0]) = 0 (YCbCr).<br>00 = Sensor input data (Byte 0 1 2 3) is Y0 U0 Y1 V0.<br>01 = Sensor input data (Byte 0 1 2 3) is Y0 V0 Y1 U0.<br>10 = Sensor input data (Byte 0 1 2 3) is U0 Y0 V0 Y1.<br>11 = Sensor input data (Byte 0 1 2 3) is V0 Y0 U0 Y1.<br>If INFMT (CAP_PAR[0]) = 1 (RGB565).<br>00 = Sensor input data (Byte 0) is {R[4:0],G[5:3]},.<br>Sensor input data (Byte 1) is {G[2:0],B[4:0]}.<br>01 = Sensor input data (Byte 0) is {B[4:0],G[5:3]},.<br>Sensor input data (Byte 1) is {G[2:0], R[4:0]}.<br>10 = Sensor input data (Byte 0) is {G[2:0],B[4:0]},.<br>Sensor input data (Byte 1) is {R[4:0], G[5:3]}.<br>11 = Sensor input data (Byte 0) is {G[2:0],R[4:0]},.<br>Sensor input data (Byte 1) is {B[4:0], G[5:3]}. |
| [1] | **SENTYPE** | **Sensor Input Type**<br>0 = CCIR601.<br>1 = CCIR656, VSync & Hsync embedded in the data signal. |
| [0] | **INFMT** | **Sensor Input Data Format**<br>0 = YCbCr422.<br>1 = RGB565. |

### Image Capture Interface Interrupt Register (CAP_INT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAP_INT** | CAP_BA+0x08 | R/W | Image Capture Interface Interrupt Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | **MDIEN** | **ADDRMIEN** | **Reserved** | **MEIEN** | **VIEN** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | **MDINTF** | **ADDRMINTF** | **Reserved** | **MEINTF** | **VINTF** |

| Bits | Description | |
|---|---|---|
| [31:21] | **Reserved** | Reserved. |
| [20] | **MDIEN** | **Motion Detection Output Finish Interrupt Enable**<br>0 = CAP_MD finish interrupt Disabled.<br>1 = CAP_MD finish interrupt Enabled. |
| [19] | **ADDRMIEN** | **Address Match Interrupt Enable**<br>0 = Address match interrupt Disabled.<br>1 = Address match interrupt Enabled. |
| [18] | **Reserved** | Reserved. |
| [17] | **MEIEN** | **System Memory Error Interrupt Enable**<br>0 = System memory error interrupt Disabled.<br>1 = System memory error interrupt Enabled. |
| [16] | **VIEN** | **Video Frame End Interrupt Enable**<br>0 = Video frame end interrupt Disabled.<br>1 = Video frame end interrupt Enabled. |
| [15:5] | **Reserved** | Reserved. |
| [4] | **MDINTF** | **Motion Detection Output Finish Interrupt**<br>If this bit shows 1, Motion Detection Output Finish Interrupt occurred.<br>Write 1 to clear it. |
| [3] | **ADDRMINTF** | **Memory Address Match Interrupt**<br>If this bit shows 1, Memory Address Match Interrupt occurred.<br>Write 1 to clear it. |
| [2] | **Reserved** | Reserved. |
| [1] | **MEINTF** | **Bus Master Transfer Error Interrupt** |

| | | |
|---|---|---|
| | | If this bit shows 1, Transfer Error occurred. Write 1 to clear it. |
| [0] | **VINTF** | **Video Frame End Interrupt**<br>If this bit shows 1, receiving a frame completed.<br>Write 1 to clear it. |

## YUV Component Posterizing Factor Register (CAP_POSTERIZE)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAP_POSTERIZE** | CAP_BA+0x0C | R/W | YUV Component Posterizing Factor Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:16] | **YCOMP** | **Y Component Posterizing Factor**<br>Final_Y_Out = Original_Y[7:0] & Y_Posterizing_Factor. |
| [15:8] | **UCOMP** | **U Component Posterizing Factor**<br>Final_U_Out = Original_U[7:0] & U_Posterizing_Factor. |
| [7:0] | **VCOMP** | **V Component Posterizing Factor**<br>Final_V_Out = Original_V[7:0] & V_Posterizing_Factor. |

## Motion Detection Register (CAP_MD)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAP_MD** | CAP_BA+0x10 | R/W | Motion Detection Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:21] | **Reserved** | Reserved. |
| [20:16] | **MDTHR** | **Motion Detection Differential Threshold** |
| [15:12] | **Reserved** | Reserved. |
| [11:10] | **MDDF** | **Motion Detection Detect Frequency**<br>00 = Each frame.<br>01 = Every 2 frame.<br>10 = Every 3 frame.<br>11 = Every 4 frame. |
| [9] | **MDSM** | **Motion Detection Save Mode**<br>0 = 1 bit DIFF + 7 bit Y Differential.<br>1 = 1 bit DIFF only. |
| [8] | **MDBS** | **Motion Detection Block Size**<br>0 = 16x16.<br>1 = 8x8. |
| [7:1] | **Reserved** | Reserved. |
| [0] | **MDEN** | **Motion Detection Enable**<br>0 = CAP_MD Disabled.<br>1 = CAP_MD Enabled. |

### Motion Detection Output Address Register (CAP_MDADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAP_MDADDR** | CAP_BA+0x14 | R/W | Motion Detection Output Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **MDADDR** | **Motion Detection Output Address Register (Word Alignment)** |

## Motion Detection Temp Y Output Address Register (CAP_MDYADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAP_MDYADDR** | CAP_BA+0x18 | R/W | Motion Detection Temp Y Output Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **MDYADDR** | **Motion Detection Temp Y Output Address Register (Word Alignment)** |

### Sepia Effect Control Register (CAP_SEPIA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAP_SEPIA** | CAP_BA+0x1C | R/W | Sepia Effect 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** |
| **UCOMP** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **VCOMP** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:8] | **UCOMP** | Define the constant U component while “Sepia” color effect is turned on. |
| [7:0] | **VCOMP** | Define the constant V component while “Sepia” color effect is turned on. |

### Cropping Window Starting Address Register (CAP_CWSP)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAP_CWSP** | CAP_BA+0x20 | R/W | Cropping Window Starting Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **CWSADDRV** | **Cropping Window Vertical Starting Address** |
| [15:12] | **Reserved** | Reserved. |
| [11:0] | **CWSADDRH** | **Cropping Window Horizontal Starting Address** |

### Cropping Window Size Register (CAP_CWS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAP_CWS** | CAP_BA+0x24 | R/W | Cropping Window Size Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26:16] | **CWH** | **Cropping Window Height** |
| [15:12] | **Reserved** | Reserved. |
| [11:0] | **CWW** | **Cropping Window Width** |

## Packet Scaling Vertical/Horizontal Factor Register (LSB) (CAP_PKTSL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAP_PKTSL** | CAP_BA+0x28 | R/W | Packet Scaling Vertical/Horizontal Factor Register (LSB) | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **PKTSVNL** | **Packet Scaling Vertical Factor N (Lower 8-bit)**<br>Specify the lower 8-bit of numerator part (N) of the vertical scaling factor.<br>The lower 8-bit will be cascaded with higher 8-bit (PKDSVNH) to form a 16-bit numerator of vertical factor. |
| [23:16] | **PKTSVML** | **Packet Scaling Vertical Factor M (Lower 8-bit)**<br>Specify the lower 8-bit of denominator part (M) of the vertical scaling factor.<br>The lower 8-bit will be cascaded with higher 8-bit (PKDSVMH) to form a 16-bit denominator (M) of vertical factor.<br>The output image width will be equal to the image height * N/M.<br>**Note:** The value of N must be equal to or less than M. |
| [15:8] | **PKTSHNL** | **Packet Scaling Horizontal Factor N (Lower 8-bit)**<br>Specify the lower 8-bit of numerator part (N) of the horizontal scaling factor.<br>The lower 8-bit will be cascaded with higher 8-bit (PKDSHNH) to form a 16-bit numerator of horizontal factor. |
| [7:0] | **PKTSHML** | **Packet Scaling Horizontal Factor M (Lower 8-bit)**<br>Specifies the lower 8-bit of denominator part (M) of the horizontal scaling factor.<br>The lower 8-bit will be cascaded with higher 8-bit (PKDSHMH) to form a 16-bit denominator (M) of vertical factor.<br>The output image width will be equal to the image width * N/M.<br>**Note:** The value of N must be equal to or less than M. |

## Planar Scaling Vertical/Horizontal Factor Register (LSB) (CAP_PLNSL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CAP_PLNSL | CAP_BA+0x2C | R/W | Planar Scaling Vertical/Horizontal Factor Register (LSB) | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **PLNSVNL** | **Planar Scaling Vertical Factor N (Lower 8-bit)**<br>Specify the lower 8-bit of numerator part (N) of the vertical scaling factor.<br>The lower 8-bit will be cascaded with higher 8-bit (PNDSVNH) to form a 16-bit numerator of vertical factor. |
| [23:16] | **PLNSVML** | **Planar Scaling Vertical Factor M (Lower 8-bit)**<br>Specify the lower 8-bit of denominator part (M) of the vertical scaling factor.<br>The lower 8-bit will be cascaded with higher 8-bit (PNDSVMH) to form a 16-bit denominator (M) of vertical factor.<br>The output image width will be equal to the image height * N/M.<br>**Note:** The value of N must be equal to or less than M. |
| [15:8] | **PLNSHNL** | **Planar Scaling Horizontal Factor N (Lower 8-bit)**<br>Specify the lower 8-bit of numerator part (N) of the horizontal scaling factor.<br>The lower 8-bit will be cascaded with higher 8-bit (PNDSHNH) to form a 16-bit numerator of horizontal factor. |
| [7:0] | **PLNSHML** | **Planar Scaling Horizontal Factor M (Lower 8-bit)**<br>Specify the lower 8-bit of denominator part (M) of the horizontal scaling factor.<br>The lower 8-bit will be cascaded with higher 8-bit (PNDSHMH) to form a 16-bit denominator (M) of vertical factor.<br>The output image width will be equal to the image width * N/M.<br>**Note:** The value of N must be equal to or less than M. |

## Packet Scaling Vertical/Horizontal Factor Register (MSB) (CAP_PKTSM)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CAP_PKTSM | CAP_BA+0x48 | R/W | Packet Scaling Vertical/Horizontal Factor Register (MSB) | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **PKTSVNH** | **Packet Scaling Vertical Factor N (Higher 8-bit)**<br>Specify the higher 8-bit of numerator part (N) of the vertical scaling factor.<br>Please refer to the register "CAP_PKTSL" to check the cooperation between these two registers. |
| [23:16] | **PKTSVMH** | **Packet Scaling Vertical Factor M (Higher 8-bit)**<br>Specify the lower 8-bit of denominator part (M) of the vertical scaling factor.<br>Please refer to the register "CAP_PKTSL" to check the cooperation between these two registers. |
| [15:8] | **PKTSHNH** | **Packet Scaling Horizontal Factor N (Higher 8-bit)**<br>Specify the lower 8-bit of numerator part (N) of the horizontal scaling factor.<br>Please refer to the register "CAP_PKTSL" for the detailed operation. |
| [7:0] | **PKTSHMH** | **Packet Scaling Horizontal Factor M (Higher 8-bit)**<br>Specify the lower 8-bit of denominator part (M) of the horizontal scaling factor.<br>Please refer to the register "CAP_PKTSL" for the detailed operation. |

### Planar Scaling Vertical/Horizontal Factor Register (MSB) (CAP_PLNSM)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAP_PLNSM** | CAP_BA+0x4C | R/W | Planar Scaling Vertical/Horizontal Factor Register (MSB) | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **PLNSVNH** | **Planar Scaling Vertical Factor N (Higher 8-bit)**<br>Specifies the higher 8-bit of numerator part (N) of the vertical scaling factor.<br>For detailed programming, please refer to the register "CAP_PLNSL". |
| [23:16] | **PLNSVMH** | **Planar Scaling Vertical Factor M (Higher 8-bit)**<br>Specifies the lower 8-bit of denominator part (M) of the vertical scaling factor.<br>For detailed programming, please refer to the register "CAP_PLNSL". |
| [15:8] | **PLNSHNH** | **Planar Scaling Horizontal Factor N (Higher 8-bit)**<br>Specifies the higher 8-bit of numerator part (N) of the horizontal scaling factor.<br>For detailed programming, please refer to the register "CAP_PLNSL". |
| [7:0] | **PLNSHMH** | **Planar Scaling Horizontal Factor M (Higher 8-bit)**<br>Specifies the higher 8-bit of denominator part (M) of the horizontal scaling factor<br>For detailed programming, please refer to the register "CAP_PLNSL". |

### Scaling Frame Rate Factor Register (CAP_FRCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAP_FRCTL** | CAP_BA+0x30 | R/W | Scaling Frame Rate Factor Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:14] | **Reserved** | Reserved. |
| [13:8] | **FRN** | **Scaling Frame Rate Factor N**<br>Specify the denominator part (N) of the frame rate scaling factor. |
| [7:6] | **Reserved** | Reserved. |
| [5:0] | **FRM** | **Scaling Frame Rate Factor M**<br>Specify the denominator part (M) of the frame rate scaling factor.<br>The output image frame rate will be equal to input image frame rate * (N/M).<br>**Note:** The value of N must be equal to or less than M. |

### Output Frame Pixel Stride Width (CAP_STRIDE)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAP_STRIDE** | CAP_BA+0x34 | R/W | Frame Output Pixel Stride Width Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [29:16] | **PLNSTRIDE** | **Planar Frame Output Pixel Stride Width**<br>The output pixel stride size of planar pipe. |
| [15:12] | **Reserved** | Reserved. |
| [13:0] | **PKTSTRIDE** | **Packet Frame Output Pixel Stride Width**<br>The output pixel stride size of packet pipe. |

### FIFO Threshold Register (CAP_FIFOTH)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CAP_FIFOTH | CAP_BA+0x3C | R/W | FIFO Threshold Register | 0x070D_0507 |

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

| Bits | Description | |
|---|---|---|
| [31] | **OVF** | **FIFO Overflow Flag** |
| [30:29] | **Reserved** | Reserved. |
| [28:24] | **PKTFTH** | **Packet FIFO Threshold** |
| [23:21] | **Reserved** | Reserved. |
| [20:16] | **PLNYFTH** | **Planar Y FIFO Threshold** |
| [15:12] | **Reserved** | Reserved. |
| [11:8] | **PLNUFTH** | **Planar U FIFO Threshold** |
| [7:4] | **Reserved** | Reserved. |
| [3:0] | **PLNVFTH** | **Planar V FIFO Threshold** |

## Compare Memory Address Register (CAP_CMPADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAP_CMPADDR** | CAP_BA+0x40 | R/W | Compare Memory Base Address Register | 0xFFFF_FFFC |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **CMPADDR** | **Compare Memory Base Address**<br>Word aligns address; ignore the bits [1:0]. |

## Current Packet System Memory Address Register (CAP_CURADDRP)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAP_CURADDRP** | CAP_BA+0x50 | R | Current Packet System Memory Address Register | 0x0000_0000 |

| 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** | **Current Packet Output Memory Address** |

## Current Planar Y System Memory Address Register (CAP_CURADDRY)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAP_CURADDRY** | CAP_BA+0x54 | R | Current Planar Y System Memory Address Register | 0x0000_0000 |

| 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** | **Current Planar Y Output Memory Address** |

## Current Planar U System Memory Address Register (CAP_CURADDRU)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CAP_CURADDRU | CAP_BA+0x58 | R | Current Planar U System Memory Address Register | 0x0000_0000 |

| 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 | Current Planar U Output Memory Address |

## Current Planar V System Memory Address Register (CAP_CURADDRV)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CAP_CURADDRV | CAP_BA+0x5C | R | Current Planar V System Memory Address Register | 0x0000_0000 |

| 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 | Current Planar V Output Memory Address |

### System Memory Packet Base Address 0 Register (CAP_PKTBA0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAP_PKTBA0** | CAP_BA+0x60 | R/W | System Memory Packet Base Address 0 Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **BASEADDR** | **System Memory Packet Base Address 0**<br>Word aligns address; ignore the bits [1:0]. |

### System Memory Packet Base Address 1 Register (CAP_PKTBA1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAP_PKTBA1** | CAP_BA+0x64 | R/W | System Memory Packet Base Address 1 Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **BASEADDR** | **System Memory Packet Base Address 1**<br>Word aligns address; ignore the bits [1:0]. |

### System Memory Planar Y Base Address Register (CAP_YBA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAP_YBA** | CAP_BA+0x80 | R/W | System Memory Planar Y Base Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **BASEADDR** | **System Memory Planar Y Base Address**<br>Word aligns address; ignore the bits [1:0]. |

### System Memory Planar U Base Address Register (CAP_UBA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAP_UBA** | CAP_BA+0x84 | R/W | System Memory Planar U Base Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **BASEADDR** | **System Memory Planar U Base Address**<br>Word aligns address; ignore the bits [1:0]. |

### System Memory Planar V Base Address Register (CAP_VBA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CAP_VBA** | CAP_BA+0x88 | R/W | System Memory Planar V Base Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **BASEADDR** | **System Memory Planar V Base Address**<br>Word aligns address; ignore the bits [1:0]. |

## 5.32 Analog to Digital Converter (ADC)

### 5.32.1 Overview

The NuMicro™ NUC970 series contains one 12-bit Successive Approximation Register analog-to-digital converter (SAR A/D converter) with eight input channels. The A/D converter supports two operation modes: 4-wire or 5-wire mode. The ADC is especially suitable to act as touch screen controller. Battery voltage detection could be easily accomplished by the SAR ADC. It has keypad interrupt signal generator.

### 5.32.2 Features

- Resolution: 12-bit resolution.
- DNL: +/-1.5 LSB, INL: +/-3 LSB.
- Dual Data Rates: 1MSPS/200KSPS.
- Analog Input Range: VREF to AGND, could be rail-to-rail.
- Analog Supply: 2.7-3.6V.
- Digital Supply: 1.2V.
- 8 Single-Ended Analog inputs.
- Compatible with 4-wire or 5-wire Touch Screen Interface.
- Touch Pressure Measurement for 4-wire touch screen application.
- Direct Battery Measurement.
- Keypad Interrupt Generator.
- Auto Power Down.
- Low Power Consumption: 4850uW(@1MSPS) / 2170uW(@200KSPS), < 1uA

### 5.32.3 Block Diagram

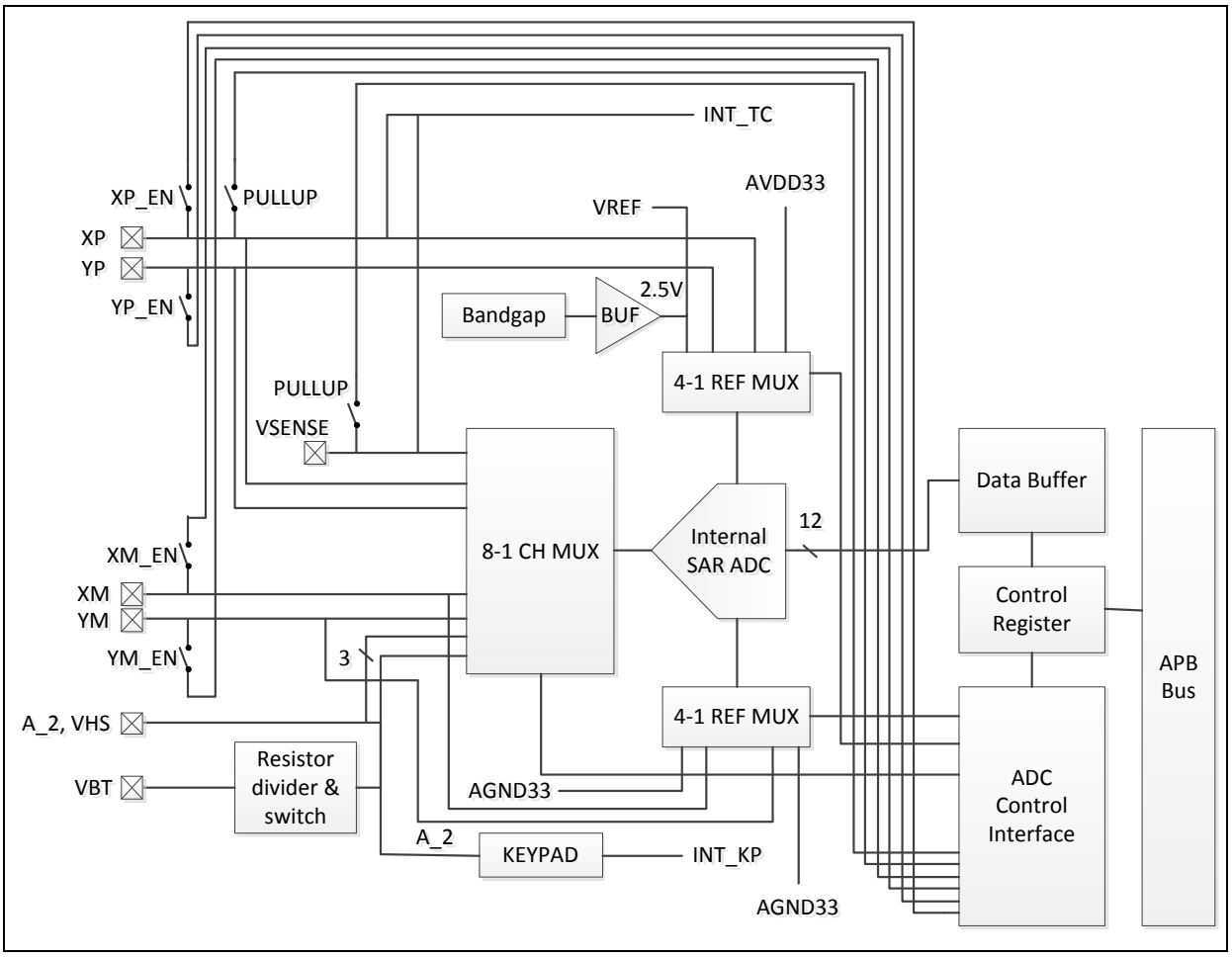



Figure 5.32-1 ADC Functional Block Diagram

### 5.32.4 Basic Configuration

The ADC peripheral clock can be enabled in ADC (PCLKEN1[24]). The ADC engine clock source is selected by ADC_S (CLKDIV7[23:16]) and ADC engine clock divider is determined by ADC_N (CLKDIV7[31:24]).

### 5.32.5 Functional Description

#### 5.32.5.1 *ADC Transfer Function*

The ADC output coding is offset in binary, 1LSB=VREF/4096, the transfer characteristic is shown in the following graph:

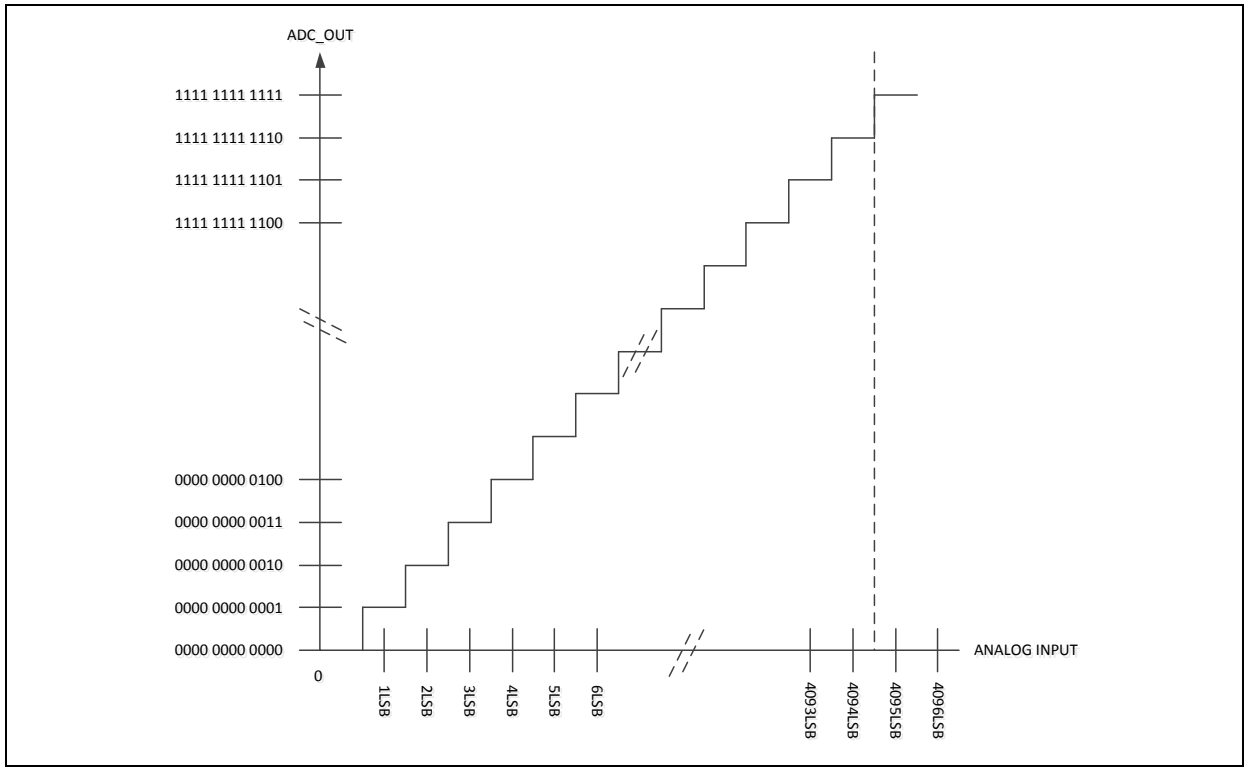



Figure 5.32-2 ADC Transfer Function

#### 5.32.5.2 *Selection of Input Signal*

| IN_SEL[2:0] | Select ADC Analog Input Signal | Description |
|---|---|---|
| 000 | VBT | ADC analog input, intended for battery voltage detection. It includes an inherent resistor divider and a switch. |
| 001 | VHS | ADC high speed input, could support 1MS/S.<br>When HSPEED is set to high, it supports 1MS/S; when HSPEED is set to low, it supports 200KS/S. |
| 010 | A_2 | ADC low speed input, could support 200KS/S; Keypad signal input pin. |
| 011 | VSENSE | ADC analog input, intended for 5-wire touch screen detection. |

| | | |
|---|---|---|
| 100 | YM | ADC analog input.<br>If used in touch screen, it should connect to the negative end of Y axis.<br>If used in 5-wire touch screen, it could connect to lower-left electrode. |
| 101 | YP | ADC analog input.<br>If used in touch screen, it should connect to the positive end of Y axis.<br>If used in 5-wire touch screen, it could connect to upper-right electrode. |
| 110 | XM | ADC analog input.<br>If used in touch screen, it should connect to the negative end of X axis.<br>If used in 5-wire touch screen, it could connect to lower-right electrode. |
| 111 | XP | ADC analog input.<br>If used in 4-wire touch screen, it should connect to the positive end of X axis.<br>If used in 5-wire touch screen, it could connect to upper-left electrode. |

The input current on analog input pins depends on analog input voltage and sampling rate. In the sampling mode, the input current charges the internal sampling capacitor. After the capacitor is fully charged, the current will be off. The internal sampling capacitor must be charged to 12-bit settling level at sampling time, under afore mentioned condition.

The signal source impedance connected to analog input needs to satisfy the following expression:

$$0.69 \times (12+1) \times C_{samp} \times 1.1 \times (R_{src} + R_{on} + R_{ref}) < \frac{1}{f_{clk}} \times 3$$

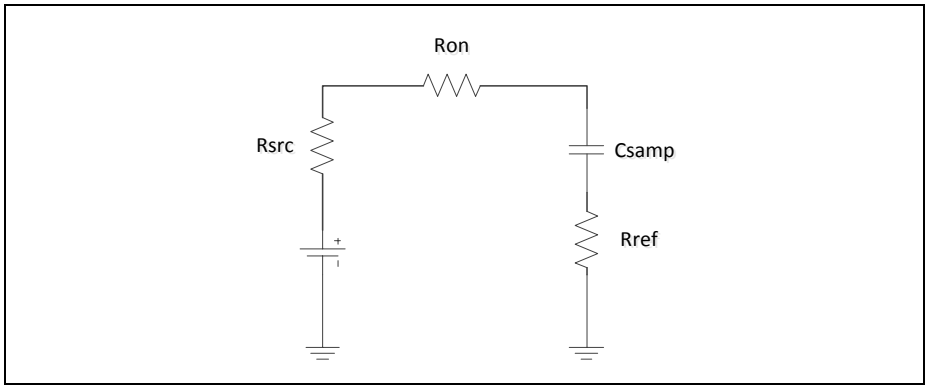



Figure 5.32-3 Simplified Sampling Diagram

5.32.5.3 *High Speed Mode(1Ms/s)*

$C_{samp} = 25.6p$

$f_{clk} = 16M, \text{frequency of the clock}$

$R_{ref} = 300$

$R_{on} = 20$

5.32.5.4 *Low Speed Mode(200ks/s)*

$C_{samp} = 25.6p$

$f_{clk} = 3.2M$, frequency of the clock

$R_{ref} = 1500$

$R_{on} = 100$

5.32.5.5 *Selection of Reference Voltage*

| REF_SEL[2:0] | ADC Analog Reference Pair Selection Signals |
|---|---|
| 00 | AGND33 Vs 2.5v buffer output, or VREF input. |
| 01 | YM Vs YP |
| 10 | XM Vs XP |
| 11 | AGND33 Vs AVDD33 |

Reference voltage is selected very flexibly, and could be selected according to the application.

- For battery voltage input, REF_SEL should be set to 00, VREF and AGND.
- For 4-wire/5-wire touch Screen Y axis conversion, REF_SEL could be set to 01
- For 4-wire/5-wire touch Screen X axis conversion, REF_SEL could be set to 10
- Others, for example, VHS inputs a sin wave, rail to rail, REF_SEL should be set to 00 or 11.

5.32.5.6 *Typical Application*

5.32.5.7 *Battery Voltage Detection*

Take VBT as input, and select internal buffer's output as the reference. For ADC configure register VBAT_EN (ADC_CONF[8]) should be set to 1.

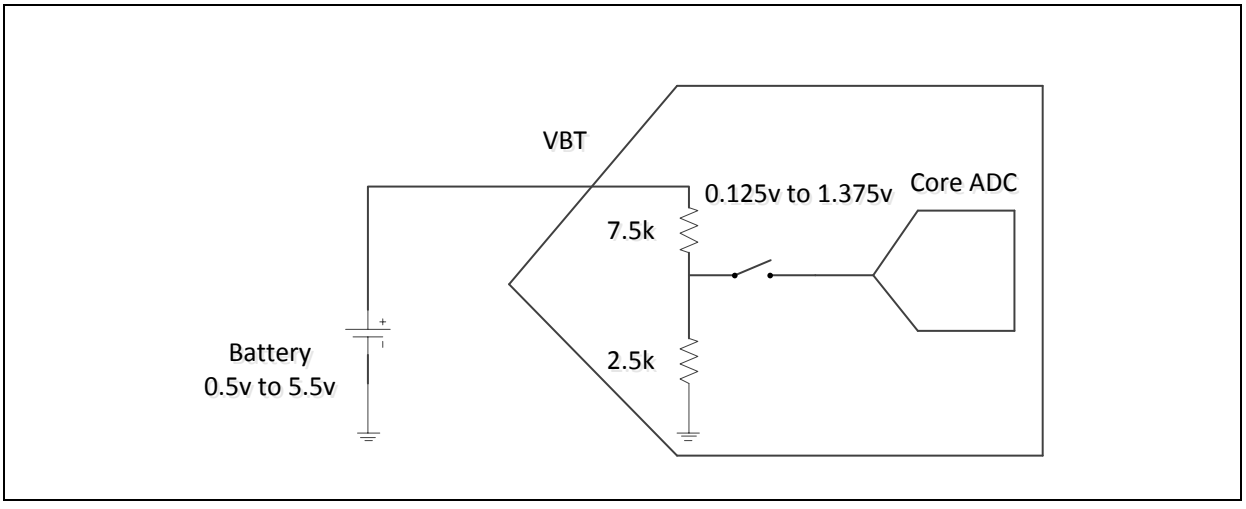



Figure 5.32-4 ADC Battery Voltage Detection Diagram

5.32.5.8 *Key Pad Scan*

Take A_2 as input, and select AVDD33 and AGND33 as the reference. For ADC configure register KPC_EN (ADC_CONF[9]) should be set to 1.

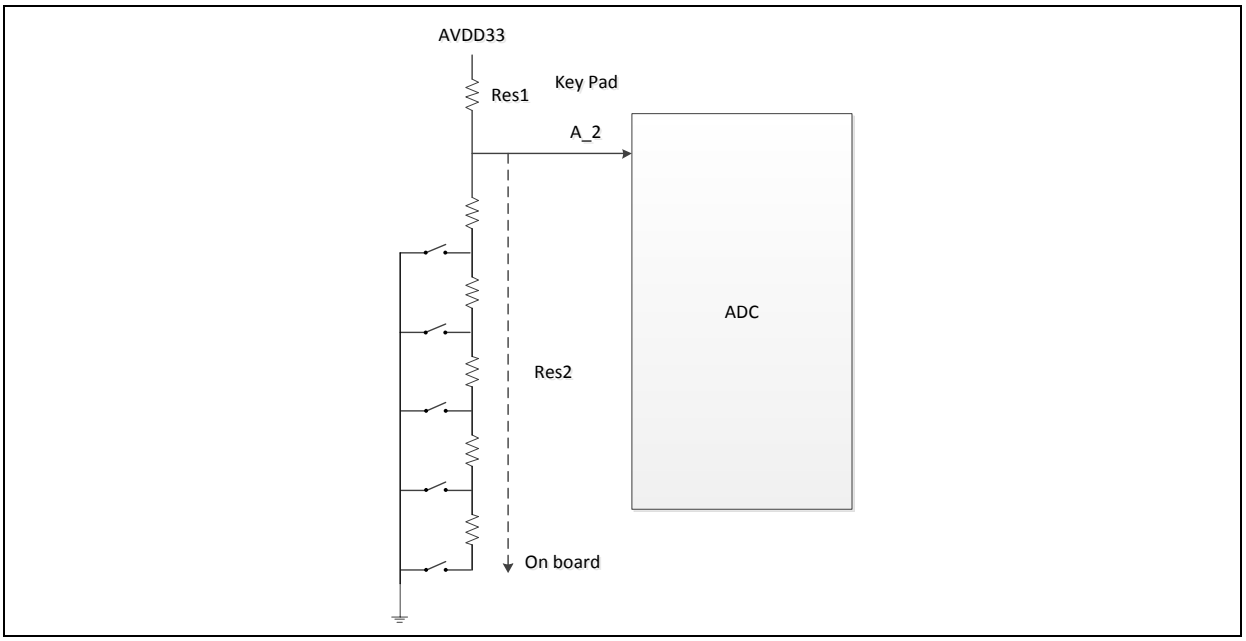



Figure 5.32-5 Key Pad Detection Diagram

If a user applies the Keypad using this structure and meanwhile, he/she needs the interrupt generator, please make sure Res1≤20K ohm and Res2<5.6*Res1. Moreover, a 0.01uF cap is recommended at A_2 on board.

If a user doesn't need the interrupt generator, please ignore the requirement for Res1 and Res2.

5.32.5.9 *4-wire and 5-wire Touch Screen*

The touch screen control logic and the switch could control the 4-wire and 5-wire type touch screen. For ADC configure register T_EN (ADC_CONF[0]) should be set to 1.ADC control register WMSWCH (ADC_CTL[16]) (Wire Mode Switch) for 5-wire/4-wire configuration. The following figures show the interface for 4-wire, 5-wire touch screen respectively.

Note that, the four switches to bias XP, XM, YP, YM have conduction resistance under 5 ohm. And the pull up PMOS have 200K ohm typically.

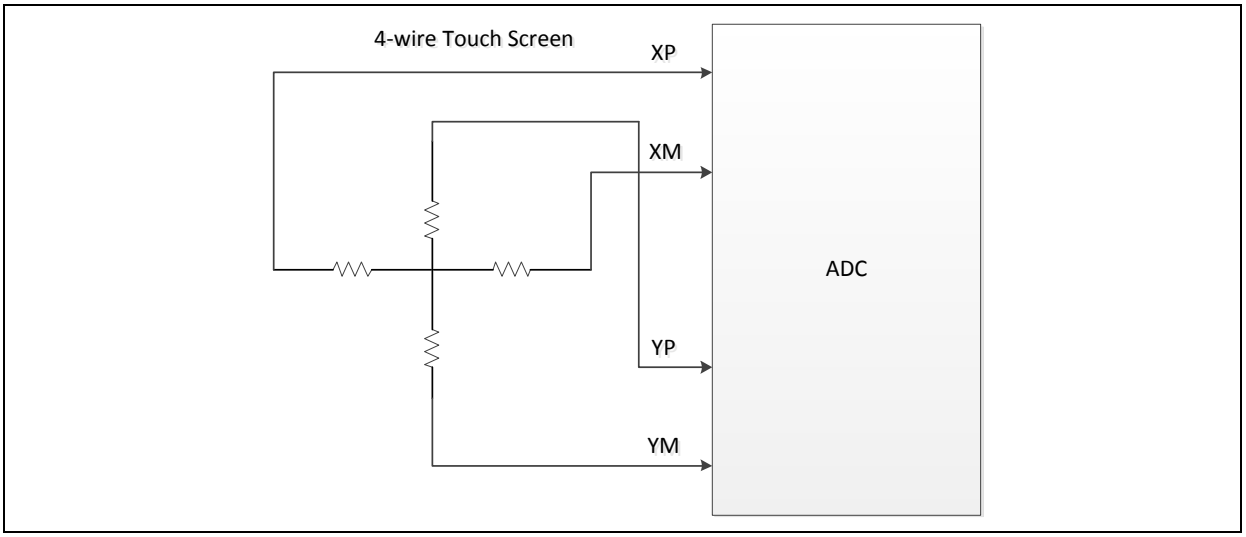



Figure 5.32-6 4-wire Touch Screen Connection Diagram

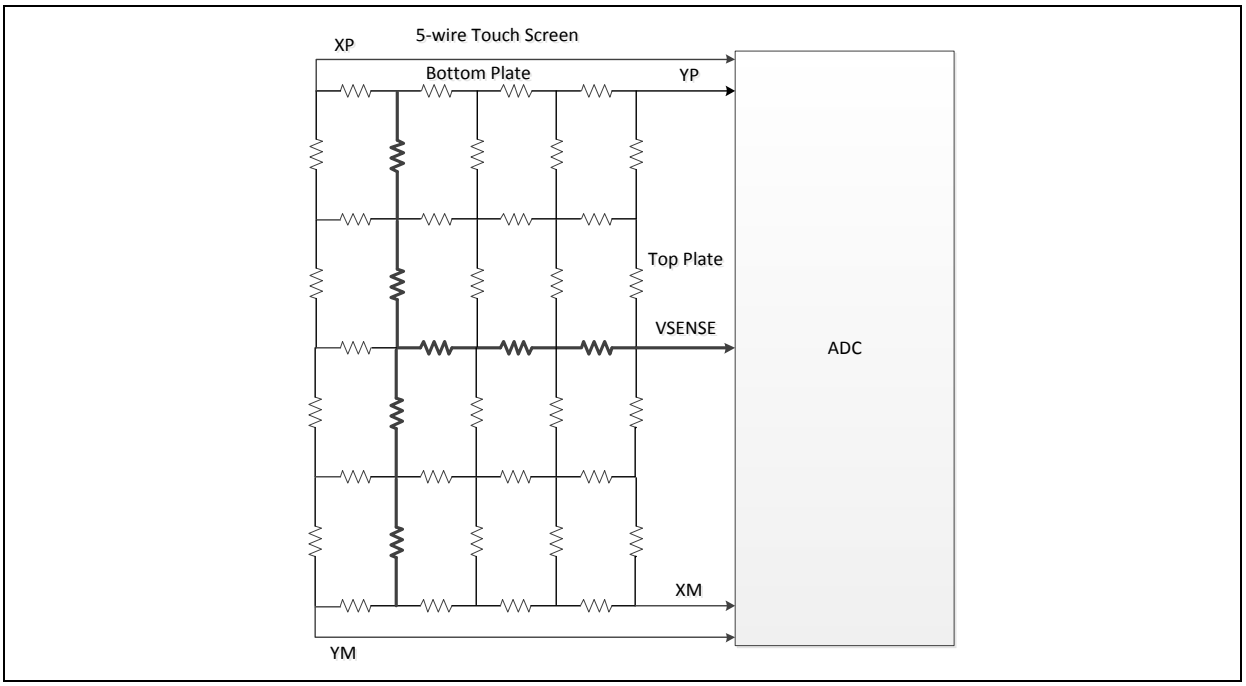



Figure 5.32-7 5-wire Touch Screen Connection Diagram

5.32.5.10 *4-wire Pressure Measurement*

To distinguish pen or finger touch, the pressure of the touch needs to be determined. The IP provides two solutions. The first method requires knowing the X-plate resistance, measurement of the X-Position, and two additional cross-panel measurements (Z1 and Z2) of the touch screen. Use the following Equation to calculate the touch resistance:

$$R_{Touch} = R_{X-plate} \times \frac{X_{position}}{4096} \left(\frac{Z_2}{Z_1} - 1\right)$$

The second method requires knowing both the X-plate and Y-plate resistance, measurement of X-Position and Y-Position, and Z1. Use the following Equation to calculate the touch resistance:

$$R_{Touch} = \frac{R_{X-plate} \times X_{position}}{4096}\left(\frac{4096}{Z_1} - 1\right) - R_{Y-plate}\left(1 - \frac{Y_{position}}{4096}\right)$$

For ADC configure register Z_EN (ADC_CONF[1]) should be set to 1. When Z_EN (ADC_CONF[1]) in ADC_FM register is set; the touch pressure measure Z will be stored in this ADC_ZDATA register.

### 5.32.6 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ADC Base Address:**<br>**ADC_BA = 0xB800_A000** | | | | |
| **ADC_CTL** | ADC_BA+0x00 | R/W | ADC Control | 0x0000_0000 |
| **ADC_CONF** | ADC_BA+0x04 | R/W | ADC Configure | 0x0000_0000 |
| **ADC_IER** | ADC_BA+0x08 | R/W | ADC Interrupt Enable Register | 0x0000_0000 |
| **ADC_ISR** | ADC_BA+0x0C | R/W | ADC Interrupt Status Register | 0x0000_0000 |
| **ADC_WKISR** | ADC_BA+0x10 | R | ADC Wake Up Interrupt Status Register | 0x0000_0000 |
| **ADC_XYDATA** | ADC_BA+0x20 | R | ADC Touch X,Y Position Data | 0x0000_0000 |
| **ADC_ZDATA** | ADC_BA+0x24 | R | ADC Touch Z Pressure Data | 0x0000_0000 |
| **ADC_DATA** | ADC_BA+0x28 | R | ADC Normal Conversion Data | 0x0000_0000 |
| **ADC_VBATDATA** | ADC_BA+0x2C | R | ADC Battery Detection Data | 0x0000_0000 |
| **ADC_KPDATA** | ADC_BA+0x30 | R | ADC Key Pad Data | 0x0000_0000 |
| **ADC_SELFDATA** | ADC_BA+0x34 | R | ADC Self-Test Data | 0x0000_0000 |
| **ADC_XYSORT0** | ADC_BA+0x1F4 | R | ADC Touch XY Position Mean Value Sort 0 | 0x0000_0000 |
| **ADC_XYSORT1** | ADC_BA+0x1F8 | R | ADC Touch XY Position Mean Value Sort 1 | 0x0000_0000 |
| **ADC_XYSORT2** | ADC_BA+0x1FC | R | ADC Touch XY Position Mean Value Sort 2 | 0x0000_0000 |
| **ADC_XYSORT3** | ADC_BA+0x200 | R | ADC Touch XY Position Mean Value Sort 3 | 0x0000_0000 |
| **ADC_ZSORT0** | ADC_BA+0x204 | R | ADC Touch Z Pressure Mean Value Sort 0 | 0x0000_0000 |
| **ADC_ZSORT1** | ADC_BA+0x208 | R | ADC Touch Z Pressure Mean Value Sort 1 | 0x0000_0000 |
| **ADC_ZSORT2** | ADC_BA+0x20C | R | ADC Touch Z Pressure Mean Value Sort 2 | 0x0000_0000 |
| **ADC_ZSORT3** | ADC_BA+0x210 | R | ADC Touch Z Pressure Mean Value Sort 3 | 0x0000_0000 |

### 5.32.7 Register Description

### ADC Control (ADC_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ADC_CTL** | ADC_BA+0x00 | R/W | ADC Control | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | **WMSWCH** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | **WKT_EN** | **WKP_EN** | **PEDE_EN** | **MST** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | | | **PWKP_EN** | **VBG_EN** | **AD_EN** |

| Bits | Description | |
|---|---|---|
| [31:17] | **Reserved** | Reserved. |
| [16] | **WMSWCH** | **Wire Mode Switch for 5-wire/4-wire Configuration**<br>0 = 4-wire mode.<br>1 = 5-wire mode. |
| [15:12] | **Reserved** | Reserved. |
| [11] | **WKT_EN** | **Touch Wake Up Enable**<br>0 = Disable touch wake up.<br>1 = Enable touch wake up. |
| [10] | **WKP_EN** | **Keypad Press Wake Up Enable**<br>0 = Disable key press wake up.<br>1 = Enable key press wake up. |
| [9] | **PEDE_EN** | **Pen Down Event Enable**<br>0 = Disable pen down event interrupt.<br>1 = Enable pen down event interrupt. |
| [8] | **MST** | **Menu Start Conversion**<br>0 = Functional menu not start.<br>1 = Start all enable bit in ADC_FM register.<br>**Note:** This bit is set by software and cleared by hardware when all the jobs listed in ADC_CONF are done. |
| [2] | **PWKP_EN** | **ADC Keypad Power Enable Control**<br>0 = Power down internal keypad circuit.<br>1 = Power on internal keypad circuit. |

| | | |
|---|---|---|
| [1] | **VBG_EN** | **ADC Internal Bandgap Power Control**<br>0 = Power down internal bandgap.<br>1 = Power on internal bandgap. |
| [0] | **AD_EN** | **ADC Power Control**<br>0 = Power down ADC.<br>1 = Power on ADC. |

## ADC Configure (ADC_CONF)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| ADC_CONF | ADC_BA+0x04 | R/W | ADC Configure | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| ADCSAMPCNT | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | HSPEED | DISZMAVEN | DISTMAVEN | REFCNT | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | SELFT_EN | KPC_EN | VBAT_EN |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| REFSEL | | CHSEL | | | NAC_EN | Z_EN | T_EN |

| Bits | Description | |
|---|---|---|
| [31:24] | **ADCSAMPCNT** | **ADC Sample Counter**<br>Set the counter value to extend the ADC start signal period to get more sampling time for precise conversion. |
| [22] | **HSPEED** | **High Speed Enable**<br>Enable ADC to high speed mode. |
| [21] | **DISZMAVEN** | **Display Z Mean Average Enable**<br>Pressure Mean average for Z1 and Z2 function enable. |
| [20] | **DISTMAVEN** | **Display T Mean Average Enable**<br>Touch Mean average for X and Y function enable. |

| | | |
|---|---|---|
| [19:16] | **REFCNT** | **ADC Reference Counter**<br>ADC reference voltage stable counter count value.<br>0 = 1 ADC clock.<br>1 = 2 ADC clock.<br>2 = 4 ADC clock.<br>3 = 8 ADC clock.<br>4 = 16 ADC clock.<br>5 = 32 ADC clock.<br>6 = 64 ADC clock.<br>7 = 128 ADC clock.<br>8 = 256 ADC clock.<br>9 = 512 ADC clock.<br>10 = 1024 ADC clock.<br>11 = 1024 ADC clock.<br>12 = 1024 ADC clock.<br>13 = 1024 ADC clock.<br>14 = 1024 ADC clock.<br>15 = 1024 ADC clock. |
| [15:11] | **Reserved** | Reserved. |
| [10] | **SELFT_EN** | **Selft Test Enable**<br>Selft test function enable. |
| [9] | **KPC_EN** | **Keypad Press Conversion Enable**<br>Keypad press conversion function enable. |
| [8] | **VBAT_EN** | **Voltage Battery Enable**<br>Battery voltage detection function enable. |
| [7:6] | **REFSEL** | **ADC Reference Select**<br>ADC reference voltage select when ADC operate in normal conversion.<br>00 = AGND33 vs 2.5v buffer output, or VREF input.<br>01 = YM vs YP.<br>10 = XM vs XP.<br>11 = AGND33 vs AVDD33. |
| [5:3] | **CHSEL** | **Channel Selection**<br>ADC input channel selection.<br>000 = VBT.<br>001 = VHS.<br>010 = A_2.<br>011 = A_3.<br>100 = YM.<br>101 = YP.<br>110 = XM.<br>111 = XP. |
| [2] | **NACEN** | **Normal AD Conversion Enable**<br>ADC normal conversion function enable |

| | | |
|---|---|---|
| [1] | **Z_EN** | **Press Enable**<br>Press measure function enable |
| [0] | **T_EN** | **Touch Enable**<br>Touch detection function enable |

## ADC Interrupt Enable Register (ADC_IER)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ADC_IER** | ADC_BA+0x08 | R/W | ADC 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** | **PEUE_IEN** | **KPUE_IEN** | **WKP_IEN** | **WKT_IEN** | **PEDE_IEN** | **KPE_IEN** | **M_IEN** |

| Bits | Description | |
|---|---|---|
| [31:7] | **Reserved** | Reserved. |
| [6] | **PEUE_IEN** | **Pen Up Event Interrupt Enable**<br>Pen up event detection interrupt enable. |
| [5] | **KPUE_IEN** | **Keypad Press Up Event Interrupt Enable**<br>Keypad press up event detection interrupt enable. |
| [4] | **WKP_IEN** | **Wake Up Keypad Press Interrupt Enable**<br>Wake up keypad press detection interrupt enable. |
| [3] | **WKT_IEN** | **Wake Up Touch Interrupt Enable**<br>Wake up touch detection interrupt enable. |
| [2] | **PEDE_IEN** | **Pen Down Even Interrupt Enable**<br>Pen down event detection interrupt enable. |
| [1] | **KPE_IEN** | **Keypad Press Event Interrupt Enable**<br>Keypad press event detection interrupt enable. |
| [0] | **M_IEN** | **Menu Interrupt Enable**<br>Function menu complete interrupt enable. |

## ADC Interrupt Status Register (ADC_ISR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| ADC_ISR | ADC_BA+0x0C | R/W | ADC Interrupt Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | INT_TC | INT_KP |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | SELFT_F | KPC_F | VB_F | NAC_F | Z_F | T_F |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | PEUE_F | KPUE_F | PEDE_F | KPE_F | M_F |

| Bits | Description | |
|---|---|---|
| [31:18] | **Reserved** | Reserved. |
| [17] | **INT_TC** | **Interrupt Signal for Touch Screen Touching Detection**<br>This signal is directly from analog macro without de-bouncing and can be used to determine the pen down touch event together with PEDE_F (ADC_ISR[2]) flag. |
| [16] | **INT_KP** | **Interrupt Signal for Keypad Detection**<br>This signal is directly from analog macro without de-bouncing and can be used to determine the keypad up event together with KPE_F (ADC_ISR[1]) flag. |
| [15:14] | **Reserved** | Reserved. |
| [13] | **SELFT_F** | **Self-test Conversion Finish**<br>Function menu self-test conversion finish.<br>**Note:** set by hardware and write 1 to clear this bit. |
| [12] | **KPC_F** | **Keypad Press Conversion Finish**<br>Function menu keypad press conversion finish.<br>**Note:** set by hardware and write 1 to clear this bit. |
| [11] | **VB_F** | **Voltage Battery Conversion Finish**<br>Functional menu battery voltage detection conversion finish.<br>**Note:** set by hardware and write 1 to clear this bit. |
| [10] | **NAC_F** | **Normal AD Conversion Finish**<br>Functional menu normal AD conversion finish.<br>**Note:** set by hardware and write 1 to clear this bit. |
| [9] | **Z_F** | **Press Conversion Finish**<br>Functional menu press measure conversion finish.<br>**Note:** set by hardware and write 1 to clear this bit. |
| [8] | **T_F** | **Touch Conversion Finish** |

| | | |
|---|---|---|
| | | Functional menu touch detection conversion finish.<br>**Note:** set by hardware and write 1 to clear this bit. |
| [7:5] | **Reserved** | Reserved. |
| [4] | **PEUE_F** | **Pen Up Event Flag**<br>Pen up event status indicator.<br>**Note:** set by hardware and write 1 to clear this bit. |
| [3] | **KPUE_F** | **Keypad Press Up Event Flag**<br>keypad press up event status indicator.<br>**Note:** set by hardware and write 1 to clear this bit. |
| [2] | **PEDE_F** | **Pen Down Event Flag**<br>Pen down event status indicator.<br>**Note:** set by hardware and write 1 to clear this bit. |
| [1] | **KPE_F** | **Keypad Press Event Flag**<br>Keypad press event status indicator. |
| [0] | **M_F** | **Menu Complete Flag**<br>Function menu complete status indicator.<br>**Note:** set by hardware and write 1 to clear this bit. |

### ADC Wake Up Interrupt Status Register (WKISR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ADC_WKISR** | ADC_BA+0x10 | R | ADC Wake Up 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** | | | | | | **WPEDE_F** | **WKPE_F** |

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [1] | **WPEDE_F** | **Wake Up Pen Down Event Flag**<br>Pen down event wake up status indicator. |
| [0] | **WKPE_F** | **Wake Up Keypad Press Event Flage**<br>Key press event wake up status indicator. |

## ADC Touch X,Y Position Data (ADC_XYDATA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| ADC_XYDATA | ADC_BA+0x20 | R | ADC Touch X,Y Position Data | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27:16] | **ADC_YDATA** | **ADC Y Data**<br>When T_EN (ADC_CONF[0]) in ADC_FM register is set; the touch y-position will be stored in this register.<br>**Note:** If the DISTMAVEN (ADC_CONF[20]) = 1, both x and y position are the results of the mean average of x and y in ADC_XYSORT0 ~ ADC_XYSORT3. |
| [15:12] | **Reserved** | Reserved. |
| [11:0] | **ADC_XDATA** | **ADC X Data**<br>When T_EN (ADC_CONF[0]) in ADC_FM register is set; the touch x-position will be stored in this register.<br>**Note:** If the DISTMAVEN (ADC_CONF[20]) = 1, both x and y position are the results of the mean average of x and y in ADC_XYSORT0 ~ ADC_XYSORT3. |

## ADC Touch Z Pressure Data (ADC_ZDATA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ADC_ZDATA** | ADC_BA+0x24 | R | ADC Touch Z Pressure Data | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | **ADC_Z2DATA** | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **ADC_Z2DATA** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | **ADC_Z1DATA** | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **ADC_Z1DATA** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [26:16] | **ADC_Z2DATA** | **ADC Z2 Data**<br>When Z_EN (ADC_CONF[1]) in ADC_FM register is set; the touch pressure measure Z2 will be stored in this register.<br>**Note:** If the DISZMAVEN (ADC_CONF[21]) = 1, both Z1 and Z2 data is the results of the mean average of Z1 and Z2 in ADC_ZSORT0 ~ ADC_ZSORT3. |
| [15:12] | **Reserved** | Reserved. |
| [11:0] | **ADC_Z1DATA** | **ADC Z1 Data**<br>When Z_EN (ADC_CONF[1]) in ADC_FM register is set; the touch pressure measure Z1 will be stored in this register.<br>**Note:** If the DISZMAVEN (ADC_CONF[21])= 1, both Z1 and Z2 data is the results of the mean average of Z1 and Z2 in ADC_ZSORT0 ~ ADC_ZSORT3. |

## ADC Normal Conversion Data (ADC_DATA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ADC_DATA** | ADC_BA+0x28 | R | ADC Normal Conversion Data | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:12] | **Reserved** | Reserved. |
| [11:0] | **ADC_DATA** | **ADC Data**<br>When NAC_EN (ADC_CONF[2]) in ADC_FM is enable, the AD converting result with corresponding channel is stored in this register. |

### ADC Battery Detection Data (ADC_VBATDATA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ADC_VBATDATA** | ADC_BA+0x2C | R | ADC Battery Detection Data | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:12] | **Reserved** | Reserved. |
| [11:0] | **ADC_VBATDATA** | **ADC Voltage Battery Data**<br>When VBAT_EN (ADC_CONF[8]) in ADC_FM register is enable, the battery voltage detect result is stored in this register. |

## ADC Key Pad Data (ADC_KPATA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ADC_KPDATA** | ADC_BA+0x30 | R | ADC Key Pad Data | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:12] | **Reserved** | Reserved. |
| [11:0] | **ADC_KPDATA** | **ADC Key Pad Data**<br>When KPC_EN (ADC_CONF[9]) in ADC_FM register is enable, the key pad detect result is stored in this register. |

## ADC Self-Test Data (ADC_SELFDATA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ADC_SELFDATA** | ADC_BA+0x34 | R | ADC Self-Test Data | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27:16] | **ADC_AGND** | **ADC Self-test AGND Data**<br>When SELFT_EN (ADC_CONF[10]) in ADC_FM register is enable, the AGND self-test result is store in this register. |
| [15:12] | **Reserved** | Reserved. |
| [11:0] | **ADC_AVDD** | **ADC Self-test AVDD Data**<br>When SELFT_EN (ADC_CONF[10]) in ADC_FM register is enable, the AVDD self-test result is store in this register. |

### ADC Touch XY Position Mean Value Sort 0 (ADC_XYSORT0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ADC_XYSORT0** | ADC_BA+0x1F4 | R | ADC Touch XY Position Mean Value Sort 0 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | **Y_SORT0** | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Y_SORT0** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | **X_SORT0** | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **X_SORT0** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27:16] | **Y_SORT0** | **Y Position Sort Data 0**<br>Y position mean average sort data 0. |
| [15:12] | **Reserved** | Reserved. |
| [11:0] | **X_SORT0** | **X Position Sort Data 0**<br>X position mean average sort data 0. |

## ADC Touch XY Position Mean Value Sort 1 (ADC_XYSORT1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ADC_XYSORT1** | ADC_BA+0x1F8 | R | ADC Touch XY Position Mean Value Sort 1 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | Y_SORT1 | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Y_SORT1 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | X_SORT1 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| X_SORT1 | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27:16] | **Y_SORT1** | **Y Position Sort Data 1**<br>Y position mean average sort data 1. |
| [15:12] | **Reserved** | Reserved. |
| [11:0] | **X_SORT1** | **X Position Sort Data 1**<br>X position mean average sort data 1. |

## ADC Touch XY Position Mean Value Sort 2 (ADC_XYSORT2)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ADC_XYSORT2** | ADC_BA+0x1FC | R | ADC Touch XY Position Mean Value Sort 2 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | Y_SORT2 | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Y_SORT2 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | X_SORT2 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| X_SORT2 | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27:16] | **Y_SORT2** | **Y Position Sort Data 2**<br>Y position mean average sort data 2. |
| [15:12] | **Reserved** | Reserved. |
| [11:0] | **X_SORT2** | **X Position Sort Data 2**<br>X position mean average sort data 2. |

## ADC Touch XY Position Mean Value Sort 3 (ADC_XYSORT3)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ADC_XYSORT3** | ADC_BA+0x200 | R | ADC Touch XY Position Mean Value Sort 3 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | **Y_SORT3** | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Y_SORT3** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | **X_SORT3** | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **X_SORT3** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27:16] | **Y_SORT3** | **Y Position Sort Data 3**<br>Y position mean average sort data 3. |
| [15:12] | **Reserved** | Reserved. |
| [11:0] | **X_SORT3** | **X Position Sort Data 3**<br>X position mean average sort data 3. |

### ADC Touch Z Pressure Mean Value Sort 0 (ADC_ZSORT0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ADC_ZSORT0** | ADC_BA+0x204 | R | ADC Touch Z Pressure Mean Value Sort 0 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | **Z2_SORT0** | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Z2_SORT0** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | **Z1_SORT0** | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Z1_SORT0** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27:16] | **Z2_SORT0** | **Z2 Position Sort Data 0**<br>Z2 position Mean average sort data 0. |
| [15:12] | **Reserved** | Reserved. |
| [11:0] | **Z1_SORT0** | **Z1 Position Sort Data 0**<br>Z1 position Mean average sort data 0. |

## ADC Touch Z Pressure Mean Value Sort 1 (ADC_ZSORT1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ADC_ZSORT1** | ADC_BA+0x208 | R | ADC Touch Z Pressure Mean Value Sort 1 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | **Z2_SORT1** | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Z2_SORT1** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | **Z1_SORT1** | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Z1_SORT1** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27:16] | **Z2_SORT1** | **Z2 Position Sort Data 1**<br>Z2 position Mean average sort data 1. |
| [15:12] | **Reserved** | Reserved. |
| [11:0] | **Z1_SORT1** | **Z1 Position Sort Data 1**<br>Z1 position Mean average sort data 1. |

## ADC Touch Z Pressure Mean Value Sort 2 (ADC_ZSORT2)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ADC_ZSORT2** | ADC_BA+0x20C | R | ADC Touch Z Pressure Mean Value Sort 2 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | **Z2_SORT2** | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Z2_SORT2** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | **Z1_SORT2** | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Z1_SORT2** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27:16] | **Z2_SORT2** | **Z2 Position Sort Data 2**<br>Z2 position Mean average sort data 2. |
| [15:12] | **Reserved** | Reserved. |
| [11:0] | **Z1_SORT2** | **Z1 Position Sort Data 2**<br>Z1 position Mean average sort data 2. |

## ADC Touch Z Pressure Mean Value Sort 3 (ADC_ZSORT3)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ADC_ZSORT3** | ADC_BA+0x210 | R | ADC Touch Z Pressure Mean Value Sort 3 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | **Z2_SORT3** | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Z2_SORT3** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | **Z1_SORT3** | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Z1_SORT3** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27:16] | **Z2_SORT3** | **Z2 Position Sort Data 3**<br>Z2 position Mean average sort data 3. |
| [15:12] | **Reserved** | Reserved. |
| [11:0] | **Z1_SORT3** | **Z1 Position Sort Data 3**<br>Z1 position Mean average sort data 3. |

## 5.33 Keypad Interface (KPI)

### 5.33.1 Overview

The Keypad Interface (KPI) is an APB slave with configurable minimum 2-row up to 4-row scan output and minimum 2-column up to 8-column scan input. Any keys in the array pressed or released are de-bounced and generate an interrupt.

The KPI supports release multiple keys, press multiple keys scan interrupt for chip reset. If the 3 pressed keys matches with the 3 keys defined in KPI3KCONF, it will generate an chip reset depend on the EN3KYRST (KPI3KCONF[24]) setting. The interrupt is generated whenever it detects any key in the keypad pressing or releasing or waking up from IDLE. User can know the interrupt source by querying KPISTATUS register.

### 5.33.2 Features

- Matrix keypad interface (maximum 4x8 array, and minimum 2x2 array).
- Programmable de-bounce time.
- Low-power wakeup mode.
- Programmable three-key reset.
- Generate interrupt and update all the keys (maximum 32 keys, minimum 4 keys) information (press/release) every time the user pressing or releasing.

### 5.33.3 Block Diagram

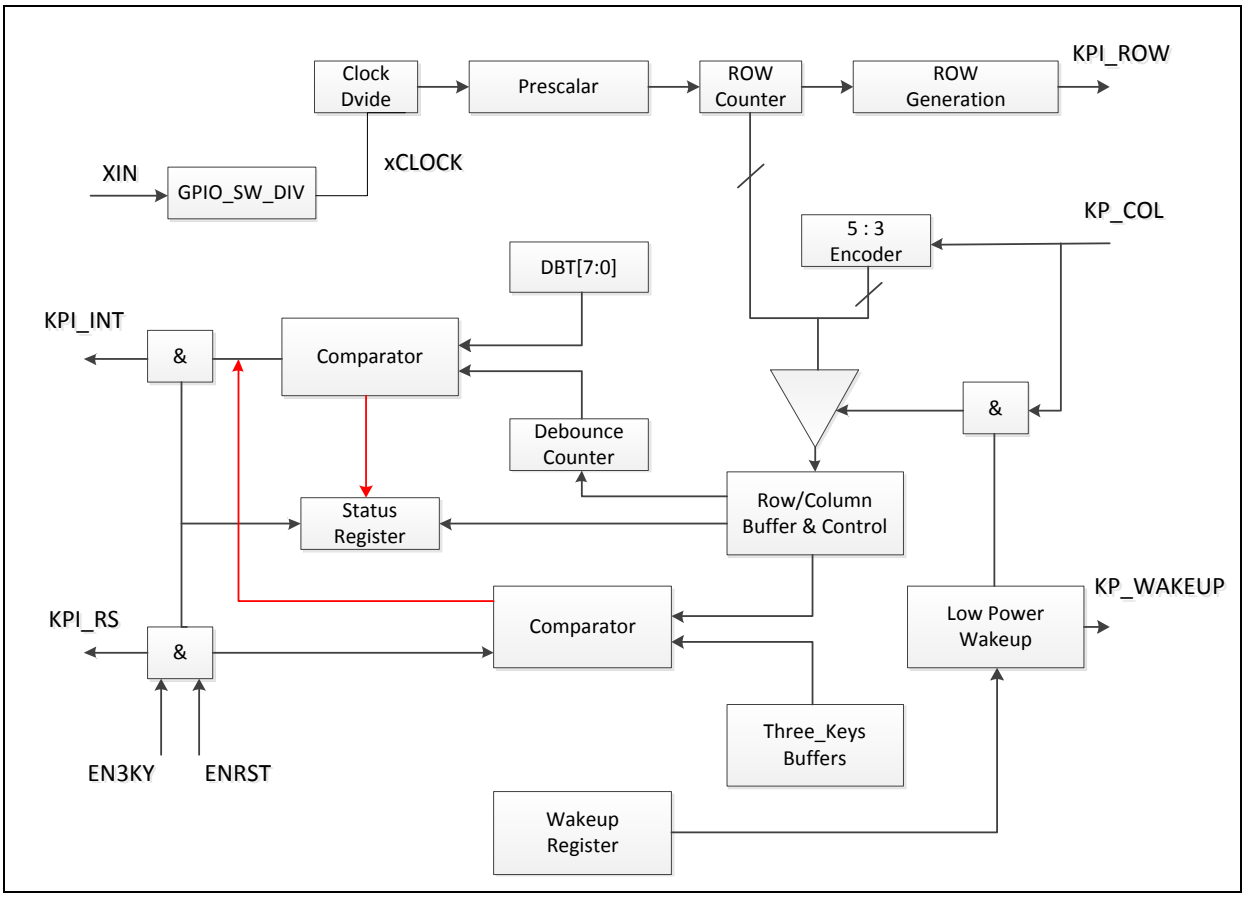



Figure 5.33-1 Keypad Controller Block Diagram

### 5.33.4 Basic Configuration

The KPI pin functions are configured in SYS_MFP_GPHL and SYS_MFP_GPHH register.

The KPI peripheral clock can be enabled in KPI (CLK_PCLKEN1[25]). The KPI engine clock source is selected by KPI_S (CLK_DIVCTL7[15]) and KPI engine clock divider is determined by KPI_N (CLK_DIVCTL7[14:8]).

### 5.33.5 Functional Description

Keypads are often used as a primary input device for embedded microcontrollers. The keypads actually consist of a number of switches, connected in a row/column arrangement as shown in Figure.

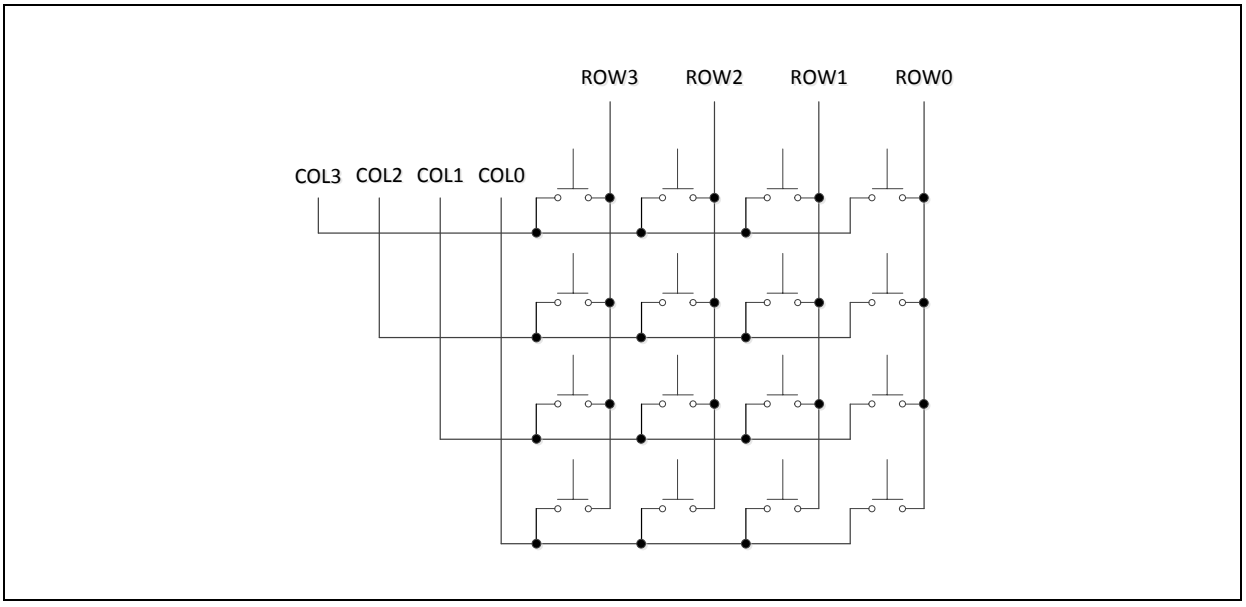



Figure 5.33-2 Keypad Connection

In order for the microcontroller to scan the keypad, it outputs a nibble to force one (only one) of the rows low and then reads the columns to see if any buttons in that column have been pressed. The columns are pulled up by the internal weak pull-up resistor.

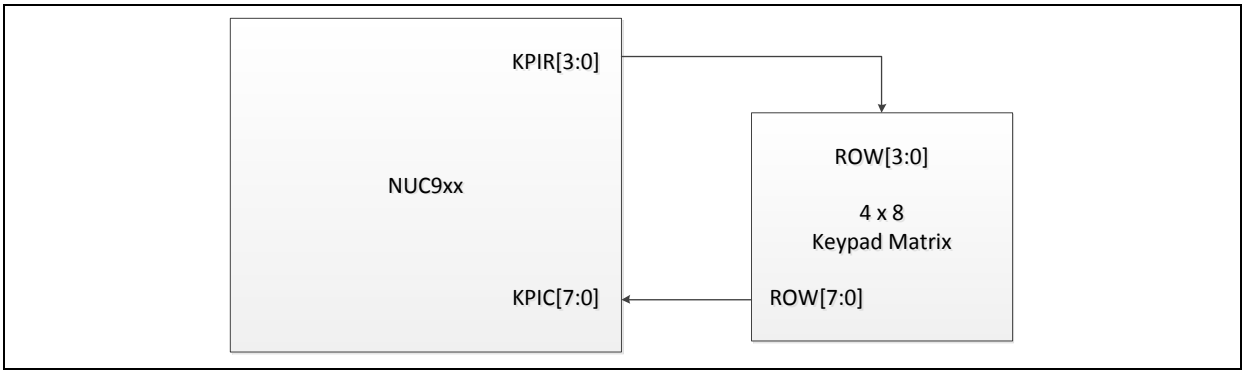



Figure 5.33-3 Keypad Interface

Consequently, as long as no buttons are pressed, the microcontroller sees a logic high on each of

the pins attached to the keypad columns. The nibble driven onto the rows always contains only a single 0. The only way the microcontroller can find a 0 on any column pin is for the keypad button to be pressed that connects the row set to 0 to a row. The controller knows which column is at a 0-level and which column reads 0, allowing it to determine which key is pressed. For the keypad scan timing as show in below.

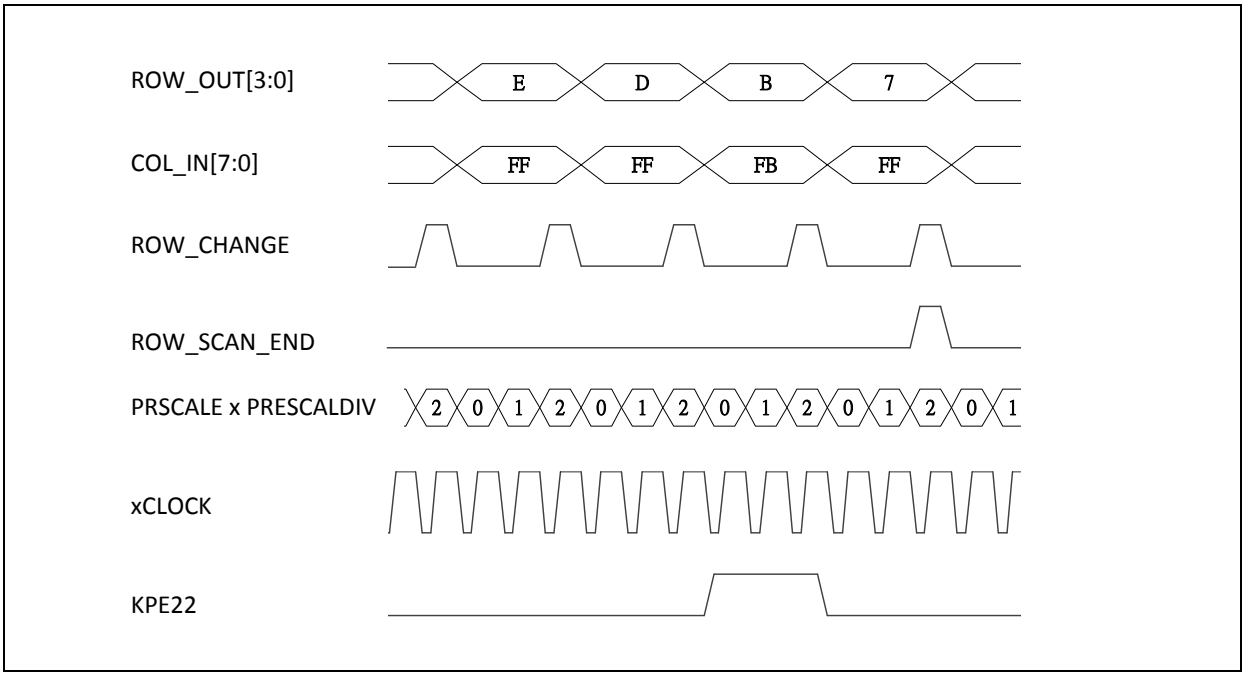



Figure 5.33-4 4x8 Keypad Scan Timing Diagram

### 5.33.6 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **KPI Base Address:**<br>**KPI_BA = 0xB800_8000** | | | | |
| **KPICONF** | KPI_BA+0x00 | R/W | Keypad Configuration | 0x0000_0000 |
| **KPI3KCONF** | KPI_BA+0x04 | R/W | Keypad 3-Keys Configuration | 0x0000_0000 |
| **KPISTATUS** | KPI_BA+0x08 | R | Keypad Status | 0x0000_0000 |
| **KPIRSTC** | KPI_BA+0x0C | R/W | Keypad Reset Period Controller | 0x0000_0000 |
| **KPIKEST** | KPI_BA+0x10 | R | Keypad Key State Indicator | 0xFFFF_FFFF |
| **KPIKPE** | KPI_BA+0x18 | R/W | Press Key Event Indicator | 0x0000_0000 |
| **KPIKRE** | KPI_BA+0x20 | R/W | Release Key Event Indicator | 0x0000_0000 |
| **KPIPRESCALDIV** | KPI_BA+0x28 | R/W | Pre-Scale Divider | 0x0000_001F |

### 5.33.7 Register Description

## Keypad Configuration (KPICONF)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| KPICONF | KPI_BA+0x00 | R/W | Keypad Configuration | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | KROW[1:0] | | Reserved | KCOL[2:0] | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | DB_EN | Reserved | DBCLKSEL[3:0] | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PRESCALE[7:0] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | WAKEUP | ODEN | INTEN | RKINTEN | PKINTEN | ENKP |

| Bits | Description | |
|---|---|---|
| [31:30] | **Reserved** | Reserved. |
| [29:28] | **KROW** | **Keypad Matrix ROW Number**<br>The keypad matrix is set by ROW x COL. The ROW number can be set 2, 3, 4.<br>000 = Reserved.<br>001 = Keypad matrix ROW number is 2.<br>010 = Keypad matrix ROW number is 3.<br>011 = Keypad matrix ROW number is 4(Default).<br>100 = Reserved.<br>101 = Reserved.<br>110 = Reserved.<br>111 = Reserved. |
| [27] | **Reserved** | Reserved. |
| [26:24] | **KCOL** | **Keypad Matrix COL Number**<br>The keypad matrix is set by ROW x COL. The COL number can be set 2, 3, 4, 5, 6, 7, or 8.<br>000 = Keypad matrix COLUMN number is 8(Default).<br>001 = Keypad matrix COLUMN number is 2.<br>010 = Keypad matrix COLUMN number is 3.<br>011 = Keypad matrix COLUMN number is 4.<br>100 = Keypad matrix COLUMN number is 5.<br>101 = Keypad matrix COLUMN number is 6.<br>110 = Keypad matrix COLUMN number is 7.<br>111 = Reserved. |
| [21] | **DB_EN** | **Scan in Signal De-bounce Enable**<br>0 = The de-bounce function is disabled.<br>1 = The de-bounce function is enabled. |

| | | |
|---|---|---|
| [20] | **Reserved** | Reserved. |
| [19:16] | **DBCLKSEL** | **Scan in De-bounce Sampling Cycle Selection**<br>0000 = Sample interrupt input once per 1 clocks.<br>0001 = Sample interrupt input once per 2 clocks.<br>0010 = Sample interrupt input once per 4 clocks.<br>0011 = Sample interrupt input once per 8 clocks.<br>0100 = Sample interrupt input once per 16 clocks.<br>0101 = Sample interrupt input once per 32 clocks.<br>0110 = Sample interrupt input once per 64 clocks.<br>0111 = Sample interrupt input once per 128 clocks.<br>1000 = Sample interrupt input once per 256 clocks.<br>1001 = Sample interrupt input once per 2*256 clocks.<br>1010 = Sample interrupt input once per 4*256 clocks.<br>1011 = Sample interrupt input once per 8*256 clocks.<br>1100 = Sample interrupt input once per 16*256 clocks.<br>1101 = Sample interrupt input once per 32*256 clocks.<br>1110 = Reserved.<br>1111 = Reserved.<br>**Note:**<br>row scan time > 2 * debounce sampling cycle.<br>row scan time = prescale * 32 (xclock).<br>xCLOCK = 1MHz ~32KHz.<br>bouncing time last for 1ms, for example,if xCLOCK = 1MHz.<br>debounce sampling cycle should choose 1024 xclock .<br>row scan time should chose 2048 xCLOCK,suppose PrescaleDivider = 0x1F,then.<br>prescale =2048/32 = 64. |
| [15:8] | **PRESCALE** | **Row Scan Cycle Pre-scale Value**<br>This value is used to pre-scale row scan cycle.<br>The pre-scale counter is clocked by the divided crystal clock, xCLOCK.<br>The PRESCALE (KPICONF[15:8]) value is from 1 to 256.<br>Eg.If the crystal clock is 1Mhz then the xCLOCK period is 1us.<br>If the keypad matric is 3x3 then each row scan time = xCLOCK x PRESCALE x PrescaleDivider.<br>Key array scan time = Each row scan time x ROWS.<br>Example scan time for PRESCALE = 0x41,and PrescaleDivider = 0x1F.<br>Each row scan time = 1us x 65 x 32 = 2.08ms.<br>Scan time = 2.08 x 3 = 6.24ms.<br>**Notes:**When PRESCALE (KPICONF[15:8]) is determined, De-bounce sampling cycle should not exceed the half of (PRESCALE x PrescaleDivider ),in the above example maximum DBCLKSEL (KPICONF[19:16]) should be 4*256 xCLOCK ,bouncing time is 1ms |
| [7:6] | **Reserved** | Reserved. |
| [5] | **WAKEUP** | **Lower Power Wakeup Enable**<br>Setting this bit enables low power wakeup.<br>0 = Not enable.<br>1 = Wakeup enable. |

| | | |
|---|---|---|
| [4] | **ODEN** | **Open Drain Enable**<br>If there are more than one key are pressed in the same column, then “short-circuit” will appear between active and inactive scan row. Software can set this bit HIGH to enable scan output KROW (KPICONF[29:28]) pins work as “open-drain” to avoid the “short-circuit”.<br>0 = Push-Pull drive.<br>1 = Open drain. |
| [3] | **INTEN** | **Key Interrupt Enable Control**<br>0 = Disable the keypad interrupt.<br>1 = Enable the keypad interrupt. |
| [2] | **RKINTEN** | **Release Key Interrupt Enable Control**<br>The keypad controller will generate an interrupt when the controller detects keypad status changes from press to release.<br>0 = Disable the keypad release interrupt.<br>1 = Enable the keypad release interrupt. |
| [1] | **PKINTEN** | **Press Key Interrupt Enable Control**<br>The keypad controller will generate an interrupt when the controller detects any effective key press<br>0 = Disable the keypad press interrupt.<br>1 = Enable the keypad press interrupt. |
| [0] | **ENKP** | **Keypad Scan Enable**<br>Setting this bit high enable the key scan function.<br>0 = Disable keypad scan.<br>1 = Enable keypad scan. |

## Keypad 3-Keys Configuration (KPI3KCONF)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| KPI3KCONF | KPI_BA+0x04 | R/W | Keypad 3-Keys Configuration | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | EN3KYRST |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | K32R[1:0] | | K32C[2:0] | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | K31R[1:0] | | K31C[2:0] | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | K30R[1:0] | | K30C[2:0] | | |

| Bits | Description | |
|---|---|---|
| [31:25] | **Reserved** | Reserved. |
| [24] | **EN3KYRST** | **Enable Three-key Reset**<br>Setting this bit enable hardware reset when three-key is detected.<br>0 = Three-key function is disable.<br>1 = Three-key function is enable. |
| [23:21] | **Reserved** | Reserved. |
| [20:19] | **K32R** | **The #2 Key Row Address**<br>The #2 means the row address and the column address is the highest of the specified 3-keys. |
| [18:16] | **K32C** | **The #2 Key Column Address.** |
| [15:13] | **Reserved** | Reserved. |
| [12:11] | **K31R** | **The #1 Key Row Address**<br>The #1 means the row address and the column address is the 2nd of the specified 3-keys. |
| [10:8] | **K31C** | **The #1 Key Column Address.** |
| [7:5] | **Reserved** | Reserved. |
| [4:3] | **K30R** | **The #0 Key Row Address**<br>The #0 means the row address and the column address is the lowest of the specified 3-keys. |
| [2:0] | **K30C** | **The #0 Key Column Address.** |

## Keypad Status (KPISTATUS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| KPISTATUS | KPI_BA+0x08 | R | Keypad Status | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | RROW3 | RROW2 | RROW1 | RROW0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | PROW3 | PROW2 | PROW1 | PROW0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | PKEY_INT | RKEY_INT | KEY_INT | RST_3KEY | PDWAKE |

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [19:16] | **RROWn** | **Release Key Row Coordinate**<br>Show the row of the release key, for example if key (2,x) was released ,x = 0,1..7,then RROW (KPISTATUS[19:16]) will show 0100.<br>**Note:**n = 0,1..3. |
| [11:8] | **PROWn** | **Press Key Row Coordinate**<br>Show the row of the press key, for example if key (2,x) was pressed , x = 0,1..7,then PROW (KPISTATUS[11:8]) will show 0000_0100.<br>**Note:**n = 0,1..3. |
| [7:5] | **Reserved** | Reserved. |
| [4] | **PKEY_INT** | **Press Key Interrupt**<br>This bit indicates that some keys (one or multiple key) were pressed.<br>Read<br>0 = None key press.<br>1 = At least one key press.<br>**Notes:**In order to clear PKEY_INT (KPISTATUS[4]), software must clear each pressing event that are shown on KPE (KPIKPE[31:0]).<br>C code example:<br>DWORD PKE.<br>PKE = reg_read(KPIKPE).<br>Reg_write(KPIKPE,PKE). |

| | | |
|---|---|---|
| [3] | **RKEY_INT** | **Release Key Interrupt**<br>This bit indicates that some keys (one or multiple key) were released.<br>Read<br>0 = None key release.<br>1 = At least one key release.<br>**Notes:**In order to clear RKEY_INT (KPISTATUS[3]), software must clear each releasing event that are shown on “key releasing event”.<br>C code example:<br>DWORD RKE.<br>PKE = reg_read(KPIKRE).<br>Reg_write(KPIKRE,RKE). |
| [2] | **KEY_INT** | **Key Interrupt**<br>This bit indicates the key scan interrupt is active when any key press or release or wakeup.<br>Read<br>0 = Not reset.<br>1 = Key press/release/wakeup interrupt occur. |
| [1] | **RST_3KEY** | **3-keys Reset Flag**<br>This bit will be set after 3-keys reset occur.<br>Read<br>0 = Not reset.<br>1 = 3 keys reset interrupt occur.<br>Write<br>0 = No operation.<br>1 = Clear interrupt flag. |
| [0] | **PDWAKE** | **Power Down Wakeup Flag**<br>This flag indicates the chip is wakeup from power down by keypad.<br>Read<br>0 = Not wakeup.<br>1 = Wakeup up by keypad.<br>Write<br>0 = No operation.<br>1 = Clear interrupt flag. |

## Keypad Reset Period Controller (KPIRSTC)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **KPIRSTC** | KPI_BA+0x0C | R/W | Keypad Reset Period Controller | 0x0000_0000 |

| 31 | 30 | 29 | 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** |
| **RSTC[7:0]** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **RSTC** | **3-key Reset Period Count**<br>The keypad controller generates a reset signal when it detects 3-key match condition, if the EN3KYRST (KPI3KCONF[24]) is set. The RSTC (KPIRSTC[7:0]) is used to control the reset period.<br>Reset period = 64 * RSTC * xCLOCK. |

## Keypad Key State Indicator (KPIKEST)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| KPIKEST | KPI_BA+0x10 | R | Keypad Key State Indicator | 0xFFFF_FFFF |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| KEST37 | KEST36 | KEST35 | KEST34 | KEST33 | KEST32 | KEST31 | KEST30 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| KEST27 | KEST26 | KEST25 | KEST24 | KEST23 | KEST22 | KEST21 | KEST20 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| KEST17 | KEST16 | KEST15 | KEST14 | KEST13 | KEST12 | KEST11 | KEST10 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| KEST07 | KEST06 | KEST05 | KEST04 | KEST03 | KEST02 | KEST01 | KEST00 |

| Bits | Description | |
|---|---|---|
| [31:0] | **KESTmn** | **Keypad State Indicator**<br>0 = Key mn is pressing.<br>1 = key mn is releasing.<br>**Note:**KESTmn: m is row number, n is column number. |

## Press Key Event Indicator (KPIKPE)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| KPIKPE | KPI_BA+0x18 | R/W | Press Key Event Indicator | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| KPE37 | KPE36 | KPE35 | KPE34 | KPE33 | KPE32 | KPE31 | KPE30 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| KPE27 | KPE26 | KPE25 | KPE24 | KPE23 | KPE22 | KPE21 | KPE20 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| KPE17 | KPE16 | KPE15 | KPE14 | KPE13 | KPE12 | KP11 | KPE10 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| KPE07 | KPE06 | KPE05 | KPE04 | KPE03 | KPE02 | KPE01 | KPE00 |

| Bits | Description | |
|---|---|---|
| [31:0] | **KPEmn** | **Press Key Event Indicator**<br>key press key event change indicator.<br>Hardware will set this bit, software should clear this bit by writing 1.<br>Software can clear PKEY_INT (KPISTATUS[4]) by writing 1 bit by bit to this register.<br>0 = None key event.<br>1 = Corresponding key have a high to low event change.<br>**Note:**KPEmn: m is row number, n is column number. |

### Release Key Event Indicator (KPIKRE)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **KPIKRE** | KPI_BA+0x20 | R/W | Release Key Event Indicator | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| KRE37 | KRE36 | KRE35 | KRE34 | KRE33 | KRE32 | KRE31 | KRE30 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| KRE27 | KRE26 | KRE25 | KRE24 | KRE23 | KRE22 | KRE21 | KRE20 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| KRE17 | KRE16 | KRE15 | KRE14 | KRE13 | KRE12 | KRE11 | KRE10 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| KRE07 | KRE06 | KRE05 | KRE04 | KRE03 | KRE02 | KRE01 | KRE00 |

| Bits | Description | |
|---|---|---|
| [31:0] | **KREmn** | **Release Key Event Indicator**<br>key release key event change indicator.<br>Hardware will set this bit, software should clear this bit by writing 1.<br>Software can clear RKEY_INT (KPISTATUS[3]) by writing 1 bit by bit to this register.<br>0 = None key event.<br>1 = Corresponding key have a high to low event change.<br>**Note:**KREmn: m is row number, n is column number. |

## Pre-Scale Divider (KPIPRESCALDIV)

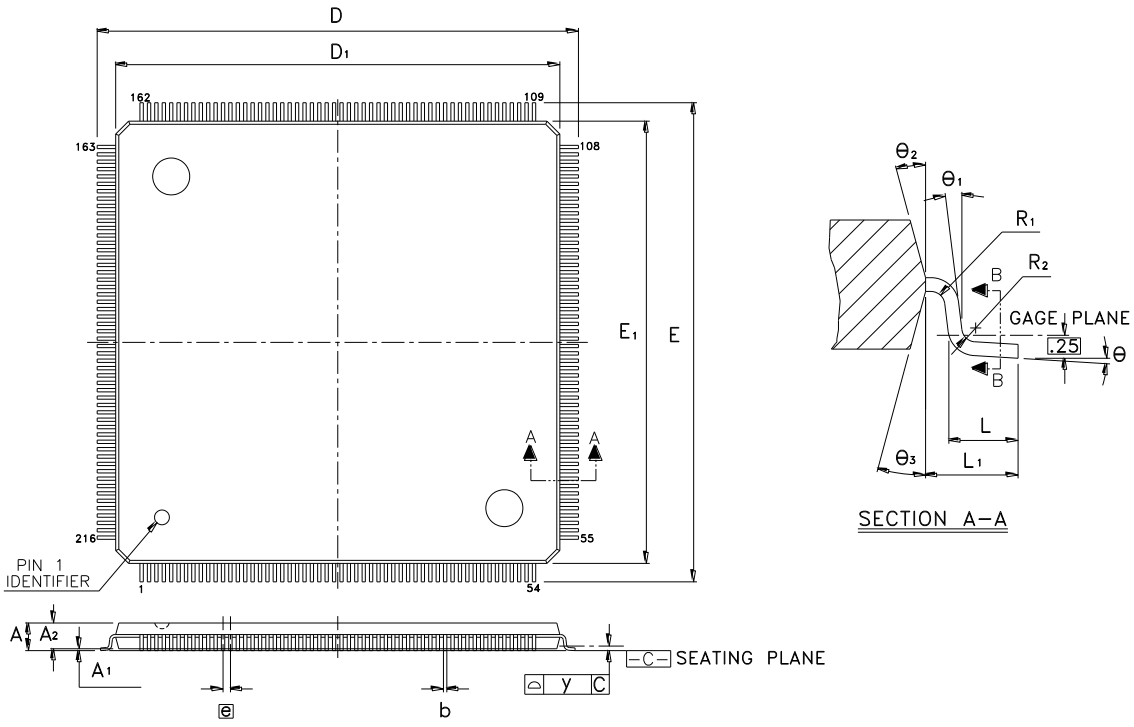
| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **KPIPRESCALDIV** | KPI_BA+0x28 | R/W | Pre-Scale Divider | 0x0000_001F |

| 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** |
| **PRESCALDIV[7:0]** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **PRESCALDIV** | **Pre-scale Divide**<br>This value is used to divide PRESCALE that is set in KPICONF[15:8]. The Prescale divider counter is clocked by the divided crystal clock, xCLOCK. The pre-scale divider number is from 1 to 256.eg.If the crystal clock is 1Mhz then the xCLOCK period is 1us. If the keypad matrix is 3x3 then<br>Each row scan time = xCLOCK x PRESCALE x PrescaleDivider.<br>Key array scan time = Each row scan time x ROWS.<br>Example scan time for PRESCALE = 0x41,and PrescaleDivider = 0x1F.<br>Each row scan time = 1us x 65 x 32 = 2.08ms.<br>Scan time = 2.08 x 3 = 6.24ms.<br>**Notes:**When PRESCALE (KPICONF[15:8]) is determined, De-bounce sampling cycle should not exceed the half of (PRESCALE x PrescaleDivider ),in the above example maximum DBCLKSEL (KPICONF[19:16]) should be 4*256 xclock ,bouncing time is 1ms |

# 6 PACKAGE DIMENSIONS

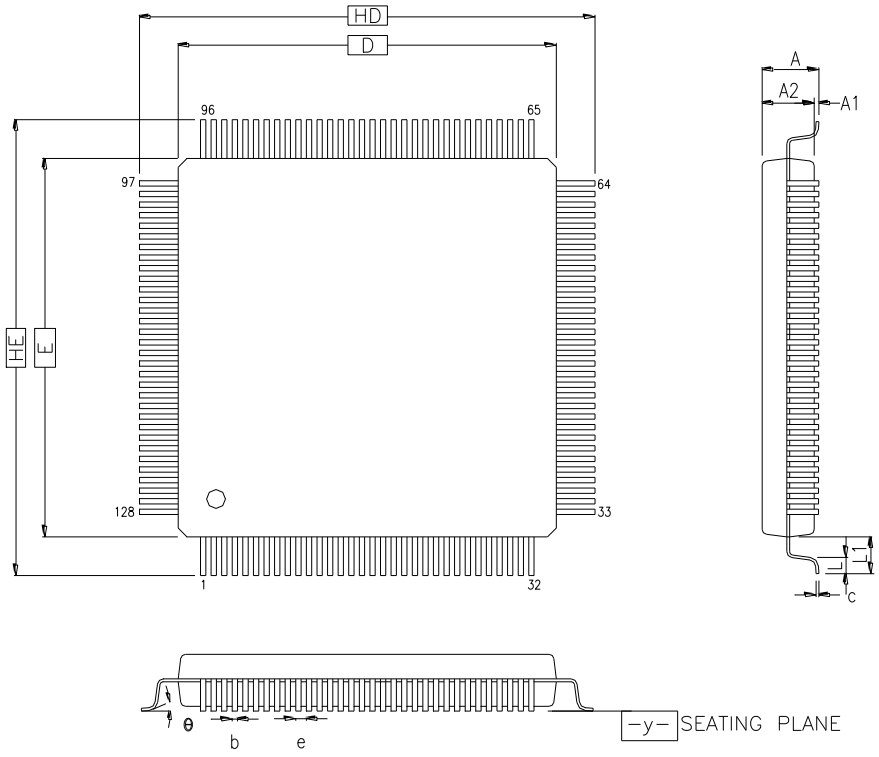
## 6.1 LQFP 216L (24x24x1.4mm footprint 2.0mm)




Controlling Dimension :Millimeters

| Symbol | Dimension in mm | | | Dimension in 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 |
| b | 0.13 | 0.18 | 0.23 | 0.005 | 0.007 | 0.009 |
| $b_1$ | 0.13 | 0.16 | 0.19 | 0.005 | 0.006 | 0.007 |
| c | 0.09 | 0.14 | 0.20 | 0.004 | 0.006 | 0.008 |
| D | 25.85 | 26.00 | 26.15 | 1.018 | 1.024 | 1.030 |
| $D_1$ | 23.90 | 24.00 | 24.10 | 0.941 | 0.945 | 0.949 |
| E | 25.85 | 26.00 | 26.15 | 1.018 | 1.024 | 1.030 |
| $E_1$ | 23.90 | 24.00 | 24.10 | 0.941 | 0.945 | 0.949 |
| e | 0.40 BSC | | | 0.0157 BSC | | |
| L | 0.45 | 0.60 | 0.75 | 0.018 | 0.024 | 0.030 |
| $L_1$ | 1.00 REF | | | 0.039 REF | | |
| $R_1$ | 0.08 | — | — | 0.003 | — | — |
| $R_2$ | 0.08 | — | — | 0.003 | — | — |
| θ | 0° | 3.5° | 7° | 0° | 3.5° | 7° |
| $θ_1$ | 0° | — | — | 0° | — | — |
| $θ_2$ | 11° | 12° | 13° | 11° | 12° | 13° |
| $θ_3$ | 11° | 12° | 13° | 11° | 12° | 13° |
| y | 0.08 | | | 0.00314 | | |

## 6.2 LQFP 128L (14x14x1.4mm footprint 2.0mm)




COTROL DIMENSIONS ARE IN MILLIMETERS.

| SYMBOL | MILLIMETER | | | INCH | | |
|---|---|---|---|---|---|---|
| | MIN. | NOM. | MAX. | MIN. | NOM. | MAX. |
| A | — | — | 1.60 | — | — | 0.063 |
| A1 | 0.05 | — | 0.15 | 0.002 | — | 0.006 |
| A2 | 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 |

# 7 REVISION HISTORY

| Date | Revision | Description |
|---|---|---|
| 2015.5.31 | 1.00 | Preliminary version. |
| 2015.6.25 | 1.10 | Add new part number series NUC978YOxxY. |
| 2015.11.23 | 1.20 | Modify NUC978YOxxY pin out, including part selection guide, pin diagram and pin description.<br>Modify REFSEL (ADC_CONF[7:6]) description. |
| 2015.12.15 | 1.30 | Remove NUC978YOxxY series related information. |

## Important Notice

**Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".**

**Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.**

**All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.**

*Please note that all data and specifications are subject to change without notice.*
*All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.*